Characterizing Semantic Web Services

Marcelo Moyano, Agustina Buccella, Alejandra Cechich
Departamento de Ciencias de la Computación
Universidad Nacional del Comahue, Neuquén, Argentina
mmoyano@thinknetgroup.com.ar, {abuccell, acechich}@uncoma.edu.ar

and

Elsa Clara Estevez
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Av. Alem 1253 – (8000) Bahía Blanca, Argentina
ece@cs.uns.edu.ar

Abstract

Semantic Web is an extension of the current web in which data contained in the web documents are machine-understandable. On the other hand, Web Services provide a new model of the web in which sites exchange dynamic information on demand. Combination of both introduces a new concept named Semantic Web Services in which semantic information is added to the different activities involved in Web Services, such as discovering, publication, composition, etc. In this paper, we analyze several proposals implementing Semantic Web Services. In order to describe them, we create a conceptual framework characterizing the main aspects of each proposal.

Keywords: Semantic Web, Ontology, Web Services

1. Introduction

Nowadays the World Wide Web (WWW) is changing to provide more useful information to users. These changes are part of the new concept called "Semantic Web". *The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computer and people to work in cooperation* [4]. At the moment HTML (hypertext markup language) is the mostly used language to publish information on the WWW. This language lets just show the information in a user-understandable form; however machines do not understand this information and therefore can not process it.

To make those differences clearer, let's introduce a simple example, in which a user – Jessica – is planning a simple schedule: she wants to go out to dinner at nine o'clock and at eleven she wants to go to the cinema. As she lives far from downtown, she must take a bus. Besides, the restaurant is far from the cinema and again she must take another bus. She then needs a plan in order to take the bus early enough to arrive at the restaurant in time and then take the bus again to arrive at the cinema before eleven o'clock. To make the plan, Jessica uses the information the Web provides. Firstly she looks up the schedules for the buses, and then she calculates the time the bus takes to arrive to the restaurant and she does the same for the cinema. Also, she makes reservations for the restaurant and books the cinema ticket.

As we can see, in order to prepare the plan Jessica has to do a series of steps consulting several web pages. In contrast, by using Semantic Web [4,5] Jessica will enter only a query in which she specifies the steps she wants to follow this night and by exploring all the semantic information provided by the semantic web pages, the system will generate the plan that she is in need of.

Of course, this plan is not generated spontaneously, several tools have worked together to give the answer. Firstly, we found the ontologies defined as a set of knowledge terms, the semantic interconnections, and the rules of inference and logic for some particular domain. And secondly, we have Web Services [25] which provide a new model of the web in which sites exchange dynamic information on demand. Then, web services might be one of the most powerful tools of the Web ontologies. With these two main tools we are actually in the presence of the Semantic Web Services [20].

Few works exist in the literature comparing Semantic Web Services. A comparison between the approach presented in [23] and other approaches is explained in [7]. But this paper is focused mainly on the advantages of the IRS approach with respect to the others.

Our work describes several approaches in the literature by introducing a framework that characterizes important aspects of Semantic Web Services, such as their semantic specification, use of standards, inference rules, and so forth. The framework and the characterization aim at highlighting advantages and disadvantages on the use of the different proposals, allowing us to suggest improvements.

This paper is organized as follows. In Section 2 we will introduce the concept of Semantic Web together with its main architectural components. In the next section, we present the Web Services concept also showing its components. Both sections 2 and 3 converge in Section 4, in which we will explain the Semantic Web Services concept together with the different approaches found in the literature to implement characteristics of this concept. In Section 5 we propose a simple framework that describes the most important aspects of the Semantic Web Services. We use this framework to characterize the approaches previously introduced. Finally, in the last section we present our conclusions.

2. Semantic Web

In order to describe the Semantic Web components, we choose the architecture presented by Berners-Lee [37]. This architecture, shown in Figure 1, is widely referred by many researches in this area, and it is based in a layered architecture, in which languages of increasing power are layered one on top of the other.

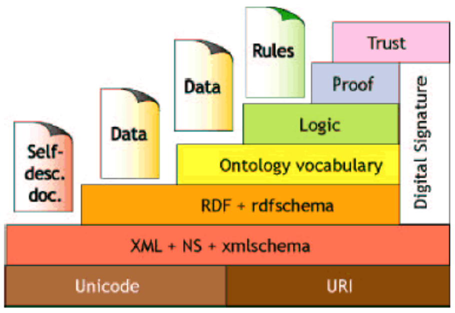

Figure 1: Semantic Web Architecture presented by Berners-Lee

We briefly describe the layers of the architecture as follows:

- *URI (Universal Resource Indicator)*: The URI is the foundation of the Web. It provides an identifier for a resource, that is, the URI is used to identify items on the Web. Resources may or may not be accessible over Internet.
- *XML (Extensible Markup Language)*: XML is a specification for computer-readable documents. XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents. The namespace (NS) indicated in Figure 1 is just a way of identifying a part of the Web. A namespace is defined for each XML document in which the user defines its elements. Finally, the XML Schema is a language for restricting the structure of XML documents and also for extending XML with data types.
- *RDF (Resource Description Framework)*: RDF is a data model for objects (resources) and relations between them. RDF provides a simple semantics for this data model. It gives a way to make statements that are machine-processable. Each RDF statement has three parts: a subject, a predicate and an object. With these three elements we can use RDF statements to say practically anything. RDF is used to tell

something about data (metadata). As with XML, an RDF model does not define the semantics of any domain. It just provides a neutral mechanism to describe data. Then, RDF Schema (RDFS) extends RDF with a semantics for generalization-hierarchies. The objects are called classes and the predicates are called properties. With RDFS we can define classes and subclasses and properties with sub-properties. But RDFS still does not provide exact semantics.

