Chapter XIII
Semantic Web Services:
Towards an Appropriate Solution to Application Integration

Antonio J. Roa-Valverde
University of Málaga, Spain

Ismael Navas-Delgado
University of Málaga, Spain

José F. Aldana-Montes
University of Málaga, Spain

ABSTRACT

The potential growth of applications distributed over a network and the large number of users has created the need for an infrastructure which can support increasing interaction among such users and applications. In this environment a scalable solution is needed. This problem known as application integration is addressed in this chapter, and it is combined with Semantic Web technology. After introducing the different techniques to overcome the application integration challenge current trends in Semantic Web Services are discussed and the most recent R&D projects and a selection of available tools are presented. Also discussed is the use of ESB as a suitable mechanism to deploy Semantic Web Services.

INTRODUCTION

Nowadays, Web Services technology has become extremely important in the distributed computing field. This technology, based on a set of standards recommended by the W3C, enables applications to communicate and exchange data over the Internet (Deitel, 2003). Web services benefit from object-oriented programming techniques because they allow developers to build applications from existing software components. This feature facilitates integration at the enterprise level which explains
why large companies in the ICT (Information and Communication Technologies) sector are involved in its development.

In parallel with the increasing Web services available, the Semantic Web has emerged. The idea of this Web is that machines are not only able to present information but can also “understand” it. It is not however an entirely new Web, rather, it is an extension of the existing one, in which the information has a well-defined meaning thereby enabling computers and people to work in conjunction.

New techniques are being developed to overcome the limitations of traditional Web Services and are brought together in the concept of Semantic Web Services (SWSI, 2002), which are capable of integrating the formalism provided by the Semantic Web to carry out automatic service discovery, composition and execution.

Currently there is a significant effort being made to achieve standardization of the technology behind the Semantic Web Services, which is apparent in the various proposals submitted by different authors to the W3C. The most relevant proposals are WSMO (WSMO, 2005) and OWL-S (OWL-S, 2004) which are reflected in recently developed work.

A prerequisite for the development of a semantic layer to evolve current applications is to know the existing technologies and tools available. An initial analysis is necessary of the features that the existing systems have for annotating Web Services. This analysis will be conducted as a comparison of the different existing technologies with emphasis on the needs we may be particularly interested in. Certain aspects to be taken into account when carrying out this study may be the number of tools available to work with the specific technology, the impact of this technology, which can be seen by the number of related works, and the possibility of it becoming a standard.

Firstly, in section 2 we provide an introduction to distributed computing. We talk about mechanisms such as RPC and related approaches such as CORBA or RMI. After that, we introduce the most important aspects of Service Oriented Architectures (SOA) and the basis of Web Services. We also discuss the application integration problem. We comment on the most important concepts in this area, namely EAI and ESB. So, we explain the existing problem in each approach and a possible solution to it. Secondly, in section 3 we describe the state-of-the-art of Semantic Web Service (SWS) technologies focusing on the current proposals. These approaches enable the life cycle of Semantic Web Services, thus we consider it relevant to describe the current trends in this area and related projects carried out by researchers in this area are discussed. We also show the best known tools for annotation, registering, discovery and composition of Semantic Web Services. In section 4 we present our ongoing work to improve the use of an ESB in application integration. In this way, we will provide a brief description of the ESB concept and how to combine this technology with Semantic Web technologies. In section 5 we discuss future trends in this area and finally, in section 6 our conclusions are presented.

**BACKGROUND**

With the expansion of the local networks and the Internet in the 90s, a need arose for applications to communicate with each other. This communication consists in exchanging data and computational effort among different processes with the ultimate goal of building a distributed system. Several solutions have been proposed to overcome the problem of communicating different applications across a network and using this network within a global environment. This problem, known as application integration, has now become a priority task among system providers.

The application integration problem is closely related to the programming style used. So, the object-oriented paradigm provides a good way to re-use solutions developed in other domains.
Object oriented software architecture is hierarchically structured to build complex and reusable software. On the lowest level, functionalities are encapsulated in an object. A set of interacting software objects is collected into a component, a fact which facilitates the task of integrating different applications in a more complex system. However, it is not enough enough to solve the integration problem and it is also necessary to facilitate the communication among applications. In order to do so, several approaches have been proposed from different providers. These already had at their disposal their own object oriented models, extended to allow communication over the network. The OMG1 established the IIOP (Internet Inter-ORB Protocol) as the standard protocol for network communication in CORBA. Microsoft, on the other hand, introduced DCOM (Distributed COM) and Sun Microsystems proposed Java RMI (Remote Method Invocation) for the Java users. These approaches all use a mechanism called RPC (Remote Procedure Call). RPC is a protocol that one program can use to request a component from a program located in another computer without having to understand the different network details.

The main drawback of the previous approaches is the lack of interoperability among different protocols, i.e., users that use RMI can easily invoke RMI compatible servers, but they encounter difficulties if they want to communicate with DCOM or CORBA. These problems increased with the “explosion” of the Web because applications had to communicate with other ones running on different platforms. So, the question of heterogeneity was added to the communication problem. Furthermore, the protocols described above may have security issues because it is necessary to open ports that usually are unknown, something, system administrators are usually against.

Thus, the use of an owner technology in such a heterogeneous environment makes the standardization process more difficult. This is the main reason for not using the previous protocols. We can conclude then, that in an environment such as the Internet the use of a particular technology cannot be restricted.

A first approach to solve the interchange of data among different applications arrived in the second half of the 90s with the appearance of XML. Developers found the possibility of expressing the information in a descriptive and uniform way. This fact allowed overcoming the heterogeneity among different operating systems and programming languages.

One of the best known protocol implementations using XML is XML-RPC which allows users/programmers to call remote procedures in a platform independent way. XML-RPC is the predecessor of SOAP (Simple Object Access Protocol), which will be described later when we introduce the basics of Web Services.

