Semantic Aspects of Web Services

Semantic Web Services Draft

Draft book chapter
<v.8>

Creation date 12/02/2003 16:44
Modification date 10/29/2003 3:23 PM

Authors: Sinuhé Arroyo
Rubén Lara
Juan Miguel Gómez
David Berka
Ying Ding
Dieter Fensel

Contents

Semantic Aspects of Web Services

Sinuhé Arroyo, Rubén Lara, Juan Miguel Gómez, David Berka, Ying Ding, Dieter Fensel
IFI Research

CONTENTS

Abstract

Semantics promise to lift the Web to its full potential. The combination of machine processable semantics provided by the Semantic Web with current Web Service technologies have coined the term Semantic Web Services. Semantic Web Services offer the means to achieve a higher order level of value-added services by automating the task driven assembly of inter-organization business logics, thus making the Internet a global, common platform where agents communicate with each other. This chapter provides an introduction to the Semantic Web and Semantic Web Services paying special attention to its automation support. It details current initiatives in the EU and US, presents the most relevant frameworks towards the realization of a common platform for the automatic task driven composition of Web Services, sketches a comparison among them to point out their weaknesses and strengths, and finally introduces the most relevant technologies to describe services and their limitations.

1.1 Introduction

Current Web technology exploits very little of the capabilities of modern computers. Computers are used solely as information rendering devices, which present content in a human-understandable format. Incorporating semantics is essential in order to exploit all of the computational capabilities of computers for information processing and information exchange. Ontologies enable the combination of data and information with semantics, representing the backbone of a new technology called Semantic Web. By using ontologies, the Semantic Web will lift the current WWW to a new level of functionality where computers can query each other, respond appropriately, and manage semi-structured information in order to perform a given task.

If ontologies are the backbone of the Semantic Web, Web Services are its arms and legs. Web Services are self-contained, self-describing, modular applications that can be published, located, and invoked over the Web [Tidwell, 2000]. In a nutshell, Web Services are nothing but distributed pieces of software that can be accessed via the Web, roughly just another implementation of RPC. The potential behind such a simple concept resides in its possibilities of being assembled ad-hoc to perform tasks or execute business processes. Web Services will significantly further the development of the Web, by enabling automated program communication. Basically, they will allow the deployment of new complex value-added services. Web Services constitute the right means to accomplish the objectives of the Semantic Web. They not only facilitate the resources to access semantically enriched information, but also its assembly and combination possibilities, enhanced with semantic descriptions of their functionalities, will provide higher order functionality that will lift the Web to its full potential.

The combination of the Semantic Web and the Web Service technology has been named Semantic Web Services. Semantic Web Services may be the killer application of this emerging Web. The Semantic Web will provide the means to automate the use of Web Services, namely *discovery*, *composition* and *execution* [McIlraith et al. 2001]. The automation of these processes will make the task driven assembly of inter-organization business logics a reality. Such automation will transform the Internet in a global, common platform where agents (organizations, individuals, and software) communicate with each other to carry out various commercial activities, providing a higher order level of value-added services. The effects of this new technological development will expand areas such as Knowledge Management, Enterprise Application Integration, and e-Business.

In a nutshell the Semantic Web promises a revolution in human information access similar to the one caused by the telephone, and comparable to the invention of the steam engine. Such a revolution will count with ontologies and Web Services as its most important champions. The way computers are seen, and the WWW is understood will be changed completely, and effects will be felt in every aspect of our daily life.

This chapter provides an overview and detailed analysis of Semantic Web Services and related technologies. The contents are organized as follows: Section 1.2 presents an overview of the Semantic Web, a little bit of history, what is the Semantic Web, and a prospective of its possibilities; Section 1.3 introduces Semantic Web Services, fundamentals, actual state of development, and future trends and directions; 1.4 presents the most important initiatives towards the development of frameworks for Semantic Web Services, why frameworks are necessary, and what benefits they provide; Section 1.5 includes an overview of the most relevant upper ontologies developed to describe Semantic Web Services from a functional and non-functional point of view; and finally section 1.6 provides a summary together with a view of the future direction the Semantic Web Services technology will take.

1.2 Semantic Web

The Semantic Web is the next generation of the WWW where information has machine-processable and machine-understandable semantics. This technology will bring structure to the meaningful content of Web pages, being not a separate Web but an augmentation of the current one, where information is given a well-defined meaning. The Semantic Web will include millions of small and specialized reasoning services that will provide support for the automated achievement of tasks based on accessible information.

One of the first one to come up with the idea of the Semantic Web was the inventor of the current WWW, Tim Berners-Lee. He envisioned a Web where knowledge is stored on the meaning or content of Web resources through the use of machine-processable meta-data. The Semantic Web can be defined as "*an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in co-operation*" [Berners et al. 2001].

The core concept behind the Semantic Web is the representation of data in a machine interpretable way. Ontologies facilitate the means to realize such representation. Ontologies represent formal and consensual specifications of conceptualizations, which provide a shared and common understanding of a domain as data and information machine-processable semantics, which can be communicated among agents (organizations, individuals, and software) [Fensel, 2001]. Many definitions of "ontology" have been given during the last years. One that really fits the purpose in the computer science field is the one give by [Gruber, 1993]: "*An ontology is a formal, explicit specification of a shared conceptualization.*"

Ontologies bring together two essential aspects that help to push the Web to its full potential. On the one hand they provide (a) *machine processability* by defining formal semantics for information making computers able

to process it; on the other, they allow (b) *machine-human understanding* due to their ability to specify real-world semantics that permit to link machine processable content with human meaning using a consensual terminology as connecting element [Fensel, 2001].

Regarding Web Services, machine processability enabled by ontologies represent their most valuable contribution. They offer the necessary means to describe the capabilities of a concrete Web Service in a shared vocabulary that can be understood by every service requester. Such a shared functionality description is the key element towards task-driven automatic Web Service discovery, composition and execution of inter-organization business logics. It will allow to (1) locate different services which solely or in combination with other will provide the means to solve given task, (2) combine services to achieve a goal, and (3) facilitate the replacement of such services by equivalent ones, that solely or in combination can realize the same functionality, e.g. in case of failure while execution.

