University of Crete
School of Sciences and Engineering
Computer Science Department

# P2P Service Provisioning In The AXML Framework

by

Themistoklis Kutsuras

Master's Thesis

Heraklion, February 2008

University of Crete
School of Sciences and Engineering
Computer Science Department

# P2P Service Provisioning In The AXML Framework

by

Themistoklis Kutsuras

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

Author:

_____

Themistoklis Kutsuras, Computer Science Department

Supervisory
Committee:

_____

Dimitris Plexousakis, Professor, Supervisor

_____

Christos Nikolaou, Professor, Member

_____

Grigoris Antoniou, Professor, Member

Approved by:

_____

Panos Trahanias, Professor
Chairman of the Graduate Studies Committee

Heraklion, March 2008

# Abstract

One of the most important aspects in computer science is the management of data. Web services provide an infrastructure for distributed computing at large, independently of any platform, system or programming language. Peer-based architectures, in which resources are shared by direct exchange between systems, offer a significant advantage and can be the base of the scalable data integration.

Data and actions that are represented through web services should be described, offering ways for automating the processes. The discovery of the appropriate services should be depended on logic-based formalizations in order to enhance their functionality and improve their proper matching. Moreover, the issue of the composition of web services is of major significance as it relies on the composition of existing business services into new and more complex services. A major limitation of the web services technology is that finding and composing services still requires manual effort. The increasing number of web services surcharges the previous limitation.

The evolution of semantic web services in addition with the development of newly frameworks has given powerful means for dealing with the previous challenges. Active XML will be presented, a new idea which having as its base the combination of XML and web services provides a simple and unified platform for the management of such active data. The combination of them appoints the suitable framework for distributed management of information. AXML is a language that empowers web services to achieve data integration by embedding calls to other web services and is structured over a peer-to-peer architecture. Thus, it is obvious to use these means having as priority to succeed in formalizing the mentioned challenges in a more automatic way.

This thesis proposes a framework that sets an interaction-based approach at the communication of peers, through the use of service calls. Its main advantages are the existence of different roles for every peer and the dynamic way that impose in the composition process. The presence of AXML serves the notion of active data as a

means of communication between peers, adding efficiently the periodical activation and materialization of service calls. Specifically, the concept of the implicit call in AXML documents is introduced and a technique for discovering and composing web services in a role-based p2p network is described. The elements that are necessary for AXML documents in order to enhance the description of the embedded services are presented, supporting the use of OWL-S descriptive characteristics. Moreover, the structure of such a network and the way of its development is presented, outlining all the characteristics and the way of involvement of its peers. The whole methodology, advancing the functionality that AXML provides, set a clear layout for data integration.

*Supervisor*: Dimitris Plexousakis
Professor

# Περίληψη

Μια από τις σημαντικότερες πτυχές στην πληροφορική είναι η διαχείριση των δεδομένων. Οι υπηρεσίες Ιστού (web services) παρέχουν την υποδομή για κατανεμημένη διαχείριση, ανεξάρτητα από οποιαδήποτε πλατφόρμα, σύστημα ή γλώσσα προγραμματισμού. Οι αρχιτεκτονικές ομότιμων συστημάτων, στις οποίες οι πόροι διαμοιράζονται από την άμεση ανταλλαγή μεταξύ των συστημάτων, προσφέρουν ένα σημαντικό πλεονέκτημα και μπορούν να είναι η βάση της ολοκλήρωσης των δεδομένων.

Τα δεδομένα και οι ενέργειες που αντιπροσωπεύονται μέσω των υπηρεσιών Ιστού πρέπει να περιγραφούν με τέτοιο τρόπο ώστε να παρέχουν τρόπους αυτοματοποίησης των διαδικασιών που εκτελούν. Η ανεύρεση των αρμόδιων υπηρεσιών πρέπει να εξαρτάται από λογικούς φορμαλισμούς προκειμένου να ενισχυθεί η λειτουργία τους και να βελτιωθεί το κατάλληλο ταίριασμά τους. Επιπλέον, το ζήτημα της σύνθεσης των υπηρεσιών Ιστού είναι μεγάλης σημασίας καθώς στηρίζεται στη σύνθεση των υφιστάμενων υπηρεσιών σε νέες και πιο σύνθετες υπηρεσίες. Ένας σημαντικός περιορισμός της τεχνολογίας υπηρεσιών Ιστού είναι ότι η εύρεση και η σύνθεση των υπηρεσιών απαιτούν ακόμα την εμπλοκή του χρήστη. Ο αυξανόμενος αριθμός υπηρεσιών Ιστού επιβαρύνει τον προηγούμενο περιορισμό.

Η εξέλιξη των σημασιολογικών υπηρεσιών Ιστού (semantic web services) σε συνάρτηση με την ανάπτυξη νέων τεχνολογικών πλαισίων παρέχει ισχυρά μέσα για την αντιμετώπιση των προηγούμενων προκλήσεων. Η ActiveXML θα παρουσιαστεί, μια νέα ιδέα που έχοντας ως βάση του το συνδυασμός υπηρεσιών XML και Ιστού παρέχει μια απλή και ενοποιημένη πλατφόρμα για τη διαχείριση τέτοιων ενεργών δεδομένων. Ο συνδυασμός τους ορίζει το κατάλληλο πλαίσιο για τη κατανεμημένη διαχείριση των πληροφοριών. Η AXML είναι μια γλώσσα που καθιστά ικανές τις υπηρεσίες Ιστού να πραγματοποιήσουν την ολοκλήρωση των στοιχείων με την ενσωμάτωση κλήσεων σε άλλες υπηρεσίες Ιστού όντας δομημένο σε μία ομότιμη αρχιτεκτονική. Κατά συνέπεια, γίνεται σαφές ότι η χρησιμοποίηση αυτών των μέσων

έχουν ως προτεραιότητα την επίτευξη της τυποποίησης των προαναφερθέντων προκλήσεων με έναν πιο αυτόματο τρόπο.

Αυτή η διατριβή προτείνει ένα πλαίσιο όπου η επικοινωνία των κόμβων (peers), μέσω της χρήσης των κλήσεων υπηρεσιών, βασίζεται σε συγκεκριμένους τρόπους αλληλεπίδρασης. Τα κύρια πλεονεκτήματά του είναι η ύπαρξη των διαφορετικών ρόλων για κάθε peer και ο δυναμικός τρόπος που επιβάλλουν στη διαδικασία σύνθεσης. Η παρουσία της AXML, εξυπηρετώντας την έννοια των ενεργών δεδομένων ως μέσο επικοινωνίας μεταξύ των peers, προσθέτει αποτελεσματικά την περιοδική ενεργοποίηση και την υλοποίηση των κλήσεων υπηρεσιών. Συγκεκριμένα, εισάγεται η έννοια της έμμεσης κλήσης στα AXML έγγραφα και περιγράφεται μια τεχνική για την ανεύρεση και σύνθεση υπηρεσιών Ιστού σε ένα ομότιμο δίκτυο όπου ο κάθε εμπλεκόμενος peer φέρει ένα συγκεκριμένο ρόλο. Παρουσιάζονται τα στοιχεία που είναι απαραίτητα για τα AXML έγγραφα προκειμένου να ενισχυθεί η περιγραφή των ενσωματωμένων υπηρεσιών, υποστηρίζοντας τη χρήση OWL-S περιγραφικών χαρακτηριστικών. Επιπλέον, αναλύεται η δομή ενός τέτοιου δικτύου και ο τρόπος ανάπτυξής του, περιγράφοντας όλα τα χαρακτηριστικά και τον τρόπο της συμμετοχής των peers. Ολόκληρη η μεθοδολογία, που προσαυξάνει την υπάρχουσα λειτουργικότητα της AXML, θέτει ένα σαφές σχεδιάγραμμα για την ολοκλήρωση των δεδομένων.

*Επόπτης*: Δημήτρης Πλεξουσάκης
Καθηγητής

*Στους γονείς μου Κωνσταντίνο και Δέσποινα και τον αδερφό μου Μαυρουδή για την αγάπη και την υποστήριξη τους σε κάθε προσπάθεια της ζωής μου.*

# Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω κατ' αρχήν τον επόπτη μου κ. Δημήτρη Πλεξουσάκη που μου έδωσε τη δυνατότητα να ασχοληθώ με ένα επίκαιρο και ενδιαφέρον θέμα. Η συνεργασία μας σε όλα τα επίπεδα ήταν εποικοδομητική καθ' όλη τη διάρκεια του μεταπτυχιακού προγράμματος και ιδιαίτερα κατά την περίοδο εκπόνησης της εργασίας όπου οι ουσιαστικές υποδείξεις και συμβουλές του με καθοδήγησαν με τον καλύτερο δυνατό τρόπο.

Επίσης, θα ήθελα να ευχαριστήσω θερμά τον κ. Χρήστο Νικολάου για την προθυμία του να συμμετάσχει στην εξεταστική επιτροπή μου και για τον ευπρόσδεκτο τρόπο με τον οποίο με δέχτηκε όσον αφορά στην χρησιμοποίηση χώρων του εργαστηρίου το οποίο διευθύνει.

Παράλληλα θα ήθελα να ευχαριστήσω ιδιαιτέρως τον κ. Γρηγόρη Αντωνίου για τη θέληση που επέδειξε να συμμετάσχει στην εξεταστική επιτροπή μου και για τις εύστοχες παρατηρήσεις του.

Ένα μεγάλο ευχαριστώ θα ήθελα να δώσω στον Μάνο Παπαγγελή, η συνεργασία μου με τον οποίο μου έδωσε τα πρώτα ερεθίσματα ώστε να ασχοληθώ με θέματα ερευνητικής φύσεως και να ακολουθήσω τη συγκεκριμένη πορεία.

Επιπλέον θα ήθελα να ευχαριστήσω τους Γιώργο Κούτρα, Παντελή Πετρίδη, Αλίνα Ψυχαράκη, Μαριάννα Καρμαζή, Γιάννη Ρουσίδη, Βαγγέλη Καλογεράκη, Ιωάννη Καντά, Βαγγέλη Παπαθανασίου για τις πολύωρες συζητήσεις μας σε ερευνητικά και μη ζητήματα οι οποίες συνέβαλαν σημαντικά στην επίλυση απορίων/προβλημάτων. Επίσης, ένα ευχαριστώ ανήκει και σε όλους τους υπόλοιπους φίλους-ες μου με τους οποίους έχουμε μοιραστεί κοινές και ευχάριστες στιγμές και εμπειρίες και οι οποίοι μου συμπαραστέκονταν σε κάθε στιγμή.

Κλείνοντας θα ήθελα να επισημάνω το σημαντικότατο ρόλο του πατέρα μου Κωνσταντίνου, της μητέρας μου Δέσποινας και του αδερφού μου Μαυρουδή σε όλη αυτή την πορεία μου τόσο στο προπτυχιακό όσο και στο μεταπτυχιακό μέρος των σπουδών μου για τους οποίους οι λέξεις συμπαράσταση, αρωγή, αγάπη είναι οι

ελάχιστες που μπορούν να περιγράψουν το μέγεθος της βοήθειας τους προς το πρόσωπό μου.

# Table of Contents

# List of Figures

# List of Tables

# Introduction

## 1.1 Motivation

One of the most important aspects in computer science is the management of data. The development of the necessary science and technology with use of central repositories has started in the early 60's. From the early days, the evolution of Internet technologies and the increase of needs for access to sources of distributed information has led to the extension of these techniques.

The relational model was one of the major subjects that the area of distributed data management has focused on for many years. Recently, the use of HTML as a standard language made the world wide publication of data much simpler and fully understood by Web browsers, on plain-text search engines and on query forms. However, HTML is a more presentational language than a data model, a fact that leaves the management of distributed information non effective.

The introduction of XML and Web services has changed totally the situation in this field. The Extensible Markup Language, XML [57] is a self-describing, semi-structured data model that is taking the place of the standard format for data exchange over the Web. XML, as a self-describing data model, raised great interest about how flexible and expressive is for the resolution of semantic heterogeneous issues. However a noticeable interoperability problem has as a source the dynamic way of data construction. The access of this kind of data was made attainable with the use of Web services.

Web services [63] provide an infrastructure for distributed computing at large, independently of any platform, system or programming language. The result of their use from various software components is the provision of a variety of functionalities

that are now accessible through the Web.Web services encapsulate programs with XML arguments having the ability to be described, published, located, invoked, and even operate with other services to create a new composed service over a network. SOAP [62] and WSDL [64] are the standards that assist firstly in the fulfilment of the previous properties of Web services and moreover in the normalisation of the way that programs are invoked over the web. This fact makes clear that centralised architectures oppose the essence of distribution that web promotes and in parallel seems week to manage its large scale. Peer-based architectures, in which resources are shared by direct exchange between systems, offer a significant advantage and can be the base of the scalable data integration.

In addition to the previous, *Semantic Web* [12], evolves techniques for enriching existing Web data with formal descriptions based on the logic of their meaning. The semantic markup is machine processable and thus accommodates access and integration of the majority of web data. The most important aspect of Semantic Web technology is *ontologies* [24]. An ontology is a formal representation of consensual world knowledge and it provides the elements for building semantic descriptions.

Web service technology has fulfilled the introduction of a new layer of abstraction that has assisted in the development of a new architecture for software. Web services represent several times business services and thus have become quite attractive as they enable the integration of different business organisations Web services, aiming in the accomplishment of the user's demands. The issue of the composition of Web services is of major significance as it relies on the composition of existing business services into new and more complex services. Integration is based on the common standards of web service interfaces, despite the languages that are used to implement the Web services and the platforms of the Web services execution.

A major limitation of the web services technology is that finding and composing services still requires manual effort. The increasing number of web services surcharges the previous limitation. To deal with this problem, there is a research direction to enrich Web services functionality with a semantic description which will appoint with a significant outfit the fulfilment of the discovery and integration processes. So, the technology of web services and Semantic Web

techniques are combined, a combination that is referred to as *semantic Web services* [46].

The use of web services in the management of data on the Web is a multifaceted topic in which many authors have studied. This study has set new challenging features such as the discovery of web services based on their functionality which has as a result the discovery of data sources that include desired data and the dynamic composition of Web services that helps in the retrieval of the dynamic/active data. A variety of techniques have been suggested following varying hypotheses and addressing different aspect of the problem. These techniques follow principles that involve the manual effort in order to address the problems.

Data and actions that are represented through web services should be described, offering ways for automating the processes. The discovery of the appropriate services should be depended on logic-based formalizations in order to enhance their functionality and improve their proper matching. The evolution of semantic web services in addition with the development of newly frameworks has given powerful means for dealing with the previous challenges. Thus, it is obvious to use these means having as priority to succeed in formalizing the mentioned challenges in a more automatic way.

## 1.2 Objective

In this work, Active XML will be presented, a new idea which having as its base the combination of XML and web services provides a simple and unified platform for the management of such active data. So, query answering acquires a new dimension for flexible use in the distributed context of the Web. The combination of them appoints the suitable framework for distributed management of information.

Active XML [53] (AXML, for short), is a declarative framework that controls and directs these emerging standards for the integration and management of distributed Web data. Moreover, AXML is a language that empowers web services to achieve data integration by embedding calls to other web services and is structured over a peer-to-peer architecture. AXML includes features in XML documents with certain properties to manage in order to succeed in retrieving dynamic data. These properties are related to the indication of the location of the service to be called and

the control of three aspects: the timing of the service invocation, the time that the results will be considered valid and the exchange of intensional and extensional data. By giving these characteristics, AXML permits us to use different styles of data integration – mediation, warehousing [6], [1] – and possible combinations of them.

An AXML document is an XML document where some of the data is given explicitly, while some parts are given only intensionally with the assistance of embedded calls to web services. The firing of these services calls results to the provision of up-to-date information. Specifically, AXML controls the activation of service calls either from the client side (pull) or from the server side (push). The approach that AXML promotes in query answering is based on *active answers*, answers that are AXML documents.

Various parameters may motivate and affect the choice of the parts of a query answer that should be given explicitly or in an active/intensional. These may contain logical reasoning, such as knowledge enrichment or context awareness, and physical reasoning like performance or capabilities.

The issue of logical reasoning of web services has found a satisfactory answer with the advent of OWL-S. Ontology Web Language for Services (OWL-S) [58] has been developed in order to provide the necessary additional parts to OWL [19], the Semantic Web language standardization at the W3C, with enhanced semantic service descriptions.

OWL-S addresses effectively the extra development of web service-related capabilities. Moreover, the existing technologies are provided with semantic expressiveness through OWL-S. Thus, the OWL-S constructs are a supplementary technology for the current dominant standards in the field of web services with which can be combined in order to provide richer functionality. Our main objective is to describe the way of use Semantic Web services, using OWL-S in conjunction with WSDL and related standards, and to present the gains of having richer semantics in the web services technology and especially in a framework of active information management such as AXML.

## 1.3 Structure

Chapter 2 presents the main concepts that are used in our work and discusses the related work to the specific domain of composition of web services as well as the approaches that have been made using p2p infrastructure.

Chapter 3 introduces us into semantic web approach, presenting an enhanced description language, OWL-S, which empowers web services with semantic reasoning of their functionality.

Chapter 4 presents the Active XML framework and its main characteristics, the motivations of its creation, the points that differ from simple XML and how it fulfils the management of resources.

Chapter 5 focuses on our methodology of combining the presented technologies, having as a result services to be expressed with an augmented way through OWL-S and integrated using some extra modules. Moreover, it highlights the structure and the relationships created in a p2p environment and how this affects services composition.

Chapter 6 summarizes the advantages of our methodology, concludes the research contributions of the thesis and draws directions for further research work.

Finally, there is a chapter where relative bibliography that is referred throughout this thesis is mentioned.

# Background Technology and Related Work

## 1.4 XML

### 1.4.1 XML in general

XML is the acronym for Extensible Markup Language. It is based on the International Standard for structured information (SGML). HTML is one more known language based on SGML. SGML has been implemented in text based languages that the use of tags enables the specification/definition of their structure. This fact offers them the characteristics of being human readable and easy understandable. XML could be presented as HTML's brother in the context that they have both SGML as their ancestor but some conceptual differences turns them to have a supplementary relation. Each language's goal is their main difference: HTML is designed to describe the way data should be represented while XML relies on data structure. HTML has a fixed set of tags on the contrary to XML which allows the programmer to define data's description so as to construct the application needed. This leads us to the basic difference between the two languages: HTML is display oriented while XML is a human readable representation of data.

A simple example and a quick view of an XML document is:

```
<address>
        <street>Rue de la Pacaterie</street>
        <batiment>499</batiment>
        <chambre>473</chambre>
        <city>Orsay</city>
        <region>Ile-de-France</region>
        <country>France</country>
        <code>91400</code>
</address>
```

The above document contains structured information about an address. The elements are obviously separated by their tags which describe their content.

Data exchange on the web before the appearance of XML was done through HTML. The operation which was usually followed included the use of an HTML page through which a distant server proposed its services. The client's browser displayed this page giving the freedom to the client to check the contained data. The occasion where a dialog with the server was established, the request was sent back to the server, the necessary processing was made and then the results were sent back to the client.

