MATCHMAKING IN SEMANTICALLY ENHANCED WEB SERVICES: INDUCTIVE RANKING METHODOLOGY

by

Selim Özyılmaz

B.S., Computer Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2008

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Dr. Ayşe Başar Bener for her expert guidance, constant encouragement and enduring patience during my graduate research. I am lucky to work in her group and have been surrounded by knowledgeable and helpful co-workers. I am grateful to Gökay Burak Akkuş for our wonderful cooperation and all the interesting discussions about Semantic Web Services throughout the preparation of this thesis.

Secondly, I would like to thank my colleagues in the Web Services Research Group for their discussions, feedback and sharing of knowledge about the concepts, we have been working on. I have learned a lot from them. I found the right direction of research with their help.

Finally many thanks to my beloved family for giving me motivation for my study. I always felt their support, courage, endless love, and belief on my success.

ABSTRACT

MATCHMAKING IN SEMANTICALLY ENHANCED WEB SERVICES: INDUCTIVE RANKING METHODOLOGY

Information is the most valuable asset of today's world, both from the perspective of industry and academia. Accumulation of information on web based mediums with increasing demand of integration between different but necessarily collaborating parties drives today's information management policies be service-driven. Semantic enhancements on web services take one more step ahead in this amazing evolution and improve integration between different applications besides removing concerns on scalability. Each emerging technology is a result of a trade-off and each trade-off yields new technologies to emerge. Once semantic web services become widely accepted, discovering the most appropriate and desired service becomes a challenging issue.

Discovery is one of the key steps in the integration of applications, not only because it figures out who the companion in the interaction is but also it forms up the basis step for composition. Once discovery of web services can be successfully managed, composition of services will yield more successful results. To identify which service is the most appropriate for given user requirements, there has to be a valid, efficient and sensitive scoring mechanism to measure appropriateness.

In this research, we focus on ranking of semantically enhanced web services. We base our strategy on individual relations between different concepts of ontology and identify the similarity degree with appropriate combination of those individual relations. We run our algorithm as well as other approaches addressing the same problem on a sample set of semantic web services and observe improvements on service discovery performance.

ÖZET

ANLAMSAL WEB SERVİSLERİNİN EŞLEŞTİRİLMESİ: TÜMEVARAN SIRALANDIRMA YAKLAŞIMI

Günümüz dünyasında bilgi hem endüstri ve hem de akademik dünyanın gözünde en önemli değer konumundadır. Bilginin web tabanlı ortamlarda birikmesi ve farklı ancak birlikte hareket etme zorunluluğu olan uygulamaların entegrasyonu için artan talep, günümüz bilgi yönetim politikalarını servis tabanlı olmaya itmektedir. Web servislerinde sağlanan anlamsal ilerlemeler bu baş döndürücü değişim sürecinde yeni bir adım olmuş ve farklı uygulamaların entegrasyonunda ilerleme sağladığı gibi ölçeklenebilirlik konusundaki endişeleri azaltmayı da başarmıştır. Gelişen her teknoloji önümüze yeni bir açmaz sunarken, her yeni açmaz da yeni teknolojilerin gelişmesine yol açar. Anlamsal web servisleri genel kabül gördükçe en uygun ve istenilen web servisinin keşfi yeni bir zorluk olarak karşımıza çıkmaktadır.

Web servislerinin keşfi farklı uygulamaların entegrasyonu sürecinde etkileşimin yapılacağı eşin bulunması açısından kilit bir adım olduğu gibi derlenme sürecindeki temel adımı oluşturması açısından da önem arz eder. Keşif ne kadar başarılı gerçekleştirilirse derlenme de o derece başarı ile neticelenecektir. Verilen gereksinimlere göre en uygun servisi bulmak doğru, verimli ve hassas bir skorlama mekanizması gerektirmektedir.

Bu çalışmamızda anlamsal web servislerinin derecelendirilmesini konusunu ele aldık. Temel stratejimiz konseptler arası tekil ilişkileri baz alan ve farklı konseptler arası ilişkiyi işbu tekil ilişkiler yoluyla elde eden bir benzerlik kıstası yaratmak oldu. Bu yaklaşımı aynı problemi adresleyen diğer algoritmalarla karşılaştırmak için örnek bir servis kümesi üzerinde test ettik ve keşif sürecindeki ilerlemeleri gözlemledik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programmers Interface
DAML	Darpa Agent Markup Language
DAML-S	Darpa Agent Markup Language for Services
DARPA	Defense Advanced Research Projects Agency
HTTP	HyperText Transfer Protocol
IRS	Internet Reasoning Service
ITL	Information Terminological Language
J2EE	Java 2 Platform, Enterprise Edition
KB	Knowledge Base
MWSDI	Meteor-S Web Service Discovery Infrastructure
OWL	Ontological Web Language
OWL-S	Ontological Web Language for Web Services
RDF	Resource Description Framework
RDF-S	Resource Description Framework for Services
SOAP	Simple Object Access Protocol
SW	Semantic Web
SWS	Semantic Web Services
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Services Description Language
WSMF	Web Services Modeling Framework
WSMO	Web Services Modeling Ontology
XML	Extensible Markup Language

1. INTRODUCTION

The paradigm of distributed computing is one of the major topics of 80's with the emergence of computer networks [73]. Applications were first split into two parts with one part, the client, initiating a distributed activity, and the other part, the server, carrying out that activity. This decentralization minimized bottlenecks by distributing the workload across multiple systems. It provided flexibility to application design formerly unknown on centralized hosts. But after some time, limitations of this two-tier architecture became observable [73].

For failover and scalability issues a third tier was introduced, separating an application into a presentation part, a middle tier containing the business logic, and a third tier dealing with the data. This three-tier model of distribution has become the most popular way of splitting applications. It makes application systems scalable. The foundation for the communication between the distributed parts of an application is the remote procedure call (RPC). To keep developers from the low-level tasks like data conversion or the byte order of different machines, a new layer of software has been used in the market. This middleware masks the differences between various kinds of hosts. It sits on top of the host's operating system and networking services and offers its services to the applications above. The first middle-wares, like DCE, were based on a procedural programming model and were superseded by the introduction of the object oriented programming model by middle-wares like CORBA,® DCOM, or RMI, which are the most popular middle-wares of today [73].

The problem with this architecture is that all tree middle-wares result in tight coupling of the client and the server. Because of the different protocols, a DCOM server can not be called from a RMI client. In addition, these middle-wares are typically used for intranet applications, and it is often an issue to cross a firewall. All of them provide HTTP tunneling to get to the server behind the firewall [73].

1.1. Motivation

As companies move from simply monitoring to proactively managing business performance, they need real-time visibility in the market, customer and competitive environment [74]. This requires integration across IT systems and business processes in a cost-effective and flexible manner. Web services technology promises to facilitate this by replacing proprietary interfaces and data formats with a standard web-messaging infrastructure. Web services have the potential to dramatically improve collaboration between customers, suppliers, partners and employees by providing a dynamic and flexible way for individuals from different companies to be active participants in the decision making process [75]. This collaboration goes far beyond simply sharing information through a Web browser. Web services help maximize the value of Business Intelligence (BI) by separating the presentation layer from the application logic layer so that BI can be embedded into business processes [73]. Web services also have the potential to lower BI development and deployment costs. For example, programmers can use Visual Basic, C, or Java to develop libraries of reusable .Net and J2EE Web services components that can be quickly assembled into solutions in a seamless and painless manner.

With common acceptance of web services maintenance of web tools becomes a major issue and trend in dealing with application protocols shift from syntactic to semantic level [77]. Data structure and sequencing information are enhanced with semantic information that encodes the definition of each data element. Key elements enabling this shift from a purely syntactic to a semantic interoperability are ontologies. They are semantic models of data and they interweave human understanding of symbols with their machine processing capabilities [77]. Besides ontology languages, semantic interoperability requires a conceptual and formal model for services. A proposal mainly driven by US based research is called OWL-S. It essentially provides an upper ontology encoded in Description Logics to capture the semantics of a service [11].

Semantic web services has great potential in business process management because of its valuable properties such as automated discovery, composition and execution of web services based on richer semantic descriptions on functional, non-functional and behavioral aspects [77]. Ontologies allow to model processes as well as data shared conceptual model.

Their formalization allows semantically enhancing information processing. Semantic annotations of web services allow to precisely detect and automatically execute suitable business functionalities as well as to maintain them [76].

Our research in this thesis focuses on ranking of semantic web services. Ranking is important in terms of discovery, finding a service that meets user requirements exactly in the same way the user wants. Discovery, on the other hand, has direct implications on application of semantic web services. Each composition process includes service discovery at each step. Therefore a fail in one link may result in total failure for the whole chain.

Ranking by itself can take into account many variables: usage patterns, client ratings, number of links given to the web service, etc. This work is intended to identify the similarity between what a service provides and what is requested by the user based on the ontology structure. There are several different approaches on this issue that are based on predefined multipliers however; our work considers concept similarity in terms of property comparisons in a well defined hierarchy of concepts so that degree of similarity is identified as objective as possible. Our proposed solution tries to figure out this degree by accumulating the similarity information of individual subsumption relations. Therefore we find it appropriate to name our work as “Inductive Ranking Methodology”.

1.2. Outline

The remainder of this thesis is organized as follows: in Chapter 2 we have explained web services and related standards in detail with semantic web extensions and examined inheritance both from object oriented and ontological point of view. Chapter 3 states current state of the art approaches for ranking semantic web services and addresses corresponding problems faced in practice. Chapter 4 details our approach to solve those problems and Chapter 5 gives information about the experiments that are performed to compare the performance of our proposed solution. Finally, Chapter 6 states our findings and finishes with future work of this research.

2. WEB SERVICES & SEMANTIC WEB

2.1. Introduction to Web Services

The concept of Web services has garnered a lot of attention and interest in software industry because of their potential to provide seamless application interoperability. Different organizations define web services differently. IBM defines Web services as "*Web services are a new breed of web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.*" [39]

A Web Service is a software entity provided by a Service Provider. It performs an action (based on inputs) on behalf of a Service Requestor and provides a result (output). An example of a service is the use of a web search engine: when a user (service requestor) types a query (input) into the search engine (service provider), the search engine tries to find relevant web sites (action) and returns the findings to the user (output).

A characteristic of a service is that it has properties, e.g. the action it performs, which are in- and outputs it supports and who performs the action [17]. These types of properties are called functional properties. Furthermore, a service also has non-functional properties, e.g. quality of service, cost, performance and security. Functional and non-functional properties characterize the service. Therefore, they can be used to describe a service.

The operational lifecycle of a service consists of three successive phases: Advertisement, Discovery and Delivery [17, 18]. In the advertisement phase, the service provider creates a service description based on the service properties. This description is used to advertise (i.e. enable the service for use) the service. In the discovery phase, the requestor tries to find (i.e. manual or automatic discovery) a service that satisfies his need. When a service is found, it is provided in the delivery phase (Figure 2.1).

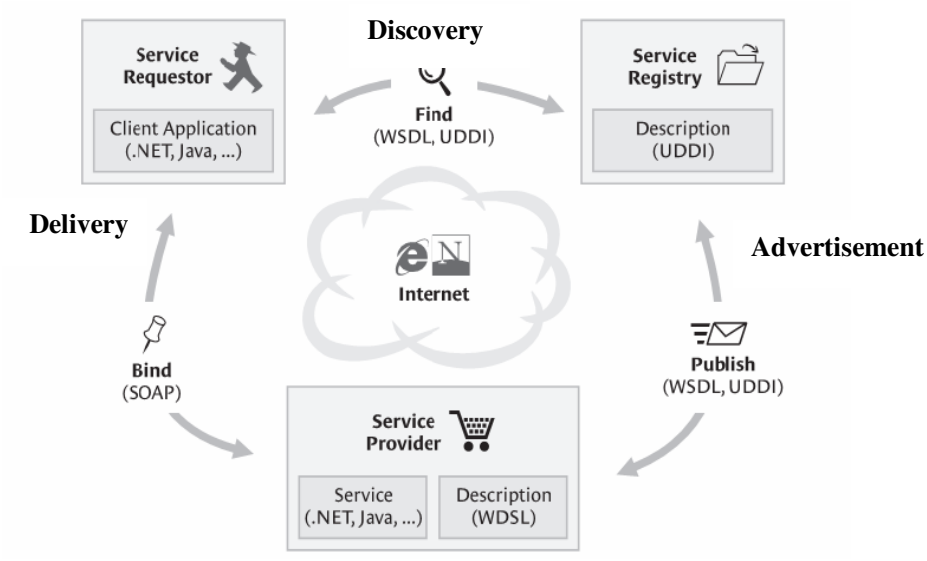

Figure 2.1. Service Lifecycle [35]

Before the delivery phase, the service requestor and service provider are unassociated with each other (may be even not aware of each other's existence). The service provider creates a service and advertises it to the world. The requestor tries to find some service it desires. With the service discovery the service requestor and some service provider are associated (Figure 2.1) with each other and both can participate in the delivery phase [79].

2.2. XML

After having outlined the web service architecture, we shall focus on the underlying technologies: XML and XML based SOAP, WSDL and UDDI.

2.2.1. XML as the Founding Technology of Web Services

The XML specification defines XML as follows: "*Extensible Markup Language, abbreviated as XML, describes a class of data objects as XML documents and partially describes the computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language.*"[19]. This

definition is not particularly practical since it depends on a different specification. Walsh gives simpler phrasing: "*XML is a markup language for structured documentation*" [20].

XML documents are electronic documents that consist of storage units called entities, which are composed of character data and so called markup [35]. The character data represents the content of the document, while the markup describes the layout and logical structure of the document and consists of declarations, elements (enclosed in opening and closing tags), comments, character references and processing instructions.

2.2.2. Structure of XML Documents

Figure 2.2 displays two sample documents that can be found in business and private applications.

From : Michael
To: Adam
Re: Confirmation

I have received your resume

ORDER
2007/07/07
Customer: Selim Ozyilmaz

Table	1x$50	$50
Chair	4x$15	$60
Total		$110

Figure 2.2. Sample Documents: A Note and an Order Receipt [35]

Figure 2.3 provides an example of a very simple XML document for the note in Figure 2.2. An XML document contains one or more elements, each of which is demarcated using tags. By assigning tags to portions of the document the semantic structure of the document becomes visible. With such structure portions of the document becomes easily processable by computer applications.

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<note>
<to>Michael</to>
<from>Adam</from>
<heading>Confirmation</heading>
<body>I have received your resume</body>
</note>
```

Figure 2.3. Note as an XML Document [35]

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<order>
        <date>2007/07/07</date>
        <customer>Selim Ozyilmaz</customer>
        <items>
                <item no=”1” price=”50” qty=”1”>Table</item>
                <item no=”2” price=”15” qty=”4”>Chair</item>
        </items>
        <due sum=”110” />
</order>
```

Figure 2.4. The Order Receipt as XML Document [35]

The XML document starts with a so-called processing instruction <?xml version=”1.0” encoding=”UTF-8” ?>. The instruction contains information for the application that parses the document. In this case the processing instruction indicates the version of the XML standard and the character encoding used [39].

What follows the instruction is a tag <order>. Tags are words enclosed in angle brackets; they are XML’s markup codes that indicate the structure of the document. The opening tag has a corresponding closing tag at the end of the document. The portion of the document that is demarcated by opening and closing tags is called an element.

Each XML document must contain exactly one root element (here: <order>). Elements that contain other elements are called parent elements; the elements that are contained by the parent elements are referred as the child elements. Elements within a document form up a tree structure. Element names are case sensitive and may consist of letters, digits and a few special characters. The text between the opening and closing tags are called the content or character data. If the content represents non-textual data such as

graphics, is usually is encoded so that byte sequences can be converted into letters and digits.

Attributes are used to attach additional information to elements. Attributes are key/value pairs intended to describe various aspects of the element in which they appear. The start tag of an element can contain any number of attributes in arbitrary order, as long as each has a unique name. Attributes are separated by spaces.

In addition to explicit text, the content of the XML document can reference so-called entities [39]. These are portions of text that are defined elsewhere (typically in the DTD document) under a specific name. Whenever a particular text should be used in the document an entity reference is inserted. Entity references are names preceded by ampersand (&) and followed by a semicolon (;).

Because XML allows designers to choose their own tag names, it is possible that two or more designers may choose the same tag names for some or all their elements. A vocabulary of an XML document, set of element names used by a particular author and for particular purpose can be denoted by defining a namespace. A namespace is identified with a reference to an URI (Uniform Resource Identifier). Typically such URI has a form similar to URLs used in web addresses as depicted in Figure 2.5.

```
<math xmlns: m=http://www.w3.org/1998/MathXML
<semantics>
        <mrow>
<!--  or  -->
<m:math>
        <m:semantics>
                <m:row>
```

Figure 2.5. Example Usage of XML Namespaces [35]

2.2.3. Defining XML Grammar: DTD and XML Schema

Each type of document with a defined purpose has some formal constraints [41]. For example a purchase confirmation issued by a car company usually contains the name of the customer, the address of the store and model of the car with prices. Each special purpose XML document should reference a definition of its grammar so that an application can

validate the structure of such document. The grammar of XML documents is defined using the DTD (Document Type Definition) or XSD (XML Schema Definition) languages. XML documents that obey the grammar are called valid [41].

```
<?xml version=”1.0” encoding=”utf-8” ?>
<xsd:schema attributeFromDefault=”unqualified”
elementFormDefault=”qualified” version=”1.0”
xmlns:xsd=http://www.w3.org/2001/XMLSchema>
        <xsd:element name=”note”>
                <xsd:complexType>
                        <xsd:element name=”body” type=”xsd:string” />
                        <xsd:attribute name=”to” type=”xsd:string” />
                        <xsd:attribute name=”from” type=”xsd:string” />
                        <xsd:attribute name=”heading” type=”xsd:string” />
                </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

Figure 2.6. Sample XML Schema that describes the Note [35]

The W3 Consortium developed the XML Schema Language that was intended as a replacement for DTD. While DTD uses a document syntax that is not XML-conformant, the Schemas are expressed using XML [42]. Documents can reference a Schema document that describes their structure and imposes constraints as for the element sequence and inclusion, data types that can be used as attributes and content within particular elements, and the cardinality of elements and attributes.

The relation between a particular XML document and an XML Schema is analogical to the correlation between a class instance and a class in object oriented programming, so XML documents that conform to a particular schema are often called instances of that schema. Figure 2.6 presents a sample schema for the note that is mentioned in Figure 2.2.

2.3. SOAP

Core features of web services – interaction and communication – are facilitated using the Simple Object Access Protocol [22]. It is a lightweight XML-based protocol (XML dialect) used for exchange of information in a distributed environment [21]. SOAP uses existing transport protocols such as HTTP or SMTP to provide a standard way of packing messages and to invoke remote procedure calls (RPCs).

SOAP documents are called messages. A SOAP message is an XML document composed of exactly one root Envelope element that must contain the body of the message and may additionally contain any header information used to describe the message. In addition, SOAP protocol defines rules for expressing user defined data types, as well as conventions for representing remote procedure calls and responses [21]. A SOAP message can also list so called actors that indicate the recipients of the entire message or of its parts.