As ontologies are built (models of how things work) it will be possible to use them as common languages to describe Web Services and the payloads they contain in much more detail [Daconta et, al 2003].

1.2.1 Related projects

Currently there are many initiatives, private and public trying to bring Berners-Lee vision of the Semantic Web to a plausible reality [Ding et al. 2003]. Among the most relevant ones finance by the European Comission are projects like Onto-Knowledge [Ontoknowledge], DIP [DIP], SEKT [SEKT], SWWS [SWWS], Esperonto [Esperonto], Knowledge Web [KowledgeWeb], and OntoWeb [OntoWeb]. In the US one of the most relevant initiatives is the DARPA Agent Markup Language [DAML], supported by the US defense department's research funding agency. Other relevant DARPA-funded initiatives are High Performance Knowledge Bases [HPKB] now completed, and its follow up Rapid Knowledge Formation [RKF]. The National Science Foundation (NSF) has also sponsored some Semantic Web efforts. On its side, the W3C [W3C] has made an important effort towards the standardization of Semantic Web related technologies.

European and US researchers have been collaborating actively on standardization efforts [Ding et al. 2003]. A result of this collaboration is the joint Web Ontology Language OWL.

Some of the initiatives presented in this section count among their main partners major software vendors such as BT or HP, providing a valuable business point of view, which allows aligning the technology with market needs, and proving the interest that exists towards its development.

1.3 Semantic Web Services

In simple terms, Web Services are a piece of software accessible via the Web. By Service can be understood any type of functionality that software can deliver, ranging from mere information providers (such as stock quotes, weather forecasts, or news aggregation) to more elaborate ones that may have some impact in the real world (such as book sellers, plane ticket sellers, or e-banking), basically any functionality offered by the current Web can be envisioned as a Web Service. The big issue about Web Services resides in their capabilities of changing the Web from a static collection of information, to a dynamic place where different pieces of software can be assembled on the fly, to accomplish users' goals, expressed perhaps, in any natural language. This is a very ambitious goal, which at the moment cannot be achieved with the current state of the art technologies around Web Services. UDDI [Bellwood et al. 2002], WSDL [Christensen et al. 2001] and SOAP [Box et al. 2000] facilitate the means to advertise, describe, and invoke them, using semi-formal natural language terms, but do not say anything about what services can do, nor how they do it in a machine understandable and processable way. ebXML (electronic business XML) [ebXML] represents an alternative to UDDI, WSDL and SOAP, allowing to exchange and transport business documents over the Web using XML. It converges to the previously cited technologies, differing with them in the degree to which it recognizes business process modeling as a core feature [Newcomer, 2002]. All these initiatives lack of proper support for semantics and therefore human intervention is needed to actually discover, combine, and execute Web Services. The goal is to minimize any human intervention, so the assembly of Web Services can be done in a task-driven automatic way.

The Semantic Web in general and ontologies in particular are the right means to bridge the gap, and actually realize a dynamic Web. They provide the machine processable semantics that added on top of current Web Services realize the idea of the Semantic Web Services. Semantic Web Services combine Semantic Web and Web Service Technology. Semantic Web Services are defined as *"Self-contained, self-describing, semantically marked-up software resources that can be published, discovered, composed and executed across the Web in a task driven automatic way".*

What really makes a difference with respect to traditional Web Services is that they are semantically marked-up. Such enhancement enables:

Automatic Web Service Discovery. "*Location of services that abides to the service requester specification for a concrete task*".

Ideally, a task will be expressed using any natural language, and then translated to the ontology vocabulary suitable for the concrete application domain. Due to the fact that there will most likely be thousands of different ontologies for a concrete domain, –different service providers will express their business logics using different terms conventions, and different service requesters will express their requirements using different vocabularies– the appropriate sup-

port for ontology merging and alignment must be available. Semantically marked-up services will be published in semantically enhanced services' repositories where they can be easily located and their capabilities matched against the user's requirements. In a nutshell these repositories are traditional UDDI registries augmented with a semantic layer on top that provides machine processable semantics for the services registered.

As an example a service requester might say, "Locate all the services that can solve mathematical equations." An automatic service discovery engine will then surf all available repositories in search for services that fulfill the given task. The list of available services will probably be huge, so the user might impose some limitations, *functional* –how the service is provided– and *non-functional* –execution time, cost, and so on– in order to get an accurate and precise set of services.

Automatic Web Service composition. *"Assembly of services based on its functional specifications in order to achieve a given task and provide a higher order of functionality"*.

Once a list of available Web Services has been retrieved, it could happen that none of the available services completely fulfils the proposed task by itself, or other –cheaper, faster, vendor-dependent– combinations of services are preferred. In this case, some of the services would need to be assembled, using programmatic conventions to accomplish the desired task. During this stage, Web Services are organized in different possible ways based on functional (*pre-conditions* –conditions that must hold before the service is executed–, and *post-conditions* –conditions that hold after the service execution–), and non-functional (processing time, cost) requirements.

As an example a service requester might say, "Compose available services to solve the following set of mathematical operations [((a * 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, the fact that multiple addition can realize multiplication, should be stated in some *domain knowledge* –characterization of relevant information for an specific area– in a way that the service composer can understand to present all different possibilities to solve our set of operations. Different alternatives that lead towards the task accomplishments are then presented to the service requester who chooses among them the composition path that suits its needs the best.

Automatic Web Service execution*. "Invocation of a concrete set of services, arranged in a particular way following programmatic conventions that realizes a given task".*

When the available services have been composed and the service requester has chosen the execution path that best suits its needs according to the non-

functional requirements, the set of Web Services is to be executed. Each Web Service specifies one or more APIs that allows its execution. The semantic markup facilitates all the information regarding inputs required for service execution, and outputs returned once this has finished.

