RANKING SEMANTIC WEB SERVICES USING SEMANTIC-DISTANCE INFORMATION

by

Mehmet Şenvar

B.S., Computer Engineering, Middle East Technical University, 2003

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

2006

ACKNOWLEDGEMENTS

Through this thesis study, I have been very thankful to many people for their support, help, guidance and ideas. First of them is my supervisor Dr. Ayşe Bener for her support, guidance and patience throughout the long stages of this study.

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 learnt a lot from them. I found the right direction of research with their help.

Continuing a thesis study was not an easy task while working in a professional business life. For their patience in this term, I would like thank to all my professional coworkers especially Özge Yalkut for her support. Also, I would like to express my special gratitude and love towards Elif Sinem Babacan for her sincere motivation.

Finally many thanks to my family for giving me motivation for my study.

ABSTRACT

RANKING SEMANTIC WEB SERVICES USING SEMANTIC-DISTANCE INFORMATION

World wide web will gain its full power when semantic meta-data descriptions are added and automatic processing of the web is realized. Web services gained importance in the e-commerce era in recent years. As the number of web services increase, discovering correct web services for user needs becomes a problem. An effective discovery is only possible when there is semantic information. Semantic web enables automatic discovery, invocation, composition and mediation of web services by users, agents and programs. The first step in semantic web is the effective discovery of web services.

Most of the current solutions and approaches on web service discovery and selection are limited in the sense that they are strictly defined, and they do not use the full power of semantic and ontological representation. This research proposes a general “semantic web service discovery framework” in which: Users or software agents should able to set the selection criteria as XML documents, defined in preliminary format, in an extensible and modifiable manner. A Matchmaking Agent (MS-Matchmaker), which is the main component of the framework, processes the service discovery query. The matchmaking is done via an efficient matching algorithm. The matchmaker will return a rated and ordered set of suitable web services to the service requestor who may be a human-user, software agent, component or other. Semantic distance of ontology concepts are the main criteria of rating the web services in parallel with user definitions, which is different from other approaches.

INDEX WORDS: Web Services, Semantic Web Service Discovery, OWL-S, SOAP, UDDI, WSDL, Semantic Web Service Discovery Algorithm, Semantic Distance

ÖZET

WEB SERVİSLERİ KEŞFİ VE SIRALANMASI İÇİN ONTOLOJİK YAKINLIĞIN KULLANILMASINA DAİR BİR SUNU

Dünya çapında ağ (World Wide Web) tüm gücünü anlamsal bilgi eklentileri yapıldığı zaman ve otomatik işlemler gerçekleştirilebildiği zaman kazanacaktır. Bu amaçla web servislerinin kullanımı son yıllarda elektronik ticaret alanında oldukça önem kazanmıştır. Artan sayıdaki web servisleri, bu servisler arasında aranılan ve doğru web servisinin bulunması problemini de beraberinde getirmektedir. Bu anlamda etkin bir web servis keşfi, anlamsal bilginin servislere eklenmesi ile mümkündür. Anlamsal web, web servislerinin otomatik olarak keşfini, çağırılmasını, birleştirilmesini ve ilişkilendirilmesini sağlamaktadır. Anlamsal webin ilk adımını bu servislerin etkin bir şekilde keşfi oluşturmaktadır.

Web servislerinin keşfi ve seçimi için sunulan yaklaşımların çoğu katı özelliklere sahip olması, anlamsal ve ontolojik yapının tüm gücünün kullanılmamasından dolayı kısıtlı imkanlara sahiptir. Bu tez çalışması, ilgili anlamlarda anlamsal web servisleri keşfi için genel bir çerçeve sunmaktadır. Kullanıcılar ve yazılım ajanları arama kriterlerini ve isteklerini XML dokümanları olarak, belirlenen formatta genişletilebilir ve değiştirilebilir şekilde tanımlayabilmelidirler. Sunulan mimarinin en temel bileşenlerinden biri olan, MS-serviseşleştiriciolarak adlandırdığımız eşleştirici ajan, servis arama sorgusunu işler ve değerlendirir. Servis eşleme etkin bir eşleme algoritması kullanılarak yapılmıştır. Eşleştirici ajan, servis talebinde bulunan insan, yazılımsal ajan veya yazılım bileşeni olabilen servis isteyicisine arama kriterleri ile en uygun şekilde eşleşen servisleri dereceli ve sıralı bir şekilde döner. Ontolojik kavramların anlamsal yakınlığının ve kullanıcının önceliklerinin servis isteyicisi tarafından tanımlanması ve bunun servis eşlenmesi sırasında kullanılması çalışmamızı diğer ilgili çalışmalardan ayıran temel özelliktir.