```
POST http://brinkster.com/twardoch/wsSquare/wsSquare.asmx HTTP/1.1
Host: eu.webmatrixhosting.net
Content-Type: text/xml; charset=utf-8 Content-Length: 299
SOAPAction : http://www.twardoch.com/ws-namespace/Square

<?xml version=”1.0” encoding=”utf-8” ?>
<soap:Envelope  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xmlns:xsd = “http://www.w3.org/2001/XMLSchema”
xmlsn:soap= http://schemas.xmlsoap.org/soap/envelope />
        <soap:Header>
            <sender>
                John Smith
            </sender>
        </soap:Header>
        <soap:Body>
            <Square xmlns=”http://www.twardoch.com/ws-namespace”>
                <x>25</x>
            </Square>
        </soap:Body>
</soap:Envelope>
```

Figure 2.7. Sample SOAP Message [35]

Figure 2.7 shows a sample SOAP message that is intended to be sent to a web service called wsSquare. The service takes one argument x of type integer and returns its square. It should be noted that the SOAP message is packaged within a HTTP POST request. Figure 2.8 displays the response of the web service to the requester.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 381

<?xml version=”1.0” encoding=”utf-8” ?>
<soap:Envelope  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xmlns:xsd = “http://www.w3.org/2001/XMLSchema”
xmlsn:soap= http://schemas.xmlsoap.org/soap/envelope />
        <soap:Body>
            <SquareResponse xmlns=”http://www.twardoch.com/ws-namespace”>
                <SquareResult>625</SquareResult>
            </SquareResponse>
        </soap:Body>
</soap:Envelope>
```

Figure 2.8. Sample Web Service Response in SOAP [35]

SOAP messages sent in realistic business-to-business scenarios include a complex body and numerous headers indicating the sender's credentials, authentication data etc. [23]. An optional element of SOAP message is the Fault element for error processing purposes. It can contain an error code and its description [24].

2.4. WSDL

While SOAP provides a mechanism for exchanging messages, another XML based language WSDL (Web Services Description Language) is responsible for describing the interfaces to web services so that applications know what messages can be sent to a service and what responses can be expected [23]. One of the aims of WSDL is information hiding. "*WSDL enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as 'how' and 'where' that functionality is offered*." [23]. A WSDL service description provides two parts of information: an abstract service description as well as some specific protocol-dependent details.

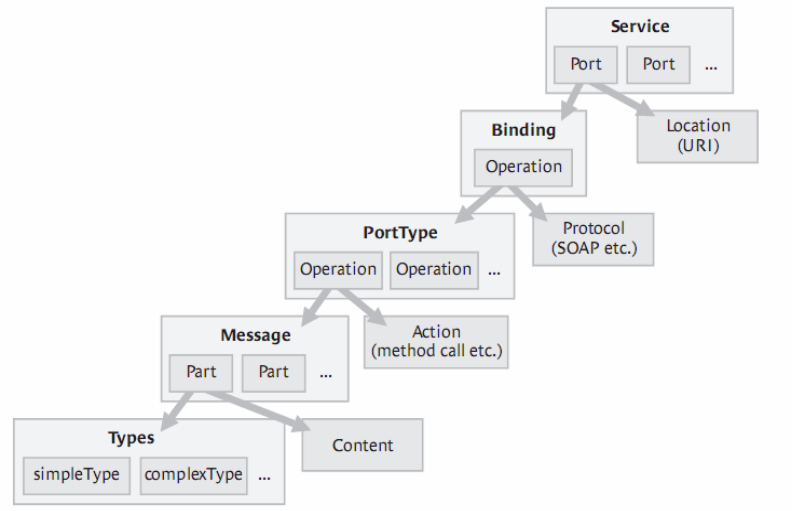

Figure 2.9. Schematic View of WSDL Document Syntax [35]

Figure 2.9 and 2.10 give a schematic view of the interdependencies within a WSDL document. The abstract definition of the interface that can be used to exchange messages with the service consists of three main components: the vocabulary, the message and the interaction.

```
<?xml version=”1.0” encoding=”utf-8” ?>
<definitons xmlns=”http://schemas.xmlsoap.org/wsdl/”
            xmlns:soap=” http://schemas.xmlsoap.org/wsdl/soap”
            xmlns:s=” http://www.w3.org/2001/XMLSchema”
            xmlns:s0=”http://www.twardoch.com/ws-namespace” >
        <types>
                <s:schema>
                        <s:element name=”Square”>
                                <s:complexType>
                                        <s:element name=”x” type=”s:int”>
                                </s:complexType>
                        </s:element>
                        <s:element name=”SquareResponse”>
                                <s:complexType>
                                        <s:element name=”SquareResult” type=”s:int”>
                                </s:complexType>
                        </s:element>
                </s:schema>
        </types>
```

Figure 2.10. Sample WSDL Document: Types Vocabulary Section [35]

The vocabulary uses the XML Schema to describe the data types within messages. In the example Square and SquareResponse that consist of a complexType definition are exhibited.

Further, messages provide an aggregated definition of data types sent to and from the service. In Figure 2.11 two messages are being defined.

```
<message name="SquareSoapIn">
        <part name="parameters" element="s0:Square" />
</message>
<message name="SquareSoapOut">
        <part name="parameters" element="s0:SquareResponse" />
</message>
```

Figure 2.11. Sample WSDL Document: Messages Section [35]

Finally interactions are defined by combining messages into operation and portType elements. Each operation is a combination of messages labeled as input, output or fault to show what role each message plays. In our example we have one portType names wsSquareSoap defined that includes one operation named Square. The operation communicates using two messages defined earlier: an input message SquareSoapIn and an output message SquareSoapOut.

```
<portType name="wsSquarePort">
        <operation name="Square">
                <input message="s0:SquareSoapIn" />
                    <output message="s0:SquareSoapOut" />
        </operation>
</portType>
```

Figure 2.12. Sample WSDL Document: Messages Section [35]

A portType is a collection of operations supported by a particular end point (port) [23]. Each portType is associated with a particular protocol by binding. Finally the collection of all ports exposed by the web service is defined as the service. (Figure 2.12)

```
<binding name=”wsSquareSoap” type=”s0:wsSquareSoap”>
        <soap:binding />
        <operation name=”Square”>
                <soap:operation />
                <input>
                        <soap:body />
                </input>
                <output>
                        <soap:body />
                </output>
        </operation>
</binding>
<service name =”wsSquare”>
        <port name=”wsSquareSoap” binding=”s0:wsSquareSoap”>
                <soap:address location=”http://www33.brinkster.com/twardoch/
wsSquare/wsSquare.asmx” />
        </port>
</service>
</definitions>
```

Figure 2.13. Sample WSDL Document: Bindings and Service Selection [35]

2.5. UDDI

Universal Description, Discovery and Integration of Web Services (UDDI) is a well known service discovery mechanism for web services [43]. In 2000, it emerged as cooperation between Ariba, IBM and Microsoft to improve integration of B2B on the internet. At this moment, the development of UDDI is performed by OASIS and versions three of the UDDI specification is released [44].

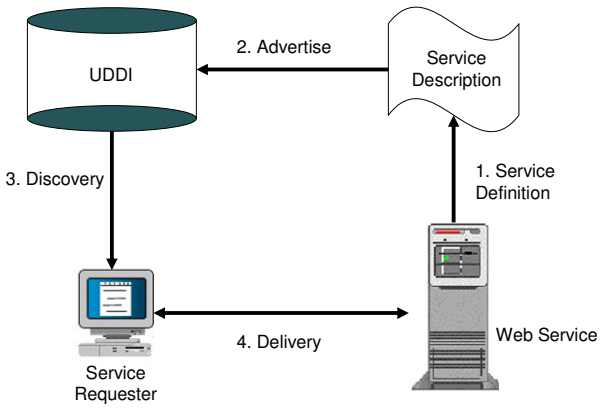

Figure 2.14. UDDI Overview [80]

2.5.1. Introduction to UDDI

The main goal of UDDI is to specify a framework for describing and discovering web services [45]. The core of the UDDI architecture is a central business registry (i.e. centralized discovery model) which functions as a naming and directory service. The service provider advertises his service to this central registry and the service requestor discovers services by querying the central UDDI registry and using the result to associating with the service. Figure 2.14 illustrates the stated architecture [80]. These steps (publishing and querying) are projected in the two major functionalities of UDDI:

(i) UDDI defines data structures and API's for publishing service descriptions.
(ii) UDDI enables the user to query the registry to look for published descriptions.
    a. Enable developers to discover information on certain services to make their clients compatible with these services.
    b. Enable dynamic binding (i.e. retrieve references to services of interest).

The information contained in an UDDI registry can be categorized using a metaphor of a telephone dictionary [45]. The information can be categorized as:

(i) White pages: listing of organizations, contact information and services these organizations provide.
(ii) Yellow pages: classifications of both companies and Web services according to standard or user defined taxonomies (i.e. tModel).
(iii) Green pages: information on how a given Web service can be invoked (e.g. WSDL specification).

This classification is reflected by the four types of information encapsulated by a UDDI registry entry [43, 44, and 45]. These information types are formatted in XML structures. One of these information types is the technical model (tModel) which will be discussed in more detail in the next section.

2.5.2. Technical Model (tModel)

tModels are abstract specifications of knowledge of web service, which is advertised in the UDDI registry. It can be seen as the 'yellow pages' of UDDI. It classifies the business or web service using standard or user-defined taxonomies. The main content of a tModel consists of a unique key and a pointer to the documentation of the service (see Figure 2.15). The documentation can reside anywhere and can be written in any available language. The documentation of the service is typically unstructured data meant for human beings and not automated systems. The human service requestor can browse the UDDI registry (UDDI entries and tModels) and can gain insight in what a certain service does and which properties and interfaces it provides.

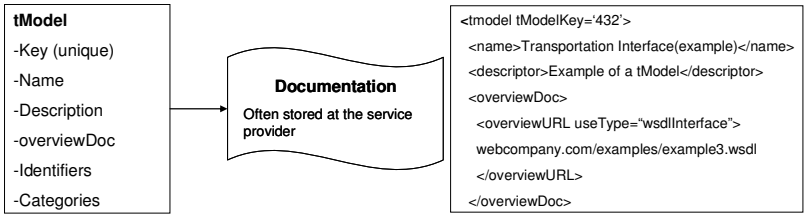

Figure 2.15. tModel [80]

A specified tModel (characterized by its unique key) can be used by multiple services. In this way, it is used to classify services.

2.6. Semantic Web

The World Wide Web emerged as a solution to the information management problem that the CERN2 research institute was facing in the late nineties [47]. The data-intensive research activity performed at CERN produced an increasing amount of digital data (about projects, researchers, software etc.) which was distributed on a network of computers running different operating systems and using particular (often incompatible) data formats. Finding and integrating this distributed and heterogeneous data was increasingly difficult. The Web was a distributed hypertext system that allowed access to heterogeneous data sources by using a set of standards for describing data (HTML),

localizing documents on a network (URI) and accessing them (HTTP). Due to the simplicity of HTML, which allowed creation of Web pages with a minimum effort, Web became a tremendous success in a relatively short time.

With the accelerated growth of Web, finding the right information becomes increasingly difficult even with support from search engines such as Google. Current search technology relies on keyword based search. This technique provides a relatively high recall (as all Web sites that mention a given keyword are retrieved) but a low semantic recall (as pages about the desired topic but not containing the keywords are ignored) [25]. Their precision is low because only few of the retrieved pages contain the information that the user needs. As a result, complex queries such as "A four-door SUV that can go faster than 100 km/h" are hard to answer because (1) such specific information is often not made available as such on the Web or, (2) if it is made available it can take different syntactic forms. A more generic query, "SUV faster than 100 km/h" will retrieve a significant number of hits due to spread distribution of keywords among web pages. On the other hand, enforcing the order of the keywords may drastically decrease the number of hits. The major cause of this instability is that keywords are treated as strings rather than meaningful entities. [80]

As a response to these limitations, Web encountered two revolutionary changes. First, the Semantic Web community aims at a semantic extension of the current syntactic Web by augmenting existing information with logics based formal descriptions of their meaning [6]. This semantic markup would be machine processable and therefore allow easier access and integration of the vast amount of the available information than what can be achieved with keyword based search. Second, Web is changing from a collection of static web pages to a dynamic environment with the advent of the Web services technology that makes software components accessible via Web protocols. The Semantic Web services technology developed at the cross road of these two technologies by applying semantic Web techniques to Web services. [6]

The goal of the Semantic Web is to solve the current limitations of the Web by augmenting Web information with a formal (i.e., machine processable) representation of its meaning [6]. A direct benefit of this machine processable semantics would be the

enhancement and automation of several information management tasks, such as search or data integration. The idea of applying knowledge representation and reasoning techniques in the context of the Web has been investigated from the mid-nineties notably by work on SHOE (Simple HTML Ontology Extensions) [48] and Ontobroker [49]. The Semantic Web term was clearly associated to this line of research in 2001 when it was defined as: "*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.*" [50]

The "well-defined meaning" of information is provided by semantic descriptions, often referred to as metadata (i.e., data about data) [79]. There are two characteristics specific to semantic metadata. First, metadata describes information in terms of a domain vocabulary whose meaning is specified by a formal domain model (ontology). Second, metadata is expressed in a representation language that can be parsed and interpreted by machines. We briefly describe ontologies and ontology languages in what follows.

2.7. Ontologies

The term ontology, originating from Philosophy [51] was adopted by AI researchers to describe formal domain models. Several ontology definitions were provided in the last decades. The most frequently cited definition is that given by Gruber in 1993 according to which ontology is "*an explicit specification of a conceptualization*" [81]. In other words, ontology is a domain model (conceptualization) which is explicitly described (specified). Later, in 1997 Borst defines ontology as a "*formal specification of a shared conceptualization*" [52]. This definition requires, in addition to Gruber's definition, that the conceptualization should express a shared view between several parties, a consensus rather than an individual view. Also, this conceptualization should be expressed in a machine readable format. In 1998, Studer et al. merge these two definitions stating that: "*An ontology is a formal, explicit specification of a shared conceptualization.*" [53]. As consensual domain models, the primary role of ontologies is to enhance communication between humans (e.g., establishing a shared vocabulary, explaining the meaning of the shared terms to reach consensus) [6]. As formal models, ontologies represent knowledge in a computer processable format thus enhancing communication between humans and computer programs or two computer programs.

2.7.1. The Need for Ontologies

SOAP, WSDL, UDDI, and BPEL4WS are the standard combination of technology to build a Web service application. However, they fail to achieve the goals of automation and interoperability because they require humans in the loop [54]. Indeed, WSDL specifies the functionality of the service only at a syntactic level. While these descriptions can be automatically parsed and invoked by machines, the interpretation of their meaning is left for a human programmer. To support reliable, large-scale interconnectivity of Web services by software, computer processable semantics are needed, which include the properties, capabilities, interfaces, and effects of the service [55].

The Semantic Web community addressed the limitations of current Web service technology by augmenting the service descriptions with a semantic layer in order to achieve their automatic discovery, composition, monitoring and execution (56, 57).

A common characteristic of all emerging frameworks for semantic Web service descriptions [58] is that they combine two kinds of ontologies to obtain a service description. First, a generic Web service ontology, such as OWL-S, specifies generic Web service concepts (e.g., Input, Output) and prescribes the backbone of the semantic web service description. Second, domain ontology specifies knowledge in the domain of the Web service, such as types of service parameters (e.g., City) and functionalities that fills in this generic framework. [6]

2.8. Ontology Definition Languages

This section presents the major enabling technologies for ontology-based applications. The discussed technologies are RDF/RDFS, DAML+OIL, and OWL.

2.8.1. RDF / RDF-S

The Resource Description Framework (RDF) specification provides a lightweight ontology system to support the exchange of knowledge on the Web [59]. It is developed by

the W3C consortium and is one of the building blocks of the so-called “Semantic Web” [60].

RDF is also an XML based ontology definition language. While XML only provides mechanisms to create structured data with elements and attributes, RDF also describes the data. It can be used to describe statements like “Tom’s gender is male” as depicted in Figure 2.16.

```
// RDF document describing “Tom’s gender is Male”
<?XML version=”1.0”?>
<rdf:RDF
// general RDF namespace
xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
// namespace points to RDF schema (RDF-S)
xmlns:example=http://www.example.com/exampleschema>
// Described resource is “Tom”
<rdf:Description rdf:about=”Tom”>
//”Tom” has predicate “Gender” which has value “Male”
<example:gender>Male</example:gender>
</rdf:Description>
</rdf:RDF>
```

Figure 2.16. Sample RDF Document [80]

In the described example, the resource that is described in RDF is “Tom”. A predicate of Tom is that he has a gender. The value of his gender is “Male”. These three elements (resource, predicate, object (i.e. value)) form the basic building blocks of an RDF document. By pointing to RDF schemas, the semantics of “gender” is defined. The namespace mechanism (i.e. xmlns, URI) of XML is used to refer to the RDF schemas. RDF schema (RDF-S) is the schema specification of RDF. While RDF is used to describe data, RDF-S is a domain-neutral way of describing the metadata that RDF uses to describe data [61]. This schema can be used to validate a created RDF document.

2.8.2. DAML+OIL

DAML+OIL is a combination of the DARPA Agent Markup Language (DAML) and the Ontology Interface Layer (OIL) [62]. Both initiatives recognized their common goal and in 2000 they merged in the combined initiative DAML+OIL. DAML+OIL extends existing standards (i.e. XML, RDF) to describe the structure of a particular domain in

terms of classes and properties. A DAML+OIL description consists of a set of axioms that asserts relationships between classes or properties (i.e. intersectionOf, unionOf, complementOf, etc). In this way the semantics of the document is captured (Figure 2.17).

```
<rdf:RDF
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
xmlns:example=http://www.example.com/exampleschema>
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
>
// This document defines the ontology
<daml:Ontology rdf:about=""></daml:Ontology>
// Class "Gender"
<daml:Class rdf:ID="Gender"></daml:Class>
// Male is "part of" Gender
<daml:Class rdf:ID="Male">
<rdfs:subClassOf rdf:resource="#Gender"/>
</daml:Class>
// Female is "part of" Gender but a female cannot be a "Male"
<daml:Class rdf:ID="Female">
<rdfs:subClassOf rdf:resource="#Gender"/>
<daml:disjointWith rdf:resource="#Male"/>
</daml:Class>
// Assign properties to classes
<daml:DatatypeProperty rdf:resource="#Gender">
<rdfs:comment>
Gender is a DatatypeProperty whose range is xsd:String.
Gender is also a UniqueProperty (can only have one
gender)
</rdfs:comment>
<rdf:type
rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#decimal"/>
</daml:DatatypeProperty>
```

Figure 2.17. Sample DAML+OIL Document [80]

2.8.3. OWL

"The OWL Web Ontology Language is designed for use with applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics." [63]. OWL is based on DAML+OIL. Generally, it uses the same language structure plus extensions. It is now being proposed as W3C standard for ontology and metadata representation. Just like DAML+OIL, OWL is based on XML and

RDF/RDF-S. It uses XML namespaces and URI's, and furthermore concepts like disjointWith, subclassOf and unionOf are present.

2.8.4. OWL-S

OWL-S recognizes that not only content but also services are offered by Web resources [64]. Users should be able to discover, invoke, compose and monitor these services. OWL-S develops an ontology for services that makes this functionality possible. OWL-S is the successor of DAML-S and is part of the DARPA agent markup language program.

One of the big tasks OWL-S tries to enable is automatic service discovery. This involves the automatic location of web services that provide a particular service that adheres to requested constraints [64]. By using the OWL-S markup of services a common description of services is created that can be matched more easily in the discovery process. Below, the top-level ontology of a service description is presented (Figure 2.18).