A World of Services

In the previous section we established that functionalities are encapsulated in an object. Components collect a set of related objects. The Service Oriented Architecture (SOA) consequently extends this hierarchical structure to distributed systems and defines services as a collection of components. From the user’s point of view a service can be defined as a set of activities that try to satisfy client necessities and have the following characteristics (Govern, 2003):

- Services are self contained and modular
- Services are discoverable and dynamically bound
- Services stress interoperability
- Services are loosely coupled, reducing artificial dependencies to their minimum
- Services have a network-addressable interface
- Services have coarse-grained interfaces in comparison to finer-grained interfaces of software components and objects
- Services can be composed
Thus, SOA is an architectural style for achieving loose coupling among interacting software agents, i.e. to minimize their artificial dependencies. Services are only a fragment of the SOA world and Web Services are the interface for interacting with these services. It is necessary therefore to distinguish Web Services from the service implementations. SOA may or may not use Web Services but Web Services provide a simple way towards SOA.

Coming back to the previous section, Web Services achieve high interoperability by using languages based on XML and standard protocols such as HTTP or SMTP. In this sense, Web Services overcome the drawbacks of technologies such as CORBA, DCOM and RMI and allow developers to build modular and auto-descriptive applications that can be published, registered and invoked from anywhere on the Web in a platform independent way. However, Web Services have some drawbacks that were solved in those technologies, i.e., quality of service and security issues. On the one hand, because they rely on the HTTP protocol, many experts have criticized the Web Services performance. This problem is inherent to TCP/IP since upper levels have a higher abstraction level but a lower performance. Experts argue that Web Services are not suitable for example for building applications which rely on high performance computation. Web Services were proposed to overcome communication problems among heterogeneous systems. On the other hand, nowadays researchers spend their efforts on improving the security of the current standards as can be seen in the recent versions. Despite these objections, Web Services have become today a commonly accepted technology among ICT companies.

Now, we are going to describe briefly the three basic concepts around the Web Service technology. Web Services are described using a formal language that is platform independent, WSDL (Web Service Description Language). These descriptions are published to enable the location and understanding of the services. Web Services are accessed through SOAP (Simple Object Access Protocol). In order to enable the location and integration of remote services the service provider

Figure 1. Web Service Stack (extracted from http://www.w3.org/TR/ws-arch/)
can make use of UDDI (Universal Description, Discovery and Integration). Figure 1 depicts the relationship among the standards supported by the W3C.

The way to use Web Services consists in three easy steps that involve each one of the standards presented above: description, discovery and invocation.

Three main actors take part in the Web Service usage process: the Web Service broker, the Web Service requester and the Web Service provider. The Web Service provider is the actor who develops and offers one or more services. To make this possible the provider must describe the offered service in an understandable way using WSDL. These service descriptions in WSDL will be stored in a public registry called UDDI to facilitate the service discovery by other requesters. UDDI acts as a service broker in the sense that it stores the necessary information to allow interaction between the Web Service requester and the Web Service provider. When a requester tries to find a Web Service that satisfies its necessities, it must look up the UDDI to obtain the service description. This service description contains the information required to allow the requester to begin a communication process by exchanging SOAP messages with the Web Service provider. Figure 2 summarizes these ideas.

Trends in Enterprise Integration

As we have observed previously, application integration has become a challenge for many ICT leaders. In addition to the previously commented distributed models other proposals have emerged to solve the integration problem in business applications. In this section, we will describe the EAI, SOA and ESB concepts in order to clearly distinguish between them. In this section we will also discuss architectural options available for enterprise integration and in which sense these options are suitable for.

Chappell (2004) states that “indications are that only a small percentage of enterprise applications are connected, regardless of the technology being used. According to a research report from Gartner Inc., that number is less than 10%. Another statistic is even more surprising of the applications that are connected, only 15% are using formal integration middleware” (p. 28). The lack of experience and knowledge in the use of these technologies may be a possible explanation for this situation. For this reason, we consider this section a need.

Enterprise Application Integration (EAI)

Enterprise Application Integration is a way of making diverse business applications communicate with each other to achieve a business objective in a seamless and reliable way, irrespective of platform and geographical location. EAI involves methodologies, processes, tools and technologies used to connect systems, data and workflows within one or between several companies. If the connection is about systems, data or processes among different companies it is called Business to Business Integration (B2Bi). In this work we
focus on the connection of systems and processes, leaving out the data integration or Enterprise Information Integration (EII).

EAI systems have evolved over time and can be classified in the architectures explained below. The first is the point-to-point integration architecture (Figure 3) also known as “accidental architecture” because it is something that nobody actually set out to create, but it is the result of years of accumulating direct connections (Chappell, 2004). This architecture is not easily extensible because for each additional/application system that needs to be integrated, the number of connectors will increase exponentially. This fact makes it very difficult to manage.

The second architecture is known as the hub & spoke integration architecture (Figure 4). This one is a centralized solution that can be managed easily, but it is not easily scalable and was developed to overcome the problems present in the accidental architecture. The hub & spoke approach aims to reduce the number of connections among heterogeneous systems, by relying on the concept of an EAI broker, which is the component that other systems use to connect to the architecture. EAI brokers provide adapters that allow heterogeneous systems built over different platforms and programming languages to communicate in real time.

The third architecture is the bus integration architecture (Figure 5). Bus architecture uses a central messaging backbone for message propagation. Each application publishes the required messages in the bus using adapters in the same way as happens with EAI brokers. The difference between bus and EAI broker approaches is in the adapters. While EAI brokers use a centralized solution, in bus topology, adapters are distributed in the application side. This later scales much better but it is more complex to maintain compared to a hub & spoke topology.

**Enterprise Service Bus (ESB)**

The different EAI topologies have tried to solve the drawbacks of the “accidental architecture”, however, they rely on the use of proprietary internal formats. This fact makes application integration specific to each provider. The ESB is based on today’s established standards in these areas and has real implementations that are already being
deployed in a number of industries. This is one of the main advantages of ESB over EAI.