The performance of simplest actions from the client side on the data it received instead of sending other requests to the service provider would have as a result to expedite the whole process. HTML was non adaptable to this perspective and thus there is no possibility of comprehension of the data's structure from the client side. The only action is the display of the data. All what he was able to do was to display it, complying to guidelines encoded in the HTML response of the server.

Supposing the followings, we present an example situation:
A service provider that offers the telephone numbers of people living in Paris is in our possession. A client, named Bob, wishes to call his friend Jean. One possible solution for trying to find Jean is to type and search for results related to her. All people named Jean in Paris create an enormous list that Bob will be lead to refine his searching criteria, taking into consideration the surname of his friend and sending the request

back to the service provider. This process is continuous till the list has a reasonable size and is easy to be scanned. This approach causes erroneous use of the service because of the performance of evadible actions during the refining process and the transmission of redundant data over the network.

If the refinement process was fully responsibility of the client, the data that would be exchanged between the two sides will be decreased only to the initial one. The evident gain is that the same operation server will not be performed over and over again and the network overhead is efficiently reduced by sending only once the results. There would be also a decrease in the server's usage and consequently there would be an increase in the number of the clients that will be able to be served in the same time interval. So, XML and its characteristic of data structure give the appropriate approximation. XML information is received by the client and the assistance of a simple XML parser makes him able to interpret it.

### 1.4.2 Main Characteristics

A problem for data communication was the variety of machines and programming languages. The use of XML which is a standard for information representation overrides it as it allows inter platform and inter language data exchanges. Text form and unicode encoding is the guarantee of parsing and interpreting any document on any kind of machine while using any alphabet we want. The main matter that communicating applications need to be aware of is what data structure to expect. The description of the data structure is possible through two mechanisms : DTD (Document Type Definition) and most recently, XML Schema [67]. They allow to determine if an XML document is valid or not and most of the XML parsers support them. This type system is helpful as it supports us with the means of figuring out the desired document format.

XML's transformability could be characterized as the write once and display in infinite ways language. The notion behind this attribute is that you can display the same content in multiple ways. The use of XSL language (Extensible Stylesheet Language) fulfils the previous notion by controlling how elements from our document

are displayed. The different way of displaying the same content only through a simple re-writing of the XSL file is the positive effect of XSL.

The hierarchical structure of XML documents makes easy their representation as trees. These trees can be traversed in multiple ways which all rely on algorithms with strong theoretical background. The result of the operations that will be executed on the tree representation will be a clear reflection of the XML document that describes it.

## 1.5 Web Services

### 1.5.1 Web Services in general

The official definition of web services states that:

*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards*

There are some characteristics of the web services that need to be highlighted:

a) Web Services do not exist for certain on the World Wide Web. The Web Services can be hosted anywhere on the network while others need only a plain method invocation in the processes running on the same computer in order to be used.
b) The implementation and deployment platform details of web services are irrelevant to the program invoking the service. The invocation mechanism is responsible for the availability of the web service (network protocol, data encoding schemes etc.).

A web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols.

A web service is an abstract notion denoting a specific functionality. Different programming languages enable a web service to have multiple implementations and to be deployed on different platforms. If XML provides the data model, Web services provide the adequate abstraction level to describe the various actors in data management such as databases, wrappers or mediators and to manage the communications between them.

Many protocols and languages are involved in this architecture. HTTP and SMTP which are standard network protocols are only a part. XML enforced itself as the standard for data transfer and data representation. But other languages work as a complementary components to the Web Services Architecture. The WSDL (Web Services Description Language) is an XML language. The basic unit of communication in the Web Services Architecture (WSA) is the SOAP message and SOAP is an XML language too.

Web Services technologies can be divided into three parts:

- The wire stack
- The description stack
- The discovery stack

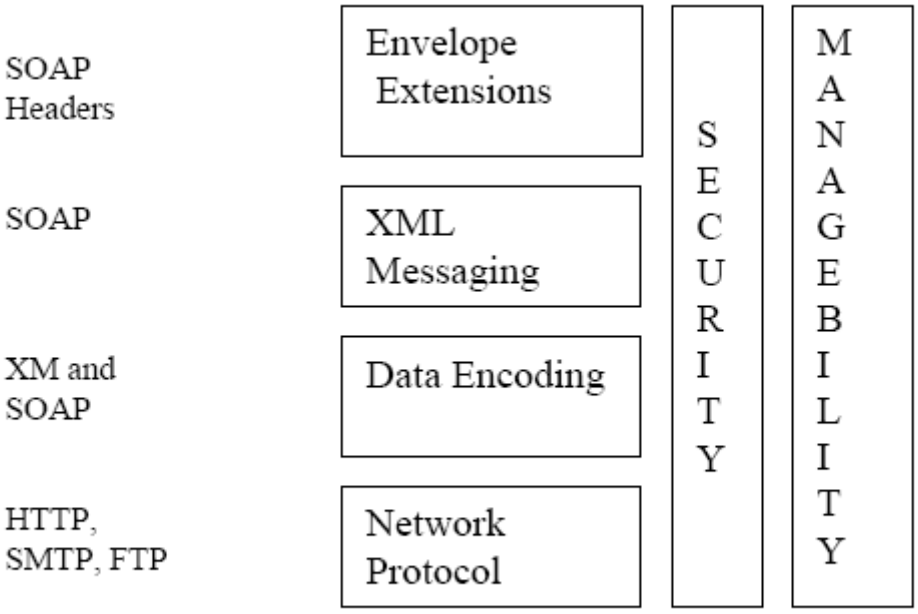


**Figure 2-1: Web service architecture – The wire stack**

The wire stack shows the technologies that participate in sending a message from the service requestor to the service provider. The network protocols at the base of the stack can be standard Internet wire-protocols like HTTP, HTTPS, SMTP, FTP. Web Services use XML for data encoding. At the top of this stack are the XML messaging layers which use SOAP in all the data encoding, interaction style and protocol binding variations. SOAP is as an XML wrapper in an envelope. On top of the SOAP enveloping mechanism is a mechanism for envelope extensions called SOAP headers which allow the association of orthogonal extensions such as digital signatures with XML digital signatures and XML cryptography.

The description stack involves mainly WSDL while the discovery stack involves UDDI (Universal Description Discovery and Integration) [66].

W3C defines WSDL as following:

*WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related*

*concrete endpoints are combined into abstract endpoints(services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.*

It describes several critical pieces of information a service client would need:

- the name of the service, including its URL
- the location the service can be accessed
- the methods available for invocation
- the input and output parameter types for each method.

### 1.5.2 Web Services Architecture

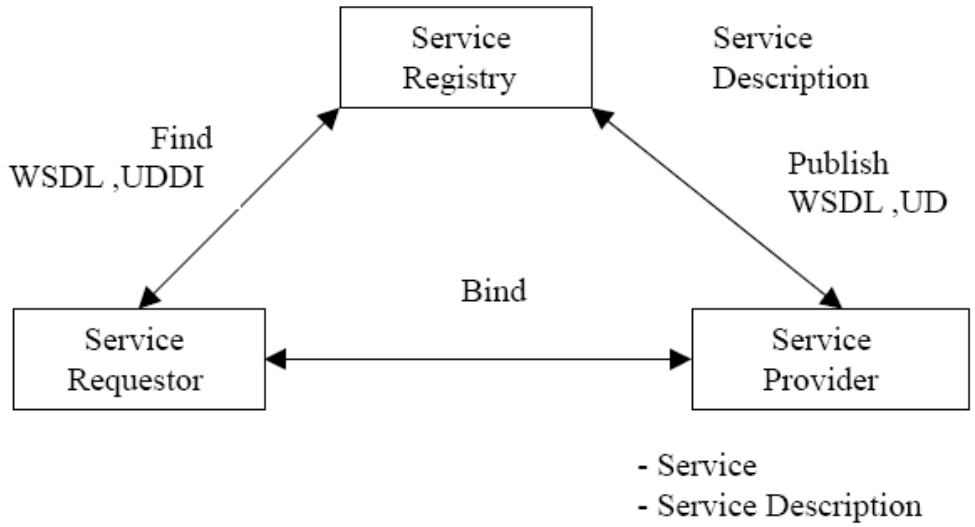


**Figure 2-2: Web services architecture**

Any service-oriented architecture includes three roles:

a) a service provider, who creates a service description, publishes it and receives web services invocation messages.

b) a service requestor who is responsible to find a service description published to one or more service registries and use it in order to bind to or invoke the web services.
c) the service registry is responsible for advertising Web service descriptions published by the service providers. Moreover, it allows the service requestors to use itself for finding the descriptions.

Three operations are involved in this architecture:

- the publish operation – act of service registration or service advertisement.
- the find operation – the service registry matches its collection of descriptions against a search criteria stated by the service requestor and returns a list of service descriptions to the service requestor.
- the bind operation which can be appeared in two forms: an on-the-fly generation of a client-side proxy based on the service description used to invoke the Web service or a very static model where the developer specifies the way the client invokes a Web service.

### 1.5.3 Simple Object Access Protocol (SOAP)

The official W3C definition of the SOAP Protocol is the following:

*SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.*

Basically, SOAP provides the following:

- a mechanism for defining the unit of communication. All the exchanged data in SOAP is packaged into SOAP messages. The SOAP envelope encloses all the information and the SOAP body can contain arbitrary XML.

- an error handling mechanism which allows the identification of the source and of the error nature through the SOAP Fault notion
- a mechanism for introducing extensions through SOAP headers
- a data representation mechanism where there is a serialization of data in some format and convention for representing abstract data structures such as programming languages data types
- a mechanism that binds SOAP and the underlying HTTP protocol
- a convention for representing Remote Procedure Calls (RPCs – the most common type of distributed computing interaction ) and responses as SOAP messages