- *Ontology*: An ontology provides the name and descriptions of entities of specific domains using predicates that represent relationship between these entities. It provides a vocabulary to represent and communicate knowledge about the domain and a set of relationship containing the term of the vocabulary at a conceptual level. We need ontologies to realize the Semantic Web vision, because they add new modeling primitives and formal semantics to RDFS. Nowadays, OWL (Web Ontology Language) [32,1] is the most popular language to represent ontologies in the Semantic Web. OWL allows us to formalize a domain by defining classes and properties of those classes, to define individuals asserting properties about them, and to reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language. OWL can be (partially) mapped to a description logic [2] making the use of existing reasoners such as FACT [16] and RACER [15] possible.
- *Logic*: The logic layer is not developed yet, but it will have any logical principle to permit a computer to reason (by inference) using these principles. As we have said above, the ontologies have inference rules to allow the computer to "understand" the meaning of the information. This layer must provide the logic to manipulate the information.
- *Proof*: Once the systems are built to follow logic, by using them it is possible to prove things. In this layer all the logical rules are used together in a proof in order to obtain useful information. For example, we define an ontology with the person, female and woman classes. The woman class is defined as a female person. If we say that Anna is a female person, the computer using the woman rule proves that Anna is a woman.
- *Digital Signature and Web of Trust*: Based on works in mathematics and cryptography, Digital Signatures let users authenticate their programs such that the computer verifies that the attached information has been provided by a specific trusted source. The Web of Trust lets users define the group of other users allowed to use their information. This process is called "trust". As all the users, a specific user trusts, trust another set of users and so on, they form a trust relationship called web of trust.

All of these layers will enable machines to comprehend semantic documents and data.

3. Web Services: An Overview

Due to the use of new technologies based on communication, Internet has emerged as a mean to share documentation all over the world. Moreover, Internet has changed from a simple container of data to a tool for solving problems and indirectly improving – or at least facilitating – user's life. From this point of view, Internet supports different electronic services – ranging from the traditional E-mail to the more recent E-commerce and E-Government. The common idea behind those services is that they are electronically supported and distributed by the Network.

On the other hand, considering business and marketing experts' point of view, E-services are identified focusing on the delivered product, so they are seen as an evolution of E-commerce. Additionally, from the point of view of Information Technology (IT) experts, E-services are seen as functional software available through the Web and frequently denominated Web-Services [33].

Architecting Web-Services is a recent challenge for IT community. In this sense, common research involving important software companies, such as IBM, Microsoft and Sun, and organizations such as the Web Service Architecture Working Group (W3C), has introduced a new architecture based on Web Services, i.e. on functional services offered through the Web.

Figure 2 (from [33]) describes the proposed architecture based on layers. The three main sections in the figure identify roles of components of the Web services oriented architecture: interaction, description and discovery agencies.

The lowest layer –*wire*– is the basement for the architecture and it determines the technologies required to transport messages from a service requestor to the supplier of the required service. The wire layer is further decomposed into three levels: (1) the *transport* layer has the function of maintaining the connection in the network using TCP/IP as standard protocol; (2) the *packaging* layer defines how data are coded in the message to be transported; and (3) the *extension* layer defines an additional group of data characteristics incorporated to the message as a header. SOAP and HTTP are the standards more commonly used to support these layers.

The next layer is the description layer, in which descriptions are specified and expressed using the language XML [36]. This layer is further described by nine layers as follows. The *Interface Description and the Implementation Description* layers define the mechanism for communicating with a Web service, and include supported operations and messages; how these messages are ordered to interoperate with the wire layer; and when these messages should be sent. The *Policy* layer is used to further describe specific information of the service, such as business aspects, security requirements, timeouts, costs and parameters of quality of the service. The *Presentation* layer describes how user's interfaces should be generated for this service. The next two layers –*Orchestation and Composition*– describe the relationships and interactions with other services. These related services can be expressed in the composition layer, including several aspects such as dependences, grouping, contention and relationships father-son. The orchestration layer is in charge of defining the order of the operations, workflows and business processes. The last two layers –*Service Level Agreements and Business Level Agreements*– describe the agreements between the requestor and the provider of the service. The Service level agreements layer defines those characteristics to which a service should adhere, such as specific performance, costs, metrics, etc. Finally, the Business Level Agreements layer describes a contractual agreement between two participants of a business that use Web services to carry out their transactions.