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
LARKS	Language for Advertisement and Request for Knowledge Sharing
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
RETSINA	Reusable Task Structure-based Intelligent Network Agents
SOAP	Simple Object Access Protocol
SW	Semantic Web
SWRL	Semantic Web Rule Language
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

1.1. Motivation

In recent years, web services became the dominant technology in providing the interoperability among different systems throughout the web. Web services are based on standardized internet protocols such as XML [1], WSDL [2] and SOAP [3]. These protocols are highly accepted and used [4, 5]. There are also mature tools and technologies for developing, deploying and invocing web services. As the usage of web services increase, the number of avaliable web services also increase leading to the problem of finding the right web service to match user needs.

If web service is used in limited business domain or with strict rules with known business partners everything will be fine. The problem of finding the right and most suitable web services for user needs emerges when open e-commerce systems are widely used and user requirements dynamically change over time.

Although there are currently proposed technologies for discovery of web services, such as UDDI [5], they do not satisfy the full discovery requirements. This discovery process is based on syntactical matching and keyword search that does not allow the automatic processing of web services.

To solve the problem of automatic discovery and processing of web services, the Semantic Web [6] vision is proposed. The Semantic Web is an effort by the W3C consortium [7], and one of its main purposes is to facilitate the discovery of web resources. As defined in the road map of semantic web [8] *:"The limitations of the Web Services infrastructure can be overcome by providing a semantic markup to the descriptions of Web services to take advantage of the information available on the Semantic Web"* [9]. To achieve this through shared conceptualizations (so-called ontologies) these web resources are given a pre-defined meaning and they will thus be machine understandable.

There are different efforts and frameworks for semantic annotation and discovery of web services [10, 11, 12]. Three main approaches have been driving the development of Semantic Web Service frameworks: IRS-II [10], OWL-S [11] and WSMF [12]. For web service discovery they also propose some techniques and algorithms. However, they mostly classify the discovered web services in set-based. They do not focus on rating the web services using semantic distance information [13].

Improvement in matching process could be gained by the use of ontological information in a useful form. With the use of this information, it can be possible to rate the services found in discovery process. As in real life, users/agents should be able to define how they see the relation of ontological concepts from their own perspective. Similarity measures have been widely used in information systems [14, 15, 16], cognitive science, software engineering and AI [17, 18, 19]. So integration of knowledge from these techniques can improve the matching process.

The goal of this research is to propose a more accurate discovery architecture using the ontological distance information defined and ranked by users. Current solutions and architectures do neither rank nor use ontological distances. Although some frameworks use this information, they use it as limiting the search level in ontological representation [20]. By using semantic distance definition information, we aim to get a rated and ordered set of web services as the general result of the discovery process. We believe that this would be better than set-based classification of discovered services. We developed the architecture in a layered form to provide further extensions to add new layers and plug-ins as they are required. The focus of the research is mapping semantic-distance weights of ontological concepts and using this information to run the matching algorithm to form the rated set. By this way, the agents and users would easily define their view of similarity and importance of concepts. Since this weight assignment is done on the ontological representation, it will be a uniform weight assignment process, and it does not require a complex task. We call the matchmaker as MS-Matchmaker, which applies the *matching algorithm* as it tries to match a web service advertisement against a service requirement.

1.2. Related Work

Currently, several algorithms and architectures are proposed for semantic web service discovery and matchmaking [20, 21, 22, 23]. However, most of them are strict architectures and they have limited capabilities. Some of them are based on formerly accepted semantic concepts such as DAML-S [24]. DAML-S is today extended as OWL-S [25] and OWL-S is widely accepted as opposed to DAML-S. Today the World Wide Web consortium (W3C) has recommended it. Current approaches also mainly focus on only input and output matching. They do not use all information of ontological representation in matchmaking either. This might be an important source of information.

In the sub sections, we will describe the current matchmaking solutions and architectures in the literature.