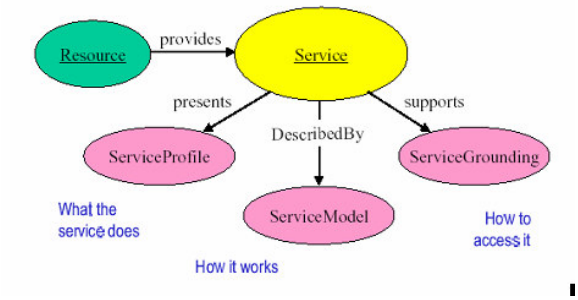

Figure 2.18. Top Level of the Service Ontology [80]

It answers fundamental questions of a service like: what does the service do (ServiceProfile), how does the service work (ServiceModel) and how can the service be accessed (ServiceGrounding) [64]. The service Profile provides a way to describe the service offering by the provider and the needed service by the requestors. An OWL-S profile describes a service as a function of three basic types of information: what organization provides the service (e.g. contact information), what functions does the service computes (required inputs, produced outputs, pre-post conditions), and a set of

features that specify characteristics of the service (e.g. service type, QoS). The service model describes how the service works. This is done by modeling the service as a process (Figure 2.19).

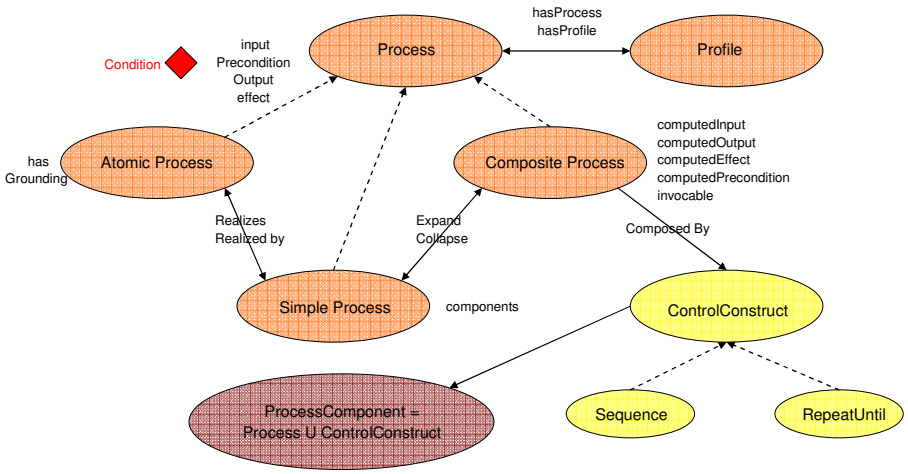

Figure 2.19. Top level of the process ontology [80]

The ServiceGrounding specifies the details on how to access the service. The details mainly describe transportation protocols and message formats, serialization, and addressing. Grounding can be thought of as a mapping from an abstract to a concrete specification of those service description elements that are required for interacting with the service. OWL-S grounding is consistent with the WSDL binding mechanism [11]. By using the extensibility elements already provided by WSDL, along with new extensibility element proposed in OWL-S, it is easy to ground an OWL-S process. In OWL-S, both the ServiceProfile and the ServiceModel are thought of as abstract representations. Only the ServiceGrounding deals with the concrete level of specification [11].

In this research, we focus our attention on concept based retrieval of semantic web services using OWL-S.

3. RELATED WORK

Before going further into the details of our proposed approach, we find it useful for the reader to review the already proposed semantic matchmaking algorithms. Most of the approaches focus on the input and output matching and they do not use the whole set of information that is provided by the ontological representation. Some of the following architectures are limited in size and its capabilities however, each of them represent the key milestones in the sophistication process of semantic matchmaking.

3.1. OWL-S Matchmaker

OWL-S Matchmaker is an entity that allows services to locate other services, provide a solution to semantic matchmaking and implement full interpretation of service providers on the web [87]. OWL-S which is an extension of OWL that describes service capabilities in detail is the most significant output of this work.

Matching process in OWL-S Matchmaker uses a minimization of candidate services by using different levels of constraints for different queries. These filters include namespace comparison, word frequency comparison, ontology similarity matching, ontology subsumption matching and constraint matching [87]. It is the user who selects and configures different levels of filters to attend the desired goal.

The semantic matchmaking filter is simply based on the matching of inputs and outputs with respect to the degree of subsumption in the ontology. This degree is not quantified however, results are categorized as fail, subsume, plug-in and exact in an increasing level of importance [87].

3.2. LARKS

LARKS (Language for Advertisement and Request for Knowledge Sharing) is a former approach that tries to use the application domain information at first place with a local ontology for each domain explored [88]. As in the case of OWL-S Matchmaker it

uses five different filters: context matching, profile comparison, similarity matching, signature matching and constraint matching. Filters in LARKS are used in a progressive manner to fine tune the result. In other words, best result is the one that takes longest time.

LARKS bases matchmaking onto input and output; precondition and effect are not considered [88]. Moreover, it used the ITL (Information Terminological Language) for ontology definition. Usage of ITL is one of the major drawbacks of LARKS since ITL falls out of scope for ontology definition languages.

3.3. Ian Horrocks and Lei Lui

The architecture proposed by Ian Horrocks and Lei Lui is based on DAML-S ontology definition language and Description Logic reasoner. Matchmaking uses inputs and outputs and categorizes the results in five strictly discrete groups: Exact, Plug-in, Subsume, Intersection and Disjoint [89]. This work is not based on OWL-S and it does not count for any partial matches.

3.4. Paolucci et al.

Quite similar with the architecture proposed in OWL-S Matchmaker, Paolucci et al. proposed a new framework for semantic matchmaking where both the requester and the provider use the same ontology [3]. Usage of a common ontology in between the different parties removes the need for different levels of filters. Matchmaking in their proposed architecture also uses input and output as basic elements of comparison. Matching results are categorized into four main groups: Exact, Plug-in, Subsume and Fail [3].

3.5. Senvar & Bener

On top of the theory proposed by Paolucci Senvar and Bener introduced the quantification of ratings in between each semantic classification group to better identify the relative importance of results. There are two major improvements in their work: the definition of semantic distance with respect to number of children and constant weight values for each semantic group. Semantic distance is a calculated cumulatively by

multiplying the number of children for each level in the ontology hierarchy tree [4]. Constant weight values are assumed to be defined by the ontology specialist. Overall rating of a concept is calculated by multiplying the weight value with the semantic distance [4].

By introducing quantification into semantic matchmaking Senvar and Bener could identify the level of importance of each service in terms of semantic matchmaking and were able to return an ordered set of results accordingly.

3.6. SAM: Semantic Advanced Matchmaker

Besides assigning constant weight values and defining semantic distance with respect to the number of children a concept has, it is also possible to enhance the scoring mechanism by adding supplementary techniques such as Word Net and attribute based comparison [85]. Word Net fine-tunes the calculated score by organizing words into synonym sets so that a semantic relation can be formed in-between. In such cases scoring only with respect to semantic distance is preceded by the classification of domain information so that only relevant services are explored in terms of semantic matchmaking. Same approach is enhanced in [90] by weighting the results of semantic matchmaking and Word Net and calculating the overall rating value by summing up the weighted components. Optimal matching is provided by bipartite matching algorithms in these approaches which maximize match score for multiple input or output cases. The following figure summarizes the applied architecture.

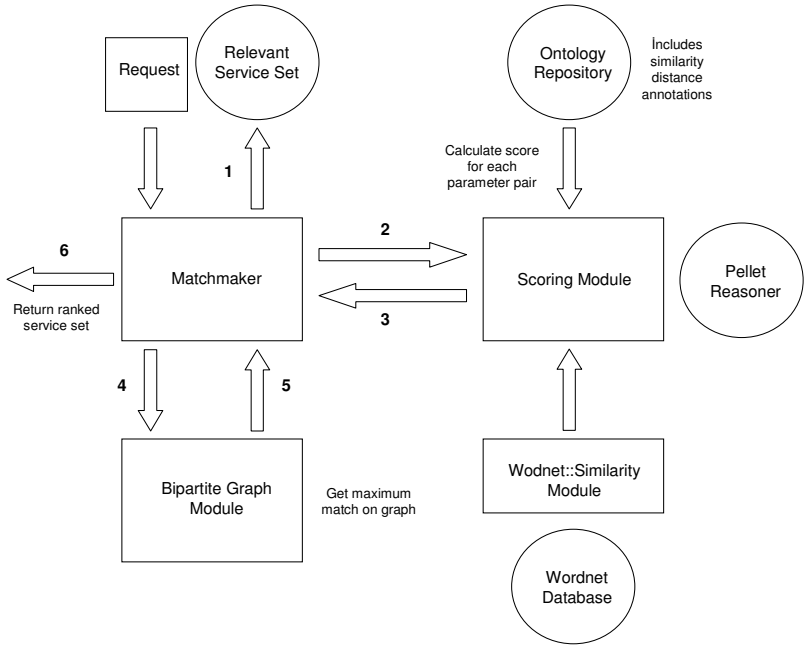

Figure 3.1. Hybrid Architecture for Semantic Matchmaking [85]

4. PROBLEM STATEMENT

4.1. Basics of Matchmaking in Web Services

Current infrastructure of web services is constituted up of three main components: SOAP (Simple Object Access Protocol), WSDL (Web Services Definition Language) and UDDI (Universal Description, Discovery and Integration) [34]. SOAP stands for exchanging messages in XML format over HTTP. It forms the foundation layer of web services stack, providing basic messaging framework that more abstract layers can be built on. *WSDL, in essence allows for the specification of the syntax of the input and output messages of a basic service, as well as other details needed for the invocation of the service* [1]. UDDI on the other hand is a global registry which facilitates businesses to define their web services, discover other web services and share information about how they interact in a global registry [2]. This structure of web services allows related functions to be performed independent of the implementation; XML based communication between different parties and searching of available web services within a registry, based on the given inputs and desired outputs (advertisement vs. request) [34]. The focus of the research in this thesis is on searching of web services in a global registry.

Within the above stated architecture, matching of given input and desired output for a web service is performed syntactically. In other words current searching mechanisms only take textual representations of input and output into account and try to figure out the best result for the query [16]. Such an approach has a crucial bottleneck: *realistically it is not possible to expect textual advertisements and requests to be semantically equivalent* [3]. Ontological approaches are introduced to web services world at this point [1, 3 and 4] and they try to match both parties on a semantic basis.

Ontologies provide a common vocabulary of terms, some specification about the meanings of those terms and an agreement for shared understanding of humans and machines. In this respect ontology represents real world objects with a hierarchically organized set of concepts. The concepts are bound with "*isA*" relation, which indicates subsumption between two concepts [6]. As a direct consequence of *isA* relation when we

move down the hierarchy more specialized concepts appear with more properties attached on them.

There exist different languages to express concepts and ontologies in XML standards [7]. Starting from BPEL4WS and WSDL-S, DAML+OIL, RDF, OWL and OWL-S are only some of the languages used for representing ontologies. BPEL4WS (Business Process Execution Language for Web Services) provides a means to formally specify business processes and interaction protocols. It provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and business-to-business spaces [30]. WSDL-S (Web Services Semantics) specification is a W3C Member Submission that defines how to add semantic information to WSDL documents. Semantic annotations define the meaning of the inputs, outputs, precondition and effects of the operations described in the web services interface. These annotations reference concepts in the ontology. Semantic annotations are used to automate service discovery, composition, mediation and monitoring [31]. DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) is an ontology language that is designed to describe the structure of the domain. It takes an object oriented approach, describing the structure in terms of classes and properties. Ontology consists of the axioms that assert subsumption relationships between classes or properties [32]. RDF (Resource description Framework) is a general framework for describing a Web Site's metadata or the information about the information on the site. It provides interoperability between applications that exchange machine-understandable information on the web. RDF relies on XML as interchange syntax, creating an ontology system for the exchange of information on the web [33]. OWL (Web Ontology Language) is designed for use by applications that need to process the content of information instead of just presenting information to humans. It is built on top of RDF to overcome its shortcomings such as local scope of properties and cardinality restrictions. It has three different versions: OWL Lite, OWL DL and OWL Full [34]. OWL-S on the other hand is built on top of OWL to realize discovery, invocation, composition and monitoring on web services with some predefined degree of automation. The research on ontologies shows that the most promising language is OWL-S. We have

also used OWL-S to represent services throughout the research in this thesis. In OWL-S concepts are represented with classes and properties are attached to these classes. In the rest of the thesis we use the terms “class” and “concept” interchangeably [12].

Matchmaking is defined as the process of finding the service provider that satisfies the server requests [36]. It is executed based on whether the web service requests and advertisements that are represented by service input and output in the functional description match or not. The matchmaking system must support input and output through the repository and enable service browsing. Semantically enhanced web service matchmaking is performed through the concepts that are already defined in the ontology. Degree of match is identified with the subsumption relation [3]. Subsumption relation defines whether the requested concept is a sub or super concept of the advertisement. Depending on the level of subsumption relation, matched services are rated and ranked accordingly [4]. Matchmaking process returns an ordered set of web services that fulfills the input of requester.

In order to visually illustrate matchmaking we are going to use the below stated sample ontology throughout the rest of our work.

4.2. Sample Ontology

Figure 4.1 displays sample vehicle ontology with 11 different concepts located at 5 different levels of subsumption. Following is a list of common statements that can be derived using the sample ontology:

(i) *Mercedes*, *Opel* and *VW* are children of *Four-Door* and grand children of *Sedan*.
(ii) *Car* is parent of *Sedan* and *Vehicle* is grandparent of *Sedan*.
(iii) *Sedan*, *SUV* and *Station Wagon* are siblings.
(iv) *Sedan* is a *Car* and isA *Vehicle*. *Sedan* is not a *Bus*.
(v) *Car* does not have all the properties *Sedan* has but *Sedan* has all the properties *Car* has.
(vi) *Sedan* inherits all the properties of *Car*. In general children inherit all the properties of *Parent*.

(vii) *Sedan* subsumes *Car*; *Car* is subsumed by *Sedan*, *SUV* and *Station Wagon*.

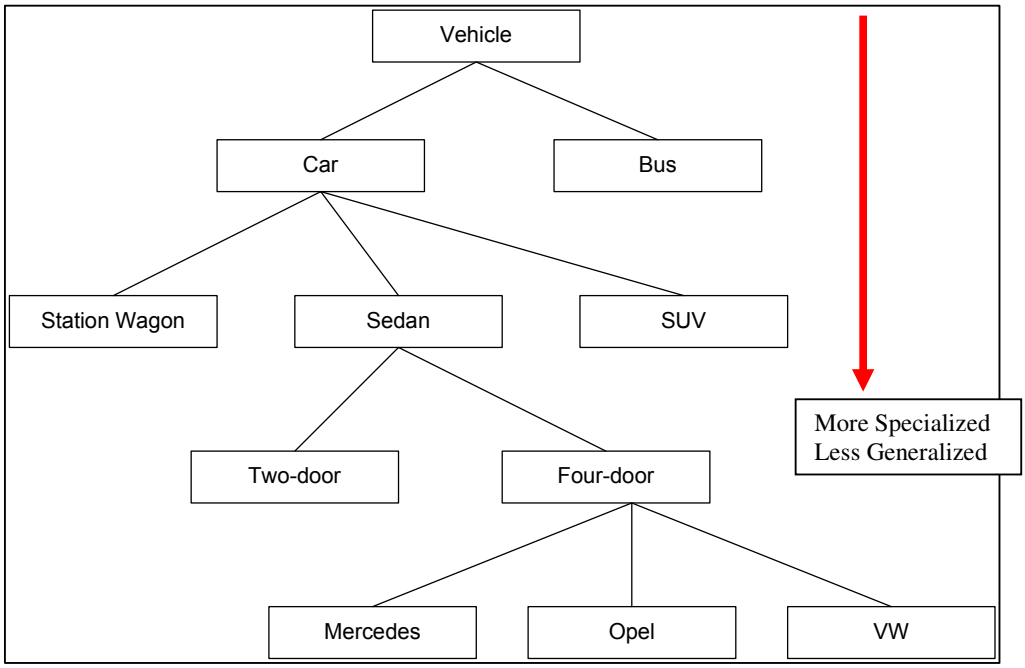

Figure 4.1. Sample Vehicle Ontology

4.3. Current Approaches in Ranking of Semantic Web Services

4.3.1. Semantic Classification

Ranking of ontology concepts with respect to the search criteria starts with the early approaches of Paolucci et al [3]. In their work concepts are grouped into four different groups with respect to their degree of subsumption. These groups are:

(i) **Exact**: If concepts in advertisement and request are the same

(ii) **Plug – In**: If advertisement subsumes request then advertisement can be plugged in request.

(iii) **Subsumes**: If request subsumes advertisement then provider does not completely fulfill the request.

(iv) **Fail**: If no subsumption relation between advertisement and request exists.

Relative importance of these groups is Exact, Plug-In, Subsumes and Fail in decreasing order. Besides the level of importance assigned to the semantic group, importance level of output and input are also differentiated such that output has greater importance than input. Candidate services are sorted according to their corresponding level of importance first and then a sorted list is returned to the requester. With this approach Paolucci et al. handle different matching patterns but miss explicit ranking of matched services. Since there is no numeric evaluation for matched services, there is not any applicable ordering schema for services that are classified in the same group [3].

4.3.2. Global Rank Constants and Semantic Distance Weight

In order to evaluate the candidate set of services, Senvar & Bener tried to calculate rank values by assigning constant values to each group of semantic classification [4]. In their work these values are assumed to be assigned by a domain expert. In addition they define semantic distance weight as *"similarity of concepts which may be the rate of coverage of sub-concepts for each concept in relation to subClassOf"* [4]. Assignment of semantic distance weight information can be done by the domain expert or by simply *1/<number of children>*. The weight information is accumulated by multiplication when traversing more than two concepts and eventually multiplied by the predefined constant value to calculate the ultimate rating. Candidate services are sorted according to this rating value and sorted list is returned to the requester. Following is the formula used for rating calculation in Senvar and Bener approach:

$$Similarity(\mathbf{T},\mathbf{C}) = \left\{ \begin{array}{l} Relation\ Constant * \prod_{i=T}^{C} (1 / \#\ of\ children\ of\ i\ ),\ where\ i\ stands \\ for\ each\ intermediate\ node\ between\ \mathbf{T}arget\ and\ \mathbf{C}andidate \\ \qquad with\ respect\ to\ subsumption\ relation. \end{array} \right\} \tag{4.1}$$

4.3.3. Sample Scenario

Let us assume that we are looking for a web service with output "Sedan" and for each of the concepts in the sample ontology there exists one and only one web service with that concept as output. The question is that: How would the candidate services be ranked?

To answer this question first thing to do is semantic classification. The classification pattern used by Paolucci et al. returns parent and grand-parents of a concept in subsume group with children and grand-children in plug-in. In other words only vertical movement along the ontology hierarchy is of concern. Results of this classification are depicted in Table 4.1. [3]

Table 4.1. Semantic Classification of Sample Ontology for Sedan (Paolucci)

Group Name	**Concepts Belonging**
Exact	Sedan
Plug-In	Car, Vehicle
Subsumes	Two-Door, Four-Door, Mercedes, Opel, VW
Fail	Station Wagon, SUV, Bus

Constant values for these groups are defined as 1, 0.8, 0.5 and 0 for exact, plug-in, subsumes and fail groups respectively by Senvar & Bener. The following graph shows the semantic distance weight assignment for each concept (Figure 4.3).