An ESB in conjunction with SOA provides the implementation infrastructure to build a highly distributed approach for integration. An ESB provides a container that could alleviate the security and quality of service problems existing in Web Services. In this way, an ESB provides support to extend the use of Web Services.

ESB draws from traditional EAI approaches in that it provides integration services such as data transformation, routing of data, adapters to applications and protocol conversion. However, in an ESB, services can be configured rather than coded. There is nothing wrong with writing code, but there is plenty of code to be written elsewhere that doesn’t have to do with interdependencies between applications and services.

The most relevant aspect of using an ESB is that it enables the abstraction of the communication model used by the applications. This fact allows us to work more comfortably without having to worry about low-level details and allows us to focus on the development of the logic that will use semantic technology. Figure 6 depicts a schematic view of an ESB.

**FROM SYNTACTIC WEB SERVICES TOWARDS SEMANTIC WEB SERVICES**

In the previous section we established that Web Services are a good approach to overcome the heterogeneity which exists among different platforms and languages. We also discussed the main criticisms about technical issues such as security, performance, quality of service and looked at how the use of an ESB can alleviate these problems.
However, we have not treated Web Services from the final user point of view.

Figure 2 shows the process that a Web Service requester should follow to use the required service. This operational policy is somewhat utopic, in the sense that all processes around the Web Service life cycle rely on syntactic descriptions. In a SOA world it is not possible to manually explore a full repository as the available amount of Web Services is constantly increasing. Thus, an automatic way to manage Web Services usage is necessary.

This is the point where Semantic Web technology comes into play. Semantic Web technologies provide the required support to build an information system in a machine understandable way. Just as the Semantic Web is an extension of the World Wide Web, Semantic Web Services (SWS) are an extension of Web Services. The aim of combining both technologies is to provide a full set of mechanisms to obtain a common framework that facilitates the use of the traditional Web Services in a SOA.

A desirable goal would be to stimulate the use of Semantic Web Services among developers without a high effort to migrate from traditional Web Services towards Semantic Web Services. Experience in computer science has shown us that this feature is the key for the success of a new technology.

In this section we talk about the proposals submitted to the W3C in this area, although the different proposals are not explained in detail because we consider that it is outside the scope of this chapter and other authors have already treated this topic in depth. Our aim here is to provide the reader with a general overview that they can understand the rest of the chapter. We discuss the most relevant current R&D projects carried out by important research groups in the Semantic Web Services field, namely DER12, ST13, KMI4 and LSDIS5. Finally, we present an overview of existing tools which can be used to put into practice these technologies.

**Current Proposals for Semantic Web Services**

The aim of the Semantic Web Service is to add meaning to traditional Web Services using semantic annotations. In the Semantic Web field these annotations rely on a stack of accepted standards recommended by the W3C. These standards allow us to conceptualize a particular portion of the world in which we may be interested. This conceptualization is known by the term “ontology” and is responsible for adding meaning to resources, thereby disambiguating the available resources and allowing users to reason about them.

In the case of Semantic Web Services it is necessary to extend this stack with other mechanisms that allow developers to model the relationship between different Web Services. These mechanisms should provide the capability to describe Web Services in order to automate processes such as advertisements, discovery, selection, composition, mediation and invocation. Advertisement refers to the process of publishing the description and endpoints of a Web Service in a service repository. Discovery is the process of locating all Web Services that match the original request. Selection is the process of selecting the most suitable Web Service out of the ones discovered, usually based on application-dependent metrics (e.g., QoS or availability). Composition is the process of integrating selected Web Services into a complex process in order to create a complex service. Invocation references the process of invoking a single Web Service or complex process, by providing it with all the necessary inputs for its execution. Figure 7 depicts the relationship among these processes in the Web Service life-cycle.

Many of the related technologies and standards are still under development, but it is important to create an awareness of this technology and to think about it today rather than tomorrow. The technology might not be at an industrial strength maturity yet, but the problems are already (Fensel, 2007).
Nowadays, technologies that try to overcome the Semantic Web Service challenge can be classified in two different approaches, namely the top-down strategy and the bottom-up strategy. The first steps towards adding semantics to Web Services follow a top-down approach. This approach includes the four mayor works submitted by W3C members in 2004-2005, namely OWL Web Ontology Language for Services (OWL-S)\textsuperscript{6}, Web Service Modeling Ontology (WSMO)\textsuperscript{7}, Semantic Web Services Framework (SWSF)\textsuperscript{8} and Web Service Semantics (WSDL-S)\textsuperscript{9}.

OWL-S was the first proposal submitted to W3C. It defines a service ontology based on OWL, which is composed of three interlinked models:

- **Service Profile**: provides information about the service and the owner that is used in the discovery phase.
- **Service Model**: provides a detailed description of the service and its behaviour.
- **Service Grounding**: how to interact with the service.

WSMO is a conceptual model that distinguishes between four different concepts to model relationships among Semantic Web Services, namely goals, ontologies, web services and mediators:

- **Goals**: model the requestor point of view about a Web Service. This information is very important in the discovery phase because it determines a correct match.
- **Ontologies**: provide the terminology used by the other concepts in WSMO.
- **Web Services**: consist of a Web Service description about its interfaces, capabilities and grounding.
- **Mediators**: describe elements that try to solve interoperability problems, e.g., ontology alignment and communication among different Web Services.

SWSF is a more recent work in this field. It relies on OWL-S (sharing its three concepts of profile, model and grounding) and the Process Specification Language (PSL), standardized by ISO 18269. SWSF is based on a conceptual model and a language to axiomatize it.

WSDL-S (Sheth, 2005) was created in the project METEOR-S10 with the aim of adding semantic annotations to WSDL and XML Schema. WSDL-S does not provide a specific model for Semantic Web Services like the other proposals because it considers that annotations are sufficient for a conceptual model. WSDL-S was taken as the basis for the Semantic Annotation for WSDL and the XML Schema recommenda-
tion (SAWSDL, 2007), the first step towards a bottom-up approach.