A web service can be seen at a simplest level as an interface that enables the interaction between two applications on different endpoints. This interaction is modelled by operating remote procedure calls (RPCs). A remote procedure call is the invocation of one application's methods from distance. The data is exchanged in XML format and the transport protocol is HTTP.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<soapenv:Header/>
   <soapenv:Body>
      <a0:getTime xmlns:a0="Time">
         <locale> it </locale>
         <timezone> Europe/Rome </timezone>
      </a0:getTime>
   </soapenv:Body>
</soapenv:Envelope>
```

**Table 2-1: SOAP request for the example of acquiring of time**

The main advantages are: a)there is no need for designing our own protocol of communication because it is already provided by HTTP and b)the exchange of XML

format data over an HTTP connection causes little overhead on both participating endpoints.

An example is the following:
Let's say we want to access a service which, through the method getTime which takes two arguments: *locale* and *timezone,* provides us with a text response representing the date and time in a town. The answer is in the country's language.
The SOAP request and the SOAP answer are shown in the tables 2-1 and 2-2:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
   <ns1:getTimeResponse
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns1="Time">
      <ns1:getTimeReturn xsi:type="xsd:string">
         martedì, 22 giugno 2004 15:41:25 Ora estiva Europa centrale
      </ns1:getTimeReturn>
   </ns1:getTimeResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**Table 2-2: SOAP answer for the example of acquiring of time**

## 1.6 Related Work

This work has as a purpose to introduce a more specific way for the description of web services using technologies that extend the capabilities of the current standards. Furthermore, there is a need for establishing an efficient way of composing web services for the achievement of a required goal using semantic specifications. Many efforts have been done in this area trying to improve the techniques used for the integration of data. Research community has offered new ideas and features so far intending in the automation of the composition.

Automated composition is the process of automatically selecting, combining, integrating and executing Web Services to achieve a user's objective. Web service composition is performed by using information provided by the service provider, in order to compose and execute a series of services to achieve a goal. The automation of Web Service composition needs this information to be encoded in a computer-interpretable form. Many efforts have been done in this direction and the most representative are presented below.

### 1.6.1 Web Service Description

The methods of services description assist software systems in the use of them. UDDI and Web Services Description Language (WSDL) are current standards developed for this reason. The services are described using an XML schema which is defined in the UDDI specification. Additionally, they are categorized with the help of keywords. Thus, UDDI does not provide a semantic search. It provides a predefined categorization of web services through keywords. Moreover, WSDL describe web services as a set of endpoints or ports operating on messages and Simple Object Access Protocol (SOAP) is the protocol for exchanging those messages between the services.

Furthermore, there has been an introduction of some new industry standards that are useful for the representation of data, the control-flow, and the transactional properties among a collection of services. Business Process Modeling Language [10], XLANG[46], Web Services Flow Language (WSFL)[27], and Business Process

Execution Language for Web Services (BPEL4WS) [55] are paradigms of such implementations.

The semantic description of services includes Web Service Modeling Framework (WSMF) in which an ontology provides the terminology used [21]. DAML-S, the ancestor of OWL-S, specifies three main components for each service: service-profile, process-model and service-grounding. A service profile is the core element of a DAML-S\specification, and is consisted of semantic descriptions of service interfaces and functions. The process-model provides information about how the service works, and the service grounding describes how the service can be accessed. There have been more techniques that attempted to add semantics in existing services. Their approach has been based in mappings between WSDL, UDDI definitions and domain ontologies [46]. Moreover, the discipline that document engineering area imposes [23], as a set of analysis and design techniques that yield meaningful and reusable models of the information exchanges within and between the web services world directs the form and the principles that a distributed framework should obey.

### 1.6.2 Web Service Discovery and Composition

#### 1.6.2.1 Semantic Web Approaches

Semantic web service discovery related work includes [15] which describe how to evaluate a degree of similarity between a service template and an actual service instance by measuring the syntactic, operational, and semantic similarity.

While the majority of discovery approaches was based on web services interfaces, [13] investigated ways to search services according to the functionality requirements, and proposed Process Query Language (PQL) to search process models from a process ontology. [37], [38] presented a method to compose web services by applying logical inferencing techniques on pre-defined plan templates. Their technique focuses on the process-centric description of service as actions that are applicable in states. Some other efforts also emphasized the need for semantic representations of state, actions, and goals for composing services [48].

OWL-S has also been used to extend the WS-BPEL execution engine with an ontology to enable run-time discovery and binding of services. The semantic discovery service (SDS) [30], which performs semantic integration customizing dynamically the existing Web Service compositions, is such an example. The customization of constraints a user gives assists in the dynamic binding of user-customized services in the workflow. However, the need for additional semantic reasoning is demanding in some occasions. OWL-S is used for the description of services and the customization of user's constraints.

An augmentation of the SDS system was recently made [36], providing the system with rich explanations of successful or not fulfillment of an integrated composition. The new system integrates Inference Web [35] with the existing discovery system to add explanation, abstraction, and browsing capabilities with the use of registry infrastructure, offering transparency with the explanation of either success or failure in the composition plans.

There are many systems that deal with the relaxed service composition problem. Efficient reasoning in the preconditions and effects of a composition plan has been made in [8] where service specifications are augmented with invariants and there is demonstration of how such assertions can be exploited for determining service effects and compositionality while in the most of the other systems preconditions and effects are ignored.

Constantinescu et al. [16] present a system that sequences of services are constructed based on partial matches of input/output types determined at runtime. Web Service composer [45] uses an interactive composition method based on parameter types to put services together, and filters the potential matches using information from the compositional context. An improved version of this approach is an end-user interactive composition system, STEER [32], which helps the average user to find, select, compose and execute local, pervasive and remote Web Services to achieve common tasks. The STEER system combines services that adhere to different ontologies by utilizing semantic mappings and transformation functions that are also published as Web Services.

#### 1.6.2.2 P2P Based Approaches

There are also many works related to p2p based approach for Web services discovery and composition. In this section we review the main approaches.
Schlosser presents in [43] an ontology-based p2p system that focused on the discovery of Web services. They propose a graph topology which allows for very efficient broadcast and search while using a globally known ontology to determine the organization of peers in the graph topology, it is allowed an efficient concept-based search. Moreover, the Speed-R system introduced in [47] makes use of ontologies to organise Web service discovery registries and is proposed as the logical partitioning of web services based on domains addresses the scalability of the discovery process.

Schmidt [44] presents a peer to peer indexing system and associated p2p storage that supports large-scale, decentralized, real-time search capabilities is developed but its discovery is based also on keywords. A Web services integration platform based on semantic Web and p2p is the result of work in [26]. The platform can process a semi-automatic composition of web services based on specific domain ontology but it does not provide high accuracy of services matching at semantic level. Additionally, a framework that uses a p2p based orchestration model to support the composition of multi-enterprise Web services is the work in [11]. Self-Serv aims to enable the declarative composition of new services from existing ones, the multiattribute dynamic selection of services within a composition, and peer-to-peer orchestration of composite service executions. Self-Serv adopts the principle that every service, whether elementary or composite, should provide a programmatic interface based on SOAP and WSDL.

This thesis proposes a framework that sets an interaction-based approach at the communication of peers, through the use of service calls. Its main advantages are the existence of different roles for every peer and the dynamic way that impose in the composition process. The presence of AXML serves the notion of active data as a means of communication between peers, adding efficiently the periodical activation and materialization of service calls.

# Chapter 2

# Semantic Web Services

## 2.1 Semantic Web

### 2.1.1 Semantic Web in general

The continuous increase of the Web has turned the processes of finding and integrating the proper data to be difficult despite the significant help that search engines offer. The keyword based search is the methodology that is widely used nowadays. The techniques that follow the previous methodology provide a quite high recall because all pages that include the asked keyword are retrieved but a low semantic recall because related web sites that though do not contain the keyword about the subject are overlooked. So, the search is of low accuracy because few of the related pages has available the desired information. Moreover, synthesized queries are impossible to answer properly because such certain information is not widely available as such on the Web or if it is available, it may rely on different syntactic forms. The high recall of a general query burdens the retrieval of the most related pages. These inefficiencies of keyword based search are reasoned by the way that keywords are treated. Keywords are seemed as strings ad not as entities with specific meanings.

The Web confronting the limitations presented introduced two important changes. Firstly, main purpose was the extension of the current syntactic Web to a semantic level by enhancing provided Web information with formal descriptions of their meaning which have a logics base [49]. The transformation of this markup into machine processable would ease the discovery and integration of data rather than the keyword based search. Furthermore, the use of web service technology changed

Web's nature into dynamic. The semantic Web services technology is a combination of the current technologies by applying Semantic Web techniques to Web services.

Semantic Web has been introduced in order to provide solutions in the instant constraints of the Web by augmenting web information with a machine processable association of its meaning. The support of web information with knowledge representation and reasoning techniques has been examined from the mid '90s both in the work on Simple HTML Ontology Extensions [29] and in Ontobroker [22].

The Semantic Web term was clearly clarified in 2001 when its definition was:

*"The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."* [12]

The need to realize the Semantic Web has raised the development of several approaches. The alternatives that presented in these approaches related with the type of data sources used for semantic description and the ways of enriching the semantic data. A first approach intend in connecting the meaning of textual data sources with rich semantic descriptions. Unfortunately, a formal representation of information being in textual documents is an insufficient effort because there was no way of recoding content in a general way.

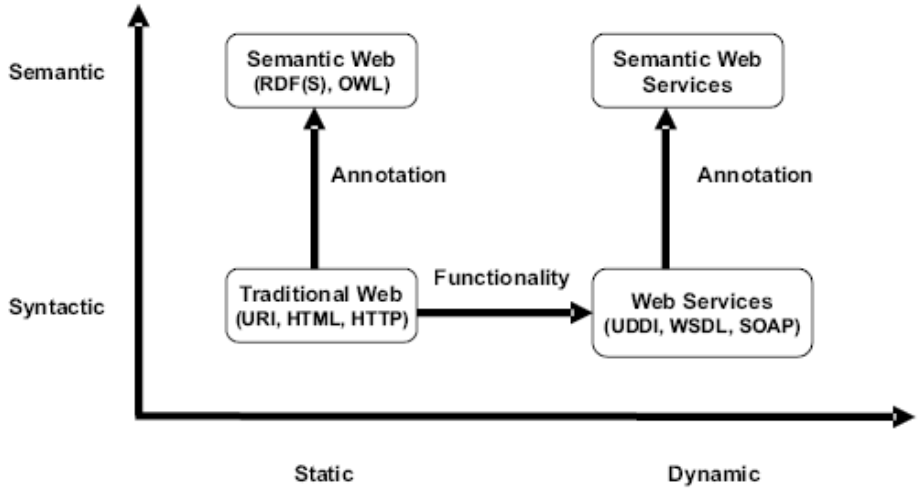


**Figure 3-3: World wide web correlations**

A different approach presented with the use of semantic Web languages with low semantic expressiveness such as XML, based on the decrease of the complexity level that semantic annotations have. The evolution of the Semantic Web has led many of the first applications using its technology to depend on little semantic information and use semantic languages with low expressivity such as XML and RDF(S).

Currently, Semantic Web research is focused in the ways of integrating and exchanging data, a field where the semantic annotations are the major supporting technique. Semantic descriptions, often called as metadata, provide a specific definition of information available. Semantic metadata has two main characteristics:

a) information are described by metadata having a domain vocabulary as a base. The meaning of this vocabulary corresponds to a formal domain model, named ontology. b) metadata is expressed in a representation language that can be parsed and interpreted by machines.

### 2.1.2 Ontologies

The adoption of the term *ontology* from Semantic Web was made to describe formal domain models. There were many ontology definitions during the years. Gruber in 1993 defines ontology as "an explicit specification of a conceptualization" while Borst in 1997 [14] claims that it is a "**formal** specification of a **shared** conceptualization". In 1998, a combination of the previous definitions was made by Studer at al [22]:

*"An ontology is a formal, explicit specification of a shared conceptualization."*

In the computer science field, ontologies are formal models which represent machine processable knowledge thus enabling an enriched communication between humans and computer programs or two computer programs.

Studer defined the main ontology attributes but there are in advance significant variations that these characteristics generate. The captured conceptualization which is implied by an ontology may differ in terms of the levels of detail and generality of the captured conceptualization. A brief discussion about these characteristics is followed in order to have a classification of the existing ontologies.

The categories that ontologies are divided taking into consideration their level of generality are:

- *Foundational or top-level ontologies* are conceptualizations that contain domain specifications and problem independent concepts and relations such as space, time, matter based on formal principles derived from philosophy, and mathematics.
- Examples of foundational ontologies are *DOLCE, the Suggested Upper Merged Ontology (SUMO) and OpenCyc5 and the Basic Formal Ontology (BFO).*
- *Generic ontologies* contain generic knowledge relative to a certain domain such as medicine, mathematics or Web services. The specification of these domain concepts is made in terms of top-level concepts, resulting in the inheritance of the general theories behind the top-level concepts. Main example of such kind of ontology is the OWL-S ontology which specifies a set of concepts that allow describing Web services.
- *Domain ontologies* are specific to a particular domain and their percent of reusability is quite low.

The level of detail is the second attribute that characterizes ontologies. Based on modelling primitives, ontologies are normally distinguished from vocabularies, taxonomies and thesaurus in terms of expressivity. Taxonomies are more useful than vocabularies because they capture hierarchical relations among terms while the other only provide a list of them. A thesaurus in turn, allows defining properties between concepts. This kind of ontologies is referred as *lightweight* ontologies while *heavyweight* ontologies involving axioms and constraints as well can model a domain more precisely [17].

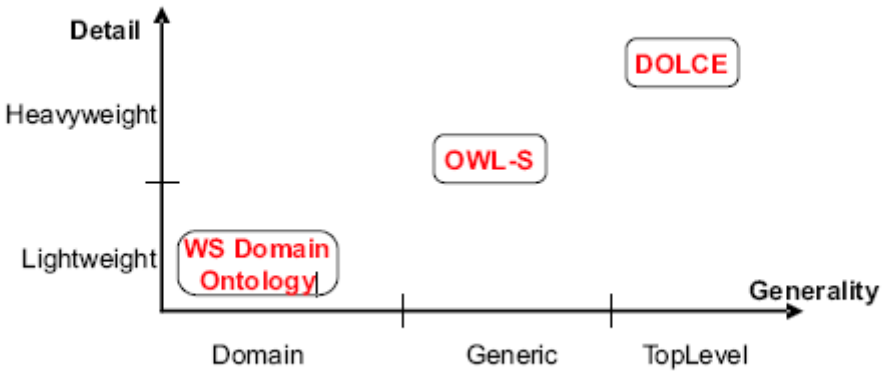


**Figure 3-4: Example ontologies classified according to the two dimensions**

### 2.1.3 Ontology languages

There are several formal languages that specify ontologies. RDF(S) and OWL are the most popular ones.

#### 2.1.3.1 Resource Description Framework (RDF)

RDF and RDF Schema represented the first attempt to establish a web based ontology language. RDF is a data model which allows the description of web resources. RDF Schema is based on RDF and allows the definition of basic ontology elements such as classes and their hierarchy, properties with their domain, range and hierarchy [33]. Consequently, RDF(S) is suitable for the expression of lightweight ontologies. Several tools were developed for RDF(S) while a large number of Semantic Web applications used it before a more expressive ontology language was developed.

#### 2.1.3.2 Ontology Web Language (OWL)

The Web Ontology Language (OWL) is a more expressive ontology language than RDF(S). OWL derived from the DAML+OIL language which was a merge of two language proposals that had as a purpose to overcome the expressivity limitations

of RDF(S): DAML-ONT9 and OIL10 [25]. OWL augments the expressivity of RDF(S) as it provides methods for the description of relations between classes such as union, intersection and value restrictions on properties such as cardinality, universal and existential quantifiers and property characteristics such as transitivity, symmetry. OWL provides the necessary constructs to encode ontological knowledge using the XML based syntax.

## 2.2 Semantic Web Services

### 2.2.1 OWL-S

The Web Services technology establishes a common and easily integrated framework in distributed environments, adding one more way in the handling of the large amount of data in the Internet. Semantic Web Services (SWSs), emanating from the evolve of Semantic Web section and the need to offer enriched descriptions for the data and the services that are available, attempts with the appropriate feed to resolve challenges associated with automated discovery, dynamic composition and other issues related with the management and the use of service-based systems. Particularly, Ontology Web Language for Services (OWL-S) [58] has been developed in order to provide the necessary additional parts to OWL [19], the Semantic Web language standardization at the W3C, with enhanced semantic service descriptions.

OWL-S, which is a descendant of DAML [56], is an attempt to establish a more effective solution for the further development of web service-related capabilities. In parallel, the adoption of existing technologies which provide semantic expressiveness and are already widely used and understood is of main purpose. Thus, the mechanisms which have been constructed for OWL-S serve the notion of being used with the Web services standards which have dominant role such as WSDL. Our main aim is to describe the way of use Semantic Web services, using OWL-S in conjunction with WSDL and related standards, and to present the gains of having richer semantics in the web services technology.

Evolving frameworks such as OWL-S, WSMO [65], IRS [39] which support semantic Web service descriptions share a common attribute. They all use two kinds

of ontologies to provide a service description. The first one is the generic Web service ontology, such as OWL-S, in which generic Web service concepts are specified such as input, output. While the generic one consists of the base of a semantic service description, the *domain ontology* enhances knowledge in the domain of the Web service, such as types of service parameters and functionalities that supplements the generic.

DAML-S and WSMO are the major alternatives generic ontologies for Web service descriptions. The first one allows the description of many parts of a Web service. Its translation from DAML to OWL resulted in the creation of OWL-S. The other one is WSMO (Web Service Modelling Ontology) which despite its significant levels of overlap with OWL-S, depends on different ideas, adding new features to OWL-S. In this thesis, OWL-S will be our major ontology.

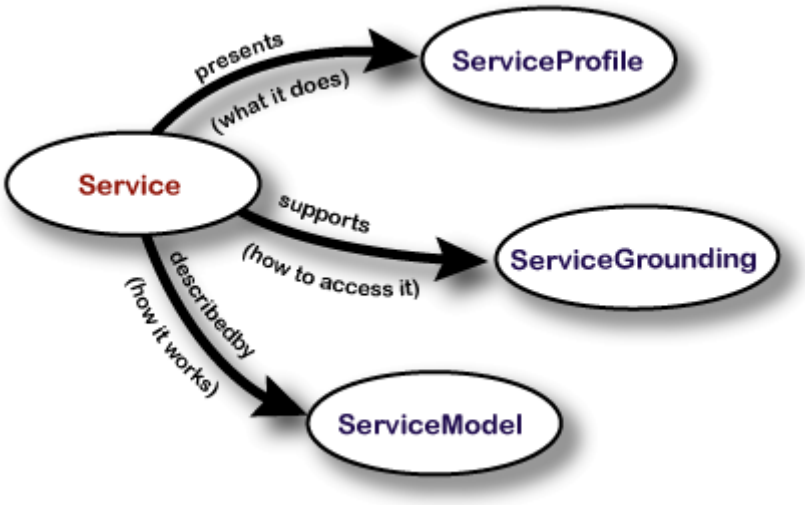


**Figure 3-5: The OWL-S Service Ontology**

The OWL-S ontology is divided into four parts detailing about what a service does, how the service works and how the service is implemented. These parts are respectively represented by the *profile*, *process* and *grounding* ontologies. *Service* is the last ontology and contains the Service concept which connects a ServiceProfile, a ServiceModel and a ServiceGrounding concept. Describing in terms of concepts, the

Service presents a ServiceProfile, is described by a ServiceModel and supports a ServiceGrounding.

The specification of the profile ontology consists of the functionality the service provides, the semantic type of the inputs and outputs, the details of the service provider and several service parameters. The discovery process is based on the above descriptive characteristics

***The Profile ontology***

*Profile* ontology has as its main concept a subclass of *ServiceProfile*.
Each *Profile* instance describes a certain process indicated by the *hasProc* relation and its functional characteristics which are Inputs, Outputs, Preconditions, Effects together with their type.

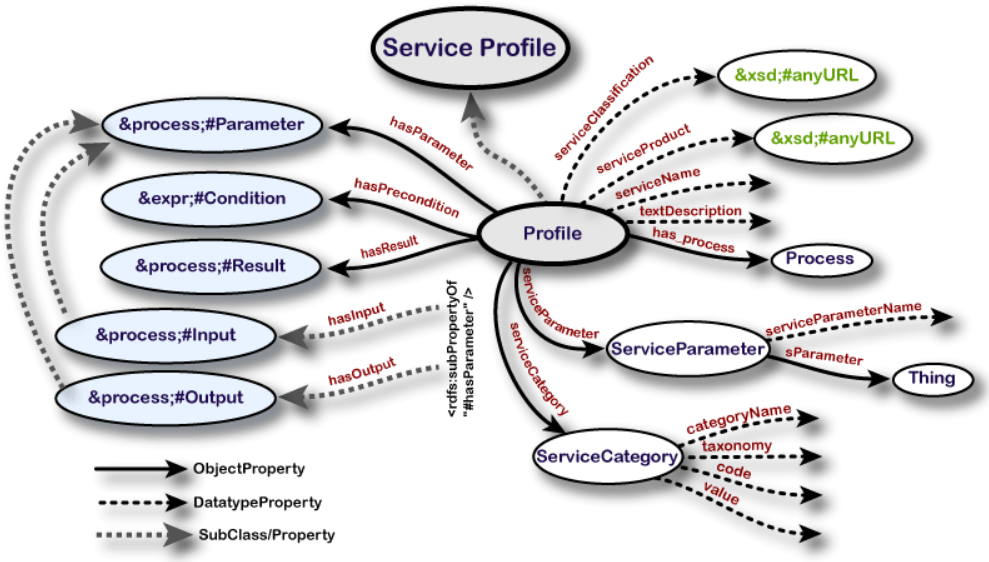


**Figure 3-6: Service profile representation**

***The Process ontology***

Services can be executed in a predefined order in order to perform a synthesized service. An example which will be presented further afterwards, is the purchasing process of a book at Amazon.com. This process contains the use of a browsing service, assisting in the selection of the books and a paying service. OWL-S enables the description of models that need more than one processes to be fulfilled.

The benefits from this attribute are important as the discovery of the appropriate product is easy, the monitoring of the processes' execution is feasible and the composition of web services becomes possible.

A *ServiceModel* concept describes the workflow of the services inside the main one and it is also specified as a *ProcessModel* concept in the Process ontology. A *ProcessModel* has a single *Process* which can be atomic, simple or composite. Composite process is a result of the composition of other atomic processes with the use of several control constructs such as Sequence, Split, Split+Join, Any-order, Choice, If-Then-Else and Iterate. Each Process has a set of Inputs, Outputs, Preconditions and Effects.

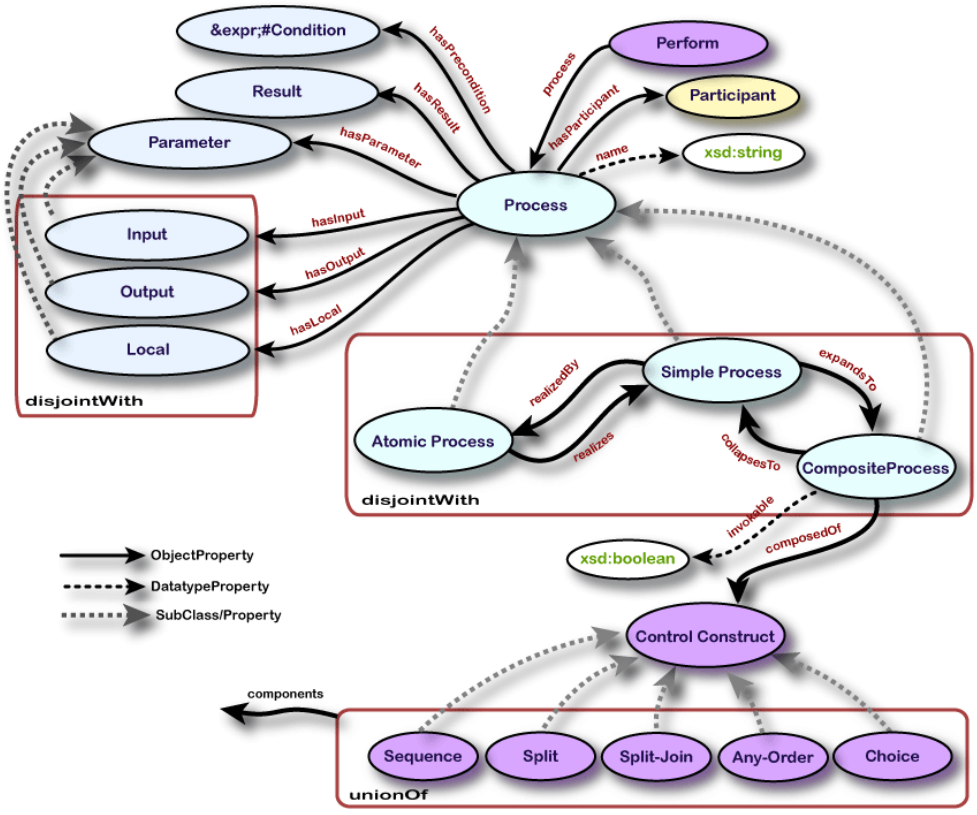


**Figure 3-7: The Service process ontology**

***The Grounding ontology***

The detailed description of the implementation of a service is provided by the grounding ontology. This description is the substantial actualization of the service's concept which is defined by the Profile and Process ontologies and is consisted of message exchange formats and network protocols. There are three specifications with which the grounding is transformed into a WSDL description.

a) Each *AtomicProcess* represents one WSDL operation.
b) Consequently to a), each input and output of an *AtomicProcess* represents a message-part in the input and output message of the WSDL operation.
c) Each type of all parts of the WSDL message can be defined as an OWL-S parameter

The Grounding ontology specifies in detail the *ServiceGrounding* as a *WSDLGrounding* which contains a set of *WsdlAtomicProcessGrounding* elements. Each grounding corresponds to one of the atomic processes specified in the *ProcessModel*.

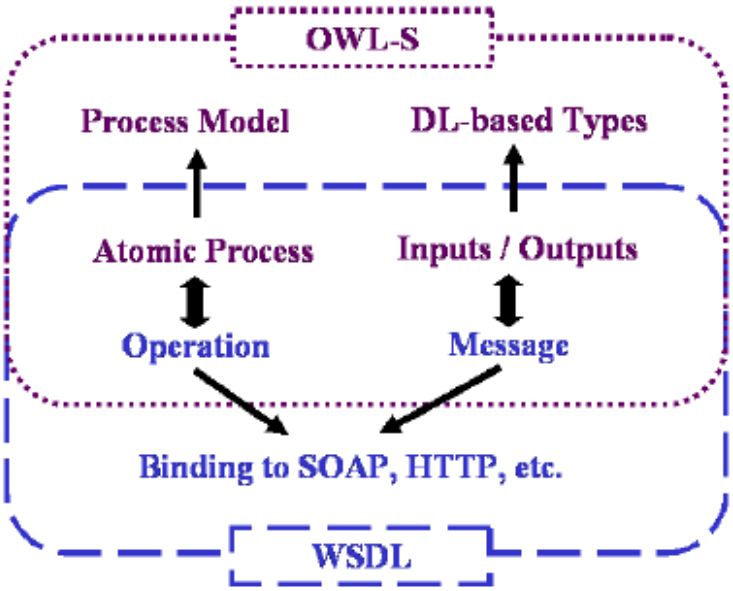


**Figure 3-8: Owl-s to wsdl mapping**

### 2.2.2 A Motivating Example

Amazon.com, having established its position in the web market, offers a web service with which client programs are able to make all the possible actions for the buying process: browsing Amazon's databases, finding books, adding them in the shopping basket. The main Amazon Web site can access the available service, using also a browser to fulfil the purchases. These processes enable the web service clients to ask for a variety of semistructured keyword searches on the shop's database, as they have available the provided WSDL specification of the service. Some of the parameters that clients search may contain are the author of a book, the manufacturer of a product. Additional information that is also available is the customer reviews to seller profiles [52].

Amazon's Web service contains a WSDL specification describing the actions that can be carried out, having as helping material tutorials and source code for sample clients. The necessity of them is the assistance with which programmers are provided for the appropriate utilization of the WSDL interface. The human effort is necessary because the WSDL specification language do not include ways to represent the semantics of the defined operations and the related parts of the messages. Amazon's WSDL operations have as type of all their inputs and outputs the string one, a fact that shows the limitation of the language in semantic issues.

The resolving of this lack of expressiveness is major objective of Semantic Web services and OWL-S. The specification of the functionality of an action, with the use of preconditions and effects, and the provision of semantic types for each of the inputs and outputs of the service is one of the contributions of OWL-S. The theory in which OWL-S is based is that specific URIs on the Semantic Web provide the definitions of these semantic concepts. The results of this assumption are the creation of means for the service and client programs to share terms between them and the enabling of finding all the relevant concepts, described in the OWL semantic language, for the clients.

A client program having both an OWL-S description of the service and the WSDL description can discriminate the kind of operation. The use of OWL for semantic typing of the elements makes the Amazon client [40] to identify the desired

inputs for the given search, to transform those elements to the appropriate string form, and to interpret the elements of a returned message.

There is a significant lack of semantic definition also in the WSDL specification of the outputs of each call to the service. The data structure of all Amazon's defined search operations return results is the same, having the name *Details*, without taking into consideration the kind of the asked product information.

The analysis of the elements that a response contains helps clients to deduce the product types. Following our example, a field *Authors* and a field *Directors* which are used to describe books and movies respectively are parts of the *Details* section and, thus, it is responsibility of the client to understand whether the values in these fields refer to a book or a movie.

Clear identification of the product type would be also deficient as there is no established way to enable such interpretations in WSDL. The provision of no semantic description of inputs and outputs in WSDL infers that the dynamic discovery and invocation of a service is impossible for the existing clients. Semantic Web services have as main purpose to transform the services operations into machine-processable in order the clients to be able to find and properly materialize services without manual efforts.

Semantic Web service clients will be able to interact with any such service. This is possible as long as there are descriptions of services' WSDL operations in a way that facilitate the notion and the representation of data in the Semantic Web.

### 2.2.3 Semantic Web Services Processes

#### 2.2.3.1 Introduction

OWL-S is an OWL ontology with three interrelated sub-ontologies, known as the profile, process model, and grounding as we have discussed before. In brief,

a) the profile is used to express "what a service does", for purposes of advertising, constructing service requests, and matchmaking
b) the process model describes "how it works", to enable invocation, enactment, composition, monitoring and recovery

c) the grounding maps the constructs of the process model onto detailed specifications of message formats, protocols, and so forth (normally expressed in WSDL).

.

WSDL 1.1 allows for the specification of *operations* as the basic building blocks of Web services. The structure related with input/output message syntax and patterns are specified is provided by the set of the operations. The construct that OWL-S provides is equivalent but somewhat more abstract. This construct is named *atomic process*, and its characteristics are its inputs, outputs, preconditions, and effects (IOPEs).

The types of inputs and outputs of an atomic process are given from the typing system of OWL, in which the use of concepts defined and shared as part of the Semantic Web is allowed.

The following sample code presents a generalized OWL-S declaration of an atomic process with its input and output specifications. In this example, AtomicProcess, input, output, and parameter-Type belong to the OWL-S process model namespace. Human, BookTitle, and ISBN are supposed to be classes defined in appropriate domain ontologies having other namespaces.