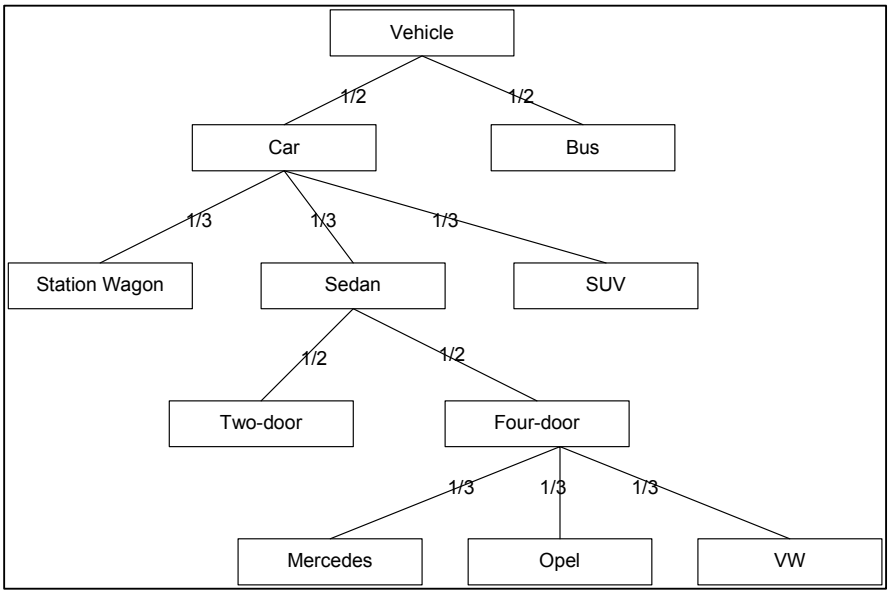

Figure 4.2. Semantic Distance Weight Assignment for Sample Ontology (Senvar)

Using the semantic distance weight values and group constants rating values are calculated as follows:

Table 4.2. Senvar & Bener Rating Values of Concepts in Sample Ontology for Sedan

Ranked Concept	**Rate Value**
Vehicle	½ *1/3 * 0.5
Car	1/2 * 0.5
Two-Door	1/2 * 0.8
Four-Door	1/2 * 0.8
Mercedes	½ *1/3 * 0.8
Opel	½ *1/3 * 0.8
VW	½ *1/3 * 0.8

4.4. Bottlenecks of Current Approaches

Our work on semantic matchmaking aims to cover most of the missing points of already proposed approaches however; it is primarily constructed on top of Senvar and Bener approach. In the following subsections, we will identify the shortcomings of this approach.

4.4.1. Predefined Multipliers

One of the major drawbacks of Senvar and Bener approach is its extensive usage of static multipliers for different semantically classified groups. Usage of a constant multiplier for all candidate services that fall in the same semantic group ignores the importance of individual concept-to-concept relations. Moreover addition or removal of a new concept in the hierarchy changes all the similarity values of existing concepts. However, this is not the case in real life. If concepts can be rated only with respect to their relations with other concepts and independent of any static predefined multiplier we believe that ranking of candidate web services would be more reasonable.

4.4.2. Horizontal Movement

Another drawback of Senvar and Bener algorithm is that it allows only vertical movement along the hierarchy tree. This restriction limits the result set and leaves such a gap that may omit more related results. It should be possible to traverse siblings of the target node in the hierarchy before going upper levels by assigning appropriate rating value and considering that node as a candidate. Such a move will not only expand the result set but also catch more relevant results particularly when the candidate set is limited in size.

4.4.3. Need for a Methodology

Researchers who are interested in matchmaking of web services have spent all their effort to proper ranking of candidate services by using algorithms that are built on subsumption relation of concepts [2, 3, 4 and 7]. Actually, this is only one side of the coin because there are many class manipulators that derive new classes from already existing

ones in ontology definition languages [12]. There is a need for a complete methodology that will take care of such operations as well as subsumption relation, calculate their rating values and return an appropriate result set.

4.4.4. Multiple Inheritance

Multiple Inheritance is a valid way of concept generation in ontology definition languages [11]. Besides with the increasing importance of information sharing among ontologies existence of multiply inherited concepts becomes inevitable particularly for the sake of minimization of information loss during transformation [71]. It should be possible to include multiply inherited concepts in ranking process and we find it worthwhile to evaluate in our approach.

4.5. Target Achievements

Our research aims to answer the following questions:

(i) Can we reach more distinct ranking results if we derive ranking from local relations instead of fixed values?
(ii) How can we increase computational efficiency?
(iii) How can we expand the result set?
(iv) How can all possible operations be performed between owl classes?
(v) Can we enable multiple inheritance ranking?

5. PROPOSED SOLUTION

In order to eliminate the above-stated shortcomings we propose a new ranking scheme which is based on comparison of property numbers of individual concepts. Before proceeding with the details of the proposed algorithm it is necessary to define *isA* relation clearly. The approach we propose defines *isA* relation as follows: If there exists *isA* relation between two concepts (i.e. concept B *isA* concept A) then child concept is a specialized form of parent concept. This specialization takes place by adding more features to parent concept (concept A). Using this definition we are trying to re-define subsumes and plug-in rate values only for individual relations without the usage of global constants. In the rest of this chapter we will discuss about the underlying logic of *isA* relation and usage of number of properties a concept has. We will then move on to proposed calculation of rating values with respect to comparison of property numbers. After clearly setting the ranking methodology we will cover horizontal movement along the hierarchy, analyze how this approach can be used for concept manipulation in OWL-S and describe how multiple inheritance is handled.

5.1. Property Based Comparison of Concepts

Senvar & Bener approach on ranking of concepts defines semantic similarity with respect to the number of child concepts, namely *1 / <number of children>*- [4]. We believe that such a calculation has two direct consequences:

(i) Similarity between a concept and its sub-concepts degrades as the number of sub-concepts increases. Asserting that number of sub-concepts is a measure of specialization, such a rating methodology assigns lower rank values to those who are able to specialize more. We believe that such a case does not make sense in real life scenarios.

(ii) Introduction of a new child concept or removal of an existing child concept changes the similarity measure between parent concept and its child concepts. We believe that valid ranking of a relation between a parent and its child has to be independent of the relations between parent and other childen.

In order to overcome the stated shortcomings of Senvar and Bener approach we overload the meaning of rating as follows: If we are considering sub-concept of a parent then rating refers to degree of specialization. On the other hand if we consider parent of a concept then rating refers to degree of generalization. Since specialization is simply addition of new properties and generalization is removal of existing ones we define rating values with respect to additional or missing properties. In parallel with our purpose, such a scheme separates the "potential to specialize" from ranking.

5.2. Rating Calculations

We preserve the semantic classification of concepts but alter the calculation formulas used in Senvar and Bener approach. Motivation behind the selection of new formulas is as follows:

(i) Possible rank values are within the interval of [0-1]. Exact matches achieve 1 point and no matches are evaluated as 0 point.

(ii) Groups of semantic classification are separated clearly. In our approach we accept that no subsume relation can have a greater rating than any plug-in relation because candidate concepts grouped as plug-in can be used interchangeably with target concept. To make this distinction we assign plug-in ratings between 0.5 and 1 whereas the interval for subsume matches is between 0 and 0.5. This distinction is represented with a scale factor in corresponding formulas.

(iii) Since ranking will be based on comparison of number of properties, we include ratio of property numbers in rating calculations.

By taking the above mentioned assumptions into account we revised the calculation formulas in Senvar and Bener approach as in Table 5.1 where T represents target concept, C represents candidate concept, ½ represents the scaling factor.

Table 5.1. Ranking Calculation for Match Groups

Semantic Classification	Rate Formula
Exact	1
Plug-in	½* (1+T/C)
Subsume	½ * C/T
None	0

Results of this ranking mechanism on the sample ontology and the scenario mentioned in previous section are detailed in Figure 5.1 and Table 5.2. The numbers next to the concepts on the ontology represents number of properties each concept has, which are assigned arbitrarily in order to display how the algorithm proceeds.

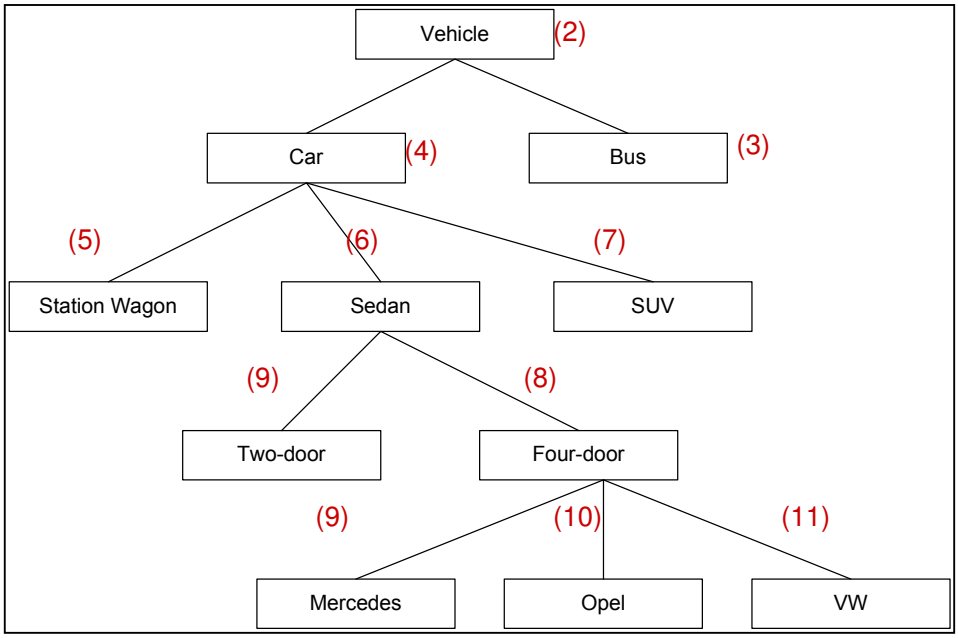

Figure 5.1.Sample Ontology with Number of Properties Included

Table 5.2. Revised Ranking Values of Concepts in Sample Ontology for Sedan

Ranked Concept	Rate Value Proposed	Rate Value (Senvar & Bener)
Vehicle	1/2 * 2/6	½ *1/3 * 0.5
Car	1/2 * 2/4	1/2 * 0.5
Two-Door	1/2 * (1 + 6/9)	1/2 * 0.8
Four-Door	1/2 * (1 + 6/8)	1/2 * 0.8
Mercedes	1/2 * (1 + 6/9)	½ *1/3 * 0.8
Opel	1/2 * (1 + 6/10)	½ *1/3 * 0.8
VW	1/2 * (1 + 6/11)	½ *1/3 * 0.8

Table 5.2 shows us that the average rating values assigned by the proposed approach are higher than the ones assigned by Senvar & Bener particularly for plug-in matches. This early result is in parallel with our expectation of better separation between subsume and plug-in matches in terms of similarity ratings.

It should also be noted that there is always a factor of subjectivity in the identification of multipliers by a domain expert. Removing those multipliers turns interpretation of the results into a more mechanical but objective format. We believe that subjectivity has to be included in similarity evaluations however; it has to be taken care not by a domain expert but the ontology builder in the construction of domain ontology.

5.3. Bipartite Matching

For multiple input and output parameters, any service input (output) may have multiple score values if it matches with more than one target input (output). In order to maximize the overall score of the service we model requested and provided parameters as a bipartite graph. A graph is called “Bipartite” if the vertex set can be partitioned into two sets such that no edge has both endpoints in the same set of bipartition [84]. A matching is a collection of edges such that every vertex is incident at most one edge. If a vertex has no edge incident to it then it is said to be exposed. A matching is perfect if no vertex is exposed, in other words if cardinality of two vertex sets are equal [84].

Bipartite matching problems may have different forms: “Find a matching of maximum (minimum) size”, “Find a matching that minimizes the cost of matching between two mutually exclusive sets” and etc. [15]. Application of bipartite matching in web services world focuses on the matching of request and results in terms of inputs and outputs. This problem can be formally expressed as follows: “Given the benefit b_{ij} for each requested input (output) i and service input (output) j find a perfect matching of maximum benefit where the benefit of matching is given by $\Sigma\ b_{ij}$”. This problem is also called Assignment Problem [84].

Hungarian algorithm solves this instance of bipartite matching problem by first transforming it into a cost matrix representation The following is the flow of algorithm used to maximize the rating values [84]:

(iii) If the matrix is not a square matrix, add zeros to make it square (nxn).
(iv) Find the largest value entry in the matrix (max_value) and replace all the others with (max_value-entry values)
(v) Subtract the entries of each column by the column minimum
(vi) Draw lines for rows and columns so that all the zeros are covered and there is no unnecessary line.
(vii) If the number of lines drawn is n, choose a combination from the cost matrix in such a way that the sum is zero.
(viii) If the number is smaller than n find the smallest element which is not covered by any other lines. Then subtract it from each entry which is not covered by the lines and add it to each entry which is covered by a vertical and a horizontal line. Then go to step 3.

Usage of bipartite matching in maximization of multiple concept matching is not a new phenomenon. This approach is used in hybrid approaches [85] that mainly aim to fine tune and enhance the quality of existing scoring schemas. Usage of WordNet increases the reliability of the results however, being language dependent and syntax-based comparison are two main drawbacks of this approach. Accumulating all the information and gathering the final result is an enhanced way of matchmaking but, it does not fix the main problem of semantic similarity with respect to constant and pre-defined values. Besides, from the

perspective of performance, usage of word net has an adverse affect on required time to respond to service requester which removes the positive effect of bipartite matching [85].

5.4. Horizontal Movement

Many research up to now have generally studied horizontal move on hierarchy tree as a supplementary issue and they used it in matching of concepts in different ontologies or resource description documents [9, 10 and 26]. Besides, there are also some studies based on sibling correlation [8]. In our research we accommodate the rules that are derived from these studies into our ranking methodology. We would like to discuss horizontal movement in four subsequent sections:

5.4.1. Horizontal Movement: A Necessity?

Concepts in an ontology tree are located with respect to their subsumption relations with other concepts. Since subsumption is a vertically defined relation between a concept and its sub-concepts, traversal of ontology tree with vertical movement is valid. However, a concept does not only have a parent or several sub-concepts, it may also have sibling concepts as well. If this is the case we can argue that it would be a mistake to consider only the concepts on the vertical path and ignore siblings at all. In other words we can ask the following question: is it worthwhile to consider sibling relations as well as parent and children relation?

If we define ranking as the degree of specialization with respect to additional or missing properties, then we can make a judgment about the rating values of the siblings. This judgment is twofold: First we know that sibling of a concept inherits all the properties of its parent. Therefore siblings have parent's number of properties in common which has a positive effect in similarity measure. Second the sibling has some additional properties from parent concept. This means that we also have to account for additional properties but these will have a negative impact on similarity measure.

5.4.2. Measurement of Rating Value for Sibling Relation

Since we treat the horizontal movement as a combination of two vertical moves corresponding score also has to reflect the properties of this combination. It should account for the common properties of siblings that are inherited from parent as well as the additional ones that make children distinct. This situation requires the rating of horizontal move to be calculated as the multiplication of subsumes relation between child and parent with the plug-in relation between parent and sibling. More formally, let *Similarity* define the similarity function with two parameters: target concept *T* and candidate concept *C*. If *T* and *C* are siblings and their parent is *P* then we define similarity measure between siblings as:

$$Similarity\ (T, C) = Similarity\ (T, P) * Similarity\ (P, C) \qquad (5.1)$$

Let us check the sample ontology at this point (Figure 4.1): Let us assume that we are looking for *Sedan*. Then rank value of *SUV* will be

$$\begin{aligned} Similarity\ (Sedan, SUV) &= Similarity\ (Sedan, Car) * Similarity\ (Car, SUV) \\ &= (1/2 * 4/6) * (1/2 * (1+4/7)) \end{aligned}$$

It should be noted that assigned rating value for sibling relation will be lower than both subsume and plug-in relations since none of the multiplied terms can have a value greater than 1.

5.4.3. Parallelism between Proposed Solution and General Rules of Horizontal Movement

There are mainly two rules that we are looking for parallelism:

(i) Depth Relative Property: “Siblings at lower parts of the ontology are more similar than siblings at upper parts.” [10]
(ii) Correlation between sibling concepts.

Depth-relative property is supported in our approach since we are evaluating similarity with respect to number of properties ratio. As number of properties increase while going deeper in the hierarchy similarity measure also increases since number of common properties that sibling concepts have also increase. On the other hand, we take care of the correlation between two siblings by accounting for the common properties they have as well as the different properties that makes them specialized. The ranking methodology covers correlation issue at this point.

5.4.4. Performance

While traversing sibling concepts in ontology we have to traverse the entire hierarchy tree. This inevitable situation arises since we have to look at the children of siblings as well. Such a flat search may lead to performance burdens which have to be prevented. To do so we propose two different ways:

(i) **Threshold Level**: It is possible to define a threshold level for ranking between compared concepts. If result of the comparison scores lower than this threshold level then there is no need to go any further in the search operation.
(ii) **Diameter**: It is also possible to define diameter of the search operation. If search operation gets longer than the defined number of steps then operation halts.

Limiting search operation in terms of rating or diameter is risky, mainly because high rating or unrealistic diameter values with respect to ontology may lead to unnecessarily expanded or over-limited candidate service sets. Therefore these two parameters are assumed to be defined by the ontology manager. On the other hand since we perform our search based on individual concept-to-concept relations and assign values to each link accordingly, we believe that above stated precautions would be sufficient in reaching a balance between the performance of algorithm and size of its corresponding result set.

5.5. Integration with OWL-S

"OWL-S is an OWL-based web service ontology, which supplies web service providers with a core set of markup language, constructs for describing the properties and

capabilities of their web services in unambiguous, computer-interpretable form" [11]. An OWL document includes optional ontology header, class axioms, property axioms and facts about individuals [12]. Classes provide an abstraction mechanism for grouping resources with similar characteristics. In other words classes in OWL represent concepts in the ontology. In this section we are going to explain how different class descriptions are mapped to our algorithm.

OWL distinguishes six types of class descriptions: a class identifier, an exhaustive enumeration of individuals that form the instance of a class together, a property restriction, intersection of two or more classes, union of two or more classes, and complement of a class description. Moreover OWL has class axioms that derive new classes from an already existing one. There are three such class axioms: subClassOf, equivalentClass and disjointWith [12].

For the purpose of our algorithm we need the total number of properties an OWL class has. In order to collect this information we have to check each of the above mentioned cases individually:

(i) **Class Identifier**: All the properties that belong to a class identifier are stated in the ontology file.

(ii) **Enumeration**: Class instances are not useful for our algorithm mainly because we do not have enough information about the properties of that instance.

(iii) **Property Restriction**: Putting a restriction on valid value range of a property does not affect the total number of properties a class has. Therefore, we accept the number of properties of the derived class as the same of its original.

(iv) **Intersection of Two or More classes**: Intersected number of properties will count for classes that are derived via intersection.

(v) **Union of Two or More Classes**: Total number of distinct properties will count for classes that are derived via union.

(vi) **Complement of a Class**: Complement of a class can only be stated if we have any information about the universal set of classes. Since we do not have such information it is not possible to identify the number of properties of a complement class.

(vii) **SubClassOf**: This relation is exactly the isA relation that is mentioned in the previous sections so we already know how to calculate the number of properties.

(viii) **EquivalentClass**: If two classes are exactly the same then their number of properties are also the same.

(ix) **DisjointWith**: As in the case of complement it is not possible to count the number of properties of disjoint classes without having information about the universal set.