Recently, efforts are focused on overcoming the logical separation between traditional and Semantic Web Services. While the previously commented top-down approach is closer to the semantic layer, a bottom-up approach is closer to the syntactic layer. In this sense, SAWSDL can be seen as a first rapprochement among the two approaches.

A recent WSMO-based ongoing work built on top of SAWSDL is known as WSMO-Lite11 (Vitvar, 2008). Authors in this work provide a light ontology based on WSMO to allow a bottom-up modelling of services. On the other hand, a recent work that grounds OWL-S in SAWSDL is described in (Paolucci, 2007). Finally, in (Kourtesis, 2008) authors attempt to build a repository of Semantic Web Services based on UDDI, SAWSDL, and OWL-DL.

### Related R&D Projects

In this subsection we describe the major projects carried out in recent years in the Semantic Web Services field. Many of these projects have inspired the creation of others projects, working groups and initiatives (e.g. SDK clusterl2 or ESSI clusterl3) to improve the Semantic Web Service technology.

Adaptive Services Grid (ASG) (http://asg-platform.org) is one of the projects in the 6th Framework Programme whose aim is to build a platform for semantic service provision. ASG provides a platform to enable discovery, composition and enactment of Semantic Web Services. The goal of ASG is related with the INFRAWEBS and METEOR-S projects responsible for the Infraweb Integrated Framework tool and the SAWSDL recommendation, respectively.

Semantic Discovery on Adaptive Services Grid (SemGrid) (http://lsdis.cs.uga.edu/projects/semgrid/) is a project funded by NSF. This project involves collaboration between the LSDIS SemDis and METEOR-S projects, and the European Commission sponsored Adaptive Service Grid (ASG) project. It investigates the use of semantic associations in Web Service discovery and Dynamic Web Process composition, and computing Semantic Associations over the grid.

Intelligent Framework for Generating Open (Adaptable) Development Platforms for Web-Service Enabled Applications using Semantic Web Technologies and Multi-Agent Systems (INFRAWEBS) (http://www.infrawebs.org/) is a European IST Project in the 6th Framework Programme. The main objective of INFRAWEBS is the development of a set of tools which will help users to create, maintain and execute Semantic Web Services. The developed platform known as the Infraweb Integrated Framework (IIF) is WSMO compliant and it relies on the Eclipse IDE.

Each set of tools is deployed within an ESB which is also considered an Infraweb Unit. The IIF defines two distinctive (but linked) parts: a Design Time Phase and a Runtime Phase. The Design Time Phase consists of the tools involved during the design of Semantic Web Services (using the Eclipse IDE), whereas the Runtime Phase consists of those tools involved during the execution of Semantic Web Services (using the ESB). Some of the tools may of course be involved in both phases. This approach ensures that system users (whether developers or normal end-users) deal with the minimum (but complete) set of tools required. Furthermore, it also provides an environment which is focused on the specific needs of the users.

The METEOR-S project (http://lsdis.cs.uga.edu/projects/meteor-s/) at the LSDIS Lab aims to extend standards such as BPEL4WS, SOAP and WSDL with Semantic Web technologies to achieve greater dynamism and scalability. This project has generated many deliverables related to the SWS life-cycle. We reference the SAWSDL recommendation as the main result of this work.
Data, Information, and Process Integration with Semantic Web Services (DIP) (http://dip.semanticweb.org/) is an Integrated Project supported by the European Union’s FP6 programme. The DIP objective has been to develop and extend Semantic Web and Web Service technologies in order to produce a new technology infrastructure for Semantic Web Services; an environment in which different Web Services can discover and cooperate with each other automatically. The long term vision of DIP is to deliver the enormous potential benefits of Semantic Web Services to e-Work and e-Commerce.

The EU IST integrated project Semantic Knowledge Technologies (SEKT) (http://www.sekt-project.com/) developed and exploited semantic knowledge technologies. Core to the SEKT project has been the creation of synergies by combining the three core research areas: ontology management, machine learning and natural language processing.

Semantics Utilised for Process Management within and between enterprises (SUPER) (http://www.ip-super.org/) is another project supported by the European Union’s FP6 programme that aims to introduce the use of Semantic Web Services into the Business Process Management (BPM). The main objective of SUPER is to raise BPM to the business level, where it belongs, from the IT level where it mostly resides now. This objective requires that BPM be accessible at the level of semantics of business experts. Semantic Web and, in particular, Semantic Web Services technology offer the promise of integrating applications at the semantic level. By combining SWS and BPM, and developing one consolidated technology SUPER will create horizontal ontologies which describe business processes and vertical telecommunication oriented ontologies to support domain-specific annotation. Therefore this project aims at providing a semantic-based and context-aware framework, based on Semantic Web Services technology that acquires, organizes, shares and uses the knowledge embedded in business processes within existing IT systems and software, and within employees heads, in order to make companies more adaptive.

The Triple Space Communication (TripCom) project is supported by the European Union’s FP6 programme (http://www.tripcom.org). The aim of TripCom is to build a coordination infrastructure for machine-to-machine interaction combining Tuple Space technology (Omair Shafiq, 2006), Semantic Web technology and Web Service technology. Moreover, Triple Space computing follows the same goals for the Semantic Web Services as the Web for humans: re-define and expand the current communication paradigm. One of the important goals is to bring Triple Spaces to an Internet-scale level. This means that any number of data providers and readers can write and read communication data over the Internet. As in the case of Web pages, there should be no limitation: the architecture is independent of system properties like response-time and throughput.

Business process fusion based on Semantically-enabled Service-oriented Business Applications (FUSION) is a project funded by the European Commission in the 6th Framework Programme (http://www.fusionweb.org/Fusion/project/project.asp). Special attention is given to e-Business, as FUSION aims at efficient business collaboration and interconnection between enterprises by developing a framework and innovative technologies for the semantic fusion of service-oriented business applications that exist within the collaborating companies. FUSION aims at the integration of research activities carried out in the areas of Business Process Management, Semantic Web and Web Services.