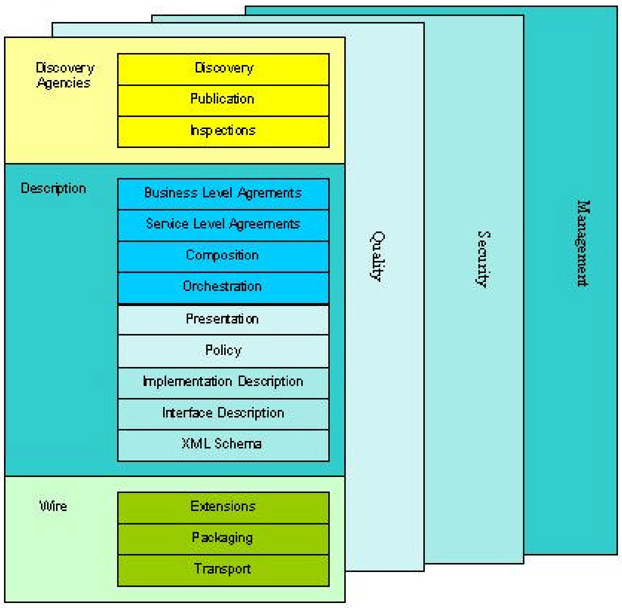

Figure 2: Web Service – Architecture based on layers

The upper layer of this architecture –*discovery agencies*– comprises the technolo gies that allow publishing the descriptions of the service to be used by possible requests. The layer supports the discovery of descriptions through advanced searches on the registrations of services and also provides the inspection of places where the services are hosted.

A Web service can also be understood as a variety of applications Web, or as independent components that are published through the Web in such way that other applications Web can find and use them [31]. These software components are identified in the network through a URI and the interaction with other agents is carried out by exchanging messages based on XML via protocols based on Internet. Particularly, documents written using the Web Services Description Language (WSDL) are used for the description of a published Web Service that then are supported and transmitted using the protocol SOAP [13].

The service oriented architecture consists of three main elements: *service provider*, *service requestor* and *discovery agencies*.

A service provider produces a WSDL description of the service that details its interface; that is, operations of the service and the input/output parameters of each operation. The service description also contains information of how the service can be located in the network. Then, a provider publishes the service description in one or more discovery agencies. In a given moment, a service requestor finds the service description in the discovery agencies and configures a client, which can communicate with the available service in the provider [13]. Figure 3 shows the elements of this Web services-oriented architecture interacting.

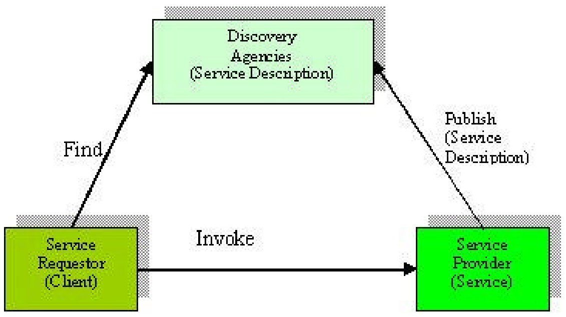
Discovery Agencies (Service Description)
Find
Publish (Service Description)
Invoke
Service Requestor (Client)
Service Provider (Service)

Figure 3: Elements of the Web services-oriented architecture interacting

Because Web services use available infrastructure on the Web, they facilitate interoperability between systems and applications on different platforms and different languages. Web services combine a series of technologies and standards, which conforms a communication service among applications at different locations. This communication service allows faster integration and collaboration than using other integration technologies on the market.

Finally, by using Web Services it is possible to guide an integration process so that multiple applications are considered as a simple piece of a business process [35].

Two important elements must be mentioned for Web Services: *UDDI* and *WSDL*. UDDI (Universal Description Discovery and Integration) is a private initiative that was developed to support Web Service discovery across technologies and platforms. UDDI is a public registry designed to store information about businesses and their services in a structured way. This information includes web service interfaces that are accurately classified. Using a set of SOAP-based API XML calls, clients can interact with UDDI at both design and run time to discover technical data, such that those services can be invoked and used. In this way, UDDI serves as an infrastructure for software based on services Web.

On the other hand, WSDL (Web Service Description Language) [17] is an extension of XML, which allows describing syntax of Web services, so that users know how to invoke them. WSDL has became a key piece in the pile of protocols of Web Services, hence it is important to describe how UDDI and WSDL collaborate. Both WSDL and UDDI were designed to clearly delineate between abstract meta-data and concrete implementations. One of the most interesting consequences of this is that there can be multiple implementations of a single WSDL interface. This design allows different systems to write implementations of the same interface.

UDDI establishes a similar distinction between abstraction and implementation through the tModels concept (technology Model). The structure tModels represents interfaces and abstract types of metadata. In contrast, tModels are binding templates that concretely implement one or more tModels. Inside a binding template, the access point for a particular implementation of a tModel is recorded. In the same way that the outline of WSDL allows us to separate interface and implementation, UDDI provides a mechanism that allows publishing tModels separately from binding templates that reference them. From the structural point of view, a file WSDL can constitute a tModel of UDDI.

Finally, a Web service cannot be accessed by everyone just because he has been registered in the UDDI. Each entry in the UDDI registry can include features of security, authorization and authentication, which controls Web services access.

4. Semantic Web Services

The Semantic Web and the Web Services concepts are combined using the advantages of each of them creating the *Semantic Web Service* concept. The Semantic Web is useful to create a repository of computer-readable data, and Web Services provides the tools for using that data automatically. Nowadays there are several research works in the literature combining Semantic Web with Web Services. In this section we will see different approaches of Semantic Web Services describing each approach with their own characteristics.