The following table summarizes corresponding number of properties for each different class descriptions:

Table 5.3. Property Numbers for Different Class Descriptions in OWL

OWL-S Operation	Property Number of New Class
Class Identifier	Check ontology file
Enumeration	N/A
Property Restriction	Same as the root class
IntersectionOf	# of common properties
UnionOf	Total # of properties
ComplementOf	N/A
SubClassOf	Root + # of new properties
EquivalentClass	Same as the root class
DisjointWith	N/A

5.6. Inheritance

Object oriented languages define multiple inheritance as *"feature in which a class can inherit behaviors and features from more than one superclass"* [14]. This definition is exactly the same for ontology concepts. Before going into further details we find it worthwhile to cover theoretical background and application of inheritance both in object oriented programming and ontology construction.

5.6.1. Object-Oriented Programming

Object-Oriented Programming is a programming paradigm that uses objects to design applications and computer programs [65]. Objects are particular instance of classes. A class defines the abstract characteristics of a thing (object), including the thing's properties (attributes) and its behaviors (methods) .

Object-Oriented Programming (OOP) has four main principles [72]:

(i) **Abstraction:** Abstraction is simplifying complex reality by modeling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem.

(ii) **Encapsulation:** Encapsulation conceals the functional details of a class from objects that send messages to it. Encapsulation is achieved by specifying which classes may use the members of an object. The reason for encapsulation is to prevent clients of an interface from depending on those parts of the implementation that are likely to change in future.

(iii) **Polymorphism:** Polymorphism allows you to treat derived class members just like their parent class's members. More precisely, Polymorphism in object-oriented programming is the ability of objects belonging to different data types to respond to method calls of methods of the same name, each one according to an appropriate type-specific behavior.

(iv) **Inheritance:** Attributes and behaviors can be inherited from the class that is being derived from. New attributes and behaviors can also be introduced in new class declaration.

These principles of OOP has the power of analyzing user requirements, designing software, constructing software while maintaining reusability, reliability, robustness and extensibility and the invention of divide-and-conquer strategy for managing complex problems. [72]

5.6.2. Inheritance in OOP

Inheritance in OOP is based on two basic terms: Sub-class and Super-class. A Sub-Class is a more specialized and hence extended version of the Super-class (Base or Parent Class). For example, Shape is a super-class for oval and square and oval is a super-class for circle in Figure 5.2. [66]

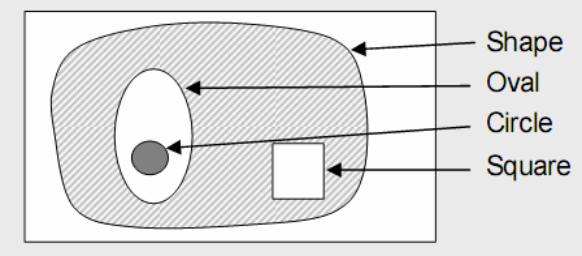

Figure 5.2. Inheritance

There are different types of inheritance [66]:

(i) A Direct Super-Class of a sub-class is that class from which it explicitly inherits within the class definition (also called parent)
(ii) An Indirect Super-Class of a sub-class is class which the super-class inherits from (directly or indirectly – also called ancestor)
(iii) Single Inheritance entails that a class can only inherit directly from one superclass
(iv) Multiple Inheritance entails that a class can inherit directly from a number of super-classes.

Inheritance is a form of software reusability in which programmers create classes that absorb an existing class data and behavior and enhance them with new capabilities. Inheritance is represented with is-a relationship [67]. An object of a derived class can also be regarded as an object of the base class. For example a car is a vehicle, so any properties and behaviors that a vehicle has also exist in car. By contrast has-a relation stands for composition. In a has-a relation an object contains one or more objects of other classes as members – for example a car has a steering wheel.

Inheritance relationships form tree-like hierarchical structure in which a base class exists with its derived classes [67]. Although classes can exist independently, once they are employed in inheritance relationship they become affiliated with other classes.

5.6.3. Inheritance in Ontology Construction

As in the case of object-oriented approach inheritance in ontology construction is maintained by subsumption relation [72]. Classes in object-oriented programming are replaced with concepts and objects of classes become instances of concepts in ontology terminology. The characteristics of a class are referred with properties of a concept and inheritance of properties is supported for derived concepts – for concepts that have is-a relation in between.

There are several different approaches in constructing an ontology hierarchy [6]:

(i) A Top-Down development process starts with the definition of most general concepts in the domain and subsequent specialization of the concepts.
(ii) A Bottom-Up development process starts with the definition of most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts.
(iii) A Combination development process is a combination of top-down and bottom-up. More salient concepts are defined first, and then generalize and specialize them appropriately.

There is no absolute true way of constructing the ontology; selection of the methodology is user dependent. The important point is correctly linking the concepts so that is-a relation holds. Is-A relation in ontology terminology is defined as follows: "*A subclass of a class represents a concept that is a kind of the concept that the superclass represents.*" [66]. For example, a sedan is a car which is a vehicle.

There is no generic way of defining the number of classifications that an ontology has to particularly when the ontology gets wider but the general rule of thumb is "*If a class has only one direct sub-class there may be a modeling problem or the ontology is not*

complete; if there are more than a dozen of subclasses for a given class then additional intermediate categories may be necessary" [68]. A new class has to be introduced to the ontology if it has additional properties that the super-class does not have or it has some restrictions on the super-class or it can participate in different relationships than the superclass [66].

5.6.4. Multiple Inheritance

Basically Multiple Inheritance means that a class or concept inherits member variable and method implementations directly from more than one parent class [68]. Being one of the fundamental principles of object oriented programming, multiple inheritance also has different points of practice [69]:

(i) **Modeling Real World Situations**: Multiple Inheritance is used to model the related but distinct roles in real world. These roles generally correspond to passive classes (classes that have only contains data with corresponding access methods) in object oriented jargon.

(ii) **Polymorphic Hijacking**: Even if there is a neat class hierarchy with objects of type A, for some reason the hierarchy tree has to be rooted at class B. If this is the situation, create a class that inherits both from A and B and treat objects as either A or B polymorphically.

(iii) **Cheap Pick Up of Functionality**: Generally to use single purpose, small classes through implementation, classes are derived from different classes which have some useful routines already defined. These classes are called mix-in classes [69].

From the perspective of object-oriented programming multiple inheritance has an important scope of usage which is favored to be more practical rather than its intended class construction purpose. On the other hand, from the perspective of ontology construction the major concern is to create new concepts [72]. In single inheritance case the relation between the inherited concept and the parent concept is generalization (specification); whereas the relation between multiply inherited concept and its parent concepts is composition [68].

Ontology definition languages support the concept of multiple inheritance however, there is no to-the point on implementation of ranking semantic web services that has multiply inherited concepts as input or output. In this research we try to figure out a ranking methodology that will also take care of such concepts and evaluate them in the result set. Figure 5.3 depicts a generic example of multiple inheritance [67]

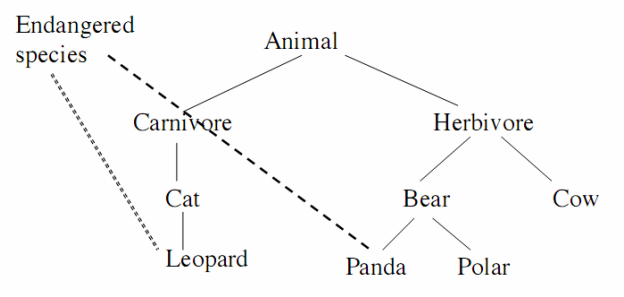

Figure 5.3. Multiple Inheritance

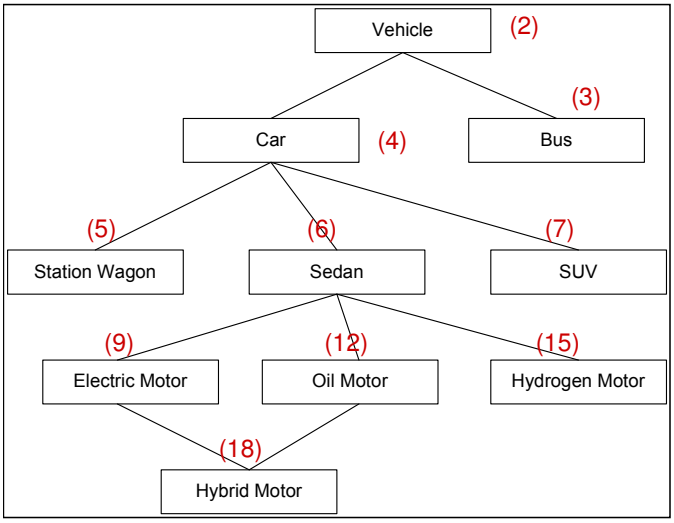

Figure 5.4. Sample Ontology with Multiple Inheritance

In Figure 5.4 we have modified the sub-tree of *Sedan* and inserted motor types instead of *Two-Door* and *Four-Door* concepts. In this new ontology we see that *Hybrid Motor* inherits all the properties of *Electric Motor* and *Oil Motor*. In this case we have to ask a question: How the similarity between *Sedan* and *Hybrid Motor* will be measured?

Since there is more than one path between these two concepts we have to traverse each path individually and calculate corresponding rating value. Then we will be able to select the maximum rated path and assign the similarity measure accordingly. However, for the sake of our sample scenario we do not have to traverse each path because each path lies on the same vertical inheritance line, in other words since *Sedan* is grandparent of *Hybrid Motor* they are on the same vertical line and this allows us to calculate the similarity measure directly, without looking at the intermediate connections. Therefore, *Similarity* (*Sedan*, *Hybrid Motor*) is calculated as 1/2 * 6/18 in a single step.

However, this might not be the case and inherited concepts may be located at different vertical lines (i.e. assume that there exists a multiply inherited concept with two parents in cousin relation and targeted concept has a relation with both parent). In this case each path directed from target concept to candidate concept should be considered individually and maximum rated path has to be selected. Calculation of different paths that do not lie on the same vertical line is a direct consequence of our horizontal movement approach since otherwise it is not possible to calculate the rating of alternative paths. The following formula formally states the calculation of multiply inherited concepts:

$$Similarity\ (T,C) = \mathbf{Max}(Similarity_1\ (T,C),\ Similarity_2\ (T,C),\ldots,\ Similarity_n\ (T,C)),$$
where $Similarity_n\ (T,C)$ *represents the rating value calculated by following the* n^{th} *path.*

(5.2)

5.7. Matching Algorithm

For the sake of comprehendibility we find it worthwhile to mention the complete algorithm at once by including all the above detailed parts as in Figure 5.5.

1. Request input and output concepts and their corresponding ontologies from the user
2. For each concept in the ontology calculate rating w.r.t requested input and output concepts
    2.1. Rate subtree of requested concept as plug-in
    2.2. Rate super tree of requested concept as subsumes
    2.3. Perform horizontal movement while traversing the super tree
    2.4. Check threshold values (min score & max number of steps) at each step of calculation
    2.5. Assign max score value calculated to each concept (multiple inheritance)
3. For each of the available services
    3.1. Identify service input and output concepts' rating values
    3.2. Apply bipartite matching to maximize the overall service score
    3.3. Check for threshold values, qualify the ones that pass
4. Rank order the result set and return to requester.

Figure 5.5. Matching Algorithm

6. SIMULATION

We have implemented the matchmaking algorithm in Java environment and used open-source tools and APIs for forming base documents and performing tests. We have made slight modifications on some base classes and created new data structures designed for our rating algorithm.

To perform our tests we have used OWLS-TC 2.1 which is a collection of web services that is intended to support the evaluation of OWL-S service matchmaking algorithms [29]. This set includes 582 indexed services classified in 7 different domains. There are totally 43 different ontologies used and test set provides 29 different queries. A fragment of sample ontology and a sample service is given in Figure 6.1 and Figure 6.2 respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
<!ENTITY books.owl "http://127.0.0.1/ontology/books.owl">
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY simplified_sumo.owl "http://127.0.0.1/ontology/simplified_sumo.owl">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="&books.owl;"
xmlns:owl="&owl;"
xmlns:rdf="&rdf;"
xmlns:rdfs="&rdfs;">

<!-- Ontology Information -->
<owl:Ontology rdf:about=""
rdfs:label="Book Ontology"
owl:versionInfo="1.0">
<rdfs:comment>An ontology containing information about books</rdfs:comment>
<owl:imports>
<owl:Ontology rdf:about="&simplified_sumo.owl;"/>
</owl:imports>
</owl:Ontology>

<!-- Classes -->
<owl:Class rdf:about="#A">
<rdfs:subClassOf rdf:resource="#Grade"/>
</owl:Class>
```

```
<owl:Class rdf:about="#Article">
<rdfs:subClassOf rdf:resource="#Text"/>
</owl:Class>
<owl:Class rdf:about="#Author">
<rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:about="#B">
<rdfs:subClassOf rdf:resource="#Grade"/>
</owl:Class>
<owl:Class rdf:about="#Book">
<rdfs:subClassOf rdf:resource="#Monograph"/>
<rdfs:subClassOf>
<owl:Restriction rdf:nodeID="b10">
<owl:allValuesFrom rdf:resource="#Author"/>
<owl:onProperty rdf:resource="#writtenBy"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#Book-Type"/>
<owl:onProperty rdf:resource="#hasType"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#Title"/>
<owl:onProperty rdf:resource="#isTitled"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Book-Type"/>
<owl:Class rdf:about="#C">
<rdfs:subClassOf rdf:resource="#Grade"/>
</owl:Class>

<owl:Class rdf:about="#Comic">
<rdfs:subClassOf rdf:resource="#Genre"/>
</owl:Class>
<owl:Class rdf:about="#D">
<rdfs:subClassOf rdf:resource="#Grade"/>
</owl:Class>
<!-- Object Properties -->
<owl:ObjectProperty rdf:about="#contains"/>
<owl:ObjectProperty rdf:about="#datePublished"/>
<owl:ObjectProperty rdf:about="#hasGenre"/>
<owl:ObjectProperty rdf:about="#hasGrade"/>
<owl:ObjectProperty rdf:about="#hasName"/>
<owl:ObjectProperty rdf:about="#hasSize"/>
<owl:ObjectProperty rdf:about="#hasType"/>
<owl:ObjectProperty rdf:about="#isTitled"/>
<owl:ObjectProperty rdf:about="#publishedBy"/>
<owl:ObjectProperty rdf:about="#timePublished"/>
<owl:ObjectProperty rdf:about="#writtenBy"/>
</rdf:RDF>
```

Figure 6.1. Fragment from sample ontology books.owl [29]

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF  xmlns:owl       = "http://www.w3.org/2002/07/owl#"
xmlns:rdfs      = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf       = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:service   = "http://www.daml.org/services/owl-s/1.1/Service.owl#"
xmlns:process   = "http://www.daml.org/services/owl-s/1.1/Process.owl#"
xmlns:profile    = "http://www.daml.org/services/owl-s/1.1/Profile.owl#"
xmlns:grounding = "http://www.daml.org/services/owl-s/1.1/Grounding.owl#"

xml:base       = "http://127.0.0.1/services/1.1/book_author_service.owls">

<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://127.0.0.1/ontology/Service.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Process.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Profile.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/Grounding.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/books.owl" />
<owl:imports rdf:resource="http://127.0.0.1/ontology/books.owl" />
</owl:Ontology>

<service:Service rdf:ID="BOOK_AUTHOR_SERVICE">
<service:presents rdf:resource="#BOOK_AUTHOR_PROFILE"/>
<service:describedBy rdf:resource="#BOOK_AUTHOR_PROCESS_MODEL"/>
<service:supports rdf:resource="#BOOK_AUTHOR_GROUNDING"/>
</service:Service>

<profile:Profile rdf:ID="BOOK_AUTHOR_PROFILE">
<service:isPresentedBy rdf:resource="#BOOK_AUTHOR_SERVICE"/>
<profile:serviceName xml:lang="en">
BookAuthorService
</profile:serviceName>
<profile:textDescription xml:lang="en">
This service returns author of the given book.
</profile:textDescription>
<profile:hasInput  rdf:resource="#_BOOK"/>
<profile:hasOutput rdf:resource="#_AUTHOR"/>

<profile:has_process rdf:resource="BOOK_AUTHOR_PROCESS" /></profile:Profile>

<process:ProcessModel rdf:ID="BOOK_AUTHOR_PROCESS_MODEL">
<service:describes rdf:resource="#BOOK_AUTHOR_SERVICE"/>
<process:hasProcess rdf:resource="#BOOK_AUTHOR_PROCESS"/>
</process:ProcessModel>

<process:AtomicProcess rdf:ID="BOOK_AUTHOR_PROCESS">
<process:hasInput  rdf:resource="#_BOOK"/>
<process:hasOutput rdf:resource="#_AUTHOR"/>
</process:AtomicProcess>

<process:Input rdf:ID="_BOOK">
<process:parameterType
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1/ontology/books.
owl#Book</process:parameterType>
<rdfs:label></rdfs:label>
</process:Input>
```

```
<process:Output  rdf:ID="_AUTHOR">
<process:parameterType
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://127.0.0.1/ontology/books.
owl#Author</process:parameterType>
<rdfs:label></rdfs:label>
</process:Output>

<grounding:WsdlGrounding rdf:ID="BOOK_AUTHOR_GROUNDING">
<service:supportedBy rdf:resource="#BOOK_AUTHOR_SERVICE"/>
</grounding:WsdlGrounding>

</rdf:RDF>
```

Figure 6.2. Sample Web Service book_author_service.owls [29]

We have used Protégé as ontology editor. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies [28]. It is used for the creation, visualization and manipulation of ontologies in various representation formats. Figure 6.3 depicts the class view of Protégé.

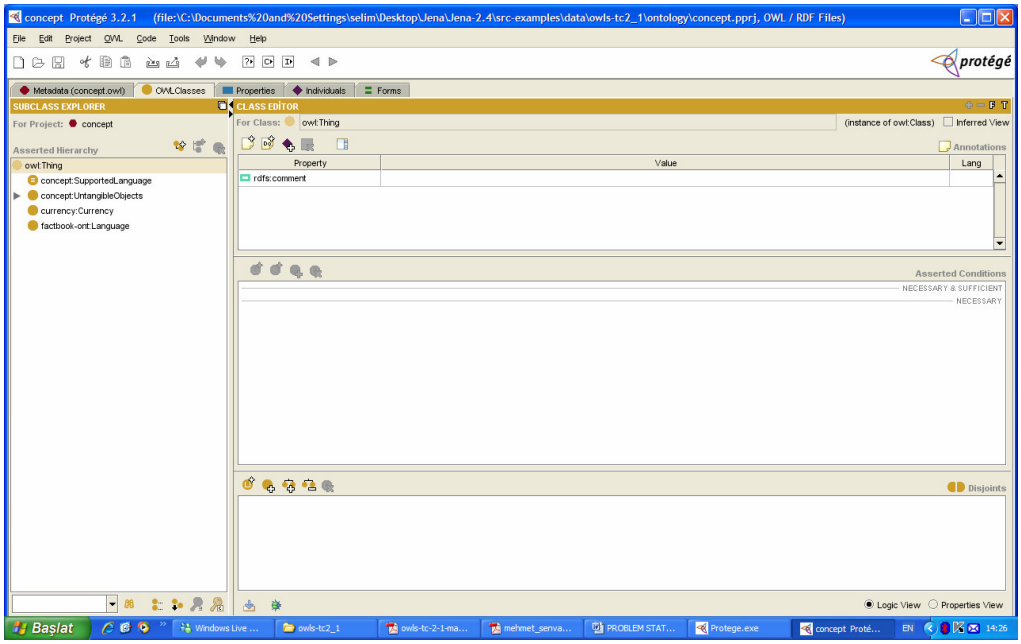

Figure 6.3. Class View in Protégé [28]

For ontology programming we have used Jena Framework. Jena provides a programmatic environment for RDF, RDF-S, OWL and SPARQL with a rule based inference engine [27]. We have used OWL API of Jena with a slight modification. We have added “Rank” and “Mark” variables to the “OntClass” definition of Jena. “Rank” variable stands for the score of the concept with respect to the target and “Mark” checks whether current concept is traversed before.