1.2.1. OWL-S Matchmaker

The OWL-S Matchmaker [20] was developed by the Intelligent Software Agents Group [26] at Carnegie-Mellon University. Semantic Matchmaker, an entity that will allow web services to locate other services, provides a solution to the problem of matching and allows for full implementation of interpretative service providers on the web [20]. Here they introduce OWL-S, an OWL-based language for describing service capabilities.

The matching engine of the matchmaking system contains five different filters for namespace comparison, word frequency comparison, ontology similarity matching, ontology subsumption matching, and constraint matching [20]. The user configures these filters to achieve the desired tradeoff between performance and matching quality.

General architecture is also based on input output matching using ontological information and subsumption relation. It also sets constraints as how much to go in similarity of concepts on the ontologic knowledge base. It does not use this information in rating and ordering of discovered services. Also the degree of similarity and path-length of *subClassOf* relation is not used in rating phase of services. The services that does not match in input/output is directly eliminated with the classificaion *fail*. The result is also

returned to the service requestor as a set based list, which is defined as *S = (exact, subsume, plug-in, fail*) where failed ones are classified but they are not known by the requestor.

1.2.2. LARKS

One of the former approaches on web service discovery is LARKS (Language for Advertisement and Request for Knowledge Sharing) [21] system proposed by Sycara et al. LARKS tries to offer a flexible architecture in a scalable manner. This system is used in a multiagent infrastructure named RETSINA (Reusable Task Structure-based Intelligent Network Agents) [27]. It offers the option to use application domain knowledge in any advertisement or request by using local ontology, written in a specific concept language ITL (Information Terminological Language), to describe the meaning in a LARKS specification. The matching process uses five different filters: context matching, profile comparison, similarity matching, signature matching and constraint matching [21]. Different degrees of partial matching can result from utilizing different combinations of these filters. However, it is not based on currently accepted standards. This increases the problem of heterogeneity. It does not use preconditions and effects of web services in the architecture as well as quality of service criteria [28].

LARKS allow a set of filters that progressively restrict the number of advertisements that are candidates for a match. The filtering mechanism allows services to strike the most adventagous trade off between the precision of matching and the time required for a match: the higher the precision the longer the time matchmaker needs before delivering an answer.

1.2.3. Ian Horrocks and Lei Lui Architecture

Another solution proposed for matchmaking is Ian Horrocks and Lei Lui's architecture [22]. Their design and implementation of a service matchmaking prototype is based on DAML-S ontology and a Description Logic reasoner to compare ontology based service descriptions. A sample representation on DAML-S is given and some revisions on service profile is done to cover larger set of queries that can be matched. Also a matching algorithm is proposed here. The degrees of matches defined in their proposed solution are Exact, PlugIn, Subsume, Intersection, Disjoint. Degrees of the match are organized in a

discrete scale. Here also matching is based on inputs and outputs. However, it is not based on OWL-S and it eliminates any mismatches on inputs or outputs. There is no match value defined either. Ranking and selection is set-based and ontologic similarity and distance measures are not evaluated in this framework.

1.2.4. WSMF Approach

The WSMF framework [12] provides a general conceptual framework for web service discovery and processing. It is based on two main principles; strong de-coupling and strong mediation. Ontologies play an important role in WSMF. Goal, pre-condition and post-condition of the web services are basic factors for service selection. Mediation is also important at different levels such as data, busisness, process and protocol. Mediation provides ontology translation to support automatic interoperation between web services. Specifically, in the WSMO architecture various mediators (e.g., OO-Mediators) address the interoperability problems that arise when various web services work together.

WSMF framework defines the principles and the appropriate conceptual model for developing and describing web services and their composition (complex web services). It defines general aspects that have to be evaluated and given importance in semantic web service discovery and processing.

In our proposed framework, we realize the mediation only conceptually and just mention the mediation at the ontological level. We use the OWL-S *sameClassOf* relation to evaluate and form mappings between different ontologies.

1.3. Outline

In the next chapter, we give background information about the topics related to our work in this research. These topics and concepts form the basis for our study. They are widely accepted dominant technologies in web services, semantic web and service discovery domains.

In chapter 3, we will define the problem of semantic web services discovery and selection. Here also some sample scenarios will be given to clearify our motivation.

In chapter 4, general properties, definitions and architecture of the proposed framework is given in detail that forms the base for the matching algorithm.

Matching algorithm with service category matching, input matching, output matching, pre/post-condition matching, plug-in and maximum number filtering is described in detail in chapter 5. Here the semantic distance information usage in input/output matching is described in detail with pseudo-code of the algorithm.