Other projects related with enterprise interoperability in the same way as FUSION are the following: ABILITIES, ATHENA, GENESIS, IMPORTNET, INTEROP, NO-REST, TRUSTCOM. Information related with these projects is available at http://cordis.europa.eu/ist/ict-ent-net/projects.htm.
The GRISINO research project is funded by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) in the FIT-IT programme (http://www.grisino.at/). It combines “Grid computing”, “Semantic Web Services” and “intelligent objects” in order to create an example of the future, enhanced infrastructure of the World Wide Web.

The main result of the project will be an experimental software prototype which will demonstrate how this infrastructure should be designed and how it will improve collaboration amongst people and machines over the World Wide Web.

Service Web 3.0 is a project financed by the European Union 7th Framework Programme (http://www.serviceweb30.eu/cms/). The aim of Service Web 3.0 is to support emerging technologies such as the Semantic Web or Web Services that try to transform the Internet from a network of information to a network of knowledge and services. To overcome this challenge, Service Web 3.0 will contribute to the implementation of framework programmes and will support the preparation of future community research and technological development. The main mission of this project is to facilitate Semantic Web Services and Semantic Web technology adoption, in particular for SMEs.

SHAPE is another project in the 7th Framework Programme (http://www.shape-project.eu/). The objective of SHAPE is to support the development and the creation of enterprise systems based on Semantically-enabled Heterogeneous Architectures (SHA). SHA extends SOA with semantics and heterogeneous infrastructures (Web services, agents, Semantic Web Services, P2P and grid) within a unified service oriented approach. SHAPE will develop a model-driven engineering (MDE) tool-supported methodology. SHAPE will take an active role in the standardization of meta models and languages for SHA. The technical results will be compliant with the proposed standards to ensure high industry acceptance. In current SOA approaches, business requirements and technical details are intertwined constraining the evolution of service-oriented business solutions. SHAPE will provide meta models and languages, methods and tools to separate the different viewpoints of SOA for the development of semantically-enabled, flexible and adaptive business services on a rich SHA infrastructure.

Single European Employment Market-Place (SEEMP) is a project supported by the European Union’s FP6 programme (http://www.seemp.org/). The mission of SEEMP is to design and implement in a prototypal way an interoperability architecture for public e-Employment services. The architecture will encompass cross-governmental business and decisional processes, interoperability and reconciliation of local professional profiles and taxonomies as well as semantically enabled web services for distributed knowledge access and sharing. In particular, the SEEMP project will develop an EIF-compliant federated architecture and interoperability middleware as well as applicative plug-in services to allow existing National/Local job market places and data warehouses to be interoperable at pan-European level by overcoming state-of-the-art limitations.

SOA4All is endorsed by the NESSI14 and aims to alleviate the problems of current SOA solutions (http://www.soa4all.org/). Its success depends on resolving a number of fundamental challenges that SOA does not address today. SOA4All will help create a world where billions of parties provide and consume services via advanced Web technology.

The outcome of the project will be a comprehensive framework and infrastructure that integrates four complimentary and revolutionary technical advances into a coherent and domain independent service delivery platform, namely Web Services, Web 2.0, semantic and context management.

COllaboration and INteroperability for networked enterprises (COIN) is an integrated project
in the European Commission 7th Framework Programme (http://www.coin-ip.eu/). The mission of the COIN IP is to study, design, develop and prototype an open, self-adaptive, generic ICT integrated solution to support the expansion of SOA. The COIN business-pervasive open-source service platform will be able to offer, integrate, compose and mash-up in a secure and adaptive way existing and innovative yet to-be developed Enterprise Interoperability and Enterprise Collaboration services, by applying intelligent maturity models, business rules and self-adaptive decision-support guidelines to guarantee the best combination of the needed services. This will be done independently of the business context, e.g., of the industrial sector and/or domain, the size of the companies involved, or the openness and dynamics of collaboration.

This way, the Information Technology vision of Software as a Service (SaaS) will find its implementation in the field of interoperability among collaborative enterprises, supporting the various collaborative business forms, from supply chains to business ecosystems, and becoming for them like a utility, a commodity, the so-called Interoperability Service Utility (ISU).

The COIN project will finally develop an original business model based on the SaaS-U (Software as a Service-Utility) paradigm where the open-source COIN service platform will be able to integrate both free-of-charge and chargeable, open and proprietary services depending on the case and business policies.

SWING is a project of the Information Society Technologies (IST) Program for Research, Technology Development & Demonstration under the 6th Framework Programme of the European Commission (http://www.swing-project.org/). SWING aims at deploying Semantic Web Service technology in the geospatial domain. In particular, it addresses two major obstacles that must be overcome for Semantic Web Service technology to be generally adopted, i.e. to reduce the complexity of creating semantic descriptions and to increase the number of semantically described services. Today, a comprehensive knowledge of logics, ontologies, metadata and various specification languages is required to describe a service semantically. SWING will develop methods and tools that can hide the complexity - and automate the creation of the necessary semantic descriptions. The objective of SWING is to provide an open, easy-to-use SWS framework of suitable ontologies and inference tools for annotation, discovery, composition, and invocation of geospatial web services. SWING builds on the DIP and SEKT IPs, by adopting, combining and reinforcing their results. A main key to the solution is adapting the SWS technology of DIP to handle geospatial services and content. Another key is utilising and advancing the technology of SEKT to annotate geospatial services with semantic information. A synergy between these two research initiatives is demanded in order to maintain and extend Europe’s leading role in SWS. The SWING framework and pilot application will increase the use of distributed and heterogeneous services in geospatial decision making. The results can be reused in other domains and will boost the availability of semantic services and bring the vision of the SWS a great leap forward. Exploitation of SWING’s results will provide Europe’s decision makers and citizens with a new paradigm of information retrieval and new business opportunities.