```
<AtomicProcess ID="AuthorSearch">
  <hasInput>
    <Input ID="Author">
      <parameterType resource="#Human">
    </Input>
  </hasInput>
  <hasInput>
    <Input ID="Title">
      <parameterType resource="#BookTitle">
    </Input>
  </hasInput>
  <hasOutput>
    <Output ID="BookID">
      <parameterType resource="#ISBN">
    </Output>
  </hasOutput>
```

**Figure 3-9: An example of an OWL-S atomic process**

The grounding for this atomic process is a plain association to a particular WSDL operation and associates each input-output element with a particular WSDL message part element. Additionally, the grounding can describe in a detailed way an XSLT script which will be useful in the transformation of each input expressed in OWL – e.g. an instance of the relevant class – to the exact syntactic form specified by WSDL, and conversely for outputs.

Semantic Web service description has a significant issue to examine which is the specification of conditions and constraints, including the preconditions and effects of a process or service. Preconditions are logical rules that need to be validated by a service requestor before the execution of the service. Effects are logical rules that assert what the results will be when there is a successful execution of the service. OWL-S effects are the side-effects of the execution of the service. There are some services that have no side effects as they provide only informative data on the contrary to some others that their execution causes the purchase of a good or the charging of a credit card.

Preconditions and effects of a process can be specified with the use of a more expressive language than OWL, such as RuleML [61], DRS [34], or the OWL Rules Language [22]. For example, one of these languages could be used to express a precondition for a bookselling service, stating that one must have a valid account and a valid credit card in order to make a purchase.

The following sections present in a detailed way OWL-S contributions in issues such as service enactment, discovery, and composition.

#### 2.2.3.2 Interaction using OWL-S

The process of interaction relies on the fact where there is an application of a declarative description of a service by a client in order to make a request and then interpret the response. The OWL-S process model published by the service along with the WSDL specifications to which it is grounded is the description being interpreted.

The finding of the appropriate inputs required by the selected service is the first step at the process of enactment in order to find the available data which will cause the successful invocation of the service by the client. The second step relies on the mapping of these input values to the corresponding elements of a WSDL message

pattern through the service grounding, proceeding in a message which establishes the communication with the service. If there is an output message from the service, it will be handled following the reverse way from what it have been described so far.

The grounding assists the client in the transformation of the received WSDL output message into an OWL-S representation of the content of that message which then can be interpreted by the client's reasoning engine.

The implementation of the theory described with the web service of Amazon requires a description of that service in OWL-S where the inputs and outputs of the service are fully represented. The creation of this description can be merely automated with the development of an initial OWL-S description of Amazon's Web service using tools such as OWL-S editor [45] that modify WSDL specifications into a partial OWL-S specification.

The inadequacy of the information provided in the WSDL description leads us to set up a complete OWL-S process model, where there are two additional steps for the generation of the description:

- giving additional semantic descriptions of each input parameter to the generated process model, and supplying any XSLT-based data transformations needed to produce the related grounded message parameter strings
- building a composite process model that connects the different operations provided by the Web service into message patterns that have semantic meaning, for example login before search or before add-to-shopping-cart.

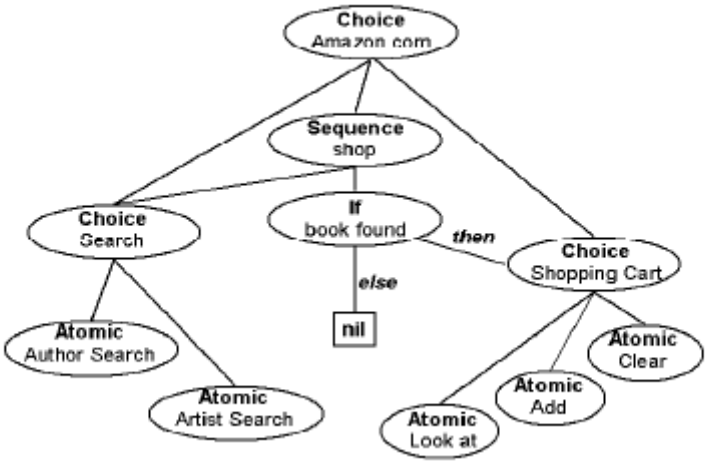


**Figure 3-10: Amazon 's web service process model**

The final process model is depicted in Fig. 10, which shows how the various operations of the service are related with the resulting OWL-S process model.

Three types of actions can be performed by the client:

a) *search* the Amazon's data bases using author search, artist search or other types of searches
b) b)*view or modify the shopping cart* by adding new items, clearing it, or looking at its contents;
c) perform the composite *shopping* process that is a combination of the previous two processed in the order they presented.

The Amazon Web service which is written in WSDL only describes the operations representing the leaves of this graph. The process descriptions described in OWL-S specify semantically the types of the data required as input, and returned as output. For example, the input of the author search should be an instance of class *Human* that stands in a particular relationship to the book being sought (*written-by).* The use of OWL classes and properties as constraints on the instances that must be identified for particular input values is of major importance for the inference process. It supports the formulation of service requests without demanding the manual writing of specific source code for each probable type of service request.

In a possible scenario where the client's purpose is to search and find the price of a specific book, it can identify the items such as author that are relevant and build the necessary elements of the search request. Then, the selection of the appropriate service to be invoked takes place as a result of the association of the information described as part of the output of the service and of this described with database elements about books. As a consequence of this fact, the client is enabled to facilitate the reverse process as well, as it is aware of what the returned data will be without having to assume it from the instantiation of the *Details* data structure.

The OWL-S grounding maps the inputs and outputs of the operations in the Amazon WSDL specification with the corresponding concepts that describe the inputs and outputs of the processes in OWL-S. The result and the main difference-advantage is the insertion of logics and ontologies that OWL provides in the reasoning process

of the Web service, something that does not violate the rules that web services should serve as the actual invocation is still consistent with Amazon's requirements.
The DAML-S Virtual Machine [40] is a way that fulfils successfully the interaction with the Amazon web service

#### 2.2.3.3 Discovery Process

Discovery is the process of finding Web services with a certain capability. In brief, discovery's requirements are the advertisement of the capabilities of the Web services with a registry, and the request of services by querying the registry for Web services with specific capabilities.

The registry plays an important role as it is responsible both for the storing of the advertisements of capabilities and the matching process between the request and the advertisements.

In this section, it is described how OWL-S may be used to express and match capabilities. Finally, it will be presented how OWL-S can be used to add capability matching to UDDI, the standard discovery registry for Web Services.

##### *2.2.3.3.1. Efficiency of Services*

The functionalities provided by Web services are associated with the capabilities that these services have. The representation of functionalities has two ways. The first one supports an extensive ontology of functions. Each class in the ontology is related with a class of homogeneous functionalities. A simple example of this kind of ontology in which a hierarchy of e-services is specified is shown below. The use of such an ontology results in the defining of Web services such as Amazon as instances of classes that represent their capabilities.

```
<owl:Class rdf:ID="e_Service">

<owl:Class rdf:ID="Information_Service">
<rdfs:subClassOf rdf:resource="e_Service"/>
</owl:Class>

<owl:Class rdf:ID="SellingService">
<rdfs:subClassOf rdf:resource="e_Service"/>
</owl:Class>

<owl:Class rdf:ID="BookSelling">
<rdfs:subClassOf rdf:resource="SellingService"/>
</owl:Class>

<owl:Class rdf:ID="AirlineTicketing">
<rdfs:subClassOf rdf:resource="SellingService"/>
</owl:Class>
```

**Table 3-3: Example of an ontology which specifies a taxonomy of e-services**

The second way of representing capabilities is the provision of a generic description of function where there is clearly formed the state transformations that the service generates. In this occasion, Amazon may own a service that needs as inputs a book title, author, address and a valid credit card number, and sends as a response a state transition where the book is delivered to address, the credit card will be charged, and the book will change ownership.

Although there are differences, both representative ways use ontologies to indicate the relation between the actual operation of a Web service and the general description of the environment in which the Web service operates.

The analysis of the tasks required leads to the selection of the right way of representation. The use of an explicit ontology of capabilities fulfils the discovery process because the matching process is reduced to subsumption between the capabilities in the ontology. On the other hand, taking into consideration all possible capabilities may be a rather difficult effort. Suppose that we desire to represent translation services from a source language A to a target language B. Having as an assumption that there are n possible languages, the possible types of translation services are $n^2$. The existence of a services' taxonomy might ease the solution of the

problem as each pair of languages that could be translated may be connected with different classes. Moreover, this taxonomy may just represent translation services as one general category, having explicit properties that allow particular services to describe the languages that they can translate from and translate to. This latter approach is in accordance with describing the capability in terms of a state transformation. The translators are distinguished by describing how they produce different kinds of results.

It is significant to point that the description of the types of the inputs and outputs of such a service is deficient to differentiate capabilities. This is obvious in the following example where a service that takes a geographic region as input and produces the names of different wines as output is used. The inputs and outputs can be used by two very different services: one that reports which wines are *produced* in a region and the other that reports the wines that are *sold* in a region.

Both representations of the capabilities of Web services are supported by OWL-S. The *Service Profile* module of OWL-S provides a high level description of services as a transformation from one state to another. A Service Profile provides a specific view of the Web service as a process. This view needs inputs, in prior to some valid precondition, and has outputs which should have some effects becoming true.

Furthermore, OWL-S has a schema with which Service Profiles can be divided in subclasses in order to describe a more accurate class of capabilities such as translation services, or wine selling services. Being precise, there are two kinds of information that are provided by a Service Profile:

a) a functional description of the Web service where it contains the transformation that the Web service causes,
b) a set of non-functional properties that include constraints on the service provided.

The first type is a description of the alteration of data as a result of what is produced for outputs for a given set of inputs and the state alteration as a result of the generation of the effects for a given set of satisfied preconditions. The second type of non-functional properties is related with either the quality of service provided by the Web service, or its security requirements [38].

OWL-S gives a full description of the service that it describes as it may support both an extensional and functional view of Web services. The main advantage is the use of ontologies of services, independently of their location, offering in parallel representation of the capabilities of a Web service. Additionally, the Web service produces transformation that can be used to describe it qualitatively or assists in the generation of descriptions of its resulting effects.

#### *2.2.3.3.2. Services Matching*

Capability matching is the comparison process between the capabilities provided by any of the advertised services with these asked by the requester. The goal of this process is the finding of the advertiser that yields the suitable results. The perfect match of the two categories of capabilities is difficult to happen, something that is obvious from the following example: the requested service may be for stock quote information, and the matching engine has to decide whether a service that provides financial news can fulfil the requested task. The work of the matchmaker is to check each of the capability advertisements and decide how possible it is for each one to perform the function required through the query.

There are numerous algorithms for the capability matching process that have been recommended for OWL-S. The comparison is made between the service descriptions in the Service Profiles and the ontologies that are published, exploiting one of the two views of the capabilities described previously. [28] and [30] propose such algorithms, having as a base how available ontologies exist that their functionalities enable the expression of the required capabilities. The fact of how similar capabilities are between the requester and the advertisements affect the number of the matching degree. [20] and [9] presented different matching algorithms, taking into consideration for the description of capabilities the state transformations generated by the Web service. The task of these matchmakers is accomplished with two matching processes, the comparison of outputs and the respective comparison of inputs. If the output required by the requester is of the same category or subsumed by the advertisement, then the check of inputs follows. If the inputs specified in the request are subsumed by the input types that the service accepts then the service is capable of accomplishing the requester's asked task.

#### 2.2.3.3.3. How OWL-S is related with UDDI

The UDDI (Universal Description Discovery and Integration) protocol is one of the major building blocks required for successful Web services. UDDI creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet. UDDI also allows operational registries to be maintained for different purposes in different contexts.

UDDI offers to businesses the advantage to register their presence on the Web. This can be done with the specification of their points of contact in terms of the ports used by the service to process requests as well as in terms of the physical contacts with people able to answer questions about the service. Furthermore, UDDI provides a language to specify a limitless set of features of services that serve the accomplishment of service location and selection as well as service invocation.

UDDI has become the standard repository of Web services. Although it has an important role, UDDI has a weak discovery mechanism. The main problem is the lack to represent capabilities such as the OWL-S Service Profile, thus, it does not support search based on capabilities of the services. Consequently, UDDI supports only the location of information about the Web service but it is unable to provide the location of a Web service, associating it with the concepts of the operations described.

OWL-S and UDDI are complementary. UDDI is an industry standard which provides a world wide distributed registry while OWL-S provides the necessary information for capability matching. A combination of the functionalities between OWL-S and UDDI is integrated with the OWL-S/UDDI matchmaker [34]. This integration is based on the mapping of OWL-S Service Profiles into UDDI Web service representations [41].

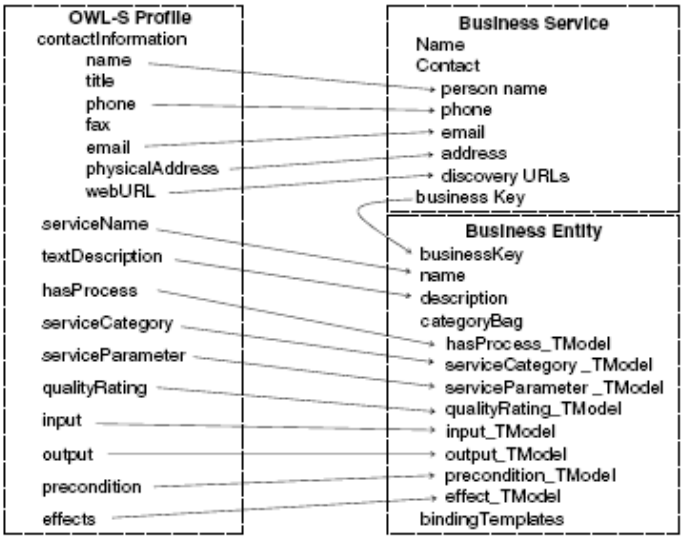


**Figure 3-11: OWL-S-UDDI representation**

The mapping function defines a set of specialized UDDI TModels - which is a set of properties that can be corresponded to a Web service specification - that facilitates the storing of OWL-S data which is impossible to be represented in the standard UDDI Web Service representation.

The integrated OWL-S/UDDI matchmaker supports the whole size of functionalities provided by UDDI using the same API, so that any client can establish a communication in order to retrieve the desired information about available Web services. In addition, the advantage of OWL-S in the representation of capabilities is provided by OWL-S/UDDI succeeding in the performance of the capability matching process. The result is a UDDI in which it is possible to search, and find, Web services by their capabilities.

# Chapter 3

# The AXML Framework

## 3.1 AXML in general

In this section, the efficacy of combining XML and Web services will be illustrated by describing AXML. In brief, parts of the data are given explicitly in AXML, while other parts comprise of calls to Web services that derive more data. AXML is based on P2P architecture [3]. Each AXML peer [70], [18] may have two action roles. Firstly, it may act as a client, by activating Web service calls embedded in its documents, and secondly may act as a server, by providing Web services that affect queries or updates over its repository of documents. The way the calls are activated vary as they can be finely controlled to occur periodically, or in reaction to some particular such as in the style of database triggers, or in a "lazy" way [5], [4] whenever they may produce update data to the answer of a query.

### 3.1.1 Logical Motivations for Active Answers

The return of active answers to queries origins from a combination of logical factors for providing more functionality. This set of factors creates substantially the list of logical motivations for the usability of active answers that are going to be analyzed as follows:

***Data reusability***

One of the first groups of motives is the ability we want to provide the client about reusing autonomously a part of the data that were sent in the query answer. The

notion behind this action is the provision of the enriched knowledge about the query's origin or more means to acquire them without having to prompt the query again.
Dynamicity is an attribute that represents the main reason for having active answers. The possibility that parts of the information given may be influenced and change over time shows the need for the active handling of data.

***Summarization***

The necessity that leads us to the use of partial answers is that the full answer may have large size, a fact that difficults the reading of it. Summarization of the data is the solution for the better use of the whole set and the natural way to achieve it is to make it intensional. Intensional or active data may be considered as a context which offers different views of a query answer and possible methods of using successfully them. AXML can easily actualize it as the service calls provided inside the AXML document can be used as navigational links to the full information. An example for this fact may be the answer of the query "XML" to Google which is very large and there is no necessity to obtain it all. A possible action could include the server to decide not to send them all but send only a small portion and a link to acquire more. This example describes in a very simplistic way the form of active data and the one that follows specifies it in more detail. A search engine, where three different data sources are combined for the query answering process, receives a query while one of the sources is not available. Using AXML, the search engine will send the concrete data of the two sources it knows of, and with a service call to the last source will allow the access to the data of the third source. The implication of this action is one more way of successful creation of intensional data where in the present example may be obtained from when the source will be back.

***Partial computations***

Usually, the result of a query is an answer with a full computed set of data. The term of active data is also relevant for data that computations are still needed to be made in order to access the full answer. For instance, a secret phone number may be sent encrypted, obliging the user to decrypt the answer in order to see it. Another paradigm can be the results lists of products descriptions. Each description includes a summary and the means to buy the product e.g. using a Web service. The results lists

as part of the whole answer may be considered as active data as it can be obtained with some extra computation [42], [7] by the data receiver making the service call related to it.

***Data enrichment***

This is the process where external tools or techniques are used in order to add value to an answer. There are different points where the process of enrichment may take place as it may affect the whole query result or only at specific parts of it e.g., an XML element. The forms that this process follows may include links with the user context, the use of the ontology theory for extraction of related concepts, transformations from one type to another, or translating the answer to a preferred language which offers better expressiveness. An active answer in all these different occasions may play the role of an agent who manages the answer and provides the enhanced form of it to the user. This motive is very important as it sets the general frame of data enrichment, giving us the opportunity to delve in these techniques and suggest some ways of enhancing not only the data itself but also and its semantics.

***Context awareness***

Information in all its forms, either in extensional or in intensional form, may rely on some context. This fact becomes tangible in the following example where a possible user receives a list of restaurants. The list's size and content may vary because of the preferences the user has, such as the taste he prefers for specific kinds or food, the way of cooking, or the restrictions that his religion imposes on his diet.

### 3.1.2 Physical Motivations for Active Answers

In the previous section, the base of the motivations presented for using active answers was the provision of more functionality. The physical motivations which will be presented in this section clarify the usefulness of active answers at the evaluation of queries in a distributed environment.

Data and intensional information in them are terms that exist in the world of web. The example of PHP or JSP pages, where HTML or XML documents have inside them source code shows one form of intensional information. This source code

is not part of the data that are exchanged through the web, but typically, is transformed into "plain" data before being sent.

In the Web services world, the materialization of the data before the exchange is not of top significance for the accomplishment of the process. The time of the invocation of service calls may be affected from physical characteristics. The current system load or the cost of communication may influence the decision whether to invoke service calls before or after the data transfer. For instance, a possible overload of the server makes the communication expensive, leading the server to the choice of sending smaller files and giving to the client the responsibility of some part of the materialization of the data.

On the other hand, compatibility issues may be created. As Web services may in principle be called remotely from anywhere on the Internet, there is a possibility the client that receives the intensional document not be able to perform them, e.g., a newspaper reader's browser may not be able to handle the intensional parts of a document, or the lack of access rights has as a result the user not have access to a specific service. These occasions makes the materialization of the data an obligatory process for the server side before the exchange of the information. Getting more general, it should be indicated that there are analogies to most of the logical level motivations.

The dynamic nature that data may have is one reason that influences in a significant way the performance and the quality of Web servers. The following example explains in a precise way, the level of affection. A Web server S publishes a pc catalogue while a comparative shopping site C, which has the role of the client, downloads the pages of the catalogue often. The cache of C will contain these pages, but information such as pc prices that change quite often will be old. Suppose that the Web server S takes the decision to turn his catalogue pages into active. The reload of a page that is now controlled by a Web service may result in sending only the different data between the old and the new page such as the change of prices, offering also savings in communication. Moreover, in an Active XML framework, C is able to be informed of the price changes without having to load the page again, making only subscription to price changes. Consequently, C will have in his possession a more precise copy of the catalogue the server provides.

Knowledge and its use in the intensional field is obvious in another example. The exchange of knowledge in the form of intensional information can accomplish tasks such as routing. A peer A requests some data from peer B and B in fact finds the data in some peer C. Then the data from C to A would be transferred through B. However, if B answers A intensionally, then A may obtain the data directly from C, which possibly results in saving in communication.

## 3.2 AXML documents

The simple idea of embedding calls to Web services inside XML documents is the base for the AXML documents. To become more specific, an XML syntax to denote service calls is defined, and elements complying this syntax are allowed to be part of the document anywhere in it. A fact which is extracted so far is that these calls constitute part of XML information that is not provided with an extensional way, but with an intensional one, by appointing means to get the appropriate data. The materialization of these service calls is feasible, meaning that the Web service which is associated to the element of the document is invoked using the SOAP protocol, and the results that return as a response to the invocation enrich the document.

The *Active XML documents* are valid XML documents, where service calls are denoted by XML elements labelled call [54]. They contain static XML data and might contain intensional data which is represented under the form of service calls. Let's consider as example a part of a document:

```
<cities axml: docName="CITY"
xmlns:axml="http://www-rocq.inria.fr/verso/AXML">
  <city locale="it" name="Rome" timezone="Europe/Rome">
    <time>
       <axml:sc doNesting="true" frequency="every 3600000"
        id="076A1B60-C353-D4A4-0056-5E9DC88450B9"
        lastCalled="1087387299111" methodName="getTime"
        mode="replace" serviceNameSpace="Time"
        serviceURL="http://staros.futurs.inria.fr:8080/axis/servlet
/AxisServlet">

          <axml:params>

             <axml:param name="locale">
               <axml:xpath>
                 ../../@locale
               </axml:xpath>
             </axml:param>

             <axml:param name="timezone">
               <axml:xpath>
                 ../../@timezone
               </axml:xpath>
             </axml:param>

           </axml:params>

       </axml:sc>
    </time>
  </city>
…
</cities>
```

**Table 4-4: Example of AXML document**

### 3.2.1 AXML Service Call Attributes

The first important element is the namespace declaration. "***http://wwwrocq.inria.fr/verso/AXML"*** is required as a namespace for all the Active XML Documents and is usually bound to the ***axml*** prefix.

The central element is the Service Call (***sc***) XML element. It is defined in the special namespace mentioned before and has a set of attributes and children XML elements defining:

- The Web Service to call
- Its parameters
- How and when to call it
- What to do with the results

The most important attributes of this element which define the web service to call are: *serviceURL, serviceNameSpace* and *methodName***.**

The *serviceURL* attribute specifies the end-point of the WebService to call, in our case: "*http://staros.futurs.inria.fr:8080/axis/servlet/AxisServlet"*.

The *serviceNameSpace* attribute specifies the namespace to use for the body element of the SOAP message (*soap:body*), more simply the method namespace URI. In our case this is *"Time"*.

The *methodName* attribute defines the name of the operation to invoke on the Web Services, in our example *"getTime"*. For the Active XML Web Service (a declarative Web Service provided by the Active XML Peer), it is not important: *"invoke"* can be used or anything else.

*Signature* and *useWSDLDefinition* attribute are some other attributes of low importance. The first one set the URL of the WSDL file defining the Web Service. It is optional since it is used only when type validation on that particular service call is wanted. The other one specifies if the file indicated by the first should be used, is optional and its default value is false.

A Service Call element has also attributes that provide information on how and when to activate the service call.

The *frequency* attribute is the most important in this category. It can have several values: a) Once b) Lazy c) On Date d) Every X

If no frequency is defined, the service call will not be activated by the Active XML Peer. *Once* means that the service call will be activated only once at the start-up of the peer. Every time the peer is restarted, the service call is activated. *Lazy* means that a service call will be activated only when its result is useful to the evaluation of a query or when the instantiation of a Service Call parameter, defined through a Xpath expression is necessary. *On Date* indicates exactly when the service call will be activated. If the format is incorrect or the date is in the past, the service call will not be activated. Example: frequency = "on 02/04/05 14:22". *Every X* means the service call will be activated periodically and the interval is given in milliseconds.

Other important attributes are:

- *id* – identifies uniquely in time and space the service call. It is automatically generated by the Active XML System.
- *Name* – has no particular meaning and is optional
- *Callable* – its value is *true* by default and if it's set to *false* causes the service call not to be activated, not even the frequency is correct.
- *lastCalled* – used to keep track of the last activation of the Service Call
- *followedBy* – allows chaining the evaluation of the Service Calls inside the same Active XML Document in a simple manner. Its value has to be a valid XPath expression returns a single ***sc*** element or the id of the Service Call.

### 3.2.2 AXML Service Call Parameters

The parameters of the service call are specified by a child element of the ***sc*** element which has the name *params* and is bounded to the Active XML namespace. It has to be present even if the service call has no parameters. Every parameter has a correspondent element *param* in the representation (bounded to the same namespace). An attribute *name* is required for this element. The parameters can be expressed directly as a value or through an XPath expression. If the service call has several parameters their order must follow and match exactly the one from the WSDL that describes the Web Service. The above statements can be resumed through the following schema representations:

```
<element name = “params”>
 <complexType>
  <sequence>
    <element ref = “axml:param” minOccurs=”0” maxOccurs=”unbounded”/>
  </sequence>
 </complexType>
</element>
```

**Table 4-5: *Params* element schema representation**

```
<element name = “param”>
 <complexType>
  <choice>
   <element ref=”axml:xpath”/>
   <element ref=”axml:value”/>
  </choice>
  <attribute name = “name” type = “xsd:NCName” use = “required”/>
 </complexType>
</element>
```

**Table 4-6: *Param* element schema representation**

### *The Value Parameter*

This value can be any well-formed XML fragment. It conforms to the following representation:

```

```

```
<element name = "value">
  <complexType mixed = "true">
    <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

**Table 4-7: *Value* parameter element schema representation**

### *The XPath Parameter*

The second way of writing an Active XML Parameter is through an XPath expression and conforming to the following schema component representation:

```
<element name="xpath">
  <simpleType>
    <restriction base="xsd:string">
       <whiteSpace value = "collapse"/>
       <minLength value = "1"/>
    </restriction>
  </simpleType>
</element>
```

**Table 4-8: *Xpath* parameter element schema representation**

Every time a Service Call needs to be activated, the XPath parameter is evaluated. The evaluation of the XPath is made starting from the corresponding ***sc*** element. So, in our example, the Service Call has two XPath parameters: ../../@locale and ../../timezone.

### 3.2.3 Service Call Result Handling

The AXML Document might be provided with the information regarding what to do with the results of an Active XML service call. The two attributes affecting this fact are: *mode* and *doNesting***.**

The *mode* attribute has possible values: *merge* and *replace.* The *merge* value means that the results will be added as the immediate right sibling of the **sc** element, considering the AXML Document as an ordered labelled tree. The previous results will be kept in the AXML Document like in the figure below. Replace means that the

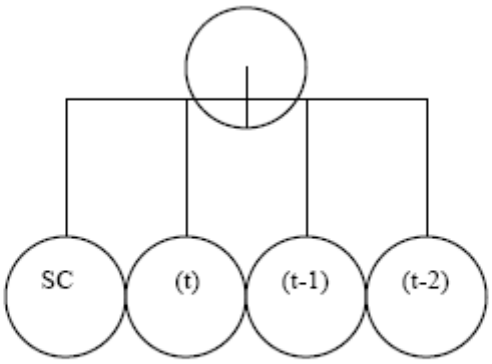


**Figure 4-12: The service call and its responses in the tree hierarchy**

previous results will be replaced by the new invocation of the Active XML service call. To keep track of the results, the Active XML system will add a special attribute to the top-level elements of the results named *origin* and bound to the Active XML namespace having as value the *id* attribute of the Active XML service call.

The *doNesting* attribute is used to keep track of the TEXT nodes which resulted from a service call. That special kind of nodes has to be handled differently: if keeping track is desired, they have to be wrapped in a special element named **text** and bound to the ActiveXML.

***Materialization***

Materialization is the process in which the invocation of the related Web service from a service call embedded in an AXML document activates the call, and the returned result is used to enrich the document. In technical terms, the SOAP

protocol is used in order to accomplish the call invocation, based on the attributes of the call element. The values of the parameters such as the children of the call element are also contained in the sent SOAP message. The returned SOAP message contains the result of the invocation, or an error message. The non-existence of errors has as a result the replacement of the service call element with the returned results from the received SOAP message.

***Tree representation***

AXML documents can have a tree representation just like the plain XML documents. AXML trees have two types of nodes: data nodes and function nodes. Data nodes correspond to the standard nodes in XML data, whereas function nodes correspond to service calls. The process of materialization leads to the replacing of a function node by the result of the related service invocation.

## 3.3 AXML Services

AXML documents following the syntactic of XML are valid XML documents. Moreover, there can be materialization of the service calls that are embedded using the existing web service standards as WSDL and SOAP. These two facts reason the characteristic of portability which results in their exchange between systems. The portability of AXML documents is crucial because it enables the introduction of AXML services [50], web services that accept AXML documents and return AXML documents.

### 3.3.1 Motivations for AXML Services

The motivations presented for AXML documents play also significant role for AXML services but their usage raises more advantages that need also to be described. These advantages can be divided to these that come from intensional results and these that come from intensional arguments.

Intensional data that come from a service invocation result is one part of the whole answer or some kind of summary of it. This characteristic is similar to this of search engines where the most proper results are returned and their location is

pointed. So, the computation of the whole answer may become on-going as there are cases where the results are provided before the fulfilment of the computation while in other cases the receiver may be obliged to invoke other service calls in the result in order to acquire the whole answer. The enrichment of an answer is another advantage of intensional data in a service invocation result as the receiver has the possibility to ask for it.

Furthermore, intensional arguments give an additional motive for the use of AXML services. The existence of intensional data in the input of a web service forces the server to carry out the process of acquiring the whole argument before executing its normal work. This case is a kind of a simple service composition where the outputs of the materialized intensional information act as inputs for other services. Section 5 will present a detailed study for how this advantage of AXML can become beneficial in the world of integration of data, combining it with enhanced descriptive specifications. So, although AXML do not provide mechanisms in order to deal with the subject of web service choreography, it supports some restricted form of web service workflow where with the necessary additional modules can turn AXML to establish a new and with more up-to-date characteristics way of service composition.

## 3.4 Current Architecture of the AXML Peer

A brief synopsis of the current architecture of an AXML peer will be given below. The main components of an AXML peer can be split in two categories, the one with these responsible for the client mode operations and the other responsible for the server mode operations. Some of them, of course, can accomplish operations at both modes.

When the AXML peer acts as a server, it receives a service request through the SOAP Wrapper, which forwards it to the Service Provider. Service Provider finds the main Service Type to execute, and orchestrates any necessary pre and post processing of the request and the answer, which may involve the execution of local or remote services through the Execution Engine. A service, that its invocation has side effects, causes updates that are performed on the persistent AXML documents of the peer by the Persistence Manager. Finally, the answer is returned through the SOAP Wrapper.

On the other side, when AXML peer acts as a client, a service call is activated by the Document Manager and scheduled by the Execution Engine. If this service is a remote one, the invocation is carried out by the SOAP wrapper. The returned answer is sent to the Document Manager, which updates the related AXML document through the Persistence Manager.

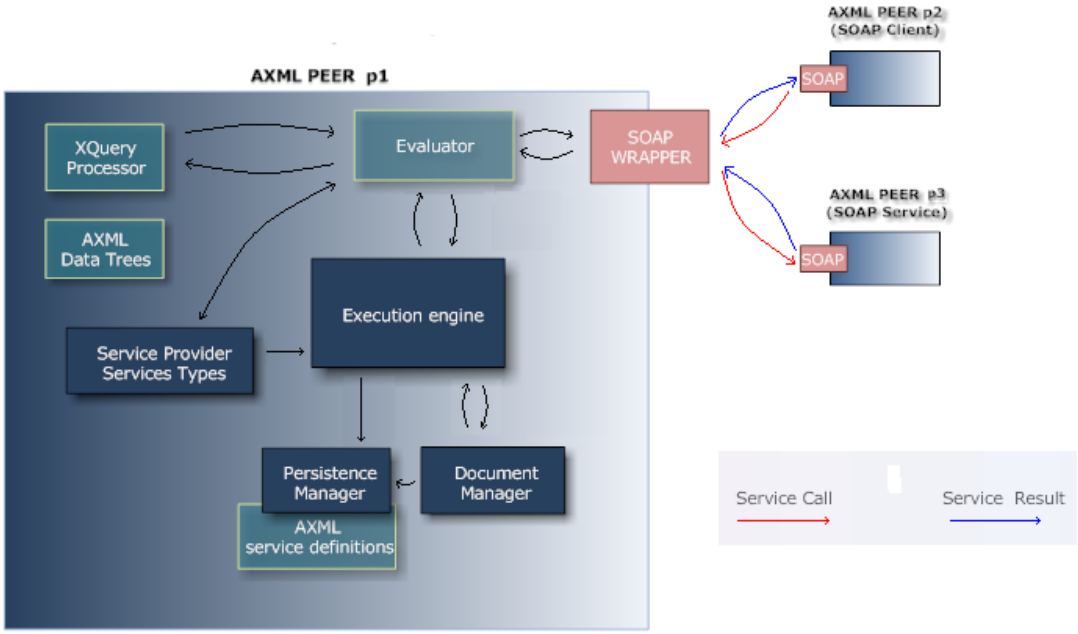


**Figure 4-13: The initial axml peer architecture**

### The AXML Service provider

This component is in charge of the server side functionality of an AXML peer by answering the service requests the peer receives. This is succeeded with the adoption of the chain of handlers design pattern. A chain is simply a sequence of handler invocations, where each handler is a program that performs some part of the required processing.

### Declarative service types

AXML focuses on declaratively specified services. Different service types are defined in the AXML service provider, one for each declarative language that is needed to support. Each service description is an XML document which references an existing service type, defines the types of the input and output parameters of the service, and gives its declarative specification, for example an XSLT transformation. The persistent repository is the storage place of service descriptions.

New service types can be added to the AXML peer without recompiling it, through a simple XML configuration file. Additionally, default implementations of these interfaces are provided, which support common functionality, such as AXML parameter handling, partial generation of the WSDL description of the service and can be extended by each service type implementation.

**The AXML Document Manager**

This component is responsible for managing the peer's persistent AXML documents and their embedded service calls. It interprets the client-side policies for the mode of activation of service calls and the validity of their .Additionally, the document manager is responsible of updating the AXML documents by merging the results of service call invocations.

## 3.5 Chapter Summary

In this chapter, AXML was introduced as a new framework in the world of data management. Its main characteristic is that the documents that comply with it, have embedded service calls to web services, something that gives them the description of active. A series of motivations have been presented for the usefulness of AXML, such as partial computations and data enrichment, highlighting the advantages that offer in contrast to simple XML. AXML contains specific elements and attributes that succeed in the implementation of the above motives. Its main extension, the axml:sc element, succeeds in the invocation of a web service, following the location it is provided through it and manipulating with the assistance of the rest of the attributes (e.g. timely with the periodic activation). Finally, the initial AXML peer architecture is presented, pointing out the main components that fulfil the data exchange and management.