In chapter 6, we describe the simulation environment tools and the cases to be evaluated for our proposed framework. We present the numerical results, selection lists and their ranking on sample simple ontologies.

In chapter 7, we evaluate our algorithm in selection and ordering; discuss the problems in simulation and evaluation.

In chapter 8, we compare our framework and algorithm with other approaches and compare the service discovery/selection results with other algorithms result conceptually.

Finally, in chapter 9, we present a summary of the research and its contributions, future work and conclude this thesis.

2. BACKGROUND

In this Chapter, we provide a brief overview of enabling technologies for semantic web and semantic web services discovery. Topics and technologies described below are dominant and largely accepted technologies today.

2.1. Web Service Discovery Architectures

Based on this semantic information, web service discovery architectures are mainly divided into three main categories, which are matchmaking, brokerage and P-2-2 architectures [29]. Matchmaker is mainly the simplest and most general kind of architecture. These architectures and solutions have different privileges and they are suitable for different kind of web service discovery needs. For example in contrast to a broker agent, a matchmaker does not deal with the task of contacting the relevant providers, transmitting the service request to the services provider and communicating the results to the service requestor in advance.

To define basically, *"Matchmaking is the process of finding an appropriate provider for a requestor through a middle agent, and has the following general form: (1) Provider agents advertise their capabilities to middle agents, (2) middle agents store these advertisements, (3) a requestor asks some middle agents whether it knows of providers with desired capabilities, and (4) the middle agent watches the request against the stored a advertisements and returns the result, a subset of the stored advertisements".* [29]

A matchmaker/yellow-pages is a middle agent that stores capability advertisements that can then be queried by requesters [29]. The requesters then choose and contact any provider they wish.

Although matchmaking systems seems to have simple and clear definition and functionality, the importance of matchmaking is high in the development of semantic web [30]. Since it forms one of the main steps in web service discovery it provides an open architecture for matching required services with the advertised ones. The matchmaking

process should be extendable, efficient, general and modular to answer different needs of service discovery problem [31]. In our research, we focused on some important aspects of matchmaking.

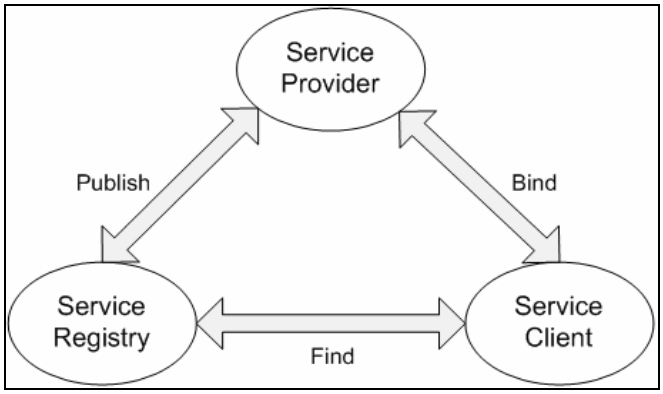

Figure 2.1. Basic Service Oriented Architecture

2.2. XML

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere [6].

XML provides a data format for documents and structured data, but does not specify the semantics of the format. To share information on different application and enterprises, shortly for interoperability, a shared set of terms defined in a common understanding is described.

XML forms the basic structured format for definition and data exchange with extensions on it using special constructs. All this semantic knowledge sharing, ontology and web services are based on XML documents having a special construct.

2.3. SOAP

SOAP [3] provides a way for applications to communicate with each other over the internet, independent of the platform and operating system.

It defines an envelope for transmitting messages and rules representing remote procedure calls. SOAP messages consist of a Body and Header parts. It is the commonly accepted simple, lightweight protocol for exchanging XML messages over the web using HTML, SMTP and SIP.

2.4. WSDL

The Web Services Description Language (WSDL) is an XML language for describing a programmatic interface to a web service [2]. A WSDL document describes a web service, its location, functionalities and input data types as well as the format of return values. However, WSDL does not support non-functional information about web services and no semantic description can be gathered from the WSDL documents. It just provides a syntactic representation of web service operation. It is standardized and formally accepted for syntactic description of web services.