As an example a service requester might say, “Solve the following mathematical operations using the execution path which contains no multiplication services [((3 * 3) + 4) – 1]”. So what the user is actually saying is solve: [((3 + 3 + 3) + 4) – 1]. The addition service that realized the multiplication will be put into a loop, which will add 3 to itself three times, the result will be added to 4 and then 1 will be subtracted to obtain the result, as stated in the equation.

Prior to execution, the service may impose some limitations to 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 whatever other requirements considered necessary as part of a concrete business logic. In case that any of the execution’s constituents fails to accomplish its goal (e.g., network breaks down, service provider server breaks down, etc) a recovery procedure must be applied to replace the failed service with another service or set of services capable of finishing the work. Once the composed service has been successfully executed, it might be registered in any of the available semantically enhanced repositories. By doing so, a new level of functionality is achieved making available for reuse already composed and successfully executed services.

Current technology does not allow to fully realizing any of the parts of the Web Services’ automation process. Among the reasons are:

- Lack of fully developed markup languages
- Lack of marked up content and services
- Lack of semantically enhanced repositories
- Lack of frameworks that facilitate discovery, composition and execution
- Lack of tools and platforms that allow to semantically enrich current Web content

Essentially the technology is not yet mature enough, and there is a long path that should be walked in order to make it a reality. Some academia and private initiatives are very interested in developing these technologies, counting already both of them, with very strong and active efforts to grow it. Among them, [Sycara et al.1999] presented a Web Service discovery initiative using matchmaking based on a representation for annotating agent capabilities, so that they can be located and brokered. [Sirin et al. 2003] have developed a prototype that guides the user in the dynamic composition of Web Services. Task-driven Automatic Web Services composition (TAWc) an initiative carried by the *next Web generation* research group at Innsbruck University aims at providing a technology mature enough as to automatically compose Web Services independently of the application domain [TAWc].

1.4 Relevant Frameworks

Various initiatives aim to provide a fully-fledged modeling framework for Web Services that allows their automatic discovery, composition and execution. Among them, the most relevant ones are WSMF and WS-CAF. The benefits that will derive from these developments will constitute a significant evolution in the way the Web and e-business are understood providing the appropriate conceptual model for developing and describing Web Services and their assembly.

1.4.1 WSMF

The Web Service Modeling Framework (WSMF) is an initiative towards the realization of a fully-fledged modeling framework for Semantic Web Services, which counts on ontologies as its most important constituents. WSMF aims at providing a comprehensive platform to achieve automatic Web Service discovery, selection, mediation and composition of complex services, that is, to make Semantic Web services a reality and to exploit their capabilities. The WSMF description given in this section is based on [Fensel & Bussler, 2002]. WSMF specification is currently evolving, but none of the key elements presented here is likely to undergo a major change.

1.4.1.1 WSMF objectives

The WSMF objectives are as follows. ***Automated discovery*** includes the means to mechanize the task of finding and comparing different vendors and their offers by using machine processable semantics. ***Data mediation*** involves ontologies to facilitate better mappings among the enormous variety of data standards. ***Process mediation*** provides mechanized support to enable partners to cooperate despite differences in business logics, which are numerous and heterogeneous.

1.4.1.2 WSMF principles

WSMF revolves around two complementary principles, namely (1) ***strong decoupling***, where the different components that realize an e-commerce application should be as disaggregated as possible, hiding internal business intelligence from public access, allowing the composition of processes based on their public available interfaces, and carrying communication among processes by means of public message exchange protocols; and (2) ***strong mediation***, which will enable scalable communications allowing anybody to speak with everybody. To allow this m:m communication style, terminologies should be aligned, and interaction styles should be intervened.

In order to achieve such principles, a mapping among different business logics, together with the ability to establish the difference between public processes and private processes of complex Web Services are key characteristics the framework should support. Mediators provide such mapping func-

tionalities and allow expressing the difference among publicly visible workflows and internal business logics.

1.4.1.3 WSMF elements

WSMF consists of four different main elements: (1) ontologies that provide the terminology used by other elements; (2) goal repositories which define the tasks to be solved by Web Services; (3) Web Services as descriptions of functional and non-functional characteristics; and (4) mediators which bypass interoperability problems. A more detailed explanation of each element is given below.

Ontologies. They interweave human understanding with machine processability. In WSMF ontologies provide a common vocabulary used by other elements in the framework. They enable reuse of terminology, as well as interoperability between components referring to the same or linked terminology.

Goal repositories. Specify possible objectives a service requester may have when consulting a repository of services. A goal specification consists of two elements:

- **Pre-conditions.** Conditions that must hold previous to the service execution and enable it to provide the service.
- **Post-conditions.** Conditions that hold after the service execution and that describe what happens when a service is invoked.

Due to the fact that a Web Service can actually achieve different goals (i.e. Amazon can be used to buy books, and also as an information broker on bibliographic information about books), the Web Services descriptions, and the goals they are able to achieve are kept separately, allowing n:m mapping among services and goals. Conversely, a goal can be achieved but by different and eventually competing Web Services.

Keeping goal specifications separate from Web Service descriptions enhances the discovery phase, since it enables goal-based search of Web Services instead of functionality based search.

Web Services. In WSMF the complexity of a Web Service is measured in terms of its external visible description, contrary to the flow followed by most description languages, which differentiate them based on the complexity of the functionality, whether they can be broken into different pieces or not. Under traditional conventions, a complex piece of software such as an inference engine with a rather simplistic interface can be defined as elementary, while a much simpler software product such as an e-banking aggregator that can be broken down into several Web Services is considered complex. This reformulation of the definition that may look trivial has some relevant consequences:

- Web Services are not described themselves, but rather their interfaces which can be access via a network. By these means service providers can hide their business logic which usually is reflected in the services they offer.

- The complexity of a Web Service description provides a scale of complexity, which begins with some basic description elements, and gradually increases the description density, by adding further means to portray various aspects of the service.