The first approach, presented in [25,26,27,28], use DAML-S to add semantics to the Web Services. DAML-S is a language based on DAML+OIL [8] ontology which is being developed by a coalition of several researchers [9]. This language has been created to bridge the gap between the specification of the format of the information to be exchanged and the specification of its meaning. DAML-S contains three main types of knowledge about a service: a *service profile*, a *service model* and the *grounding* .A *service profile* provides a high-level description of a service and its provider. It is used to request or advertise services with discovery services and capability registries. Service profiles consist of three types of information: a description of the service and the service provider; the functional behavior of the service; and several functional attributes tailored for automated service selection. A *service model* enables an agent to perform a more in-depth analysis of whether the service meets its needs, compose service descriptions from multiple services to perform a specific task, coordinate the activities of different agents, and monitor the execution of the service. In conclusion, the service profile provides the information needed for an agent to discover a service, whereas the service model provides enough information for an agent to make use of a service [9]. Finally, the *grounding* describes how atomic processes; which provide abstract descriptions of the information exchanges with the requesters, are transformed into concrete messages that can be exchanged on the net, of through procedure call.

Two types of mappings are proposed by the authors: UDDI representation on DAML-S and WSDL on DAML-S. The first one has the objective of allowing a semantic description and matching services within UDDI. A mapping completely embedded of DAML-S profiles into UDDI representations is shown [25] allowing that all the search functionalities provided by UDDI can be used to retrieve information about services that are represented as DAML-S services. Besides, a matching engine has been implemented to perform inferences based on DAML+OIL logics and effectively adds capability matches to UDDI. In order to do so, two architectures [25,26] are presented: DAML-S/UDDI Matchmaker architecture (Figure 4) and DAML-S matching-engine architecture (Figure 5).

The Matchmaker receives advertisements of services and requests for service in DAML-S format from outside through the *Communication Module*. Then, the Communication Module sends it to the *DAML-S/UDDI Translator* that constructs a UDDI service description using information about the service provider, and the service name. The result of the registration with UDDI is a reference ID of the service. This ID combined with the capability description of the advertisement are sent to the DAML-S Matching Engine that stores the advertisement for capability matching. The requests for the opposite direction are: the Communication Module sends them to the DAML-S Matchmaker that performs the capability matching. The result of the matching is the advertisement of the providers selected and a reference to the UDDI service record. The combination of UDDI records and advertisements is then send to the requester.

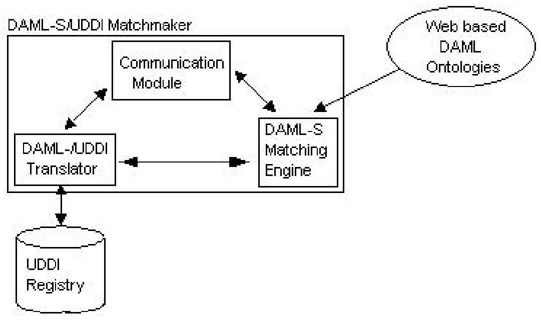

Figure 4: DAML-S/UDDI Matchmaker architecture

In Figure 5, the DAML-S based matching engine is presented. After a request is received, the *Matching Engine* component selects the advertisements from the *AdvertisementDB* that are relevant for the current request. Then it uses the *DAML+OIL Reasoner* to compute the level of match. In turn the DAML+OIL Reasoner uses the *OntologyDB* as data to be used to compute the matching process. The AdvertisementDB also takes advantage of the *OntologiesDB* to index advertisements for fast retrieval at matching time.

In the second type of mapping, researchers have built a tool for the translation of WSDL into DAML-S specification [26,27,28]. The tool called WSDL2DAMLS takes as input a WSDL specification and it returns as output a partial DAML-S description of the Web service. The results of this translation are a complete specification of the grounding and an incomplete specification of the profile and service model.

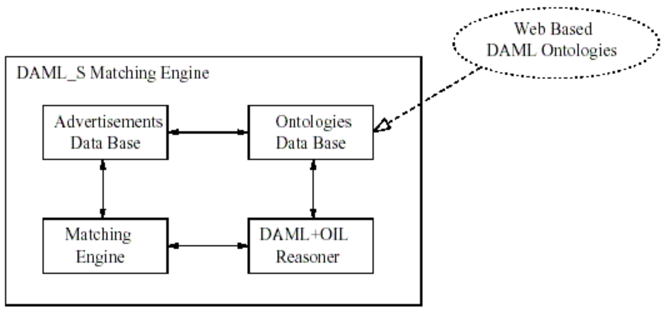

Figure 5: DAML-S matching-engine architecture

The second approach, presented in [20,22], describes a framework to Semantic Web services. DAML-S has been used to make the machine understandable and use-apparent. The authors have developed an agent technology that exploits the DAML-S language to support automated Web service composition and interoperability. Thus, this new technology enables three main services: *discovery*, *execution*, and *composition and interoperation*. *Automatic Web service discovery* involves automatically locating Web services that provide a particular service. To do so, a semantic markup (DAML-S) of service has been included allowing to specify the information necessary for Web service discovery as computer-interpretable semantic markup at the service Web sites. Then, a service registry search engine can automatically locate appropriate services. This approach associates properties with services that are relevant to automated service classification and selection. Thus, the semantics added provides a declarative advertisement of service properties and capabilities, which is computer–interpretable and can be used for automatic service discovery. *Automatic Web service execution* involves a computer program or agent automatically executing an identified Web service. Semantic markup of Web services provides a declarative, computer-interpretable API for executing services. The markup tells the agent what input is necessary, what information will be returned, and how to execute, and potentially interact with, the service automatically. *Automatic Web service composition and interoperation* involves the automatic selection, composition, and interoperation of appropriate Web services to perform some task, given a high-level description of the task's objective. With semantic markup of Web services, the information necessary to select, compose, and respond to services is encoded at the service Web sites. Software can be written to manipulate this markup, together with a specification of the task's objectives, to achieve the task automatically.