# Chapter 4

# Composition in the AXML Framework

## 4.1 Introduction

The rapid growth of internet has resulted in the existence of a huge number of web services. The current evolvement of the Web relies on the distributed existence and integration of data from various information resources. Moreover, there has been a significant increase in the necessity of using the available web services in such a way that more complicated services would be created in order to provide additional functionalities. This fact is a result of simple service's weakness to fulfil on its own the requirements that a user asks. Thus, the accomplishment of a more complex goal is feasible using more than one of the existing services.

Currently, users either utilize some services that they know already or find services by looking it up in a keyword-based search engine - e.g. google.com or by looking it up in a web services registry such as UDDI. Moreover, both the discovered services which are composed and the feasible dataflow they produce are mostly fulfilled manually, weakening the effectiveness of the process. The problem stands at the way the web services abstraction are modelled and the way of composing the available services.

The current approach has as a purpose to bring forward how the complexity of the process in distributed environments will be reduced, offering an enriched model for the integration of data. In general, the process of services composition can follow four dimensions of implementation [43] which:

a) if a user is involved in the specification of composition
b) if the composition process follows specific patterns
c) if the composition process is fulfilled dynamically

d) if a user is involved in the fulfilment of the composition

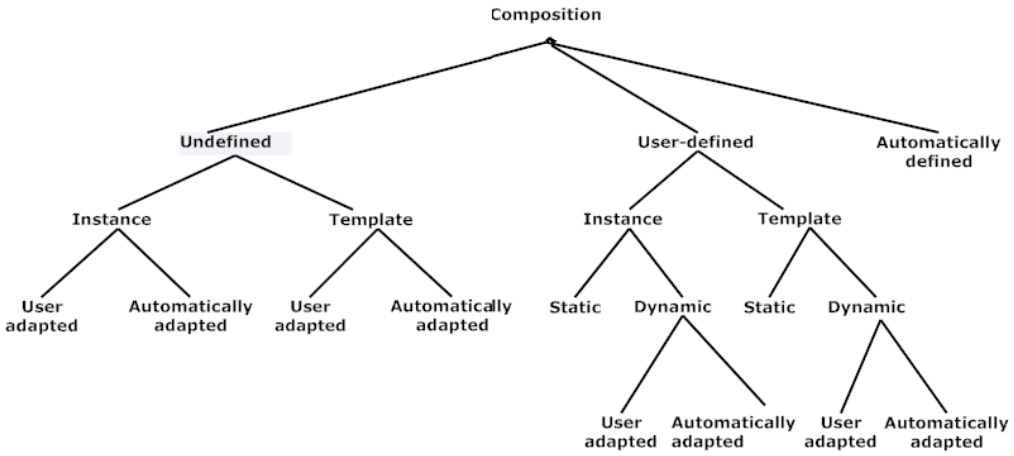


**Figure 5-14: Web service composition methods**

The first dimension focuses on the whole definition of the composition process from a user, something that includes both the management and data-flow and the services which are composed. On the contrary to the previous, the automatic way of composition has no human interference as the system is responsible totally for the whole process.

The full automation of the composition process is an issue that still raises many challenges in the computers science research, that are related with the way of mapping of what user needs to a set of correlated services where there is an output-input matching between them, resulting in a composition that satisfies the initial requirements. Moreover, the user-defined and the automatic composition categories may use either actual service instances or service templates. The discovery and the integration of individual service instances take place at execution time for a given plan [47] in the automatic technique. Additionally, the definition of a composition plan may not become at design-time but may be built dynamically at execution time. Finally, there are techniques that incorporate characteristics from both sides and offer a semi-automatic adaptation of the composition process that is more feasible with the current web services technologies and standards.

The efficiency of this work can be focused on the improved characteristics for the discovery of the available services, the enabling of the composition process in an

enhanced way that OWL-S offers in the description of the services and the introduction of the active nature in the integrated data with the use of the AXML.

## 4.2 Implicit Calls Using OWL-S

### 4.2.1 General Characteristics

As it has been indicated in previous sections, the service call which is used in AXML is explicit as the element axml:sc in an axml document is the place where the location of the calling service is defined explicitly. This definition is followed by a number of attributes that indicate the appropriate service. Additionally, the element axml:sc may contain an implicit service call where its exact location is unknown but a description of the service is provided. The execution of this kind of service calls has to be integrated in the AXML framework in terms that serve the principles of discovery and composition with semantic characteristics.

A semi-automated discovery or composition of services requires the semantic definition of their functionality. The functionality is clearly described with the use of ontologies, a fact that underlines their important role. Following this subsumption, the successful invocation of an implicit service call in AXML is premised by the ontology based description of the services. The way of the services' annotation is of major significance as it has to be determined the attributes of the service that are required to be included in order the search process of the candidate services to be fulfilled.

The inputs of the service have to be provided as they are needed for the invocation of the proper service. The description of the outputs is also necessary as an AXML document may be considered as an incomplete materialization where plain XML and dynamic information acquired from service calls is combined together. The matching of a service with a service query is a result of the matching of their inputs and outputs. Moreover, the description of the service domain affects positively the integration process as the search space for the service to be invoked is reduced. This fact leads us to use descriptions of the service query which are represented by the definition of its domain, outputs and inputs.

The use of OWL-S and its characteristics offer a crucial assistance in the representation of the query enriching it with semantic information. The gains of this representation are that service is reflected in the discovery process as it fulfils the matching between profiles. Furthermore, the description of the preconditions and effects in OWL-S, as it has been presented in the previous chapter, extends the composition capability. The definition of inputs and outputs of services for composition is our major concern in this work.

### 4.2.2 Annotation of an Implicit Service Call

The matching of a service query described in terms of AXML with the offered service described in OWL-S needs the annotation of the service call from the requester side with means that serve the capability matching between the two sides. Table 5-1 presents an example of an implicit service call which obeys different structure from the explicit service call in the following parts:

a) The attributes *serviceURL*, *serviceNameSpace*, *methodName*, *signature*, and *useWSDLDefinition* of a normal service call in AXML that fulfil the identification of the requested service are not specified in the implicit service call. The replacement of the functionality is succeeded by the introduction of a new attribute named *serviceCat*. This attribute enables the service call to describe in detail the domain of a service, reducing in this way the search area for the proper service.

b) Furthermore, there is an addition to the *param* element of two more attributes that are called *param_type* and *param_data_type*. *Param_type* specifies the type of a parameter while *Param_data_type* provides the description of the classes where the values of the parameter belong to. The implicit service call is defined as having two parameters:

- the *publisher* being the input of the service whose value is of type *http://localhost:8080/axml/owls/Concepts.owl#publisher*
- the *booklist* being the output of the service whose value is of type *http://localhost:8080/axml/owls/Concepts.owl#booklist*

```
<?xml version="1.0" encoding="UTF-8"?>
<Inventory axml:docName="Inventory"
 xmlns:axml="http://www rocq.inria.fr/verso/AXML">
   <publisher>Addison-Wesley</publisher>
   <publisher>Morgan Kaufmann Publishers</publisher>
   <books>
    <book year="1999">
      <title>The Economics of Technology and Content for Digital TV</title>
      <editor>
        <last>Gerbarg</last><first>Darcy</first>
        <affiliation>CITI</affiliation>
      </editor>
      <publisher>Kluwer Academic Publishers</publisher>
       <price>129.95</price>
    </book>
    <axml:sc serviceCat= "http://localhost:8080/axml/owls/hierarchicalProfile.owl#Book"
       frequency="every 3600000" mode="replace" >
        <axml:params>

            <axml:param name="publisher"
       param_type= "http://www.daml.org/services/owl-s/1.1/Process.owl#Input"
       param_data_type= " http://localhost:8080/axml/owls/Concepts.owl#publisher">
             <axml:xpath> ../../publisher/text()</axml:xpath>

           </axml:param>

           <axml:param name="booklist"
       param_type="http://www.daml.org/services/owl-s/1.1/Process.owl#Output"
       param_data_type=" http://localhost:8080/axml/owls/Concepts.owl#booklist">
           </axml:param>

       </axml:params>
    </axml:sc>
</books>
</Inventory>
```

**Table 5-9: An example of an implicit service call**

### 4.2.3 Data Model for Implicit Service Calls

AXML documents and services should be a part of a data model where the integration of the information following semantic annotation will be possible. The description of such a model should serve the rules we presented so far in order to establish a full framework where AXML is enriched with semantic logics. An AXML instance consists of a number of peers, each one containing some AXML documents that are active. The evaluation of these documents generates calls between these peers and possibly results in new documents being evaluated at each peer.

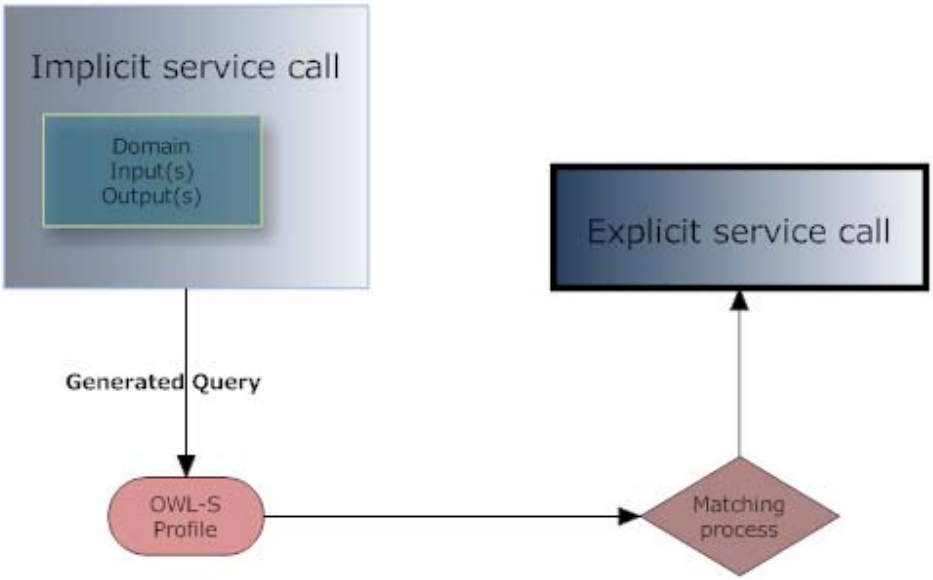


**Figure 5-15: Evaluation process of query with implicit service call**

An instance I consists of a number of peers $p_1$, ..., $p_n$. The content of a peer $p_i$ is defined by a triple (R,F,W), where $R_i$ the peer's repository, is a set of persistent AXML documents, F, the peer's services, is a set of AXML service definitions, and W, the peer working area, is a set of AXML temporary documents. All these sets are assumed to be finite. Each document *d* in the working area W of a peer $p_i$ represents the computation of some service call in $p_i$.

An AXML document is modelled by a labelled tree with nodes representing the document elements/attributes and the edge represents the component of relationships among documents. Some of tree leaves are special implicit service call nodes and is defined by a tuple $< p,f,x_1, ..., x_n >$, where *p* is the peer that contains the expected service, *f* is the domain of the expected service, and $x_1, ..., x_n$ are the inputs and outputs annotated by concepts of the expected service.

This representation model will be followed in the evaluation process of a given query. Its structure will be clearly understood after having introduced all the components that need to be added in the current AXML peer architecture, an issue we are going to examine below.

## 4.3 P2P Composition

### 4.3.1 The Use of p2p Architecture

The formalization of the query with the OWL-s profile indicates that the discovery and the composition process can be started following a certain methodology. These two processes can follow two different kinds of computing, the centralized [44] and the distributed one [26].

The centralized one has as its main component a centralised registry which has the role of the advertiser for the web services. A newly existing web service establishes its presence being in the registry and moreover makes its functionalities known through it to the candidate requesters. The gathering of information of a specific service from a possible requester is fulfilled after the necessary contact with the registry. This fact leads to the acquisition of the proper details in order to discover the needed service or the needed services in order to compose them.

Although the centralized computing type offers a guaranteed discovery of the available services in the registry affecting positively the composition process, it still has sufficient problems such as performance bottlenecks and costly synchronization between the service providers and the registry - e.g. the large number of stored advertisement on centralized registries causes delays in the capabilities update process- that prevent it from being dominant in the web services management area**.**

On the other hand, the distributed kind of computing changes the centralized type of the registry, transforming it into a distributed one. Following this assumption, web services are nodes in a network of peers, which may or may not be Web services. In the discovery process, a requesting web service queries its neighbours in the network. Possible matching states the end of the discovery with the sending of the appropriate reply while failure in matching, results in querying its own neighbouring peers and the query propagates through the network where message propagation is usually bound by a Time To Live (TTL) number that limits the distance a message can travel. Any node in the p2p computing type will send response to the requested queries on the contrary to the centralised one and moreover single points of failure are reduced because of the high level of connectivity which ensures the delivery of the

message to the provider. In addition, there is lack of bottlenecks due to the indexing of the available web services that each node has.

The AXML framework allows every peer in the network to provide its own data to the rest of the peers as far as web services are published over the AXML documents in their repository. Therefore, the services and their functionalities that are available at each AXML peer change frequently, subsuming that numerous implicit service calls can exist.

### 4.3.2 The Structure of a p2p Composition Network

The complexity of the p2p composition is a matter that needs to be confronted. An attempt to confront this problem is the adopting of a 2-dimension network in order to fulfil the whole process. Each peer in such a network advertises several Web services which belong to specific domains. One of its main characteristics is the grouping of the peers whose web services belong to the same domain. Every peer in the network belongs at least to one domain which, from its side, has both a master peer and a backup peer. Each one of these peers plays a significant role in this architecture. Firstly, the master peer of each domain has as its responsibility the maintenance of two lists: a) the list that contains all the master and backup peers of other domains and b) the list that contains all peers that belong in the master peer's domain, their advertised services with the inputs they need for their invocation and the

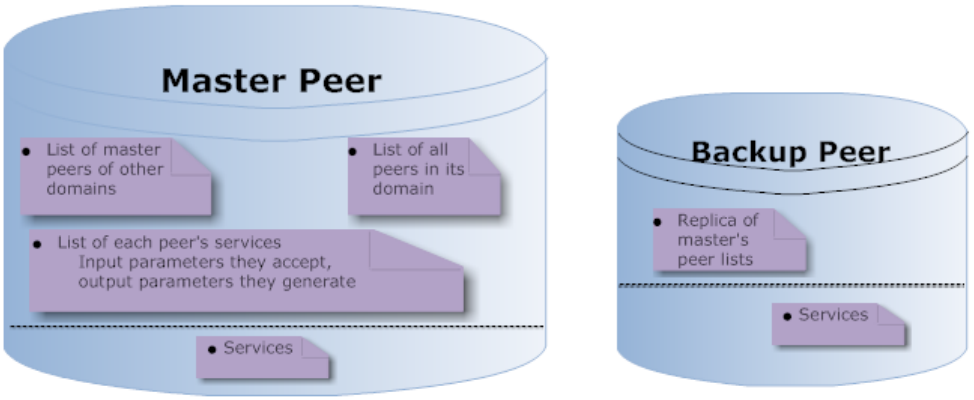


**Figure 5-16: The content of master and backup peer**

output parameters they produce. The role of backup peers is to maintain a duplicate of the previous lists.

Additionally, each peer besides the information it has about its master and backup peer, it supports the maintenance of two more lists: the predecessor-successor lists for its related services. A predecessor of a service refers to the outputs that match the inputs of this service. Respectively, a successor of a service refers to the inputs that match the outputs of this service. Thus, the participation of the candidate peers in the composition process becomes feasible with the assistance of the predecessor-successor relationships that are reported in these lists. The followed process begins from the discovery of the query's outputs that are provided by one or more network's peers and continues to the finding of those which can accept the inputs the query provides and are necessary for the composition.

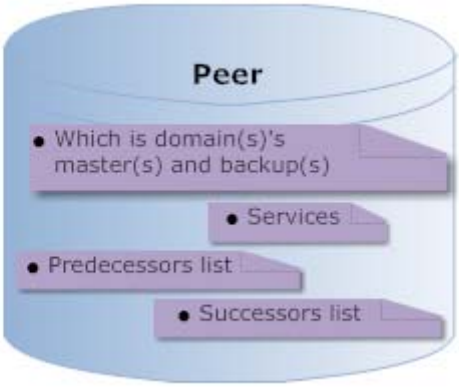


**Figure 5-17: The content of a simple peer**

### 4.3.3 The Evolution of the Network

In a p2p network, every peer has as its role the provision of web services. Each one of these services belongs to a specific domain as we mentioned before and each peer belongs to one group of peers at least, where the term group refers to peers whose services are of the same domain. The groups are structured obeying the rules of the ontologies structure where hierarchy is the main characteristic. Having this assumption, the structure and the way a p2p composition network evolves may be presented clearly.

There are two issues that affect the evolvement of a p2p network**:** the first one is related with the domain which is associated with the available services while the second one refers to the services input-output matching relationships. Obviously, the first issue refers to the master peer's maintenance of the two lists we described before while the second to the peers which participate in a predecessor-successor relationship. The successor peer always attends this relationship as this information is significant enough for the progress of the composition process. So, network is organised in a way where peers of the same domain are grouped together and each peer is aware of the dependencies among them.

A case network with the characteristics that have been described before is presented below. The network contains only one peer in its initial mode, p1. Then, p2 contacts p1 in order to become member of the network. Based on their common

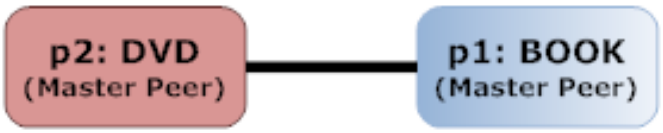


**Figure 5-18: Initial state of p2p network - Master peer assignment for different domains**

ontology, they decide what domains they provide services for, in this example '"Book" and "DVD". The fact that they are the first peers with available services for the specific domains turns them into the master peers, recording it in their lists.

A new entrance in the network becomes when p3 contacts p2 and, following the previous rules among p1 and p2, p2 decides that p3 belongs in the "Book" domain. As p2 is a master peer of its domain, it is responsible for the redirection of p3 to the proper peer with the help of the lists that maintain which is p1 in this situation. The result of the redirection is the contact between p3 and p1 and the recording of p1 as its master peer. Additionally, the fact that "Book" group consists of p3 apart from p1 leads to the automatic assignment of the backup peer role to p3 for this domain.

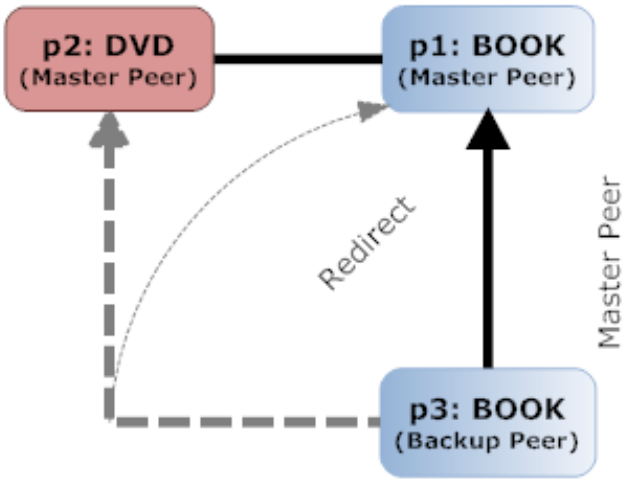


**Figure 5-19: Join of p3 into the network – Redirection and Backup Peers assignment**

The self-organizing characteristic enables the network to confront several situations such as the contact between a candidate peer with a non-master or back up peer – something that takes place at the time of p5 joining in the network - or the join of peers from different sub-groups within a group.