As can be inferred from the previous paragraph, the framework describes services as black boxes, i.e., it does not model the internal aspects of how such a service is achieved, hiding all business logic aspects. The black box description used by WSMF consists of the following main elements:

- *Web Service name*. Unique service identifier used to refer to it.
- *Goal reference* Description of the objective that can fulfill stored in a goal repository.
- *Input and output data*. Description of the data structures required to provide the service.
- *Error data.* Indicates problems or error states.
- *Message exchange protocol*. Provides the means to deal with different types of networks and their properties providing an abstraction layer.
- *Non functional parameters*. Parameters that describe the service such as execution time, price, location, or vendor

In addition to this basic Web Service description, WSMF considers other properties of the service such as: (1) *failure*: when an error occurs affecting one of the invoked elements of a service and recovery is not possible, information about the reason of the failure must be provided; (2) *concurrent execution*: if necessary it should allow the parallel execution of different Web Services as a realization of the functionality of a particular one; (3) *concurrent data input and output*: in case input data is not available while a service is executing, it enables providing such input at a later stage and if required pass it from one invoker to another until reaching the actual requester; (4) *dynamic service binding*: in case that a service requires to invoke others to provide its service, a new proxy call is declared, the proxy allows referral to a service without knowing at define time to which concrete service is bound; (5) *data flow:* refers to the concrete proxy ports where data has to be forwarded; (6) *control flow*: defines the correct execution sequence among two or more services; (7) *exception handling*: upon failure services may return exception codes, if this is the case the means to handle a concrete exception must be defined; and (8) *compensation:* upon failure of an invoked Web Service a compensation strategy which specifies what to do can be defined.

Mediators. Deal with the service requester and provider heterogeneity, performing the necessary operations to enable full interoperation among them. It uses a peer to peer approach by means of a third party, facilitating this way a higher degree of transparency and better scalability for both requester and provider. Mediators facilitate coping with the inherit heterogeneity of Web-based computing environments, which are flexible and open by nature. This heterogeneity refers to:

- Data mediation. In terms of data representation, data types and data structures.

- Business logics mediation. Compensate for the mismatches in business logics.
- Message protocols mediation. Deals with protocol's heterogeneity.
- Dynamic service invocation mediation. In terms of cascading Web Service invocation. It can be done in a hard-wired way, but also, it can be more flexible by referring to certain (sub)-goals

These main elements of the framework together with, message understanding, and message exchange protocol layers are put together to provide automatic Web Service discovery, selection, mediation and composition of complex services.

1.4.2 WS-CAF

The Web Service Composite Application Framework (WS-CAF) [Bunting et al. 2003], represents another initiative that tries to address the application composition problem, by the development of a framework that will provide the means to coordinate long-running business processes in an architecture and transaction model independent way. WS-CAF has been designed with the aim of solving the problems that derive from the combined use of Web Services, to support information sharing and transaction processing. The discussion of WS-CAF is based on a draft of the specification made available on July 28th 2003 by Arjuna Technologies, Fujitsu Software, IONA Technologies, Oracle and Sun Microsystems. Due to the draft nature of the specification is very likely that changes will occur, even though the main elements of it may remain stable.

1.4.2.1 WS-CAF objectives

The main objectives of WS-CAF are: (1) ***interoperability***, to support various transaction models across different architectures; (2) ***complementarity***, to accompany and support current state of the art of business process description languages standards such as BPEL [Andrews et al. 2002], WSCI [Arkin et al. 2002] or BPML [Arkin, 2002] to compose Web Services; (3) ***compatibility***, to make the framework capable of working with existing Web Service standards such as UDDI [Bellwood et al. 2002], WSDL [Christensen et al. 2001], or WS-Security [Atkinson et al. 2002]; and (4) ***flexibility*** based on a stack architecture, to support the specific level of service required by Web Services combination.

1.4.2.2 WS-CAF principles and terminology

The framework definition is based on two principles, namely: ***interrelation*** or how participants share information and coordinate their efforts to achieve predictable results despite failure; and (2) ***cooperation*** to accomplish shared purposes. Web Services cooperation can range from performing operations over a shared resource, to its own execution in a predefined sequence.

The WS-CAF terminology makes use of the concepts of *participant*, *context*, *outcome*, and *coordinator* throughout the specification:

- **Participants**. Cooperating Web Services that take part in the achievement of a shared purpose.
- **Context**. Allows storing and sharing relevant information to participants in a composite process, to enable work correlation. Such data structure includes information like identification of shared resources, collection of results, and common security information.
- **Outcome**. Summary of results obtained by the execution of cooperating Web Services.
- **Coordinator**. Responsible for reporting participants about the outcome of the process, context management, and persisting participants outcome

1.4.2.3 WS-CAF elements

The framework consists of three main elements: (1) *Web Services Context (WS-CTX)* represents the basic processing unit of the framework, allowing multiple cooperating Web Services to share a common context; (2) *Web Services Coordination Framework (WS-CF)* builds on top of WS-CTX, and distributes organizational information relevant for the activity to participants, allowing them to customize the coordination protocol that better suits their needs; and (3) *Web Services Transaction Management (WS-TXM)* represents a layer on top of WS-CTX and WS-CF, defining transaction models for the different types of B2B interaction [Bunting et al. 2003].

WS-CTX. It is a lightweight mechanism that allows multiple Web Services participating in an activity to share a common context. It defines the links among Web Services through their association with a Web Service Context Service, which manages the shared context for the group. WS-CTX defines the context, the scope of the context sharing, and basic rules for context management [Bunting et al. 2003a].

WS-CTX allows stating starting and ending points for activities, provides registry facilities to control which Web Services are taking part in a concrete activity, and disseminates context information. WS-CTX is composed of three main elements, namely:

- *Context Service:* Defines the scope of an activity and how information about the context can be referenced and propagated.
- *Context:* Defines basic information about the activity structure. It contains information on how to relate multiple Web Services with a particular activity. The maintenance of contexts and its association with execution environments is carried by the Context Service which keeps a repository of contexts.
- *The Activity Lifecycle Service (ALS)*: Is an extension of the Context Service, which facilitates the activity's enhancement with higher-level interfaces. Whenever a context is required for an activity and it does not exactly suit the necessities of the particular application domain, the Context Service issues a call to the registered ALS which provides the required addition to the context.