This approach is based on model based programming [21] and agent programming language as ConGolog [10]. The objective of the approach is to provide a DAML-S-enabled agent programming capability that supports writing generic procedures for Web service-based tasks. Model-based programs comprise a *model*, the agent's KB (Knowledge Base), and a *program*, the generic procedure we wish to execute. When a user requests a generic procedure, the agent populates its local KB with the DAML-S Web service markup that is relevant to the procedure. The model-based programs, such as the generic procedures, are written in ConGolog without prior knowledge of what specific services the agent will use or of how exactly to use the available services. They capture what to achieve but not exactly how to do it. They use procedural programming language constructs (if-then-else, while, and so forth) composed with concepts defined in the DAML-S and constraints ontologies to describe the procedure. The agent's model-based program is not executable as is. It must be deductively instantiated in the context of the agent's KB, which includes properties of the agent and its user, properties of the specific services we are using, and the state of the world. The instantiation is performed by using deductive machinery.

The third approach, presented in [6,11], describes the Web Service Modeling Framework (WSMF) in which the concept of SWWS (Semantic Web Enabled Web Services) is introduced. The SWWS objective is the transformation of the Web from a static collection of information into a distributed device of computation on the basis of Semantic Web technology making content within the World Wide Web machine-processable and machine-interpretable. Various aspects have been taken into account to achieve an intelligent and automatic Web service discovery, selection, mediation and composition into complex services: *Document Types*, that use ontologies to describe the content of business documents; *Syntax*, in which the documents can be represented using for example XML; *Semantics*, in which an ontological language can be used to represent the correct values of the elements of document types; *Transport Binding*, a service requester as well as a service provider have to agree on the transport (HTTP/S, FTP, etc.) to be used when service requests are executed; *Security*, each message exchange should be private and unmodified between the services requester and service provider; etc. With all of these aspects in mind, the WSMF has been built. The WSMF provides the model for developing and describing Web services and their composition. A model in WSMF consists of four main different elements: *Ontologies*, *Goal repositories*, *Web services descriptions* and *Mediators*. The *ontologies* are used to define the vocabulary that is used by other elements of WSMF specifications (document types). They enable reuse of terminology as well as interoperability between components referring to the same or linked concepts. The *goal repositories* define the problems that should be solved by web services. The description of a goal specifies objectives that a client may have when he consults a Web service. The same web service can serve different goals and obviously different (competing) web services can serve the same goal. The *Web services descriptions* define various aspects of a web service. For example, distinguish between elementary and complex Web services. WSMF identifies more than ten aspects. The *mediators* which bypass interoperability problems, for example, they mediate between different interaction styles of components.

The WSMF defined in this approach is close to the work that DAML-S done. Both use the Semantic Web's key enabling technology of ontologies as their basis. A detailed comparison between DAML-S and WSMF is provided in [14]. Besides, the SWWS Conceptual Architecture [6] has been designed to enable semantic-driven Web service applications. The ontologies can be represented in RDF or OWL.

This approach is being refined by developing a formal ontology and a formal language. The Web Service Modeling Ontology (WSMO) [30] has been developed to show which elements are used for each WSMF element (ontologies, goals, web service descriptions and mediators). Many properties of goals, mediators, ontologies and web services are considered to be axioms, defined by logical expressions. These logical expressions as being described in F-Logic [19].

In the fourth approach, presented in [18], UDDI is considered as a search engine with a restrictive functionality, because it is based on keyword retrieval only. Therefore, authors developed a Semantic Services

Matchmaker to enhance the UDDI search functionality by using semantic elements such as ontology and constraints. The goal of the work in [18] is to combine semantic search with standard specifications such as SOAP, WSDL and UDDI, and to investigate the feasibility of using semantics along with Web Services.

The Semantic Services Matchmaker is based on the LARKS [34] algorithm, but it also uses some ideas from the DAML-S matching algorithm [29]. Specifically, it adopts the LARKS filtering approach, which uses complex information retrieval mechanisms to locate Web services advertisements in UDDI when no semantic information has been provided. The experiment introduced in [18] tries to show the contribution and the limits of information retrieval techniques to Web services discovery, and the contribution of ontological information to improve matching processes.

In the architecture of Matchmaker integrated with UDDI Registry, Matchmaker is inserted between the users or requesters of the services and the UDDI registry. The Matchmaker API is equivalent to the UDDI API that facilitates the user's connection. Moreover, format of results returned by both applications is the same.

Ideally, when a requester looks for a service, matchmaker will retrieve a service that matches exactly the service that the requester expects. In practice it is very unlikely, so matchmaker will retrieve a service whose capabilities are similar to the capabilities expected by the requester. Furthermore, matchmaker should be able to characterize the distance between the service requested and the services found, so that the requester can make a decision on which service invokes. To address this problem, matchmaker identifies three levels of matching: *Exact* match resulting when the two descriptions are equivalent, *Plug-in* match resulting when the service found is more general than the service requested, and *Relaxed* match that is used to indicate the degree of similarity between the advertisement and the request.