The concepts defined in WSDL are the operation, message, type, port type, binding port and service. These are syntactical definitions about the web service and they do not cover semantic meaning abut the web services for discovery. WSDL aims provide an interface for the web service to be called by other users. WSDL and SOAP forms the basic description and messging parts of the Web Services Stack as given in Figure 2.2.

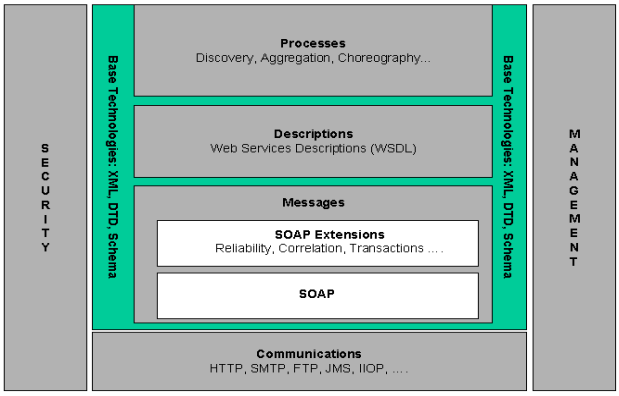

Figure 2.2. Web Services Stack

2.5. UDDI

UDDI (Universal Description and Discovery Interface) [4] is a cross-industry effort driven by a set of major platform and software providers, as well as marketplace operators and e-business leaders [6]. The aim of UDDI is to create a global, platform-independent, open framework to enable businesses to discover each other, define how they interact over the Internet and share information in a global registry that will more rapidly accelerate the global adaptation.

Through the UDDI registries, web services are defined on three different conceptual structures: white, yellow and green pages [5]. White pages cover general service Provider Company and contact information with general service description. Green pages consist of the information businesses use to describe how other businesses can conduct electronic commerce with them. Yellow pages are name-value pairs defined by structures named t-attributes to allow valid taxonomic identifier to be attached to the white pages as an extra.

UDDI provides an API for registering, publishing and discovering web services to users, however it does not cover semantic information and discovery is done on the syntactical level. UDDI does not represent service capabilities and only models may be used to provide a tagging mechanism and search on them is performed by string matching on some fields defined [22]. Thus, it is of no use for locating services based on a semantic specification of their functionality. In addition, it does not define how the registries will be

differentiated on different service domains and how the architectural location of different UDDI registries will be formed.

2.6. RDF

The Resource Description Framework (RDF) integrates a variety of applications from library catalogs and worldwide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events using XML as interchange syntax. The RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web [32].

RDF was developed because XML lacks formal semantics. RDF provides a framework, which allows encoding any type of information into "resources". A resource is an entity being described. It is given by a Uniform Resource Locater (URI). An RDF-file can be seen as a graph, the nodes of which are resources and atomic values. The links are properties. A RDF-file can also be seen as a set of triples of the form (resource, property, resourcelatomicValue). A sample RDF document is given in Figure 2.3 with mainly *subClassOf* property.

There exist several programs, which can read RDF-documents, among others SiLRI (Simple Logic-based RDF-Interpreter), SiRPAC (Simple RDF Parser and compiler), IMB's RDF, Perl RDF: Parser [33].

```
<rdf:RDF xml:lang="en"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntaxns#"

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description ID="MotorVehicle">
<rdf:type resource="http://www.w3.org/2000/01/rdfschema#
Class"/>
<rdfs:subClassOf
rdf:resource="http://www.w3.org/2000/01/rdfschema#
Resource"/>
</rdf:Description>

<rdf:Description ID="PassengerVehicle">
<rdf:type resource="http://www.w3.org/2000/01/rdfschema#
Class"/>
<rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="Truck">
<rdf:type resource="http://www.w3.org/2000/01/rdfschema#
Class"/>
<rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>


<rdf:Description ID="Van">
<rdf:type resource="http://www.w3.org/2000/01/rdfschema#
Class"/>
<rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="MiniVan">
<rdf:type resource="http://www.w3.org/2000/01/rdfschema#
Class"/>
<rdfs:subClassOf rdf:resource="#Van"/>
<rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
```

Figure 2.3. A Sample RDF document on Vehicle Ontology

2.7. OWL-S

"OWL-S is a OWL-based [11] 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 [9]. The aim of OWL-S is to semantically describe web services for discovery and invocation.