Essentially WS-CTX allows the definition of activity in regard to Web Services, provides the means to relate Web Services to one another with respect to a particular activity, and defines Web Services mappings onto the environment.

WS-CF. It is a sharable mechanism that allows management of lifecycles, context augmentation, and guarantees message delivery, together with coordination of the interactions of Web Services that participate in a particular transaction, by means of outcome messages [Bunting et al. 2003b].

WS-CF allows the definition of the starting and ending points for coordinated activities, the definition of points where coordination should take place, the registration of participants to a concrete activity, and propagation of coordination information to activity participants.

The WS-CF specification has three main architectural components, namely:

- *Coordinator*: Provides the means to register participants for a concrete activity.
- *Participant*: Specifies the operation(s) performed as part of the coordination sequence processing.
- *Coordination Service*: Determines a processing pattern used to define the behavior of a particular coordination model.

In a nutshell WS-CF defines the core infrastructure for Web Services Coordination Service, provides means to define Web Services mappings onto the environment, delineates infrastructure support, and finally allows concreting the responsibilities of the different WS-CF subcomponents.

WS-TXM. It comprises protocols that facilitate support for various transaction processing models, providing interoperability across multiple transaction managers by means of different recovery protocols [Bunting et al. 2003c].

WS-TXM provides the core infrastructure for Web Services Transaction Service, facilitates means to define Web Services mappings onto the environment, defines an infrastructure to support an event communication mechanism, and finally establishes the roles and responsibilities of the WS-TXM components.

Roughly speaking WS-CAF enables the sharing of Web Service's context viewed as an independent resource, provides a neutral and abstract transaction protocol to map to existing implementation, presents an application transaction level dependent upon application needs, includes a layered architecture that allows applications to use the level of service they need, not forcing them to work with more functionalities than required, and includes a great degree of interoperability thanks to the use of vendor neutral protocols.

1.4.3 Frameworks comparison

Both of the initiatives introduced so far present a solution to compose applications out of multiple Services using different approaches, and currently at different development stages. While WSMF adopts a more formal philosophy that focuses on how Web Services should be described to achieve composition based on the paradigms of the Semantic Web, providing an extensive covering of all the different aspects in the field, WS-CAF takes a more hands-on approach from a service execution point of view and its requirements to address the same problem.

WS-CAF puts special interest to deal with failure, context management, transaction support, and effort coordination, while WSMF uses ontologies as a pivotal element to support its scalable discovery, selection, mediation and composition aim.

WS-CAF already counts with a layered architecture for services execution which shows a more mature state of development, while WSMF efforts have focused on establishing the pillars of what a complete framework to model Semantic Web Services should look like, providing a broader coverage of the subject.

Both initiatives count on the support of major software vendors which will result in the development of a solid technology with a clear business and user driven aim.

Table 1 summarizes the functionality provided by both frameworks regarding its automation support. As a comment the mediation must account for process, protocol, data, and service invocation mediation, while execution must take care of failure, context management, transaction support, effort coordination, concurrent execution, concurrent input and output, exception handling and compensation strategies in case of failure.

Framework	Automation support
WSMF	- Discovery - Selection - Mediation - Composition - Execution
WS-CAF	- Mediation - Composition - Execution

Table 1.*Summary of intended purpose of WSMF and WS-CAF*

The Semantic Web is the future of the Web. It counts on ontologies as a key element to describe services and provide a common understanding of a domain, and Semantic Web Services as its killer application. The development foreseen for this technology presents WSMF as a stronger candidate from an impact and visibility point of view, as well as a long term runner, due to the use of the paradigms of the Semantic Web. WS-CAF doesn't use ontologies in its approach, but puts strong emphasis in the coordination of multiple and possibly incompatible transaction processing models across different architectures, which in the short term, will bridge a technology gap, and will enable the cooperation of both frameworks in a near future.

1.5 Epistemological ontologies for describing services

Upper level ontologies supply the means to describe the content of on-line information sources. In particular, and regarding Web Services, they provide the means to mark them up describing their capabilities and properties in unambiguous, computer interpretable form. In the coming sections a description of DAML-S and its actual relation with OWL-S is presented.

1.5.1 DAML-S and OWL-S

The Defense Advanced Research Projects Agency (DARPA) Agent Markup Language (DAML) for Services (DAML-S) [DAML-S 2003] is a collaborative effort by BBN Technologies, Carnegie Mellon University, Nokia, Stanford University, SRI International and Yale University to define an ontology for semantic markup of Web Services.

DAML-S sits on top of WSDL at the application level, and allows the description of knowledge about a service in terms of *what* the service does –represented by messages exchanged across the wire between service participants–, *why* does it do it, and *how* it does it [DAML-S 2003].

The aim of DAML-S is to make Web Services computer interpretable enabling their automated use. Current DAML-S releases (up to 0.9) have been built upon DAML+OIL, but to ensure a smooth transition to OWL (Web Ontology Language), 0.9 release and subsequent ones will be based also on OWL. Roughly speaking, DAML-S refers to the ontology built upon DAML+OIL, while OWL-S refers to the one built upon OWL.

1.5.2 DAML-S elements

DAML-S ontology allows the definition of knowledge that states what the service requires from agents, what it provides them with, and how does it do it. To answer such questions it uses three different elements: (1) ***Service Profile*** which facilitates information about the service and its provider to enable its discovery, (2) ***Service Model***, which makes information about how to use the service available, and (3) ***Service Grounding***, which specifies how communications among participants are to be carried on and how the service will be invoked.