Available Tools

The number of tools available for using with a particular technology is a good indicator of the importance of such technology. There are several tools to work with Semantic Web Services that try to put into practice the more important proposals, namely OWL-S, WSMO and SAWSDL. Usually, these tools are distributed under GPL or LGPL license. In the following, we list the tools available for each proposal.
OWL-S Based Tools

Figure 8 depicts an overview of the existing tools for OWL-S. These tools can be classified into two main groups, namely necessary tools at design time (e.g. editor) and necessary tools at run time (e.g. matchmaker). Generally, these tools are developed for academia and so they are not well known in the business world.

WSDL2OWL-S provides translation between WSDL and OWL-S. The results of this translation are a complete specification of the Grounding and partial specification of the Process Model and Profile. The incompleteness of the specification is due to differences in information contained in OWL-S and WSDL. Specifically WSDL does not provide any process composition information, therefore the result of the translation will also lack process composition information. Furthermore, WSDL does not provide a service capability description. Therefore the OWL-S Profile generated from WSDL is also necessarily sketchy and must be manually completed. Nevertheless the output of WSDL2OWL-S provides the basic structure of an OWL-S description of Web services and saves a great deal of manpower (Paolucci, Srinivasan, Sycara, & Mishimura, 2003).

The Java2OWL-S Converter provides a partial translation between Java and OWL-S 1.1. The results of this translation are a complete specification of the Grounding, partial specification of the Process Model and Profile, and an OWL Class file, when at least one of arguments in the java class method is a user defined class. This tool extends the WSDL2OWL-S converter, instead of taking a WSDL file, it takes a java class file as input and converts it into a WSDL file, using apache axis’ Java2WSDL converter, and then feeds the generated WSDL file as input to the WSDL2OWL-S converter (Paolucci, Srinivasan, Sycara, & Mishimura, 2003).

OWL-S2UDDI Converter converts the OWL-S profile descriptions to corresponding UDDI advertisements, which can then be published.
in any UDDI registry. This converter is part of the OWL-S/UDDI registry. The OWL-S/UDDI registry enhances UDDI registry with OWL-S matchmaking functionalities (Paolucci, 2002a).

The Matchmaker Client can be used to interact with the OWL-S/UDDI matchmaker (Srinivasan, 2002; Paolucci, 2002a), which is responsible for matching OWL-S descriptions with the UDDI information. An online version of the matchmaker client is available at http://www.daml.ri.cmu.edu/matchmaker (Paolucci, Srinivasan, Sycara, & Mishimura, 2003).

OWL-S IDE is a development environment that supports a semantic web service developer through the whole process from the Java generation, to the compilation of OWL-S descriptions, to the deployment and registration with UDDI.

OWL-S VM is an environment developed with the aim of executing Semantic Web Services (Paolucci, Srinivasan, Sycara, & Mishimura, 2003).

OWL-S Broker provides the necessary logic for the discovery process. This component is strongly related with the Matchmaker Client and OWL-S/UDDI matchmaker (Sycara, 2004; Paolucci, 2004).

WSMO Based Tools

The main criticisms received for these tools are to do with the number of them required to cover the SWS life-cycle. Advocates of other proposals, such as WSMO, have tried to collect the implemented tools in a unified environment. Components that have become WSMO in a suitable approach for Semantic Web Services are WSML15 and WSMX16.

The Web Service Modeling Language (WSML) is a language that formalizes the Web Service Modeling Ontology (http://www.wsmo.org/TR/d16/d16.1/v0.21/). It uses well-known logical formalisms in order to enable the description of various aspects related to Semantic Web Services and consists of a number of language variants, namely WSML-Core, WSML-DL, WSML-Flight, WSML-Rule and WSML-Full.

The Web Services Execution Environment (WSMX) is an execution environment for dynamic matchmaking, selection, mediation and invocation of semantic web services based on WSMO. WSMX tries to cover the SWS life-cycle. A downloadable version of WSMX is available at http://sourceforge.net/projects/wsmx/.

The Web Services Modeling Toolkit (WSMT) may be the best known tool compliant with WSMO. WSMT is a framework for the rapid deployment of graphical administrative tools, which can be used with WSMO, WSML and WSMX. WSMT is built over the Eclipse IDE and it brings together the following components among others.

- **WSML2Reasoner Framework** (http://tools.deri.org/wsml2reasoner/) is a highly modular architecture which combines various validation, normalization, and transformation functionalities essential to the translation of ontology descriptions in WSML with the appropriate syntax of several underlying reasoning engines.

- **WSMO Studio** (http://www.wsmostudio.org/) is a Semantic Web Service editor compliant with WSMO. WSMO Studio is available as a set of Eclipse plug-ins that can be further extended by third parties.

- **WSML Rule Reasoner and WSML DL Reasoner** are two reasoner implementations for WSML available at http://dev1.deri.at/wsml2reasoner/.

- **WSML Validator** is a WSML variant validation implemented as part of wsmo4j. An online validation service is available at http://tools.deri.org/wsmi/validator/.

- **WSMO4J** is an API and a reference implementation for building Semantic Web Services applications compliant with the Web
Service Modeling Ontology. WSMO4J is compliant with the WSMO v1.2 and WSML 0.2 specifications. A downloadable version is available at http://wsmo4j.sourceforge.net.

Other WSMO compliant environments are the Internet Reasoning Service (IRS III) and INFRAWEBS Integrated Framework. The Internet Reasoning Service is a KMi Semantic Web Services framework, which allows applications to semantically describe and execute Web services. The IRS supports the provision of semantic reasoning services within the context of the Semantic Web. The core of IRS III uses the language Common Lisp to build the reasoner component. More information is available at http://kmi.open.ac.uk/projects/irs/.