Three modules characterize OWL-S [11]: Service Profile, Process Model and Grounding. Three properties of the service class, as given in Figure 2.4, are:

presents: Service class *presents* a *Service Profile*. The *Service Profile* provides detailed description of the service and it is a provider in a human readable way. Inputs, outputs, preconditions and post conditions of services are defined in service profile. It is used for populating service registries, automated service discovery and hence crucial in the web discovery process.

describedBy: Service class is described by *Service Model*. The *Service Model* describes what the service does, how it works, and what functionality it provides as process.

supportedBy: Service class is *supported* by a *Service Grounding*. The *Service Grounding* provides information about service access specifications, such as; communication and transportation protocols. In other words, service grounding describes information about how a client can access the service.

OWL-S aims users and software agents to be able to discover, invoke, compose, and monitor web resources offering particular services and having particular properties, and providing a high degree of automation if desired, describing the web services semantically. It is generally accepted as semantic description language and layer for web services. It provides an API [35] for users and software agents to be used in the discovery process. So our proposed semantic web service discovery framework will be based on OWL-S semantic descriptions.

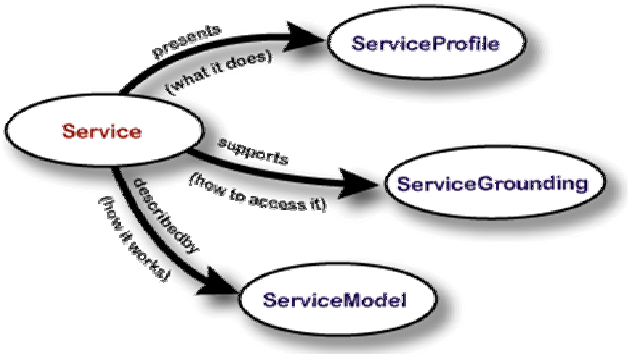

Figure 2.4. Top Level of the Service Ontology

Especially owl-s profile part is used in the matchmaking process of our work. Service category, inputs, outputs, pre and post conditions are used parts in the service profile.

2.8. Ontologies

"In the context of knowledge sharing, I use the term ontology to mean a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And it is certainly a different sense of the word than its use in philosophy." [36]

From the point of view of web services, ontology function as universal dictionaries and knowledge base so that all web services share the same interpretation of the terms contained in the messages that they exchange [37]. Ontology play a key role in the semantic web by providing vocabularies that is used by the applications in order to understand shared information. These ontologies are defined in standardized syntaxes such as RDF. As given in figure 2.5 Ontologies and RDF forms the middle layer of Semantic Web Stack.

However, there may be different ontologies defined in the same domain. Enterprises/agents may have a subset of a global ontology, which could be more useful for themselves. In this case, a third ontology may play the role of semantic bridge between the local and global ontology; which is called as the mediation [38]. This ontology mapping and matching is done via the Ontology Mediation component of our proposed framework. However, local ontology would require a mapping file to be defined to the global ontology. The mediation is an important part of the general framework, but it is outside the scope of our research in this thesis. For simplicity it will be mentioned but simple global ontology on different domains will be assumed.

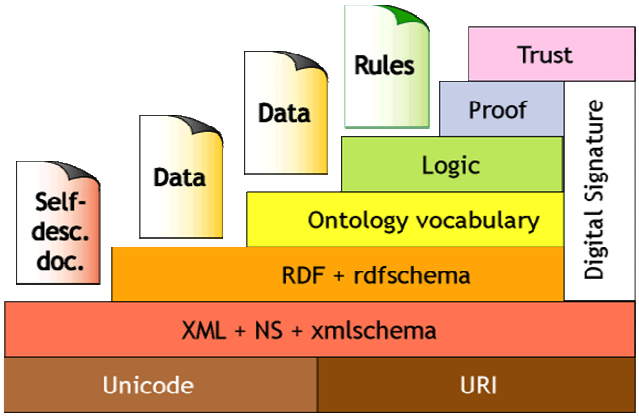

Figure 2.5. Semantic Web Stack

2.9. Semantic Web

The Web as it is known today provides its vast amount of information mostly in forms of human understandable web pages, i.e. plain text. As a consequence, the underlying structure of the data is not evident to computer programs or robots browsing the Web [42]. The Semantic Web (SW) approach means to develop languages and mechanisms for expressing information in machine understandable form. These languages emerge in form of ontologies that will describe web resources easily processed by computer programs. Useful resources about the Semantic Web can be found at semantic web review [39]. With the semantic web may activities such as publishing, discovery, composition will be possible (Figure 2.6).