The second aspect of the matchmaker is to provide a set of filters that help with the matching process. Basically, these filters are: *Namespace Filter,* that determines whether the requested service and the registered ones have at least one shared namespace; *Text Filter*, that looks for human-readable service explanation parts such as comments; *Domain Filter*, that checks whether each registered service and the requested one belong to an ontology domain; *I/O Type Filter*, that checks if the definitions of the input and output parameters match; and *Constraint Filter*, that compares restrictions to determine if the registered service is less constrained than the requested.

Finally, authors of [18] describe the Web Service Semantic Profile (WSSP). WSSP is a way to encode semantic information in WSDL and it is inspired by the DAML-S Service Profile [9]. The aim of the WSSP is to enrich WSDL with the semantics it needs to represent capabilities of web services. As in a WSDL file, WSSP defines each message and each parameter of those messages. Moreover, WSSP offers the possibility to add constraints to inputs and outputs of a web service. It gives the possibility to represent a web service more accurately, and it allows the search engine to perform a more accurate search. Constraints are added to the WSDL specification through the element *constrainedBy*.

In the fifth approach, presented in [23], it is described another framework to include semantic in Web Services called IRS-II (Internet Reasoning Service). Furthermore, IRS-II is an implemented infrastructure for semantic Web services. The main goal of IRS-II is to support the publication, location, composition and execution of heterogeneous web services, augmented with semantic descriptions of their functionalities.

IRS-II has three main classes of features which distinguish it from other works on semantic web services. Firstly, it supports automatic transformation of standalone software (written in Java or Lisp programming languages) into web services. Secondly, IRS-II builds a knowledge model based on reusable components and as a result, its architecture explicitly separate task specifications (the problems which need to be solved), from the method specifications (ways to solve problems), and from the domain models (where these problems need to be solved). Finally, IRS-II services are web service compatible, i.e. standard web services can be published though the IRS-II and any IRS-II service automatically appears as a standard web service to other Web Service infrastructure.

IRS-II was developed as a part of the IBROW project [3], whose goal was to support application development through the automatic configuration of reusable knowledge components, available from distributed libraries on the Internet. These libraries are structured according to the UPML framework [12]. UPML identifies three classes of components: *Domain models* describing the domain of an application, *Task models* providing a generic description of the task to be solved (specifying the input and output types, the goal to be achieved and the preconditions), *Problem Solving Methods* (PSMs) providing abstract descriptions of reasoning processes, and *Bridges* specifying mappings between the different models within an application. Each class of component is specified by means of an appropriate ontology.

Basically, in the IRS-II architecture, we distinguish three main components: the Server IRS, the IRS Publisher and the IRS Client, which communicate through a SOAP-based protocol. IRS server stores descriptions of semantic web services. A knowledge level description is stored using the UPML framework of tasks, PSMs and domain models. They are represented internally in OCML [24], an Ontolingua-derived language which provides both the expressive power to express task specifications and service competencies, as well as the operational support to reason about them. The IRS Publisher plays two roles in the IRS-II framework. Firstly, it links web services to their semantic descriptions within the IRS server. Secondly, it automatically generates a set of wrappers which turn standalone Lisp or Java code into a web services described by a PSM. Finally, the IRS Client supports web service invocation through capability driven, which is a key feature of IRS-II. An IRS-II user simply asks for a task to be achieved and the IRS-II broker locates an appropriate PSM and then invokes the corresponding web service.

5. Discussion

In this section we present a simple framework to characterize the approaches described in the last section. Our framework is based on five important aspects needed to implement Semantic Web Services. Figure 6 shows the framework elements.

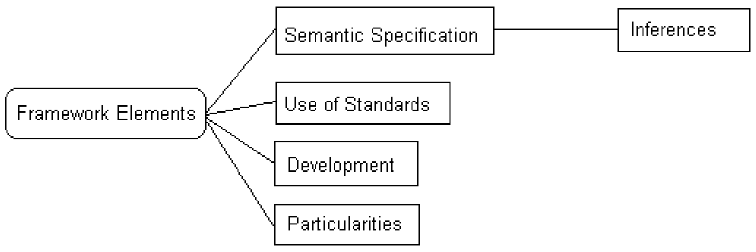

Figure 6: Framework Elements

As we can see, the framework has four main elements: *Semantic Specification*, *Use of Standards*, *Development* and *Particularities*. *Semantic Specification* indicates the language or framework used to add semantics to the specification of the Web Services. This element has another related element: *Inferences*. Each approach, in accordance with the language used for the semantics, uses an different inference engine. These inferences are used to map requested services with available services. The *Use of Standards* element indicates which approaches apply semantics in Web Services standards. The *Development* element indicates the developed aspects for each approach in order to implement a Semantic Web Service. For example, if the approach develops a framework; or a framework and an infrastructure; or the tools to support some characteristic of its system. Finally, the *Particularity* element indicates the individual aspects of each approach in order to implement their proposals.