Service Profile. It plays a dual role; service providers use it to advertise the services they offer, while service requesters can use it to specify their needs. It presents a public high-level description of the service that states its intended purpose in terms of: (1) *service description*, information presented to the user, when browsing service registries, about the service and its provider, that helps to clarify whether the service meets concrete needs and constraints such as security, quality requirements, and locality; (2) *functional behavior,* description of duties of the service; and (3) *functional attributes*, additional

service information such as time response, accuracy, cost or classification of the service.

Service Model. It allows a more detailed analysis of the matching among service functionalities and user needs, enabling service composition, activity coordination and execution monitoring. It permits the description of the functionalities of a service as a *process*, detailing control and data flow structures. The Service Model includes two main elements, namely: (1) *Process Ontology*, which describes the service in terms of inputs –information necessary for process execution–, outputs –information that the process provides–, preconditions –conditions that must hold prior the process execution–, effects–changes in the world as a result of the execution of the service–, and if necessary component sub-process; and (2) *Process Control Ontology* that describes process in terms of its state –activation, execution and completion–. Processes can have any number of inputs, outputs, preconditions and effects. Both outputs and effects can have associated conditions.

The process ontology allows the definition of *atomic*, *simple* and *composite* processes:

- **Atomic processes.** They are directly invocable, have no subprocesses and execute in a single step from the requester perspective.
- **Simple processes.** They are not directly invocable, and represent a single-step of execution. They are indented as elements of abstraction that simplify the composite process representation, or allow different views of an atomic process.
- **Composite processes.** They decompose into other processes, either composite or non-composite, by means of control constructs.

Service Grounding. Specifies details regarding how to invoke the service, (protocol, messages format, serialization, transport and addressing). The grounding is defined as mapping from abstract to concrete realization of service's descriptions in terms of inputs and outputs of the atomic process, realized by means of WSDL and SOAP. The service grounding shows how inputs and outputs of an atomic process are realized as messages that carry inputs and outputs.

In a nutshell DAML-S is an upper ontology used to describe Web Services and includes elements intended to provide automated support for the Semantic Web Service's tasks. Table 1 summarizes the facilities covered by each upper level concept in the DAML-S ontology:

Upper level concept	Automation support
Profile	- Discovery
Model	- Planning - Composition - Interoperation -Execution monitoring
Grounding	- Invocation

Table 2.*Summary of intended purpose of DAML-S upper level concepts*

The following example (adapted from [Ankolekar, et al. 2002]) provides detailed information on how to use the DAML-S ontology to describe the pieces of software that can build a service and how to define the grounding of each one of these basic processes. The example presents a Web aggregation service. Given the user's login, his password, and the area of interest to which the aggregation is to be performed, it consolidates information disseminated over different Web sources, (i.e., banks in which the user has an account, telephone companies the user works with, news services to which the user has signed), and presents it avoiding the process of login and browsing each one of the sources, in search for the desired piece of information.

First the service must be described in terms of the different constituents that comprise it, specifying the type of process (atomic, simple, composite). In this case a description of the news aggregation service, as an atomic process, is provided.

```
<daml:Class rdf:ID="NewsAggregation">
        <rdfs:subClassOf rdf:resource="&process;#AtomicProcess"/>
</daml:Class>
```

Then, the set of different properties associated with each one of the programs of the service must be defined. An input for the news aggregation service could be the language used to write the gathered news.

```
<rdf:Property rdf:ID="language">
        <rdfs:subPropertyOf rdf:resource="&process;#input"/>
        <rdfs:domain rdf:resource="#NewsAggregation"/>
        <rdfs:range rdf:resource="&xsd;#string"/>
</rdf:Property>
```

Next the grounding must be defined and related to the service constituent. For this purpose a restriction tag is used establishing that the NewsAggregation program has *grounding* and that it is identified by the name NewsAggregationGrounding. Basically we are stating that every instance of the class has an instance of the hasGrounding property, with the value NewsAggregationGrounding.

```
<daml:Class rdf:about="NewsAggregation ">
        <daml:sameClassAs>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#hasGrounding"/>
                        <daml:hasValue rdf:resource="# NewsAggregationGrounding"/>
                </daml:Restriction>
        </daml:sameClassAs>
</daml:Class>
```

Finally and example of a DAML-S grounding instance is presented. It is important to notice that URIs (#ConsolidateNews, #NewsAggregationInput, etc) correspond to constructs in a WSDL document which is not shown here.

```
<grounding:WsdlGrounding rdf:ID=" NewsAggregationGrounding ">
        <grounding:wsdlReference rdf:resource="http://www.w3.org/TR/2001/NOTE-wsdl-
20010315">
        <grounding:otherReferences rdf:parseType="daml:collection">
```

```
            "http://www.w3.org/TR/2001/NOTE-wsdl-20010315"
            "http://schemas.xmlsoap.org/wsdl/soap/"
            "http://schemas.xmlsoap.org/soap/http/"
        </grounding:otherReferences>
        <grounding:wsdlDocuments rdf:parseType="daml:collection">
            "http://service.com/aggregation/newsAggregation.wsdl"
        </grounding:wsdlDocuments>
        <grounding:wsdlOperation
            rdf:resource="http://service.com//newsAggregation#ConsolidateNews"/>
        <grounding:wsdlInputMessage
            rdf:resource="http://service.com//newsAggregation.wsdl#NewsAggregationIn
        put"/>
        <grounding:wsdlInputMessageParts rdf:parseType="daml:collection">
            <grounding:wsdlMessageMap>
                <grounding:damlsParameter rdf:resource=#NewsLanguage>
                <grounding:wsdlMessagePart
                    rdf:resource="http://service.com//newsAggregation.wsdl#agg
                regatedNews">
            </grounding:wsdlMessageMap>

            ... other message map elements ...

        </grounding:wsdlInputMessageParts>
        <grounding:wsdlOutputMessage
            rdf:resource="http://service.com//newsAggregtion.wsdl#NewsAggregationOut
        put"/>
        <grounding:wsdlOutputMessageParts rdf:parseType="daml:collection">

        ... similar to wsdlInputMessageParts ...

</grounding:wsdlOutputMessageParts>
<grounding:WsdlGrounding>
```

1.5.3 Limitations

DAML-S and OWL-S early beta stage can be seen as a gruyere cheese with lots of holes to fill [Payne, 2003]. It has numerous limitations that will be overcome in a near future:

- **One-to-one mapping between Profiles and Service Models**. It prevents the reuse of profiles and n:m mappings between profiles and concrete services.
- **Separation between public and private business logic**. It does not allow hiding the internals of the service, exposing its business logic to requesters.
- **Lack of conversation interface definition**. The service model could be used as the conversational interface if the service is published as a composite service, but this is not stated as the intended purpose of the service model.
- **Interface overloading.** The defined WSDL grounding doesn't support publishing one single service interface for a service accepting different inputs.
- **Pre-conditions and effects.** No means to describe pre-conditions and effects are given, and the definitions of these elements are not clear enough.