INFRAWEBS Integrated Framework was described in the previous section when we mentioned the European project INFRAWEBS (http://www.iit.bas.bg/InfrawebsDesigner/Index.html). In this section we want to show the INFRAWEBS Designer, a graphical, ontology-based, integrated development environment for designing WSMO-based SWS and goals. The INFRAWEBS Designer does not require any preliminary knowledge of WSML. This issue simplifies the user’s effort in a top-down design.

SAWSDL Based Tools

Finally, we look briefly at the available tools to work with SAWSDL. All of these tools have been developed by LSDIS and are available at http://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL/.

The SAWSDL4J API interface is an implementation of the SAWSDL specification. The API would allow developers to create SAWSDL based applications. SAWSDL4J extends the WSDL4J API for WSDL1.1.

Woden4SAWSDL is a WSDL 2.0 parser, based on Apache Woden, with API classes that allow for SAWSDL parsing and creation.

Radiant is an Eclipse plugin that allows creating and publishing SAWSDL and WSDL-S service interfaces. Users can add annotations to existing service descriptions in WSDL using the user friendly graphical interface of Radiant.

Lumina is other Eclipse plugin that allows for discovery of Web services based on semantics. Lumina is a graphical user interface built on top of the METEOR-S Web Services Discovery Infrastructure engine. In the MWSDI engine, a semantically enhanced services registry is created based on UDDI.

APPLYING SEMANTIC TECHNOLOGY TO ENTERPRISE INTEGRATION

In the background to this chapter we established that ESB has become the most promising solution to build a SOA infrastructure from several heterogeneous sources. We also mentioned that there are a lot of projects and work being carried out by researchers in the area of Semantic Web Services. However, many of those works have focused on the search for a suitable technology to model traditional Web Services applying the acquired knowledge in the Semantic Web. Proof of this fact can be seen in a number of proposals submitted to the W3C.

Although there is not an official standard for modelling Semantic Web Services we believe that developed approaches must be put in practice. In this way it is possible to use Semantic Web Services in conjunction with an ESB to overcome the problem of enterprise integration.

The use of the Semantic Web Services technology in enterprise would not be possible without the existence of an infrastructure that allows covering the life cycle of Web Services using semantic annotation techniques. This necessary infrastructure could be an ESB, which would facilitate the integration of various heterogeneous systems. An ESB allows the cooperation and the exchange of data between services. It is a logical architecture
based on the principles of SOA, which aims to define services explicitly and independently of the implementation details. It also pays close attention to securing a transparent location and excellent interoperability (Chappell, 2004).

An ESB makes Web Services, XML, and other integration technologies immediately usable with the mature technology that exists today. The core tenets of SOA are vital to the success of a pervasive integration project, and are already implemented quite thoroughly in the ESB. The Web Service standards are heading in the right direction, but remain incomplete with respect to the enterprise-grade capabilities such as security, reliability, transaction management, and business process orchestration. The ESB is based on today’s established standards in these areas, and has real implementations that are already being deployed across a number of industries. The ESB is quite capable of keeping in step with the ongoing evolution of the Web Services equivalents of these capabilities as they mature.

It would be interesting to maintain these capabilities over the use of Semantic Web Services. For this reason it has been proposed to construct an infrastructure composed of different layers where the ESB is the foundation on which the others are based. The objective is to define a Semantic Enterprise Service Bus (SESB), providing mechanisms to collect all these technologies together and acting as a layer to access services through the invocation paradigm based on goals. This SESB would therefore be responsible for the process of coordinating individual services available on the Web. Figure 9 shows an overview of the proposed infrastructure.

The first step towards designing the system is to select a possible implementation of an ESB, which will be built on the remaining layers of the architecture. The fact that this tool is the pillar of the design means that the choice of implementation is of great significance with regard to possible integration with other technologies to be used in the upper layers.

On top of the first layer we can find the Web Services layer, which provides the necessary mechanisms for executing such services. Basically, this is the logic that acts as an intermediary between the lower layer and the semantic layer.

We are now working on developing the semantic layer. Our aim is to introduce semantics into the core of the ESB. This issue provides reasoning capabilities to the artefacts deployed on the ESB. Although we have not obtained experimental results we believe that the ongoing prototype will facilitate the deployment of new (semantic) services.

**FUTURE TRENDS**

Nowadays, the main challenge among researchers in the Semantic Web Service field lies in overcoming the technological gap between the use of syntactic technology and semantic technology. As we can see, many R&D projects are ongoing with the aim of bringing semantics into SOA. In many cases, the goal of these projects is to build a platform enabling the use of Semantic Web Services (INFRAWEBS, WSMX, IRS-III). These platforms cover the whole Semantic Web Service life-cycle enabling discovery of services based on goals, composition, registry and mediation. The developed tools rely on a top-down design, i.e., they assume that users start with an ontology implementation from which Web Services will be annotated. This way, the first step would be to specify goals, mediators and others semantic
restrictions in order to build a semantic layer that Web Services will use.

Despite being the most common way to begin a new project based on Semantic Web Service technology in future, now it is not the most suitable way to evolve current SOA applications towards a Semantic SOA. The reason for this is that with a top-down design it is not possible to take advantage of the existing Web Services. Nowadays, a bottom-up design is necessary, which allows developers to re-use the existing applications and adapt them to the Semantic Web.

At present, some researchers have noticed this necessity and have begun to provide solutions. In this sense, a first approach has been the recent recommendation of SAWSDL by W3C, after developing a mechanism to enable semantic annotation of Web Service descriptions in WSDL 2.0. SAWSDL is independent of the language used for the ontology description. This fact makes SAWSDL suitable for being combined with different approaches such as WSMO or OWL-S as we can see in the recent works in this field (Vitvar, 2008; Paolucci, 2007; Kourtesis, 2008).

Current effort shows that researchers have become aware of the actual technological transition in SOA. In this sense, it is difficult to know when Semantic Web Services may be used among ICT enterprises without any limitation. For the moment, researchers should postpone the development of new platforms that cover the SWS life-cycle focusing their effort on obtaining a solution to overcome the current transition problem between SOA and Semantic SOA.