Table 1 shows how the five main approaches implement the frameworks' elements. The first approach by Paolucci et al [25,26,27,28], presents two mappings: DAML-S with UDDI and DAML-S with WSDL. For the first mapping, the authors generate an infrastructure of a DAML-S/UDDI Matchmaker in which the semantics is added in the UDDI registry. For the second mapping, the semantic is added in the service specification messages, that is, WSDL. Both mapping use DAML-S as semantic language. To make the inferences in order to map the requested services with the available services, DAML+OIL logic is used. Besides, this approach defines an architecture and creates a tool (WSDL2DAMLS) in order to transform a WSDL specification into a partial DAML-S description of the Web service. Note that it is very important that the authors introduce DAML as a language to specify Web Services.

The second approach by McIlraith et al [20,22], presents an implementation of a set of agents that allow the discovery, execution , and composition and interoperation of Web Services. Each agent contains a KB local in which the semantics is specified with DAML-S. In order to make the inferences, the approach uses ConGolog programs allowing us to define generic procedures that will be invoked by clients.

The approach by Fensel et al [6,11] and refined by Roman et al [30], presents a different alternative. Firstly, it does not use DAML-S, rather it describes a framework for the service specification (WSMF). This framework has all the semantic information to implement a Semantic Web Service. A SWWS architecture has been created to enable semantic-driven Web service applications. The particularity of this approach is the use of mediators to map the ontologies. Besides, a refinement has been developed in order to formalize the WSMF elements. Each element can be written using the F-Logic language.

Approaches	Semantic Specification	Inferences	Use of Standards	Development	Particularities
Approach by Paolucci et al [25,26,27,28]	DAML-S	DAML+OIL	UDDI WSDL DAML-S	Infrastructure, framework and a tool to partially map WSDL to DAML-S	Introduce the use of DAML in Web Services
Approach by McIlraith et al [20,22]	DAML-S	ConGolog	DAML-S	Infrastructure and framework	Use of agents and locals KB
Approach by Fensel et al [6,11]. Refined by Roman et al [30].	WSMF with WSMO	F-Logic	WSMO	SWWS Framework and WSMO	Use of mediators
Approach by Kawamura et al [18]	DAML-S and others	DAML+OIL and others	UDDI WSDL	Infrastructure and framework	Combine LARKS filters with DAML-S
Approach by Motta et al [23]	UPML	Ontolingua OCML	-	Infrastructure, framework and a tool to transform applications to Web Services automatically	A tool to generate Web Services based on stand-alone programs

Table 1: Framework Elements with the five approaches

The approach by Kawamura et al [18], implements a framework and an infrastructure based on the Matchmaker defined in the first approach (Paolucci et al). The particularity is the use of LARKS filters combined with DAML-S specification in order to improve the service discovery. Equal to the Paolucci et al approach, this approach uses DAML-S in the UDDI registry and in WSDL but the inferences can be done using OWL, DAML+OIL or RDFS logics.

Finally, the approach by Motta et al [23], generates another framework (IRS-II) with an implemented infrastructure. This approach is related with Fensel et al approach because it use the framework WSMF instead of DAML-S. It only use SOAP as communication standard. As particularity, a tool to generate web services based on stand-alone programs written in Java or Lisp is implemented. IRS-II uses OCML logic in order to make the inferences.

6. Conclusion

In this paper, we have presented both an analysis and a characterization of five proposals that implement Semantic Web Services. In order to do so, we have created a conceptual framework with four main elements: *Semantic Specification*, *Use of Standards*, *Development* and *Particularities*. Based upon our description, we have found some elements in common and also original aspects.

Our work aims at contributing to the Semantic Web Services community by giving and analyzing different aspects of five proposals which have been used as a reference for further research. Moreover, this work is useful for anyone who is introduced in this topic.

All of these proposals are still under development, and currently the research community is focusing on different aspects to improve both the framework as well as the system infrastructure. To do so, some approaches are based on Web Services standards such as WSDL and UDDI. We consider that the use of standards is useful to make the use of Semantic Web Services be supported by all software community. Therefore, approaches that apply semantic through standards have a major projection in the software industry.