- **Constructs**. The constructs defined for the process model may not be sufficient.

The path followed by this standard seems to be correct one. Lots of efforts are being put to work to overcome many of the actual problems and limitations, which will undoubtedly be solved in the near future, providing a solid standard to consistently describe Semantic Web Services.

1.6 Summary

The Semantic Web is here to stay. It represents the next natural step in the evolution of current Web. It will have a direct impact in areas such as e-Business, Enterprise Application Integration, and Knowledge Management, and an indirect impact on many other applications affecting our daily life. It will help create emerging fields where knowledge is the most precious value, and will help to further the development of existing ones.

The Semantic Web will develop a complete new concept of Web by extending the current one, alleviating the information overload problem, and bringing back computers to its intended use as computational devices, and not just information rendering gear. Ontologies are the backbone of this revolution due to its potential for interweaving human understanding of symbols with machine processability [Fensel, 2002]. Ontologies will enable to semantically enhance Web Services, providing the means that facilitate the task-driven automatic discovery, composition and execution of inter-organization business logics, and thus making Semantic Web Services the killer application of the Semantic Web.

A shared functionality description is the key element towards task-driven automatic Web Service discovery, composition and execution of inter-organization business logics. It will allow to (1) locate different services which solely or in combination with others will provide the means to solve given task, (2) combine services to achieve a goal, and (3) facilitate the replacement of such services by equivalent ones, that solely or in combination can realize the same functionality – such as in case of failure while execution.

Many business areas are giving a warm welcome to the Semantic Web and Semantic Web Services. Early adopters in the biotechnology or the medicine field are already aware of its potentials to organize knowledge and infer conclusions from available data. The juridical field has already realized of its benefits to organize and manage large amounts of knowledge in a structured and coherent way. The interest of professionals from this sector towards the Semantic Web is rapidly gaining momentum, being forecasted as an area where the Semantic Web will make a difference. Editorials, libraries and basically anyone dealing with information will soon realized of the benefits of Semantic Web Services due to the new business paradigm it provides. Business

services will be published on the Web and will be marked up with machine processable semantics, to allow its discovery and composition, providing a higher level of functionality, adding increasingly more complex layers of services, and relating business logics from different companies in a simple and effective way.

The hardest problem the Semantic Web Services must overcome is its early development stage. Nowadays the technology is not mature enough to accomplish its promises. Initiatives in Europe and US are gaining momentum, and the appropriate infrastructure, technology, and frameworks are currently being developed. Roughly speaking there is a gap between the current Web and the Semantic one in terms of annotations. Pages and Web resources must be semantically enriched in order to allow automatic Web Services interoperation. Once this is solved, the Semantic Web and Semantic Web Services will become a plausible reality and the Web will be lifted to its full potential causing a revolution in human information access. The way computers are perceived, and the Web is understood will be completely changed, and the effects will be felt in every detail of our daily life.

References

[Andrews et al. 2002] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Ivana Trickovic, Sanjiva Weerawarana, "Business Process Execution Language for Web Service (BPEL4WS) 1.1", http://www.ibm.com/developerworks/library/ws-bpel, August 2002.

[Ankolekar, et al. 2002] Anupriya Ankolekar, Mark Burstein1, Jerry R. Hobbs, Ora Lassila, David Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, Katya Sycara, "DAML-S: Web Service Description for the Semantic Web" International Semantic Web Conference, ISWC 2002.

[Arkin, 2002] Assaf Arkin. "Business Process Modelling Language", http://www.bpmi.org/, 2002.

[Arkin et al. 2002] Assaf Arkin, Sid Askary, Scott Fordin, Wolfgang Jekeli, Kohsuke Kawaguchi, David Orchard, Stefano Pogliani, Karsten Riemer, Susan Struble, Pal Takacsi-Nagy, Ivana Trickovic, Sinisa Zimek, "Web Service Choreography Interface 1.0", http://wwws.sun.com/software/xml/developers/wsci/wsci-spec-10.pdf, 2002.

[Atkinson et al. 2002] Bob Atkinson, Giovanni Della-Libera, Satoshi Hada, Maryann Hondo, Phillip Hallam-Baker, Johannes Klein, Brian LaMacchia, Paul Leach, John Manferdelli, Hiroshi Maruyama, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, John Shewchuk, Dan Simon, "Web Services Security (WS-Security) 1.0", http://www-106.ibm.com/developerworks/library/ws-secure, 2002.

[Bellwood et al. 2002] Tom Bellwood, Luc Clément, David Ehnebuske, Andrew Hately, Maryann Hondo, Yin Leng Husband, Karsten Januszewski, Sam Lee, Barbara McKee, Joel Munter, Claus von Riegen, "UDDI Version 3.0. Published Specification", http://uddi.org/pubs/uddi-v3.00-published-20020719.htm, 2002.

[Berners et al. 2001] Tim Berners-Lee, James Hendler, and Ora Lassila, "The Semantic Web". *Scientific American*, 284(5):34-43, 2001.