For evaluation purposes, we have implemented Senvar and Bener algorithm and performed simulation for two major cases. First we have selected two of the ontologies for input and outputs respectively in order to remove any ontology dependent effect from simulation. Then we have checked how other ontologies respond and compared our results.

6.1. Evaluation Criteria

In order to accurately measure the effectiveness of our approach, we have defined several different key performance indicator (KPI) categories [83]. Table 6.1 summarizes the identified categories and the underlying reasoning in their selection.

Table 6.1. Key Performance Indicator Categories

KPI Category	Motivation
Heterogeneous Result Set	The optimum matching set should include the services that are scored as highest according to the request. In addition, services in the result set should be scored as apart as possible so that degree of similarity can be differentiated by the requester. To measure the heterogeneity we have tracked the number of input and output score clusters formed in the result set.
Size of Target Result Set	The optimum matching set should include as much related services as possible. The proposed approach is assumed to attend this goal by including the similarity relation of sibling nodes. Assuming that data set has dissimilar services we have tracked the number of sibling concepts scored in ontology tree for each query.
Percentage of Qualification	The optimum matching set should include as much services as possible that are over a certain degree of relatedness. To measure the effectiveness we have tracked the number of services matched for different threshold values.
Flexibility	The optimum matching set should be as flexible as possible for ontologies of varying average degree of similarity. To measure the flexibility we have tracked the number of services matched for different ontology structures.

Table 6.2 summarizes our expectations from the proposed approach for each KPI category:

Table 6.2. Expectations from the Proposed Approach

KPI Category	Expectation
Heterogeneous Result Set	Separately scored concepts that are more apart than what other approaches propose, which in turn means more accuracy in terms of concept similarity
Size of Target Result Set	Increase in size of result set due to horizontal movement
Percentage of Qualification	Increase in percentage of qualified services for different threshold levels
Flexibility	Protection of the upwards trend in result set size for ontologies of varying differentiation ratios

6.2. Threats to Validity

Every scientific research addresses a problem, observes the root causes, proposes a solution and evaluates the proposed solution with respect to the causes identified. Nevertheless to say, this process is not always as smooth as expected, there are always some risks faced and some obstacles introduced throughout the journey to success.

In our study we find the simulation service set as the most vulnerable aspect of the research due to three different reasons. First of all, size of the service set that is used in the simulation is limited to reflect real-life case; hence we can confidently assess the scalability of the algorithm. Second, some of the services in the service set are not found to be valid when compared with other services that are also in the same set (i.e. referred concepts in referenced ontologies are not found). This may yield confusions in the interpretations of matched services; therefore we have taken these out in the simulation. Third, the service set does not completely comply with the precondition of the algorithm which states that each inherited concept has to have some difference with its parents and its siblings. To remedy this situation we have defined *differentiation ratio* and *fluctuation rate* variables that represent the average degree of parent-children and sibling-sibling rate of difference respectively.

In addition to the above stated factors, it also has to be noted that increasing the number of simulation scenarios will have a positive impact on the comprehensibility and precision of any scientific research. This rule is also applicable for our study. Currently we have defined six different scenarios with respect to changing values of two simulation constants. It is always possible to define new alternatives to run the simulation however; we believe that the identified scenarios are the ones that depict the features of our proposed algorithm in the best possible way.

6.3. Simulation Constants

Since Senvar and Bener algorithm requires the definition of global weight values for subsume and plug-in relations, in parallel with their assumption we define the constants as 0,5 and 0,8 respectively. In addition, we have defined several different simulation constants as follows:

6.3.1. Threshold Values

There are two different threshold values used in the implementation: The first one is used for checking whether the overall score of the service is enough to be listed and the second one is used for checking whether the score of the last explored concept is enough to explore its sub-tree. After numerous trials on values of each kind, we decided to set three different threshold values for the former case: *No Threshold* meaning that there is no bottom line for service scores; *Low Threshold* meaning that the threshold value is set to 0,1 and *High Threshold* meaning that the threshold level is set to 0,6. The latter threshold value on the other hand is set to 0,01.

6.3.2. Maximum Number of Steps

Starting from the target node we have also defined the maximum number of steps that search operation can be performed for simulation purposes. After several trials on the selection of an appropriate value, we have decided to set it as 10 for sub-tree and super-tree traversal and 4 for horizontal movement. Through our simulations we have observed that values greater than these may lead to performance burdens or out-of-memory exceptions.

6.3.3. Differentiation Ratio

Our algorithm assumes that each of the concepts existing in ontology is well defined and has its own set of properties. However vast of the concepts used within the test set does not obey this rule and there is not a hierarchical increase in the number of properties that an ontology has. In order to simulate well-defined concepts with already existing service set we have defined *Differentiation Ratio* as the average percentage of specialization between a parent and its children. We expect that usage of differentiation ratio will give us an idea about the performance of the algorithm under changing degrees of specialization. In order to simulate this assumption and staying within the limits that our service set allows, we have defined two different values for differentiation ratio. *Low Differentiation* meaning that value is assigned to 0,1 and *High Differentiation* meaning that value is assigned to 0,6.

Moreover in order to discriminate each child from one another we have defined *Fluctuation Rate* which is a randomly assigned variable that either increases or decreases the effect of Differentiation Ratio. By this way score of each children is guaranteed to be different.

6.4. Simulation Results

6.4.1. Query Set

Queries that are used throughout the simulation with corresponding inputs and outputs are listed in Table 6.3 for single ontology case and in Table 6.4 for multiple ontology case. For multi-ontology case query numbers are assigned as in the manual of OWLS-TC [29].

Table 6.3. Queries Used for Single Ontology Case

Query #	# of inputs	Ontology name	Input name	# of outputs	Ontology name	Output name
1	1	books	PrintedMaterial	1	concept	Price
2	1	books	RomanticNovel	1	concept	Price
3	1	books	Recommended-Short-Story	1	concept	Price
4	1	books	PrintedMaterial	1	concept	TaxedPriceInDollar
5	1	books	RomanticNovel	1	concept	TaxedPriceInDollar
6	1	books	Recommended-Short-Story	1	concept	TaxedPriceInDollar

Table 6.4. Queries Used for Multiple Ontology Case

Query #	# of inputs	Ontology name	Input name	# of outputs	Ontology Name	Output name
1	1	my_ontology	Car	1	Concept	Price
2	1	books	Book	1	Concept	Price
3	2	my_ontology	DVDPlayer	1	Concept	Price
			MP3Player			
4	2	my_ontology	Car	1	Concept	Price
			Bicycle			
6	1	concept	MaxPrice	1	my_ontology	Cola
9	1	concept	Recommended Price	2	my_ontology	Coffee
					my_ontology	Whiskey
10	2	books	Science-Fiction-Novel	1	Concept	Price
		books	User			
13	1	books	Title	1	my_ontology	ComedyFilm
14	1	books	Title	1	my_ontology	VideoMedia
15	1	portal	Researcher-In-Academia	1	Portal	Address
16	1	portal	University	1	Portal	Lecturer-In-Academia
18	1	portal	Publication-Number	1	Portal	Publication
19	1	books	Novel	1	Books	Author
22	1	travel	Surfing	1	Travel	Destination
23	2	travel	Hiking	1	Travel	Destination
		travel	Surfing			
29	1	books	Title	1	Books	Book

6.4.2. Clustering and Horizontal Movement

As mentioned in section 3.4.1 Senvar and Bener algorithm does not discriminate children of a concept in its scoring process. To prove this assertion we have simulated the

algorithm and considered same scored concepts as clusters. For our proposed approach, we have defined a cluster as a collection of two or more concepts where each couple in the collection has a score difference smaller than 0.01. We have measured the number of distinct scored concepts of each algorithm both for single and multiple ontology cases.

We have also collected the number of concepts scored due to horizontal movement along the ontology hierarchy during our simulations. Primarily we do not prefer to collect the number of services that are scored due to horizontal movement because this data can lead to false statements when there is not any corresponding service. Effect of horizontal movement and clustering are both depicted in Table 6.5 and Table 6.6. It is the ratio of distinctly scored concepts over total number of concepts scored which displays the ability of the applied algorithm to differentiate concepts in terms of similarity.

Table 6.5. Clustering and Horizontal Movement Comparison for Single Ontology

(IC: Input Concepts Scored, OC: Output Concepts Scored, DS: Distinctly Scored, HM: Horizontal Movement)

Query # LDNT	Senvar & Bener Approach						Proposed Approach					
	# of IC	# of DS	# of HM	# of OC	# of DS	# of HM	# of IC	# of DS	# of HM	# of OC	# of DS	# of HM
1	20	7		14	4		20	18	0	14	13	0
2	7	5		14	4		15	14	8	14	13	0
3	9	5		14	4		16	14	7	14	13	0
4	20	7		4	3		20	18	0	14	13	10
5	7	5		4	3		15	14	8	14	13	10
6	9	5		4	3		16	14	7	14	13	10

Table 6.6. Clustering and Horizontal Movement Comparison for Multiple Ontology

(IC: Input Concepts Scored, OC: Output Concepts Scored, DS: Distinctly Scored, HM: Horizontal Movement)

Query #	Senvar & Bener Approach						Proposed Approach					
	# of IC	# of DS	# of HM	# of OC	# of DS	# of HM	# of IC	# of DS	# of HM	# of OC	# of DS	# of HM
1	19	4	0	14	4	0	30	30	11	14	13	0
2	15	6	0	14	4	0	20	19	5	14	13	0
3	7	7	0	14	4	0	20	18	13	14	13	0
4	24	8	0	14	4	0	39	36	15	14	13	0
6	3	2	0	5	4	0	14	13	11	13	13	8
9	6	4	0	10	6	0	14	13	8	24	24	14
10	9	7	0	14	4	0	21	19	12	14	13	0
13	3	2	0	5	4	0	5	4	2	13	12	8
14	3	2	0	7	3	0	5	4	2	35	35	28
15	26	19	0	4	3	0	29	25	3	30	23	26
16	9	7	0	15	11	0	12	9	3	28	22	19
18	4	2	0	13	5	0	4	3	0	71	60	58
19	9	5	0	3	2	0	19	18	10	6	5	3
22	3	3	0	16	4	0	12	11	9	16	14	0
23	6	6	0	16	4	0	29	27	23	16	14	0
29	3	2	0	15	6	0	5	4	3	20	19	5

Analyzing Table 6.5 and 6.6, we have observed that on the average 54% of the output concepts and 34% of the input concepts that are scored are distinct and remaining portion of the concepts share same score level. This significant ratio apparently mentions the effect of ignored individual relations between concepts. On the other hand formation of input clusters is approximately 10% when the proposed approach is simulated where a cluster is defined as group of concepts within score proximity of 0.01.

Besides the mentioned effect of clustering, we have found that 46% of the input concepts and 36% of the output concepts are scored due to the horizontal movement on the average. If we calculate same ratio not for concepts but for resultant services this value becomes 27%. This means that candidate service set for any query is significantly expanded with the introduction of horizontal movement. We believe that horizontal movement will play an essential role for expertise searching in the network even if queries of the test set do not include such a case.

6.4.3. Scenarios

We have tested the above mentioned queries for 6 different scenarios of proposed approach and 3 different scenarios of Senvar&Bener approach. Table 6.7 represents the scenarios with corresponding simulation constants.

Table 6.7. Simulation Scenarios

	Proposed Approach						**Senvar&Bener A.**		
	NTLD	**NTHD**	**LTLD**	**LTHD**	**HTLD**	**HTHD**	**NT**	**LT**	**HT**
Threshold	*No*	*No*	*Low*	*Low*	*High*	*High*	*No*	*Low*	*High*
Differentiation	*Low*	*High*	*Low*	*High*	*Low*	*High*	*N/A*	*N/A*	*N/A*

For each of the above stated scenarios we have collected the number of services qualified, average input score and average output score for each query. Tables 6.8, 6.9 and 6.10 display the collected information for single ontology case whereas Tables 6.11, 6.12 and 6.13 display the collected information for multiple ontology case:

Table 6.8. Simulation Data for No Threshold Scenarios – Single Ontology

No Threshold	Senvar and Bener Approach			Proposed Approach LD			Proposed Approach HD		
Query #	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score
1	39	0.21	0.73	39	0.87	0.98	39	0.51	0.90
2	27	0.09	0.73	39	0.27	0.96	39	0.10	0.96
3	25	0.11	0.71	39	0.18	0.97	39	0.10	0.95
4	27	0.22	0.05	39	0.83	0.22	39	0.53	0.17
5	19	0.10	0.05	39	0.23	0.22	39	0.03	0.93
6	17	0.11	0.05	39	0.18	0.18	39	0.05	0.03

Table 6.9. Simulation Data for Low Threshold Scenarios – Single Ontology

Low Threshold	Senvar and Bener Approach			Proposed Approach LD			Proposed Approach HD		
Query #	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score
1	25	0.31	0.715	39	0.81	0.98	39	0.56	0.94
2	2	0.33	1	34	0.27	0.97	3	0.1	1
3	23	0.12	0.69	34	0.17	0.97	0	0	0
4	2	0.25	0.16	39	0.81	0.20	27	0.63	0.13
5	0	0	0	34	0.30	0.21	0	0	0
6	2	0.12	0.25	34	0.19	0.23	0	0	0

Table 6.10. Simulation Data for High Threshold Scenarios – Single Ontology

High Threshold	Senvar and Bener Approach			Proposed Approach LD			Proposed Approach HD		
Query #	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score
1	2	1	1	39	0.81	0.98	5	0.83	0.91
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

Table 6.11. Simulation Data for No Threshold Scenarios – Multiple Ontology

No Threshold	Senvar and Bener Approach			Proposed Approach LD			Proposed Approach HD		
Query #	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score
1	34	0.671	0.803	35	0.866	0.992	32	0.9065	0.9306
2	40	0.526	0.693	40	0.88	0.974	37	0.8632	0.9026
3	20	0.761	0.53	20	0.86	0.985	22	0.7311	0.8926
4	35	0.41	0.809	35	0.522	0.990	33	0.488	0.929
6	17	0.698	0.685	39	0.622	0.60	39	0.5364	0.5822
9	21	0.539	0.752	39	0.595	0.62	39	0.5833	0.5585
10	41	0.308	0.664	45	0.378	0.975	36	0.3322	0.9262
13	10	1	0.437	18	0.967	0.474	10	1	0.4616
14	8	0.906	0.562	26	0.979	0.482	26	0.9706	0.2632
15	13	0.49	0.70	17	0.642	0.883	14	0.69775	0.8253
16	8	0.511	0.9375	17	0.659	0.696	14	0.64815	0.6026
18	12	0.75	0.393	13	0.978	0.914	12	0.9572	0.8578
19	20	0.503	0.866	23	0.687	0.847	11	0.9614	0.8919
22	37	0.827	0.579	40	0.845	0.957	29	1	0.8917
23	40	0.545	0.585	40	0.569	0.966	40	0.5537	0.8856
29	8	1	1	8	1	1	8	1	1

Table 6.12. Simulation Data for Low Threshold Scenarios – Multiple Ontology

Low Threshold	Senvar and Bener Approach			Proposed Approach LD			Proposed Approach HD		
Query #	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score
1	16	1	1	32	0.9341	0.9821	32	0.904	0.9233
2	13	1	1	37	0.9406	0.9738	34	0.85	0.929
3	5	0.5	1	18	0.9296	0.9710	17	0.947	0.895
4	16	1	1	29	0.5870	0.9910	29	0.5563	0.9128
6	8	1	0.833	37	0.5999	0.61	8	1	0.952
9	8	0.833	1	23	0.706	0.8928	11	0.9865	0.8488
10	10	0.70	1	13	0.8179	0.9845	14	0.793	0.9228
13	5	1	0.733	10	1	0.6910	5	1	0.9263
14	3	1	1	26	0.9769	0.4686	6	1	0.816
15	5	1	0.8	15	0.718	0.877	15	0.6866	0.8517
16	4	1	0.875	12	0.736	0.680	6	0.824	0.98
18	3	1	1	13	0.98	0.931	12	0.94	0.976
19	9	1	0.851	22	0.7065	0.8456	9	0.99	1
22	17	1	1	40	0.857	0.96	37	0.86	0.86
23	19	0.602	1	32	0.656	0.976	32	0.65	0.90
29	8	1	1	8	1	1	8	1	1

Table 6.13. Simulation Data for High Threshold Scenarios – Multiple Ontology

High Threshold	Senvar and Bener Approach			Proposed Approach LD			Proposed Approach HD		
Query #	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score	Size of service set	AVG. Input Score	AVG. Output Score
1	16	1	1	29	0.989	0.984	29	0.96	0.96
2	13	1	1	34	0.966	0.977	31	0.90	0.952
3	6	0.958	1	17	0.96	0.96	13	0.976	0.89
4	1	1	1	7	0.8758	0.987	6	0.882	0.971
6	6	1	1	13	0.964	0.998	8	1	0.944
9	6	1	1	8	0.99	1	8	0.95	1
10	4	1	1	9	0.968	0.972	7	1	0.922
13	3	1	1	5	1	0.98	5	1	0.932
14	3	1	1	3	1	0.9266	5	1	0.895
15	3	1	1	6	0.99	0.99	6	0.97	0.97
16	3	1	1	4	1	0.984	4	1	0.9445
18	3	1	1	12	0.978	0.96	12	0.93	0.91
19	7	1	1	9	0.9822	1	9	0.963	1
22	17	1	1	29	1	0.96	27	1	0.93
23	4	1	1	10	1	0.938	5	1	0.959
29	8	1	1	8	1	1	8	1	1

6.4.4. Differentiation Ratio and Threshold Value

In order to properly measure the effect of changing differentiation ratio and threshold values we have grouped different scenarios and compare them.