References

1. Antoniou, G., Harmelen F. Web Ontology Language: OWL. Handbook on Ontologies in Information Systems. Staab & Studer Editors. Springer-Verlag, 2003.
2. Baader, F., Calvanese, D., McGuiness, D., Nardi, D. and Patel-Schneider, P. editors. The Description Logic Handbook - Theory, Implementation and Applications. Cambridge University Press, ISBN 0-521-78176-0, 2003.
3. Benjamins, V., Plaza, E., Motta, E., Fensel, D., Studer, R, Wielinga, B., Schreiber, G., Zdrahal, Z., Decaer, S. An intelligent brokering service fro knowledge-component reuse on the Word-Wide Web. *11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Canada 1998.
4. Berners-Lee, T., Hendler, J., Lassila, O. The Semantic Web. Scientific American, May 2001.
5. Berners-Lee, T. Weaving the Web. Harper, San Francisco,1999.
6. Bussler, C., Fensel, D., Maedche, A. A Conceptual Architecture for Semantic Web Enabled Web Services *SIGMOD Record*. Vol.31(4), pp. 24-29, 2002.
7. Cabral, L., Domingue, J., Motta, E., Payne, T. and Hakimpour, F. Approaches to Semantic Web Services: An Overview and Comparisons. In proceedings of the *First European Semantic Web Symposium (ESWS2004)*; May 10-12, 2004, Heraklion, Crete, Greece.
8. Connolly, D. ,Van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L. DAML+OIL Reference Description. W3C, December 18, 2001. Available at http://www.w3.org/TR/daml+oil-reference.
9. DAML-S Coalition: A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAMLS: Semantic markup for Web services. In Proc SWWS, pp. 411-430, 2001.
10. De Giacomo, G., Lesperance, Y. and Levesque, H. ConGolog, a Concurrent Programming Language Based on the Situation Calculus. *Artificial Intelligence*, Vol 1–2(121), pp. 109–169, Aug. 2000.
11. Fensel, D., Bussler, C. and Maedche, A.. Semantic Web Enabled Web Services. In Proceedings of the *International Semantic Web Conference 2002*, LNCS, Springer, pp. 1-2, 2002.
12. Fensel, D., Motta, E. Structured Development of Problem Solving Methods. *IEEE Transactions on Knowledge and Data Engineering*. 13(6), pp. 913-932. 2001.
13. Ferris, C., Farrel, J. What Are Web Services. *Communications of the ACM*. Vol 46(6), 2003.
14. Flett, A. A comparison of DAML-S and WSMF. In *Internal Report, Vrije Universiteit Amsterdam*, 2002.
15. Haarslev, V. and Moller, R. RACER system description. In P. Lambrix, A. Borgida, M. Lenzerini, R. Moller, and P. Patel-Schneider, editors, Proceedings of the International Workshop on Description Logics, number 22 in CEUR-WS, Linkoeping, Sweden, July 30-August 1 1999.
16. Horrocks, I. The FaCT system. In H. de Swart, editor, Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98, number 1397 in Lecture Notes in Artificial Intelligence, pages 307–312. SpringerVerlag, Berlin, May 1998.
17. Januszewski, K. Web Service Description and Discovery Using UDDI. Microsoft http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service10032001.asp. October 2001.
18. Kawamura, T., De Blasio, J., Hasegawa, T., Paolucci, M., Sycara, K.Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry. *ICSOC 2003*, LNCS 2910, pp. 208-224, 2003.
19. Kifer, M., Lausen, G. and Wu, J. Logical foundations of object oriented and frame-based languages. Journal of the *ACM*, Vol.42(4), pp. 741-843, 1995.
20. McIlraith, S., Son, T., Zeng, H. Semantic Web Services. *IEEE Intelligent Systems*, pp. 46-53, March/April, 2001.
21. McIlraith, S. Modeling and Programming Devices and Web Agents. In *Proc. NASA Goddard Workshop Formal Approaches to Agent-Based Systems*, Lecture Notes in Computer Science, Springer-Verlag, New York, 2001.
22. McIlraith, S.,Martin, D. Bringing Semantics to Web Services. *IEEE Intelligent Systems*, pp. 90-93, Jan/Feb, 2003.
23. Motta, E., Domingue, J., Cabral, L., Gaspari, M. IRS-II: A Framework and Infrastructure for semantic Web Services. *2nd International Semantic Web Conference (ISWC2003)*. Florida, USA. 20-23 October, 2003.
24. Motta, E. Reusable Components for Knowledge Modelling. *IOS Press. Amsterdam*. The Netherlands, 1999.
25. Paolucci, M., Kawamura, T., Payne, T., Sycara, K. Importing Semantic Web in UDDI. *WES 2002*, LNCS 2512, pp.225-236, 2002.
26. Paolucci, M., Sycara, K. Autonomous Semantic Web Services. IEEE Internet Computing, pp. 34-41, Sept/Oct, 2003.
27. Paolucci, M., Sycara, K., *Kawamura, T. Delivering Semantic Web Services. Technical report CMU-RI-TR-02-32, Robotics Institute, Carnegie* Mellon University, May, 2003..
28. Paolucci, M., Srinivasan, N., Sycara, K., Nishimura, T. Towards a Semantic Choreography of Web Services: from WSDL to DAML-S. In Proceedings of *ICWS'03*, 2003.
29. Paulocci, M., Kawamura, T., Payne, T., Sycara, K. Semanitc Matching of Web Services Capabilities. Proceedings of *First International Semantic Web Conference (ISWC 2002)*, IEEE, pp. 333-347, 2002.
30. Roman, D., Lausen, H., Keller, U. Web Service Modeling Ontology - Standard. WSMO Working Draft March 06, 2004. Available at http://www.wsmo.org/2004/d2/v0.2/20040306/#L3907.
31. Roy, J., Ramanujan, A. Understanding Web Services. *IT Profesional* Vol.3(6), pp. 68-73, November, 2001.
32. Smith, M.K.,Welty, C., McGuinness, D.L. OWL Web Ontology Language Guide. W3C, http://www.w3.org/TR/2004/REC-owl-guide-20040210/. 10 February 2004.
33. Stafford, T. E – Services. *Communications of the ACM*. Vol. 46(6), pp- 27–28, 2003.
34. Sycara, K., Widoff, S., Klusch, M., Lu J. LARKS: Dynamic Matchmaking among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agents Systems*, Vol 5, pp. 173-203, 2002.
35. Watson, S., Dunlop, G. A Definition of Web Services. Itercea (www.Intercea.co.uk). August 2002.
36. XML Org. http://www.xml.org/.
37. http://www.w3.org/2000/Talks/1206-xml2k-tbl/.