Therefore we propose the development of adapters or wrappers over existing SOA applications as a future extension of the described work. These adapters will allow the application of a semantic layer over implemented Web Services which will be reusable in the proposed Semantic ESB. In this work we expect to implement an automatic or semi-automatic tool to annotate Web Services using SAWSDL over concepts in an ontology. This tool will be incorporated into the Semantic ESB to facilitate the deployment of not annotated Web Services.

CONCLUSION

In this chapter we have presented an overview of the topics of application integration and Semantic Web Services. First, we listed the different approaches for solving the application integration problem. Subsequently, we introduced the basis of SOA, Web Services and ESB, and later we described the main ideas in Semantic Web Services. We have commented on the existing proposals and classified them into two groups, namely top-down and bottom-up strategies. We also discussed the available tools for developing Semantic Web Services and provided a set of related projects with this technology.

Later, we briefly described our work on improving the use of an ESB in application integration.

The aim of this chapter has not been to compose a detailed manual in application integration nor Semantic Web Services because these topics have been treated in depth by other authors. However, we provided an overview of the work in this area and many useful references throughout the discussion for the interested reader.

REFERENCES


Semantic Web Services


KEY TERMS AND DEFINITIONS

**CORBA:** The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

**DCOM:** Distributed Component Object Model (DCOM) is a proprietary Microsoft technology for communication among software components distributed across networked computers. DCOM extends Microsoft’s COM and provides the communication substrate under Microsoft’s COM+ application server infrastructure.

**EAI:** Enterprise application integration (EAI) is the process of linking applications within a single organization together in order to simplify and automate business processes to the greatest extent possible, while at the same time avoiding having to make sweeping changes to the existing applications or data structures.

**ESB:** Enterprise Service Bus (ESB) provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. Contrary to the more classical enterprise application integration (EAI) approach the foundation of an enterprise service bus is built of base functions broken up into their constituent parts, with distributed deployment where needed, working in harmony as necessary.

**GIOP:** General Inter-ORB Protocol (GIOP) is the abstract protocol by which object request brok-ers (ORBs) communicate. Standards associated with the protocol are maintained by the Object Management Group (OMG).

**IIOP:** Internet Inter-Orb Protocol (IIOP) is the implementation of GIOP for TCP/IP. It is a concrete realization of the abstract GIOP definitions.

**OMG:** The Object Management Group, Inc. (OMG) is an international organization supported by over 600 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. Primary goals are the reusability, portability, and interoperability of object-based software in distributed, heteroge-
OMG Task Forces develop enterprise integration standards for a wide range of technologies, and an even wider range of industries. OMG’s modeling standards enable powerful visual design, execution and maintenance of software and other processes. OMG’s middleware standards and profiles are based on the Common Object Request Broker Architecture (CORBA) and support a wide variety of industries.

**ORB:** An object request broker (ORB) is a middleware technology that manages communication and data exchange between objects. ORBs promote interoperability of distributed object systems because they enable users to build systems by piecing together objects - from different vendors - that communicate with each other via the ORB (Wade, 1994). The developers are only concerned with the object interface details. This form of information hiding enhances system maintainability since the object communication details are hidden from the developers and isolated in the ORB (Cobb, 1995).

**RMI:** Remote Method Invocation is the Java implementation for performing the object equivalent of remote procedure calls. The Java Remote Method Invocation system allows an object running in one Java virtual machine to invoke methods on an object running in another Java virtual machine. RMI provides for remote communication between programs written in the Java programming language.

**RPC:** Remote procedure call (RPC) is an inter-process communication technology that allows a computer program to communicate with a procedure running in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction.

**SAWSDL:** The Semantic Annotations for WSDL and XML Schema (SAWSDL) W3C Recommendation defines mechanisms using which semantic annotations can be added to WSDL components. SAWSDL does not specify a language for representing the semantic models, e.g. ontologies. Instead, it provides mechanisms by which concepts from the semantic models that are defined either within or outside the WSDL document can be referenced from within WSDL components as annotations.

**SOA:** Service Oriented Architecture (SOA) is a software architecture where functionality is grouped around business processes and packaged as interoperable services. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. The aim is a loose coupling of services with operating systems, programming languages and other technologies which underly applications.

**SOAP:** SOAP is a simple and lightweight XML-based mechanism for creating structured data packages that can exchanged between network applications. SOAP consists of four fundamental components: an envelope that defines a framework for describing message structure, a set of encoding rules for expressing instances of application-defined data types, a convention for representing remote procedure calls (RPC) and responses, and a set of rules for using SOAP with HTTP. SOAP can be used in combination with a variety of network protocols; such as HTTP, SMTP, FTP, RMI/IIOP, or a proprietary messaging protocol.

**UDDI:** Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative, sponsored by OASIS, enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet.

**WSDL:** The Web Services Description Language (WSDL) is an XML-based language that provides a model for describing
ENDNOTES

1 Object Management Group
2 http://www.deri.org/
3 http://www.sti2.org/
4 http://kmi.open.ac.uk/
5 http://lsdis.cs.uga.edu/
6 http://www.w3.org/Submission/2004/07/
7 http://www.w3.org/Submission/2005/06/
8 http://www.w3.org/Submission/2005/07/
9 http://www.w3.org/Submission/2005/10/

10 http://lsdis.cs.uga.edu/projects/meteor-s/

11 http://www.wsmo.org/TR/d11/v0.2/20080304/
12 http://sdk.semanticweb.org/
13 http://www.essi-cluster.org/
14 NESSI Open Framework is one of the main challenges of the European Platform on Software and Services. It has a consortium of 16 partners and is led by Atos Origin. More information is available at http://www.nessi-europe.com/Nessi/.
15 http://www.wsmo.org/wsml/
16 http://www.wsmx.org/