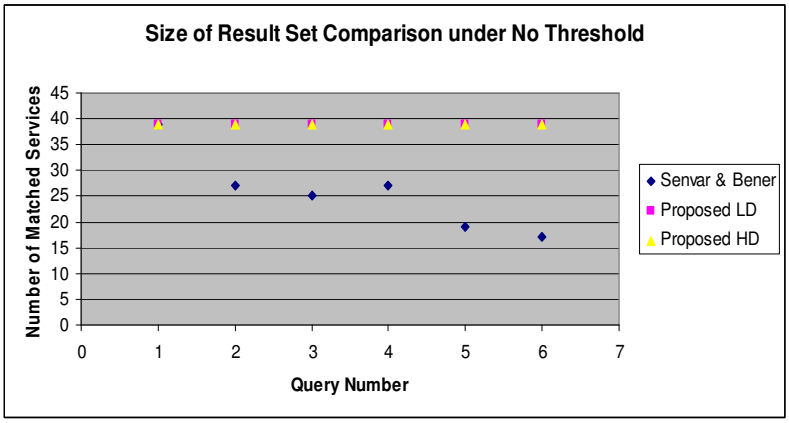

Figure 6.4. Effect of Differentiation Ratio for No Threshold – Single Ontology

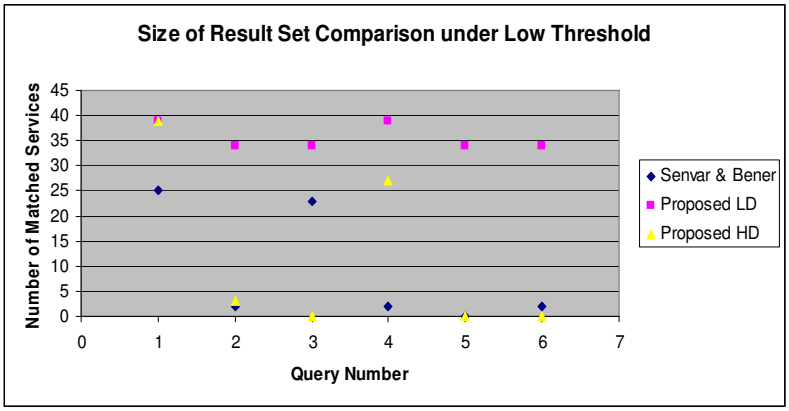

Figure 6.5. Effect of Differentiation Ratio for Low Threshold – Single Ontology

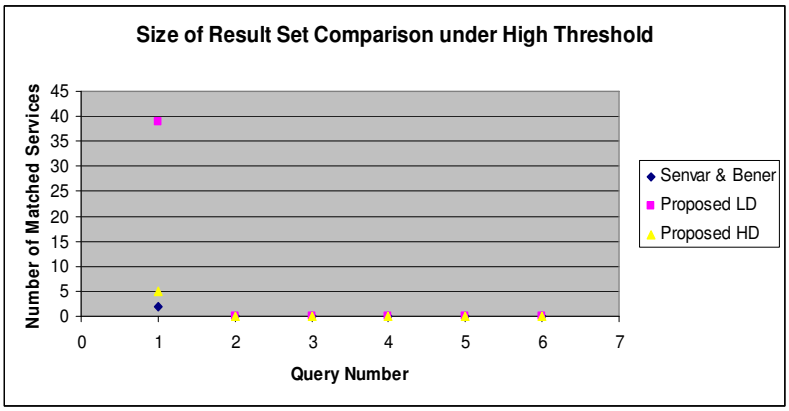

Figure 6.6. Effect of Differentiation Ratio for High Threshold – Single Ontology

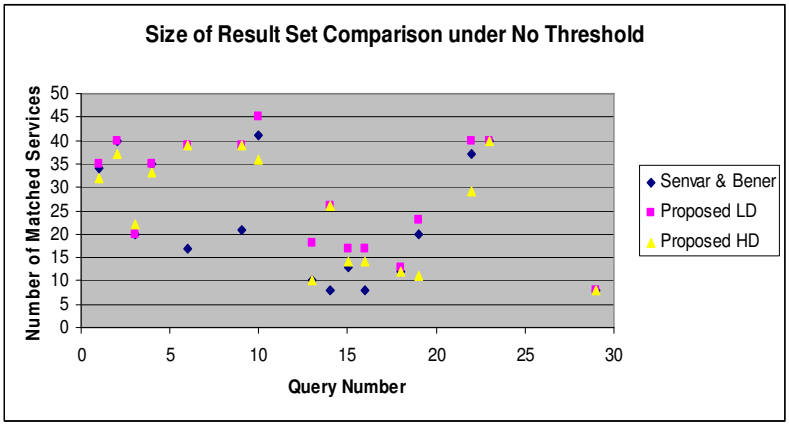

Figure 6.7. Effect of Differentiation Ratio for No Threshold – Multiple Ontology

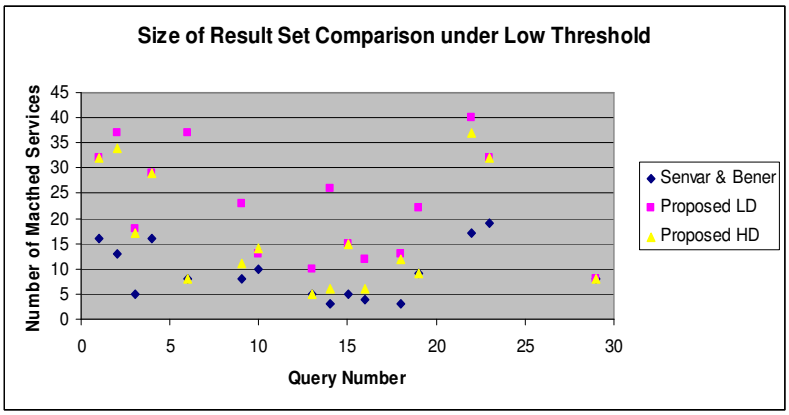

Figure 6.8. Effect of Differentiation Ratio for Low Threshold – Multiple Ontology

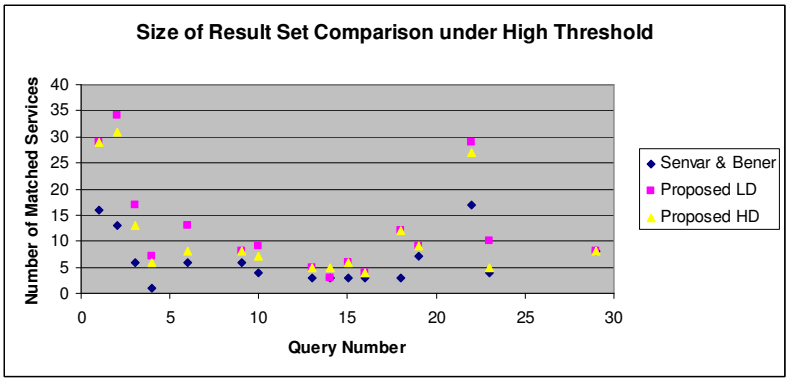

Figure 6.9. Effect of Differentiation Ratio for High Threshold – Multiple Ontology

Effect of differentiation ratio under constant threshold value is depicted in Figures 6.4, 6.5 and 6.6 for single ontology case and in Figures 6.7, 6.8 and 6.9 for multiple ontology case respectively. Analyzing the above figures our findings are stated as follows:

(i) Comparing proposed approach with low differentiation and proposed approach with high differentiation, it is observed that size of result set decreases as the differentiation rate is increased. In other words as the average degree of specialization of an ontology is increased, average similarity between compared concepts decreases, which in turn decreases service rating values. Once service ratings start to decrease, number of services that can qualify under a certain threshold value decreases and this situation drives result sets be smaller in size.

(ii) For No Threshold case, proposed approach with low differentiation has larger result set than Senvar&Bener approach and Senvar&Bener approach has a larger result set than proposed approach with high differentiation. In other words there is no significant value proposed approach brings in addition to Senvar&Bener approach under No Threshold case.

(iii) For Low Threshold and High Threshold cases, proposed approach with low differentiation has a larger result set than proposed approach with high differentiation and the smallest result set belongs to Senvar&Bener approach. Explanation of this increase in result set size is twofold: First, average rating calculated from proposed approach are higher than average ratings calculated from Senvar&Bener approach. In other words, usage of individual comparisons better reveals concept similarity than the usage of number of children as a similarity indicator. Second, services that are rated as a result of horizontal movement plays an important role in this increase.

(iv) As threshold values are getting higher, both approaches converge to a same size of result sets. In other words, for no or low threshold cases it is possible for a low-score subsumes service to qualify therefore there is a margin in between the results set sizes of two approaches. On the other hand for high threshold case only exact and some high-scored plug-in matches are able to qualify therefore both approaches returns the same exact match service set.

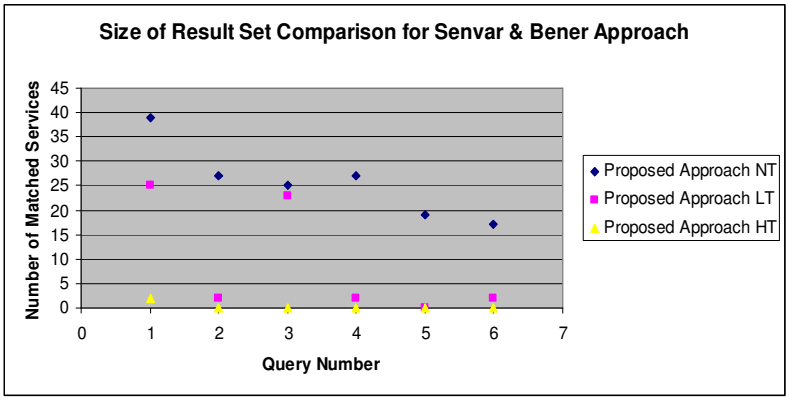

Figure 6.10. Effect of Threshold for Senvar&Bener Approach – Single Ontology

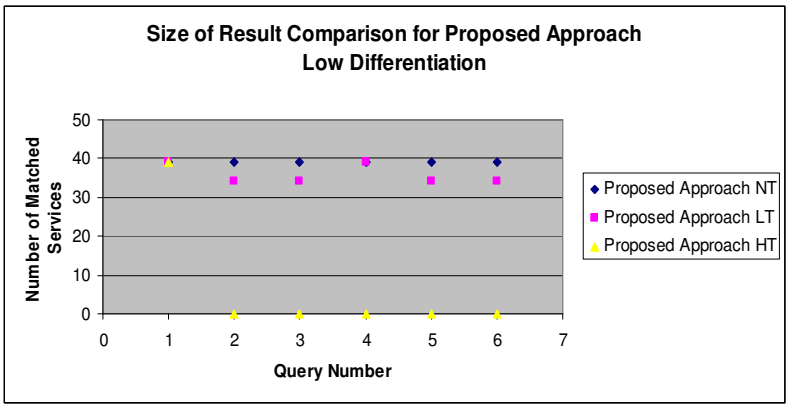

Figure 6.11. Effect of Threshold for Proposed Approach LD – Single Ontology

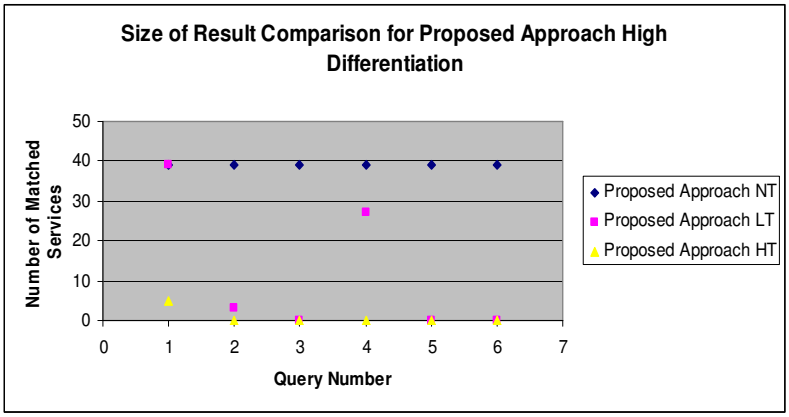

Figure 6.12. Effect of Threshold for Proposed Approach HD – Single Ontology

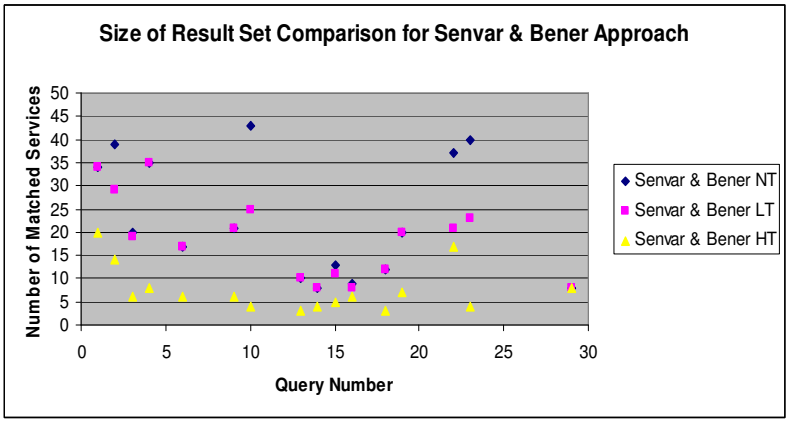

Figure 6.13. Effect of Threshold for Senvar&Bener Approach – Multiple Ontology

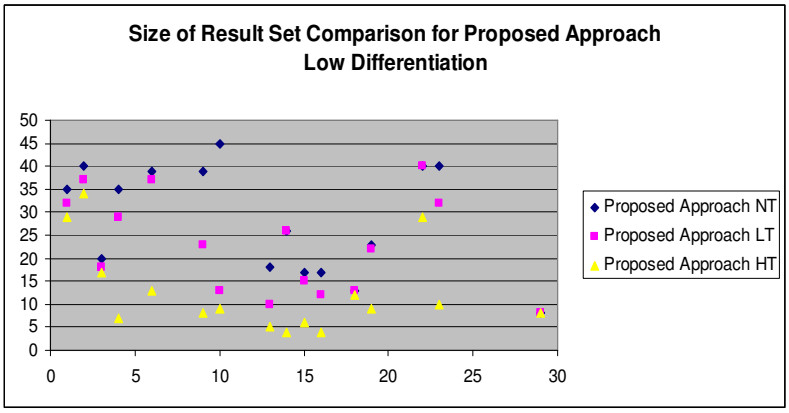

Figure 6.14. Effect of Threshold for Proposed Approach LD – Multiple Ontology

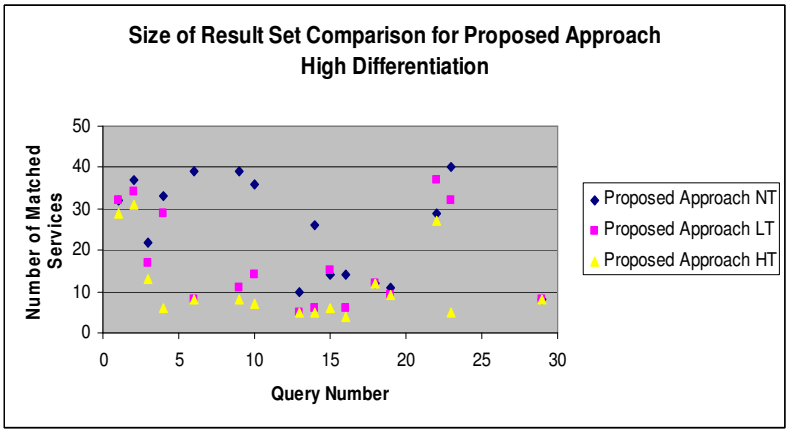

Figure 6.15. Effect of Threshold for Proposed Approach HD – Multiple Ontology

Analyzing Figures 6.10, 6.11 and 6.12 for single ontology case and Figures 6.13, 6.14 and 6.15 for multiple ontology case it is apparent that increasing the threshold value decreases the size of result set for both approaches. The amount of decrease is highest for Senvar&Bener approach and low differentiation has a higher decrease than high differentiation case for our proposed approach. This situation is also a consequence of scoring algorithm and horizontal movement as mentioned before.

6.5. Statistical Significance

In order to validate the results of our simulation we have performed Mann-Whitney Significance Test [86]. Mann-Whitney Significance Test is a non parametric test for assessing whether two samples come from the same distribution. The null hypothesis is that the two samples are drawn from a single population and therefore their probability distributions are equal. It requires two samples to be independent and the observations to be ordinal or continuous. The values observed from two different sample sets are first ranked and ordered, and then the difference between the observed and expected mean value is calculated. Transforming the random variable into a normal distribution variable completes the chain of calculations and degree of statistical significance is identified. The following table summarizes the calculations and formulas used for the Mann-Whitney Test for Senvar and Bener Approach and the proposed approach under low differentiation for high threshold case.

Table 6.14. Mann-Whitney Signifance Test

	Senvar & Bener	Rank	Proposed LD	Rank
	16	26,5	29	30,5
	13	25,5	34	32
	6	13,5	17	28,5
	1	1	7	16,5
	6	13,5	13	25,5
	6	13,5	8	19
	4	8,5	9	21,5
	3	4,5	5	11
	3	4,5	3	4,5
	3	4,5	6	13,5
	3	4,5	4	8,5
	3	4,5	12	23,5
	7	16,5	9	21,5
	17	28,5	29	30,5
	4	8,5	10	22,5
	8	19	8	19
Sum	197		328	
Mean	12,3125		20,5	
Standard Deviation	26,533		26,533	
Expected Sum	264		264	
z	-2,5066		2,223897	

The z value corresponding to 2.22 is below 0.02 for normal distribution which means that our proposed approach is found to perform better with 2% confidence interval. The application of the same test for other comparison cases falls in the range of 2-5% which proves the statistical validity of our study.

6.6. Multiple Inheritance

In order to simulate multiple inheritance in our simulation we have modified 2 of the original ontologies in the service set with 6 of the corresponding services. Figure 6.16 displays the final versions of the modified ontologies and Table 6.15 summarizes the ontologies and services modified, name and rating of the multiply inherited concept and matched service scores under high differentiation ratio and high threshold value.

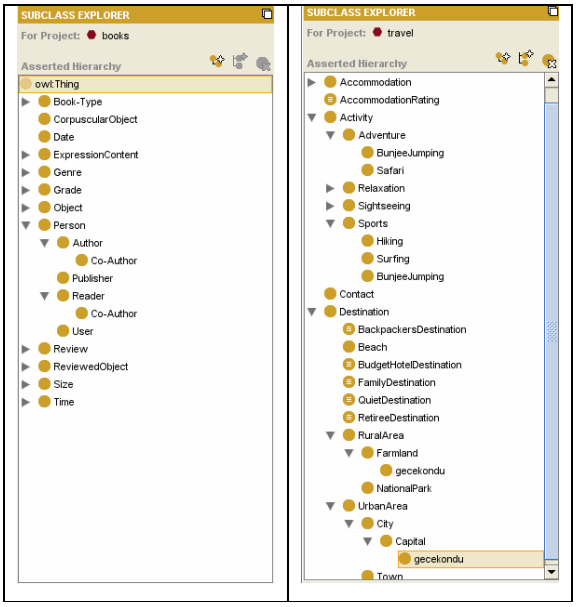

Figure 6.16. Modified Ontologies to Simulate Multiple Inheritance

Table 6.15. Multiple Inheritance in Simulation

Query Number	Modified Ontology	Name of multiply Inherited Concept	Rating of multiply inherited concept	Number of Matched Services
10	books	Co-Author	0.2274	7
22	Travel	BunjeeJumping	0.0163	22
23	Travel	BunjeeJumping	0.1338	10
19	books	Co-Author	0.7366	9
22	Travel	gecekondu	0.5597	26
23	Travel	gecekondu	0.6481	9

6.7. Evaluation

We have summarized the findings of our simulation with respect to the identified set of criteria in Table 6.16.

Table 6.16. Evaluation of Simulation Results

Evaluation Criteria	Property Matching	Semantic Distance
Heterogeneous Result Set	✓	
Size of Target Result Set	✓	
Percentage of Qualification	✓	
No Threshold	◯	◯
Low Threshold	✓	
High Threshold	✓	
Flexibility	✓	
Low Differentiation	✓	
High Differentiation	✓	

✓ : improved ◯: no significant change

7. CONCLUSION & FUTURE WORK

The main purpose of this thesis is to create a new ranking scheme for discovery of semantically enhanced web services. Ranking is directly related with concept ontology, where concepts are used to identify the entities both in input and output of a service. User request is also mentioned in the form of a sequence of concepts and matchmaking is done between what is requested by user and what is provided by service. The degree of match is referred as rating and classifying web services with respect to their ratings is referred as ranking.