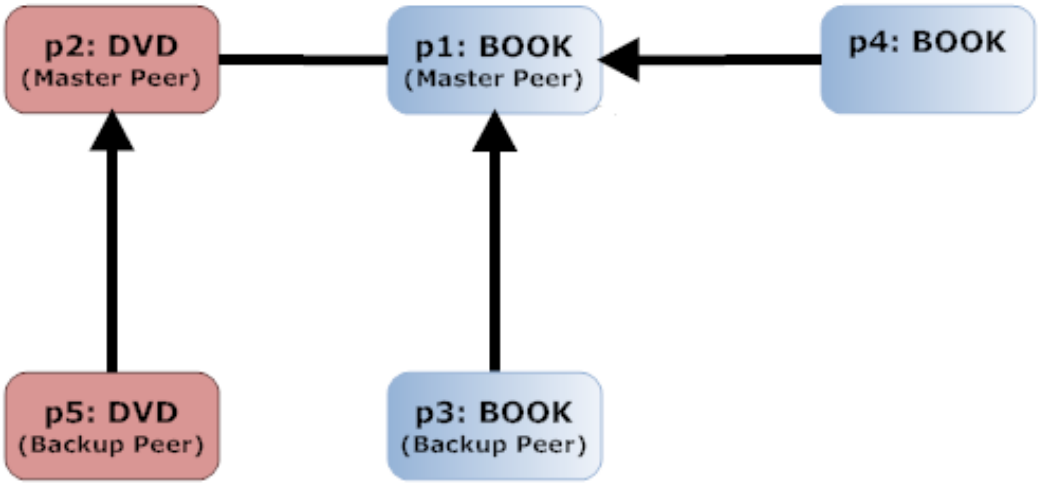


**Figure 5-20: Join of p4 and p5 into the network**

The entrance of every peer in the network, besides the way it is organised and connected with the other peers, results also in a parallel process that is fulfilled by its master. The master peer queries other master peers of different groups in order to

discover whether the outputs of the new peer matches their inputs or whether there are outputs that matches the new peer's inputs.

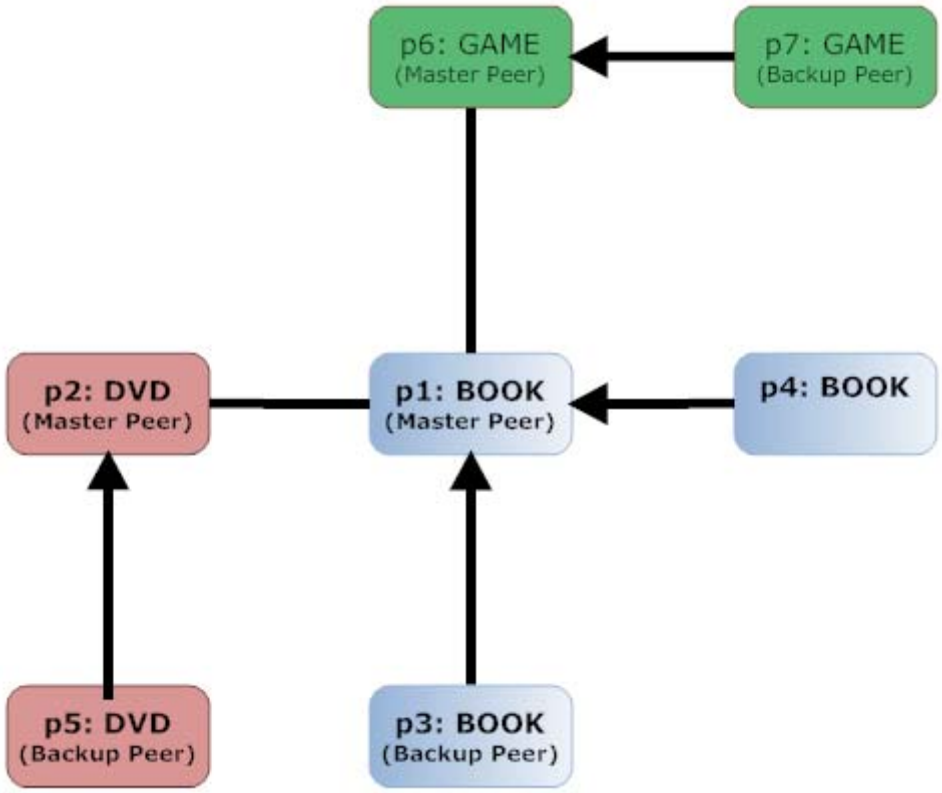


**Figure 5-21: Join of p6, p7 into the network**

Successful match in the first occasion makes the new peer a "predecessor" of any such peer while in the second one becomes a ''successor'' of any such peer. The existence of such a matching relationship is recorded in the master peer of the successor that is involved.

### 4.3.4 AXML Architecture for Implicit Service Calls

The implicit service calls that have been described previously can be achieved in the AXML system with the embedment of an extra element suitable for the p2p discovery and composition processes. Its structure is based on WSPDS [40]. WSPDS (Web services peer-to-peer discovery service) is a distributed discovery service implemented as a cooperative service, fulfilling semantic level capability matching and offering an efficient and compatible discovery in distributed environments.

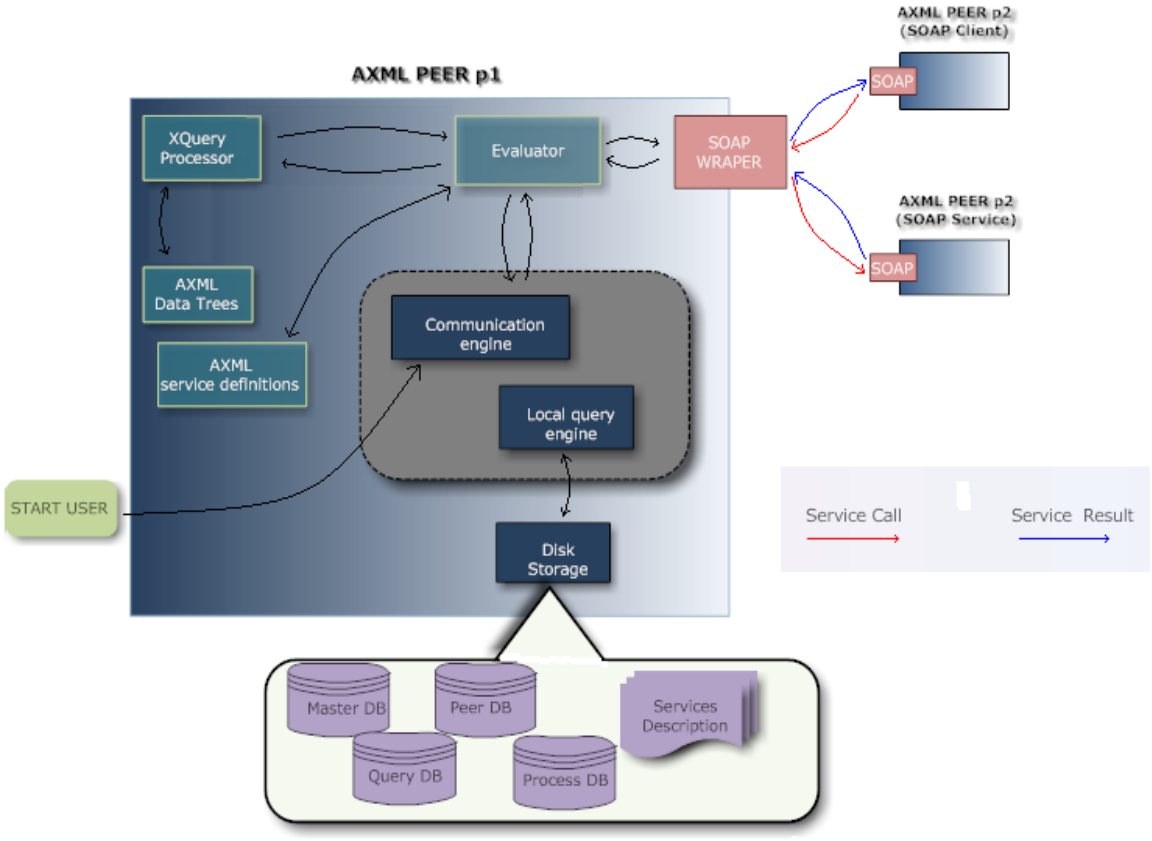


**Figure 5-22: AXML peer architecture**

The additional element is the searchService component. This is consisted of two engines, the communication engine and the local query one, each of these having crucial role in the composition of user's desired goal.

The communication engine is responsible for the provision of services ' interfaces to the AXML evaluator, to the user and to the other peers. One of this engine's characteristics is to receive service queries from the evaluator, sending responses for them either locally with the conduciveness of the local query engine or globally through the other peers. The result of this phase is the gathering of various answers and the merging of them, giving the user a number of choices of the existing or the composite services he wills to invoke. The final phase of this activity leads to the submission to the evaluator of the list of the desired services with their location.

Another task that is supported is the receiving of queries from the other peers in the p2p network. The communication engine with the help of the local query one analyzes the queries and responds to the caller the appropriate answer. Furthermore, it redirects

the query, which it has not been expired yet according to the TTL measure, to the related peers. The TTL measure, as it has been introduced earlier, controls the propagation of a query through a p2p network, reducing the level of dissemination of the query through it.

The local query engine is responsible for the answering of the query that communication engine sends to the local peer. The elements that constitute it are the following: *Service-Cat* which refers to the service domain the service belongs to and the *Outputs*, *Inputs* which are respectively responsible for the outputs and inputs matching between the OWL-S profile description of the query and those of existing services. The matching process which will be described is based on the semantics that the services impose. The ontology description that is used for the semantics of the services guides us to apply the matching process between the services only with the matching between the concepts that describe the services, including inputs and outputs.

The process for the matching of the candidate services which will participate in the composition comply with the following rules. Obligatory requirement for successful result in the matching process is the matching of every output of the query with an output of the published services. The process results in failure if there is no matching for one of the query's outputs. The inputs between a published service and a query sent are matched following the same procedure as outputs, differing only in the order, as the inputs of the service are examined against the query's ones.

### 4.3.5 Composition Algorithm Used by searchService Component

The algorithm that is presented below gives a precise description of the methodology that is followed in discovery and composition in searchService component.

First of all, the implicit call model introduced before is the base of the generation of the appropriate requested query. In brief, implicit service calls embedded in an AXML document are represented as leaves of an AXML tree. Their definition obeys the following tuple structure: $< p,f,x_1, ..., x_n >$, where $p$ is the peer that contains the expected service, $f$ is the domain of the expected service, and $x_1, ..., x_n$ are the inputs and outputs annotated by concepts of the expected service.

**Algorithm 1 Composition Algorithm – searchService**

SLL: Service Location List

**STEP 1:**
The query is generated from the evaluator and is transferred in the communication engine of the searchService component. The query is forwarded to its master

**STEP 2:**
This query is forwarded to the master peers whose services are in the same domain of the query.

**STEP 3:**
**if** ∃ query ∈ QueryDB is similar to this query **then**
Return the results of the similar query
**goto STEP 8**
**else**
The query is forwarded to the member peers that provide the services whose outputs match those of the query.

**STEP 4:**
The TTL of the query is reduced

**STEP 5:**
**if** the inputs of candidate service match those of the query **then**
The SLL is updated adding the service in it.
The calculated query with its results is stored in its queries data bases.
**goto STEP 7**

**else if** ∃ predecessors for the candidate service and the TTL > 0 **then**
Forward this query to its predecessors.
Add the candidate service to the compositions in the list SLL returned by its predecessors.
**goto STEP 6**

**STEP 6:**
There is fusion between the local (initial's master) composition list with those returned by their predecessors. This process is fulfilled by the communication engine.

**STEP 7:**
The master of the initial peer receives the results (SLL)

**STEP 8:**
The initial peer's communication engine receives the list from its master's and sends the result (SLL) returned to the evaluator to the user.

**Table 5-10: Composition algorithm used by searchService component**

The evaluation of the service call requires its exact location. The successful invocation of the service needs both the evaluation of the service's location and the processing of it. First of all, the evaluator component should be responsible for taking the parameters that represent the inputs of the service and generate the desired query. The query is based on OWL-S profile descriptions that the evaluator can access and leads to the return of the query's location to it.

When a query is generated, the first action is its transition from the evaluator to the searchService component. Afterwards, searchService transmits the query to the master in its domain. Besides this action, it is also responsible for transferring the answers of the master peer to the user and for returning the list of the user's selections from the available compositions to the evaluator.

The master of the initial peer takes the decision of which are the candidate domains for the query and then broadcasts it to the master peers of the related domains. The available composition plans are discovered and the master peers of the involved domains return the list of them to the master peer of the initial one. So, the master has as its last action to deliver the received list to its peer.

The master peers of the participating domains when they receive the query from the initial's master peer examine their query database if there is any stored query that has been already answered and matches the given query. The existence of such a query has as a result the immediate sending of the proper answer. Conversely, they start searching in their peer database to check whether there are available services in their domain which ensure the exact match between their outputs and the query's ones. The finding of such services leads to the forwarding of the query to their host peers where the compositions plans are created. Consequently, the query database of the master peers is updated.

The role of the host peers is the fulfilment of the input matching process. The inputs of the query are checked with their services' inputs. If there is a successful matching, the host is able to give an answer for the query and add the service to the list of compositions. In different occasion, it broadcasts this query to the peer providing the predecessor of this service expecting an answer from it.

## 4.4 Chapter Summary

In this chapter, the concept of the implicit call in AXML documents was introduced and a technique for discovering and composing web services in a role-based p2p network was described. It was clearly presented the elements that are necessary for AXML documents in order to enhance the description of the services that contain, supporting the use of OWL-S descriptive characteristics. Moreover, the structure of such a network and the way of its development was presented, outlining all the characteristics of the involved peers. In brief, each peer affects the discovery and the composition process for the fulfilment of the implicit calls according to its content and its role. The initial peer that receives the requested query, its master peer and the master peers of the domains with the candidate services for the answer of the query are used from the algorithm presented in order to provide a more flexible and expressive solution for the integration of service calls, and respectively of services, in the AXML framework. Composition plans are feasible only when sufficient matching criteria are existed between candidate services. The whole methodology set a clearer layout for resolving the composition problem with a semi-automatic way with the contribution of the searchService component while it maintains the active nature of the data that AXML provides.

# Chapter 5

# Conclusions

## 5.1 Summary

The general conclusions of this thesis are presented in this chapter. In particular, an evaluative comparison will be made between the proposed direction for the integration of data and the current techniques that are used. The additional characteristics in the process of composition and the use of AXML data for the fulfillment of it are analyzed, highlighting their differences from the existing ways and their advantages. Moreover, some open issues for future work are mentioned.

## 5.2 The Use of Ontology

Ontological approach gives an advanced solution to the description of web services. The use of common ontologies in knowledge based systems enables the understanding between them on a concept level, confronting the data heterogeneity issue. Ontology in general offers

- a common vocabulary which every participant can use
- an explication of what concept has not been described - the formalization of all the notions provides the necessary explanation to the machine in order to substitute the human involvement as much as it can in the discovery and in the composition processes.

### 5.2.1 OWL-S Contribution

In this thesis, OWL-S was described as the framework where ontology is used in such a way that web services have more precise specifications of what their capabilities are. The use of OWL-S for the description of services is an important factor in the proposed framework as it provides enhanced features that are necessary for the attempt of automating the service related processes.

Specifically, OWL-S includes three parts. The service profiles are the first part which has the role of descriptions vocabulary. The main advantage of this part is the provision of high-level descriptions for web services enabling both their advertisement and their sufficient reasoning about how useful they are for the accomplishment of a requested goal.

The second part of OWL-S can create detailed specifications for service models. The descriptions that models offer are related with the IOPE of a service. Preconditions and results especially provide an additional and more precised feature than WSDL language in the matching process of the users' demands with the available services. The effectiveness of this way of representation is clearly shown in the composition process as the choice of the suitable services result in the creation of composition plans that fulfill accurately the initial requirements.

Service grounding is the third part of OWL-S which its role is the mapping of the inputs and the outputs of a service to messages at the communication level and as specific mapping descriptions to WSDL messages and ports are followed, it is obvious that also associates the atomic processes of the service with the operations needed to be described in a WSDL file.

The relationship of OWL-S grounding with WSDL elements can be shown with the creation of an instance of OWL-S grounding class where there is included all the necessary information about the associated OWL-S and WSDL constructs. The existence of specific OWL-S extensions that are relevant to WSDL elements are quite useful to describe how a WSDL construct grounds OWL-S. Specifically, extensions are provided for WSDL types, messages, operations and binding elements making OWL-S more expressive, while its descriptions can contain enhanced characteristics that so far WSDL could not support.

OWL-S, as we have seen, has no intention to replace current web service standards but to comply with them, enhancing their utility and adding semantic features in order to direct the integration of data to become less human dependent.

### 5.2.2 Insufficiencies of OWL-S

On the other hand, although OWL-S has introduced extra characteristics in the world of web services which assist in the support of their semantic direction, there is a series of issues that have not been included yet. Firstly, descriptions for security and QoS properties for web services cannot exist. Secondly, the process model is not able to handle exceptions, a fact that would lead to further alignment with WSDL faults. Moreover, there is a grand need the existing ontologies to become more specific in order to cover a wider and more specialized range of service categories. One possible solution would be the creation of more domain-level ontologies, and especially the development of specializations of current OWL-S profiles.

Finally, as more and more services fulfill sales transactions, the need for modules that support the specification of cost models, negotiations, contracts and guarantees is of high importance as the provider and the consumer should know the cost of transaction accomplishment in order to be aware whether its use is worthless or not. Becoming more general, a general set of quality of service metrics is necessary in order consumer to know the level of reliability and consistency of the service which will be executed. Such modules are also crucial because they assist in the understanding of the possible alterations that need to be made in order to improve the metrics they refer to.

## 5.3 The AXML Choice

The introduction of AXML as the standard for the management of the documents is a fact that gives to our approach extra capabilities. AXML document, as we have previously mentioned, are XML documents with embedded calls to web services. This kind of documents is enriched by the results of the service calls they embed. One first contribution of this idea is that since service invocations is based

completely on current web service standards, portability is succeeded, a fact that results the enabling of the exchange of data.

AXML documents as the mean of exchanging data between systems leads to effective data-oriented schemes for distributed computation. The distributed environment gives the opportunity to the peers that participate into it to co-operate for the performance of data management task with a dynamic way, proceeding in the discovery of new relevant data at run-time. So, AXML documents and services are fully suitable for a p2p architecture where each peer can take both the role of the client invoking the service calls which AXML documents contain or the server role having available services over these documents.