[Box et al. 2000] Don Box, Ddavid Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendel-sohn, Henrik F. Nielsen, Satish Thatte, Dave Winer, "Simple Object Access Protocol (SOAP) 1.1", http://www.w3.org/TR/SOAP/, 2000.

[Bunting et al. 2003] Doug Bunting, Martin Chapman, Oisin Hurley, Mark Little, Jeff Mischkinsky, Eric Newcomer, Jim Webber, and Keith Swenson, "Web Service Composite Application Framework (WS-CAF), http://developers.sun.com/techtopics/Webservices/wscaf/primer.pdf, 2003.

[Bunting et al. 2003a] Doug Bunting, Martin Chapman, Oisin Hurley, Mark Little, Jeff Mischkinsky, Eric Newcomer, Jim Webber, and Keith Swenson. "Web Service Context (WS-Context)", http://developers.sun.com/techtopics/Webservices/wscaf/wsctx.pdf, 2003.

[Bunting et al. 2003b] Doug Bunting, Martin Chapman, Oisin Hurley, Mark Little, Jeff Mischkinsky, Eric Newcomer, Jim Webber, and Keith Swenson. "Web Service Coordination Framework (WS-CF)", http://developers.sun.com/techtopics/Webservices/wscaf/wscf.pdf, 2003.

[Bunting et al. 2003c] Doug Bunting, Martin Chapman, Oisin Hurley, Mark Little, Jeff Mischkinsky, Eric Newcomer, Jim Webber, and Keith Swenson. "Web Service Transaction management (WS-TXM)", http://developers.sun.com/techtopics/Webservices/wscaf/wstxm.pdf, 2003.

[TAWc] TAWc, www.nextWebgeneration.org/projects/TAWc.

[Christensen et al. 2001] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, "WSDL Web Services Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl, 2001.

[Daconta et, al 2003] Michael C. Daconta, Leo J. Obrst and Kevin T. Smith, "The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management", Wiley Publishing, Indianapolis, Indiana, 2003.

[DAML-S 2003] The DAML services coalition: "DAML-S: Semantic Markup for Web Services (version 0.9)", http://www.daml.org/services/daml-s/0.9/daml-s.pdf, 2003.

[DAML] DARPA Agent Markup Language (DAML), www.daml.org.

[Ding et al. 2003] Ying Ding, Dieter Fensel, and Hans-Georg Stork, "The Semantic Web: from Concept to Percept", to appear in Austrian Artificial Intelligence Journal (OGAI), 2003.

[DIP] Data, Information and Process Integration with Semantic Web Services, informatik.uibk.ac.at/nextWebgeneration/projects/dip.

[ebXML] electronic business XML (ebXML), www.ebxml.org/specs.

[Esperonto] Esperonto, esperonto.semanticWeb.org.

[Fensel, 2001] Dieter Fensel, "Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce", Springer-Verlag, Berlin, 2001.

[Fensel & Bussler, 2002] Dieter Fensel, Christoph Bussler "The Web Service Modeling Framework WSMF", *Electronic Commerce Research and Applications*, 1(2), 2002.

[Fensel, 2002] Dieter. Fensel, “Semantic Enabled Web Services” XML-Web Services ONE Conference, June 7, 2002.

[Gruber, 1993] Thomas R. Gruber, “A Translation Approach to Portable Ontology Specifications”, *Knowledge Acquisition*, 5:199-220, 1993.

[HPKB] High Performance Knowledge Bases (HPKB) reliant.teknowledge.com/HPKB.

[KowledgeWeb] Knowledge Web, knowledgeWeb.semanticWeb.org

[Martin et al. 2001] James Martin, “Web Services: The Next Big Thing”, *XML Journal 2*, 2001. http://www.sys-con.com/xml/archivesbad.cfm.

[McIlraith et al. 2001] Sheila A. McIlraith, Tran C. Son, Honglei Zeng, "Semantic Web Services". *IEEE Intelligent Systems*, 16(2), March/April, 2001.

[Newcomer, 2002] Eric Newcomer, “Understanding Web Services: XML, WSDL, SOAP UDDI”. Addison Wesley, 2002.

[OntoKnowledge] Ontoknowledge, www.ontoknowledge.org.

[OntoWeb] OntoWeb, www.ontoWeb.org.

[Payne, 2003]. Terry Payne, The First European Summer School on Ontological Engineering and the Semantic Web. http://minsky.dia.fi.upm.es/sssw03, Cercedilla (Spain) July 21-26, 2003.

[Peer, 2002] Joachim Peer. “Bringing Together Semantic Web and Web Services”. *First International Semantic Web Conference*, Sardinia, Italy, June 2002.

[RKF] Rapid Knowledge Formation (RKF), reliant.teknowledge.com/RKF.

[SEKT] Semantic Knowledge Technologies, sekt.semanticWeb.org.

[Sirin et al. 2003] Evren Sirin, James Hendler, Bijan Parsia, "Semi-automatic Composition of Web Services using Semantic Descriptions", Proceedings of the 1st Workshop on Web Services: “*Modeling, Architecture and Infrastructure*” (WSMAI-2003), In conjunction with ICEIS 2003, Angers, France, pp. 17-24, April 2003.

[Sycara et al.1999] Katya Sycara, Matthias Klusch, Seth Widoff, “Dynamic Service Matchmaking among Agents in Open Information Environments”, *ACM SIGMOD Record*, vol. 28, no. 1, pp. 47–53, Mar. 1999.

[SWWS] Semantic Web Enabled Web Services, swws.semanticWeb.org.

[Tidwell, 2000] Doug Tidwell, “Web Services: the Web's next revolution”, http://www-106.ibm.com/developerworks/edu/ws-dw-wsbasics-i.html.

[W3C] World Wide Web Consortium (W3C), www.w3.org/2001/sw.

Acknowledgements

The authors would like to thank SungKook Han, Holger Lausen, Michael Stolberg, Jos de Brujin, and Anna Zhdanova for the many hours spent discussing different issues that helped in great manner to make this work possible. Also thanks to Alexander Bielowski for his feedback regarding English writing.

Index