Ontology, by structure is a tree-shaped hierarchy of concepts where concepts are related with subsumption relation. Rating of a service with given user request is evaluated with respect to the corresponding locations of service input and output in ontology. Regarding the location of a concept, there can be four different sets of relation as Paolucci et Al. suggests: Exact, Subsume, Plug-in, No Match. Our ranking approach also accepts this classification.

Ranking of concepts has to take the corresponding group of relation into account. In order to give numerical values to compare ratings, the concept of semantic distance is defined by Senvar and Bener. However, the concept of semantic distance assumes that degree of similarity between a parent and a child concept degrades once parent has more child concepts. To remedy this false assumption, our approach bases its ranking strategy on individual relations of concepts and defines rating between any concepts by accumulating the interim individual relation information. Similarity of individual relations is calculated based on the ratio of property numbers. We have specified rating formulas for different sets of relation groups accordingly.

Yet another point that we take into account is the horizontal movement within ontology. Existing ranking algorithms only considers vertical movement, which is a direct result of dominating subsumption relation. However, it should be noted that siblings also have a significant amount of resemblance with target concepts rather than most of the grand-children and grand-parents. To cover this missing point we propose to evaluate the

similarity of a sibling node by multiplying the ratings of corresponding individual relations.

In parallel with our approach in calculating ratings, we have focused on definition variations of concepts particularly in OWL-S. For each variation that affects the number of properties we have identified a way to calculate the similarity measure. By doing so, we aim to cover all the issues regarding similarity in OWL-S and form up a complete methodology.

The last point of our work on semantic web services focuses on usage of multiple inheritance. In object oriented design multiple inheritance generally refers to the re-usage of code fragments but for ontology construction composition of concepts is a more important point. This composition is done by merging the properties of component concepts and ontology languages such as OWL-S gives multiple inheritance support in this sense. In implementing our algorithm we find it worthwhile to keep an eye on multiply inherited concepts. For such concepts, rating of each path from target concept to the multiply inherited concept is calculated individually and maximum rating among different paths is taken into account.

To evaluate the success of our approach, we have implemented our approach in Java as well as Senvar and Bener algorithm and performed our tests on a sample set of semantic web services provided by OWLS-TC 2.1. The services in this set are not well defined in terms of property definitions. We have marked the concept of “well-defined” as follows: if a concept subsumes another concept, then it has to have more properties in number than the subsumed concept. However when we check the validity of our assumption, we observed that there were many cases ignoring this rule. To remedy the situation, we idealized our results by introducing two different variables to our algorithm. *Differentiation Ratio* is the average percentage of specialization between a parent and its children and *Fluctuation Rate* is an additional variable that guarantees each child to be different from one another. Moreover, to prevent performance burdens in ontology search, we define two different kinds of thresholds: one for maximum number of hops that a search operation can take place and one for controlling the minimum rating value so that search operation can continue. Finally we applied bipartite matching to catch the maximum

rating available in case there are multiple matches in service inputs (outputs) for a given user input (output).

Simulation on the service set showed us that assigning constant values for different relation types yields rating clusters to be formed in discovered services. Our approach takes care of this situation by taking only individual relations into account and its inductive strategy prevents cluster formations. In addition we have performed our test on six different scenarios, with respect to changing values of differentiation ratio and threshold value to see the effect of these parameters. In conclusion we observed that match scores and number of matched services above the stated threshold value dramatically increases due to the horizontal movement performed in the hierarchy and more results are returned to the user.

7.1. Contribution

Discovery constitutes up a major issue in the phenomenon of semantically enhanced web services. With common usage of semantic web services it becomes a fundamental need to select the most appropriate web service from a bunch of offered services. Ontology dependent ranking schemes for semantic web services emerge to meet this need. The proposed method has an inductive approach based on the individual concept-to-concept relations and overcomes the problems due to over-generalization of concepts for similarity measurement. Besides, it introduces the idea of horizontal movement to ontology searching and proves that siblings may have greater similarity than most of the grandparents and grand children. Proposed methodology also takes into account the concept of multiple inheritance in terms of concept matchmaking and gives a room for it in rating calculations.

7.2. Future Work

Although the research in this thesis proposes a novel and conceptual architecture for service discovery there is some additional work to be done. First, identification of similarity measure with respect to property numbers that compared concepts have is a valid methodology for many cases but exceptionally it is possible for some of the properties to have more importance than others in terms of uniqueness. To address this issue, it is

possible to add a new element to property descriptions indicating the corresponding priority of that property. In such a scheme, not only the number of properties would be a concern for similarity measurement but also the priority of properties would be taken into account.

Second, ranking of semantic web services should not only be a matter of technical similarity. User ratings and usage patterns also has to be included to reflect subjective ideas as well as the objective ones. Combination of historical data with technical analysis will improve the correctness of ranking but figuring out the corresponding weight values for historical data and technical similarity is another challenge to face at his point.

Third, the proposed approach has to be validated against real user preferences in order to compare and measure the quality of service ranking. This will not only give us a room to evaluate our approach with respect to subjective user evaluations but also bring the opportunity to evaluate the semantic acceptance of ontologies used since our proposed approach assumes that it is the responsibility of the ontology builder to create semantically valid ontologies.

8. REFERENCES

1. Martin, D., M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, "Bringing Semantics to Web Services: The OWL-S Approach", *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, California, USA, 2004.

2. Sajjanhar, A., J. Hou and Y. Zhang, "Algorithm for web services matching", *APWeb*, Vol. 3007, pp. 665-670, 2004.

3. Paolucci, M., T. Kawamura, T. Payne and K. Sycara, "Semantic Matching of Web Services Capabilities", *Proceedings of the First International Semantic Web Conference (ISWC)*, Sardinia, Italy, pp. 333 - 347, 2002.

4. Senvar, M. and A., Bener, *Ranking Semantic Web Services Using Semantic Distance Information*, M.S. Thesis, Boğaziçi University, 2006.

5. Uschold, M. and M. Gruninger, "Ontologies: principles, methods and applications", *Knowledge Engineering Review* 11, 93-155, 1996.

6. Sabou, M., *Building Web Service Ontologies*, M.S. Thesis, Information and Knowledge Systems, Dutch Graduate School, 2006.

7. Sirin, E. , B. Parsia, and J. Hendler, "Filtering and Selecting Semantic Web Services with Interactive Composition Techniques", *IEEE Intelligent Systems*, Vol. 19, No. 4, pp. 42-49, 2004.

8. Shin, H.W., E.H. Hovy, D. McLeod, and L. Pryor, "Generality: A New Criterion for Measuring Generality of Documents", *Tech Report 05-849*, University of Southern California Department of Computer Science, 2005.

9. Zhang, L. and J. Gu, “Ontology based semantic mapping architecture”, *Proceedings of 2005 International Conference on Machine Learning and Cybernetics,* Guangzhou, *IEEE Computer Society*, p. 2200-2205, 2005.

10. Knappe, R., *Measures of Semantic Similarity and Relatedness for Use in Ontology-based Information Retrieval*, Ph.D. Thesis, Department of Computer Science, Roskilde University, 2006.

11. *OWL-S: Semantic Markup for Web Services*, http://www.w3.org/Submission/OWL-S, 2004.

12. *OWL Web Ontology Language Reference*, http://www.w3.org/TR/owl-ref, 2004.

13. Shan, Y., T. Cargill, B. Cox, W. Cook, M. Loomis and A. Synder, “Is Multiple Inheritance essential to OOP?”, *ACM Sigplan Notices*, Vol. 28, No. 10, pp.360-363, 1993.

14. *Multiple Inheritance*, http://en.wikipedia.org/wiki/Multiple_inheritance, 2007.

15. *Massachusetts Institute of Technology, A Labeling Algorithm for the Maximum-Flow Network Problem*, http://web.mit.edu/15.053/www/AMP-Appendix-C.pdf, 2007.

16. Paolucci, M., K. Sycara and T. Kawamura, "Delivering Semantic Web Services", Technical Report CMU-RI-TR-02-28, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2002.

17. O'Sullivan, J. , D. Edmond, and A. H. M. ter Hofstede. “Service Description: A survey of the general nature of services”, *Technical FIT-TR-2003-01*, Queensland University of Technology, Brisbane, January 2003.

18. M. Klein et al., “Stepwise Refinable Service Descriptions: Adapting DAML-S to Staged Service Trading”, *Springer Lecture Notes For Computer Science*, p 178-193, 2003.
19. *The SGML History*, http://www.sgmlsource.com/history, 2002.

20. *IBM*, *SOA and Web Services*, http://www-106.ibm.com/developerworks/webservices, 2007.

21. *Simple Object Access Protocol (SOAP) 1.1,* http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2000.

22. Coyle, F., “XML, Web Services and the Data Revolution”, *Boston. Addison-Wesley Information Technology Series*, 2002.

23. Curbera, F. et al. "Unraveling the Web Services Web: An Introduction to SOAP, WSDL and UDDI", *IEEE Internet Computing 2* (2002), pp.86-93, Washington, 2002.

24. Chappell, D., “Understanding .NET: A Tutorial and Analysis”, *Addison Wesley*, 2002.

25. Klein, M. and A. Bernstein, "Towards High-Precision Service Retrieval", *IEEE Internet Computing*, Vol. 8, No. 1, p. 30-36, January 2004.

26. Zager, L. and G. Verghese, *Graph Similarity and Matching*, M.S. Thesis, Massachusetts Institute of Technology, 2005.

27. *Jena – A Semantic Framework for Java*, http://jena.sourceforge.net/, 2007.

28. *The Protege Ontology Editor and Knowledge Acquisition System,* http://protege.stanford.edu/, 2007.

29. *SemWebCentral: Project Info OWL-S Service Retrieval Test Collection,* http://projects.semwebcentral.org/projects/owls-tc/, 2007.

30. *Business Process Execution Language for Web Services 1.1,* http://www.ibm.com/developerworks/library/specification/ws-bpel/, 2007.

31. *Web Services Semantics – WSDL-S,* http://www.ibm.com/developerworks /webservices/library/specification/ws-wsdl-s/, 2007.

32. Horrocks, I., "DAML+OIL: A Description Logic for the Semantic Web", *IEEE Computer Society Technical Committee on Data Engineering*, 25(1), pp. 4-9, 2002.

33. *Resource Description Framework*, http://www.angelfire.com/anime3/ internet/data.htm, 2007.

34. Dean M., G. Schreiber, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider and L. Stein, "Web Ontology Language (OWL)", ht-tp://www.w-3.org/TR/owl-ref/, 2004.

35. Adam, T., "Web Services Technologies: XML, SOAP and UDDI", *.Net and Web Services Seminar European University,* Frankfurt, 2003.

36. Okkyung C., S. Han and A. Abraham, "Semantic Matchmaking Services Model for the Intelligent Web Services", *International Conference on Computational Science and Applications*, UK, IEE Press, UK, pp. 146-148, 2006.

37. SOAP version 1.2, *w3c working draft 17 December 2001*, http://www.w3.org/TR/20-01/WD-soap12-part0-20011217, 2001.

38. Goldfarb, C., P. Prescod, F. Charles, "Charles Goldfarb's XML Handbook", pp. 32, *Prentice-Hall*, Upper Saddle River, New Jersey, December 2001.

39. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, http:www.w3.org/TR/xml, 2006.

40. *W3C XML Protocol Working Group,* http:www.w3.org/2000/xp/Group, 2006.

41. *XML Schema Tutorial,* http://www.w3schools.com/schema/default.asp, 1999.

42. *DTD Tutorial,* http://www.w3schools.com/dtd/default.asp, 1999.

43. *Universal Definition Discovery Integration*, http://www.uddi.org, 2004.

44. *UDDI Technical White Paper*, http://www.uddi.org/whitepapers.html, 2000.

45. Agarwal, S., S. Handschuh, S. Staab and Fensel, D. et al. (Eds.), "Surfing the Service Web", *ISWC*, LNCS 2870, Springer-Verlag Berlin Heidelberg, pp.211–226, 2003.

46. Van Der Weide, Th. P., "Information Discovery", http://osiris.cs.kun.nl/iris/web-docs/edu/ir1/ir1.pdf, April 2001.

47. Berners-Lee, T., "Information Management: A Proposal, in-house Technical Document", *CERN*, http://www.w3.org/History/1989/proposal.html, 1989.

48. Luke, S., L. Spector, and D. Rager, "Ontology-Based Knowledge Discovery on the World-Wide Web", *Working Notes of the Workshop on Internet-Based Information Systems at the 13th National Conference on Artificial Intelligence (AAAI96)*, 1996.

49. Fensel, D., M. Crubezy, F. van Harmelen and I. Horrocks, "OIL & UPML: A Unifying Framework for the Knowledge Web", *In Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI'00*, Berlin, Germany, August 20-25, 2000.

50. Berners-Lee, T., J. Hendler, O. Lassila, "The Semantic Web", *Scientific American*, May 2001.

51. Kivela, A. and E. Hyvonen, "Ontological theories for the Semantic Web", *[M]. Helsinki HIIT Publications*, pp. 111-136, 2002.

52. Borst, P. and H. Akkermans, "An Ontology Approach to Product Disassembly", *EKAW 97*, Sant Feliu de Gu5xols, Spain, October 15-19th, 1997.

53. Studer, R., V. R. Benjamins, and D. Fensel, “Knowledge engineering: Principles and methods”, *Data and Knowledge Engineering (DKE)*, Vol. 25(1-2), pp.161-197, 1998.

54. Burstein, M., J. Hobbs, O. Lassila, D. Martin, D. McDermott, S., McIlraith, S. Narayanan, M. Paolucci, T., Payne and K., Sycara, “DAML-S: Web Service Description for the Semantic Web”, *International Semantic Web Conference,* pp. 348-363, 2002.

55. Van der Aalst, W.M.P, “Don’t go with the flow: Web Services Composition Standards Exposed”, *IEEE Intelligent Systems*, pp.72-79, 2003.

56. McIlraith, S., T. Son and H. Zeng, “Semantic Web Services”, *IEEE Intelligent Systems*, Vol.16, No.2, pp.46-53, March 2001.

57. Stuckenschmidt, H., and M. Klein, “Integrity and Change in Modular Ontologies”, *Proceedings of the International Joint Conference on Artificial Intelligence-IJCAI '03*, pp. 900-905, Acapulco, Mexico, 2003.

58. Hakimpour, F., J. Domingue, E. Motta, L. Cabral and Y., Lei, “Integration of OWL-S into IRS-III”, *In proceedings of the First AKT Workshop on Semantic Web Services (AKT-SWS04) KMi*, The Open University, Milton Keynes, UK, December 8, 2004.

59. W3C, “RDF”, http://www.w3c.org/RDF/, Visited: 2007.

60. Fensel, D., J. Hendler, H. Lieberman, W. Wahlster, “Spinning the Semantic Web”, *MIT Press,* pp. 478, 2003.

61. Powers, S., “Practical RDF”, *Oreilly & Associates*, 2003.

62. *DAML Language,* http://www.daml.org/language, Visited: 2007.

63. W3C, “Web Ontology Working Group”, http://www.w3c.org/2001/sw/WebOnt/, Visited: 2007.

64. The OWL Service Coalition, “OWL-S: Semantic Markup for Web Services”, http://www.daml.org/services/owl-s/1.0/owl-s.html, 2003.

65. *Object Oriented Programming*, http://en.wikipedia.org/wiki/Object-oriented_programming, 2007.

66. “Principles of object oriented programming: Part 2: Inheritance and Polymorphism”, *Computer Science*, p.33-39, May 2005.

67. Deitel, M. and P. J. Deitel, *Object-Orinted Programming,* Deitel & Deitel, 2007.

68. Wiliam, H., "Composition and Multiple Inheritance in Object Oriented Design", *Proceedings of 2000 Conference on Object-Oriented Programming Systems, Languages, and Applications*, Minneapolis, 2000.

69. University of Waterloo, *Multiple Inheritance and Interfaces,* plg.uwaterloo.ca/~migod/246/lectures/08-MultipleInheritance.pdf, 2006.

70. Rosen, J. P., "Object Oriented Paradigms: OOD vs. Inheritance", *Ada-Europe Conference*, Athens, May 1991.

71. *Object Oriented Languages Basics – Java World,* http://www.javaworld.com/javaworld/jw-09-2001/jw-0907-java101.html, 2001.

72. Seiji, K., J. Aasman and S. Haflich, "OWL vs. Object Oriented Programming", *4th International Semantic Web Conference (ISWC 2005)*, 6-10 November, Galway, Ireland, 2005.

73. Hunzer, G., “Introduction to Web Services”, *Borland White Paper*, March, 2002.

74. Hyperion Solutions Corporation, *Web Services and Business Intelligence,* 2003.

75. Papazoglu, M., "Web Services and Business Transactions", *World Wide Web, Internet and Web Information Systems*, Vol. 6, pp. 49-91, 2003.

76. John, D., M. Stollberg, "Semantic Web Services for Business Process Management", *2nd International Conference on Internet and Web Applications and Services (ICIW 2007),* May 13-19, 2007.

77. Della Valle E., D. Cerizza, V. Bicer, Y. Kabak, G, Laleci and H. Lausen, "The Need for Semantic Web Service in the eHealth", *In W3C workshop on Frameworks for Semantics in Web Services*, 2005.

78. Liliana, C., J. Dominque, E. Motta, T. Payne, and F. Hakimpour, "Approaches to Semantic Web Services: An Overview and Comparisons", *1st European Semantic Web Symposium*, Heraklion, Greece, 2004.

79. Oundhakar, S., *Semantic Web Service Discoveriy in Multi-Ontology Environment*, M.S. Thesis, University of Georgia, Athens, Greece, 2004.

80. Broens. T., *Context-Aware Ontology Based Semantic Service Discovery*, M.S. Thesis, University of Twente, Netherlands, 2004.

81. Gruber, T., R., "Towards Principles for the Design of Ontologies used for Knowledge Sharing", *International Workshop on Formal Ontology*, March 1993.

82. Papadimitriou, C., K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", *Prentice Hall*, 1982.

83. Küster, U, H. Lausen, B. König-Ries, "Evaluation of Semantic Service Discovery – A Survey and Directions for Future Research", *2nd Workshop on Emerging Web Services Technology*, Halle, Germany, 2007.

84. *Massachusetts Institute of Technology, Combinatorial Optimization*, http://math.mit.edu/~goemans/18433S07/matching-notes.pdf, 2007.

85. Ilhan, E. S., A. Bener, "Improved Service Ranking and Scoring: Semantic Advanced Matchmaker (SAM) Architecture", *ENASE 2007*, Barcelona, Spain, 2007.

86. Lowry, R., *The Mann Whitney Test*, http://faculty.vassar.edu/lowry/ch11a.html, 2007.

87. *OWL-S Matchmaker*, http://www.cs.cmu.edu/~softagents/daml_Mmaker/damls_matchmaker.htm, 2006.

88. Sycara, K., S. Widoff, M. Klusch and J. Lu, "LARKS: Dynamic Matchmaking Among Heterogenous Software Agents in Cyberspace", *Autonomous Agents and Multi-Agent Systems,* Vol.5, pp. 173-203, 2002.

89. Li, L. and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology", *Proceedings of the Twelfth International World Wide Web Conference,* pp.331-339, Budapest, 2003

90. Umesh, B., K. Roshan, "Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching", IEEE International Conference, pp. 86-93, 2007