The use of AXML in our approach extents the current capabilities of data management and integration not only in the way information acquired and used but also in the performance of the operations which is a result of the logical motives that have been described earlier in Section 4. An example of what we claim would contain an online shop S which advertises its electronic catalog and another e-shop S' receives the catalog of S in certain periods. Part of the catalog's information such as product prices may often be not up-to-date, something that makes S' to download the entire catalog each time it wants. The intensional representation of data through AXML enables S to provide them with that way while S' is enabled for updating the information about the prices of the product without having to get all the descriptions of the products. Dynamicity, as described earlier, affects in the reducing of communication costs and in empowering the client side to determine with the help of the frequency attribute the period that he wants to retrieve the updated intensional data.

The AXML framework also supports the definition of various kinds of services, including the definition of AXML services which are web services that exchange AXML documents. Moreover, the current implementation has the advantage of accepting new kind of service types, something that gives us the ability to enhance OWL-S in the AXML world as we've already described. This fact empowers us to suggest an extent to the current implementation that advances the way of describing the provided services. An important extension to our approach is the combination of the expressiveness that OWL-S offers with the characteristics of AXML. AXML service calls elements point so far to specific attributes in order to

describe the service which has to be invoked. These attributes provide useful information about the service such as its location, its namespace and the method which is needed to be invoked. All these refer to a certain service, having a weakness in specifying a general concept for the desired service. This limitation is overcome with the introduction of new attributes in the service call element of AXML. The new attributes not only set the sense of concept with the assistance of the advantages that OWL-S induces but also enable a mechanism for supporting the composition of the services that can fulfill the initial requested service. The first one is succeeded with the introduction of the serviceCat attribute in the axml:sc element and the param_type, param_data_type in the param element which is nested in axml:sc, making possible the description of an implicit service call and furthermore enhancing the descriptive characteristics that AXML provided for its services. The second one is accomplished with the insertion of the searchService component as this described in chapter 5. The algorithm presented establishes a procedure that takes fully advantage of the distributed environment in which is applied and of the description of the services in OWL-S, finding the appropriate service or services that can fulfill the request. This gives a significant advance in AXML, as beyond the features that already has, it turns into a framework that the integration of data is possible, offering a more expressive and rich solution for the description of them.

## 5.4 The p2p Choice

### 5.4.1 Centralized vs p2p Architecture

Peer-to-peer technology combined with semantic web raised a new area of challenges offering benefits to the sharing of knowledge. The involvement of peer-to-peer lead to innovative paths where the centralized solutions lack in effectiveness in the discovery, the retrieval and the integration of data from solutions that adopt decentralized approaches.

The existence of metadata in p2p environments is quite important. P2p networks provide data resources that they are not organized following hypertext like structures, when their management could be made through navigation, but follow the

decentralized structure where participant peers are repositories of them, expecting the request of an appropriate query in order to retrieve and send them.

Centralized approaches suffer from many drawbacks such as poor scalability and less consistency when the size of the registry is enormous. . A large number of service advertisements cause insufficiencies in the timely update process of the registry with the changes in the capabilities and in the availability UDDI as a classical example of centralized registry, acts as yellow pages, indexing the services it contains together with their location. Moreover, the searching process is limited, as only keyword-based search is allowed while the search based on similarity of services is not supported.

So, the existence of a centralized repository, as UDDI, for web services descriptions is not sufficient enough while many providers of web services has the tendency to create their own registries in order its use to become more flexible for them. But this fact prohibits the discovery of services as it is highly costly to replicate the information from all private registries to the central one.

The alleviation of this problem is based on combining semantic web with p2p where centralized design is replaced from a distributed one by connecting private registries, thus, creating a p2p network. P2p architecture substantially results in the development of a virtual global registry where its participants are all the local registries. Besides, the logical association of all registries, maintaining though their autonomy, p2p architecture provides decentralization of the control of data management. Furthermore, it supports efficient way of querying, forwarding the query to the relevant peer(s). Specifically, the methodology presented in this thesis, resolve the peer selection service problem as the peers that participate are categorized according to the domains that belong the services they provide. So, the discovery of a candidate service becomes ontology-based, as a possible query is forwarded to the suitable domains that its additional serviceCat element implies. The notion of concept is adopted and is formalized in such a way that the provided solution makes a step further from the current methodologies.

### 5.4.2 Changes in the AXML Architecture

A new architecture for AXML is necessary in order the AXML system to be able to handle implicit service calls. Figure 5.9 presents clearly the changes that have to be made in the architecture of AXML. The system is extended from its initial structure with the addition of two components which are the following:

a) The first one is the *searchService* component. It is consisted of two parts which are the communication engine and the local query engine. The role of searchSearvice component was made clear above in the presentation of the composition process as it is responsible for receiving the query from the evaluator and computing it, if it is possible, with its peer's services or forwarding it to the appropriate peer in order to be computed.

These two modules enhance the capabilities of the current AXML peer architecture as it implements the composition process presented. The communication engine providing services ' interfaces to the AXML evaluator, to the user and to the other peers can receive service queries from the evaluator and can send responses for them either locally with the assistance of the local query engine or globally. Communication engine differs from the initial Execution engine as it carries out processes that fulfil the composition of services that an implicit service call needs in order to be answered, such as the gathering of various answers, the merging of them, and the submission to the evaluator the list of the desired services with their location. The matching process between the generated query and the advertised services is executed from the local query engine resulting in the answering of a possible match to the communication one. The two engines that searchService component contains extend the Execution engine and the way of data exchange in a p2p network resolving the problem of proper answering of the query model we presented.

b) The second component is the *storage* one. Its role is the maintenance of the parts that are necessary for the description of the peer, as each peer in such a network should consist of:

- A registry that should contain the OWL-S descriptions of the services provided by the peer. Descriptions of services are quite useful as they are used in the matching process between the received query and the available services.

• A database named Process DB which its role is the maintenance of the predecessor-successor relations dealing with the services that the peer provides.

The master peers and backup peers consist of three new components in the storage:

a) A database named Peer DB which stores the peers that have services of the same domain. Actually, it has the role of the service registry of a specific domain.

b) A database named Master DB which maintains the master peers and backup peers of the other domains. Every peer needs this database in order to continue the procedure of the discovery of the service which will satisfy the query's requirements. The query received by a peer cannot always be computed by its provided services. This fact results in the transmitting of the query to the appropriate master peers which share the same domain as the query.

c) A database named Query DB which stores every query that has been answered by the peer's services. The answer is also maintained with the query, something that offers the ability to use again the answer of a same query, either simple or composed, to compute other queries.

## 5.5 Challenges and Future Work

Nowadays, it is noticeable that semantic web becomes popular as it brings solutions to the problem of the sufficient management of the huge information provided in the web. Semantic web promises highly automated processes for the management and the integration of data. These innovations have to be encoded in machine-understandable descriptions and relations between them in order to form a Web of Sense. Also Semantic Web utilizes concept of ontologies to provide interoperability between intelligent systems that function in the Web of semantically described resources.

Research community has also great interest to p2p communication model since its characteristics are suitable for current information systems. P2p enables a total decentralization in the communications between systems, providing scalability and fault tolerance.

The improvement of information sharing is based on the necessity of organizing data into terms of the semantic web. The description of service's characteristics - e.g. its type, its inputs, its outputs - on a common ontology base is the desired goal for turning the integration processes to be automated.

AXML with the contribution of OWL-S, as presented in this work, offers a powerful tool for data integration. AXML on its own provide a variety of integration processes related with data storage and calculations in a decentralized environment having AXML documents as its exchange mean. Moreover, the introduction of implicit calls extends the current framework's capabilities as it enables dynamic data source discovery and dynamic data retrieval following an ontology-based approach.

Furthermore, this work focuses on the addition of some modules in the AXML system in order to enhance the expressivity of services description and to adapt integration processes into p2p environment. Firstly, OWL-S is used to annotate the query in the implicit service call while, secondly, the use of p2p composition service with the exact definition of the role among the involved peers provides with a significant solution the area of dynamic composition of data. Inputs and outputs are taken into consideration in the discovery and composition process where the specification of preconditions and effects in the implicit call should be made in future in order to improve accuracy in data obtaining. Moreover, an issue that need to be examined is the confrontation of the lack of ontological precision and ontological drift, a fact that relies on the constant maintenance of ontologies and the absence of updating mechanisms of their definitions.

AXML as an infrastructure suitable for distributed p2p environments sets its presence in the web services world. Having as its base the documents and the enrichment of them through the firing of the service calls that they have, it can be described as service oriented. This is a result of the importance that services play in AXML because the way that they are described, the time of their firing affect crucially the data that are obtained and their usage, enabling AXML to become the field of adapting a variety of services issues approaches.

This work is a full critical proposal of advancing the capabilities of AXML framework. The provided features through its first implementation from the GEMO team of INRIA institute has established the scene while this work goes one step further the current architecture in order to be able to provide implicit calls and

therefore a composition process through them. Although this work is a step forward in services composition, as it is applied with a new framework with extra characteristics in the management of data, there is still some distance to cover in order to succeed in proceeding to a fully automated composition. The use of OWL-S for the description of web services and extensions in planning related technologies, the desired goal will be closer in the near future.

# Bibliography

[1] Abiteboul, S. "Managing an XML warehouse in a P2P context" *Int'l Conference on Advanced Information Systems Engineering*, 2003

[2] Abiteboul, S., Benjelloun, O., Manolescu, I., Milo, T., Weber, R. "Active XML: Peer-to-Peer Data and Web Services Integration (demo)", *Proc. of VLDB*, 2002

[3] Abiteboul, S., Benjelloun, O. et al. "Active XML: A Data-Centric Perspective on Web Services", *BDA '02*, 2002

[4] Abiteboul, S., Benjelloun, O., Milo, T. "Positive Active XML", *Proc. of ACM PODS*, 2004.

[5] Abiteboul, S., Benjelloun, O., Cautis, B., Manolescu, I., Milo, T., Preda,N. "Lazy query evaluation for Active XML", *Proceedings of the ACM SIGMOD Conference on Management of Data,* 227–238, 2004

[6] Abiteboul, S., Cobena, G., Nguyen, B., Poggi, A. "Construction and Maintenance of a Set of Pages of Interest (SPIN)", *Conference on Bases de Donnees Avancees,* 2002

[7] Abiteboul, S., Manolescu, I., Taropa, E. "A framework for distributed xml data management", *EDBT*, 1049–1058, 2006

[8] Alevizou, V., Plexousakis, D "Enhanced Specifications for Web Service Composition" *ECOWS 2006: 223-232*

[9] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S. (Editor), Trickovic, I., Weerawarana, S. "Business Process Execution Language for Web Services", Version 1.1, 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[10] Arkin, A., Business Process Modeling Language (BPML), Working Draft 0.4, 2001, http://www.bpmi.org/

[11] Benatallah, B. et al., "The self-serv environment for web services composition," *Internet Computing, vol. 7, no. 1*, 2003

[12] Berners-Lee, T., Hendler, J., and Lassila, O. "The Semantic Web", *Scientific American*, 284(5):34 – 43, 2003

[13] Bernstein, A. and Klein, M. "Discovering Services: Towards High Precision Service Retrieval", *Proceedings of the CaiSE workshop on Web Services, e-*

*Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications*, Toronto, Canada, 2002

[14] Borst, W. "Construction of Engineering Ontologies", *PhD thesis*, University of Tweente, Enschede, NL–Centre for Telematica and Information Technology

[15] Cardoso J, Sheth A. "Semantic e-Workflow Composition", *Journal of Intelligent Information Systems (JIIS)*,2002

[16] Constantinescu, I., Faltings, B., Binder, W. "Large scale, type-compatible service composition", *Proceedings of the 2nd International Conference on Web Services (ICWS)*. San Diego, CA, USA, 2004

[17] Corcho, O., Fernandez-Lopez, M., and Gomez-Perez, A. "Methodologies, tools and languages for building ontologies: Where is the meeting point?", *Data and Knowledge Engineering*, 46(1):41 – 46, 2003

[18] Cremarenco, C. "Implementation of the active XML peer for the J2ME platform", *Internship report*, 2003

[19] Deborah L. McGuinness and Frank van Harmelen. "OWL Web Ontology Language Overview", *World Wide Web Consortium (W3C) Candidate Recommendation*, 2003. http://www.w3.org/TR/owl-features/

[20] Di Noia, T., Di Sciacio, E., Donini, M.F. and Mongiello, M. "Semantic Matchmaking in a P-2-P Electronic Marketplace", *SAC 2003*, pp. 582-586, 2003

[21] Fensel, D, Bussler C, Ding Y, Omelayenko B "The Web Service Modeling Framework WSMF", *Electronic Commerce Research and Applications,* 2002

[22] Fensel, D., Decker, S., Erdmann, M., and Studer, R. "Ontobroker: Or How to Enable Intelligent Access to the WWW", In *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based System Workshop (KAW98)*. Banff, Kanada, 1998

[23] Glushko, R. J., McGrath, T., "Document Engineering: analyzing and designing the semantics of Business Service Networks**"** *Proceedings of the IEEE EEE05 international workshop on Business services networks,* 2005

[24] Gruber, T. "A Translation Approach to Portable Ontology Specifications" *Knowledge Acquisition*, 5(2):199 – 220, 1993

[25] Horrocks, I., Patel-Schneider, P., and van Harmelen, F. "From SHIQ and RDF to OWL: The Making of a Web Ontology Language", *Journal of Web Semantics*, 1(1):7 – 26, 2003

[26] Juanzi, L., Bin, X., Wenjun, X., Dewei, C., Po, Z., Kehong, W. "Sewsip:semantic based web services integration in p2p," *IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*, 2005

[27] Leymann, F., Web Service Flow Language (WSFL 1.0), 2001, http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[28] Li, L., Horrocks, I. "A Software Framework for Matchmaking Based on Semantic Web Technology", In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 331-339, ACM, 2003

[29] Luke, S., Spector, L., and Rager, D. "Ontology-Based Knowledge Discovery on the World Wide Web". In *AAAI96 Workshop on Internet-based Information Systems*, 1996

[30] Malone, T. W., Crowston, K., Jintae Lee, B.P., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C.S., Bernstein, A., Herman, G., Klein, M. and O'Donnell E. "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes", *Management Science*, 45(3):425--443, March, 1997

[31] Mandell, D., McIlraith, S. "Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation", *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, pp. 227–241. Sanibel Island, FL, USA, October 2003, http://projects.semwebcentral.org

[32] Masuoka, R., Labrou, Y., Parsia, B., et al. "Ontology-enabled pervasive computing applications", *IEEE Intell. Syst. 18(10)*, 68–72, 2003

[33] McBride, B. "The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS", In *(Staab and Studer, 2004),* pages 51 – 66, 2004

[34] McDermott, D., Dou, D. "Representing Disjunction and Quantifiers in RDF" *Proceedings of the First International Semantic Web Conference* (ISWC2002), 2002

[35] McGuinness, D.L., Da Silva, P.P. "Explaining answers from the semantic web: The inference web approach", *Journal of Web Semantics 1(4)*, 397–413, 2004

[36] McGuinness, D.L., Mandell, D., Mcllraith, S., et al. "Explainable semantic discovery services" *Stanford Networking Research Center Project Review*, Stanford. CA, USA, 2005

[37] McIlraith S, Son T "Adapting golog for composition of semantic Web services", *Proc KRR* 482–493, 2002

[38] McIlraith S, Son T, Zeng H "Semantic Web services", *IEEE Intel Sys March/April*, 2001

[39] Motta, E., Domingue, J., Cabral, L., and Gaspari, M. "IRSII: A Framework and Infrastructure for Semantic Web Services", In *(Fensel et al., 2003),* pages 306 – 318, 2003.

[40] Paolucci, M., Ankolekar, A., Srinivasan, N. and Sycara, K. "The DAML-S Virtual Machine". In *Proceedings of the Second International Semantic Web Conference* (ISWC 2003), pp 335-350, 2003

[41] Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. "Importing the Semantic Web in UDDI", In *Proceedings of E-Services and the Semantic Web (ESSW02)*, 2002

[42] Ruberg, G., Manolescu, I. "Towards cost-based optimization for data-intensive web service computations", *SBBD 2004*, 283–297, 2004

[43] Schlosser, M., Sintek, M., Deker, S. and Nejdel, W. "A scalable and ontology-based p2p infrastructure for semantic web services," *2nd International conference on P2P Computing (P2P'02)*, 2002

[44] Schmidt, C., Parashar, M., "A peer-to-peer approach to web service discovery," *WWW Journal, vol. 7, no. 2*, 2004

[45] Sirin, E., Parsia, B., Hendler, J. "Filtering and selecting semantic web services with interactive composition techniques" *IEEE Intell. Syst. 19(4)*, 42–49, 2004

[46] Sivashanmugam K., "The METEOR-S Framework for Semantic Web Process Composition", *Master Thesis, Computer Science. University of Georgia*, 2003

[47] Sivashanmugam, K., Verma, K., Mulye, R. and Zhong, Z. "Speed-r: Semantic p2p environment for diverse web services registries,"

http://webster.cs.uga.edu/ mulye/SemEnt/final.html

[48] Srivastava B, Koehler J "Web Service Composition – current solutions and open problems". *ICAPS 2003 Workshop on Planning for Web Services*, Trento, Italy, 2003

[49] Sycara, K. et al., "Automated Discovery, Interaction and Composition of Semantic Web Services," *J. Web Semantics*, vol. 1, no. 1, pp. 27–46, 2003

[50] Tarora, E., "Services Continus en Active XML" *Ecole Polytechnique Promotion X2000 Raport de Stage d'option Scientifique*, 2004

[51] Thatte S., "XLANG: Web Services For Business Process Design", *Microsoft Corporation*, 2001
http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[52] The Amazon Web Services, http://www.amazon.com/webservices

[53] The AXML project, www.activexml.net

[54] The Active XML team "Active XML User's Guide"
http://www.activexml.net/AXML%20Guide.pdf

[55] The Business Process with BPEL4WS, http://www.ibm.com/developerworks/edu/ws-dw-ws-bpelws-i.html

[56] The DARPA Agent Markup Language (DAML), http://www.daml.org/

[57] The Extensible Markup Language (XML), http://www.w3.org/XML/

[58] The OWL-S Coalition. OWL-S 1.0 Release, http://www.daml.org/services/owl-s/1.0/

[59] The OWL-S editor, http://owlseditor.semwebcentral.org/

[60] The OWL-S/UDDI Matchmaker, http://www.daml.ri.cmu.edu/matchmaker/

[61] The Rule Markup Initiative, http://www.dfki.uni-kl.de/ruleml/

[62] The SOAP Specification, version 1.2, http://www.w3.org/TR/soap12/

[63] The W3C Web Services Activity, http://www.w3.org/2002/ws/

[64] The Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl/

[65] The Web Service Modeling Ontology, http://www.wsmo.org/

[66] Universal Description, Discovery and Integration of Web Services (UDDI), http://www.uddi.org/

[67] XML Typing Language (XML Schema), http://www.w3.org/XML/Schema