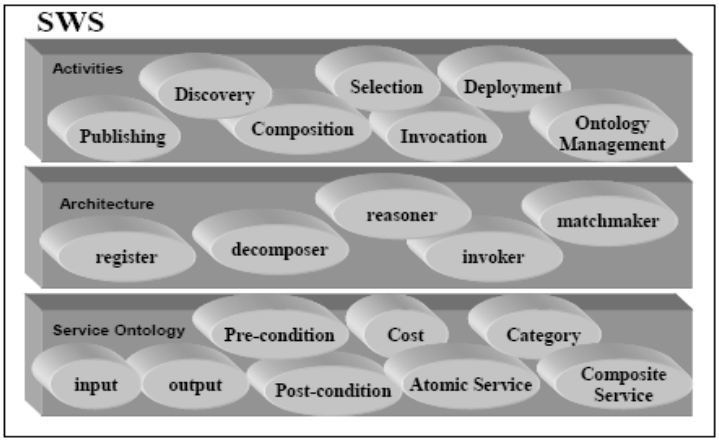

Figure 2.6. Semantic Web Services Infrastructure Dimensions

2.10. Summary

In this chapter, we presented a brief overview of enabling technologies for semantic web and web services discovery. These technologies are dominant today and most frameworks and research are based on them. We also base our proposed framework on these technologies and we aim to use all the available information in these data structures in a uniformed way.

3. PROBLEM STATEMENT

Web services have been proposed and largely accepted as a solution for interoperability in business transactions on the web. The goal of semantic web is that these web services will be machine understandable and processable. Using semantic annotations, web services discovery based on user preferences will be possible as a first step towards semantic web. However, today there is the big problem of finding the required semantic web services: with its general name "semantic web service discovery and selection" problem [65].

Today, existing state-of-the-art and accepted technologies for publishing and discovering web services (WSDL [2], UDDI [5]) does not use semantic descriptions and annotations. These technologies are based on syntactic matching of concepts and matching of some taxonomic relations. These are far from satisfying the requirements and they do not provide the goal of automatic services discovery [57].

There is an ongoing research on semantic web and semantic web services discovery as the semantic technologies mature [6]. Many researchers already proposed different solutions and frameworks for semantic web service discovery and selection [20, 21, 22]. Most of these approaches do the matching in a discrete scale and they do not use all the available ontological information. In addition, matchmaking algorithms mostly classify the resulting services on a pre-defined set and they do not offer any ranking or ordering to user/agent. Therefore, selection is mostly left to the user with little information provided.

There are also some other problems in the current solutions that are proposed. First, there might be cases where matching process returns a large set of services that meet the requirements partially. In this case, without an ordering, the service requestor cannot ensure to choose the optimal service and it has to re-evaluate the offered services in its view. Returning service results in a category of "match" or "does not match" or some other discrete-scale matching category causes the problem of choosing the best among the offered services. So that, if the matching process provides a ranked result instead of a discrete scale categorization, the requestor can decide the best matching service according

to these rankings. In addition, current solutions use the subsumption relation of concepts in matching but they do not use the subsumption level and distance of concepts in ontological representation. This important information should be used to get a better matching result.

Second problem in the proposed algorithms is threshold appliance [62]. By threshold appliance, we mean that many threshold values are required from the user to use in the matching process. It is hard for the user to know these values and which values to apply. Also based on this threshold values matching results may vary very much. Threshold appliance may cause some problems in the matching process as s result. Consider a matching result that returns either match or mismatch on a particular threshold of a matching degree. If this process returns no services, the service requestor might need to weaken this threshold value to get some services. This may need a re-execution of the whole matching process, which is a costly task. On the other hand, return of no results may be a result of no suitable service occurrence. In either case, service requestor does not have any chance of knowing the case so this causes an undeterminist situation. Therefore, threshold appliance might not be a right choice for matching.

In this research, we aim to provide an efficient and accurate matchmaking algorithm using the ontological-distance information in a general semantic web service discovery framework. To achieve this accuracy, we allow the users to specify their view of concepts, in a name user preference, which we name as semantic-distance on the local ontologies. Rather than just having a discrete scale service categorization for the discovered services, we provide a numerical result for the matching process. This matching process and ranking procedure is also based on user preferences and weights assigned by the user/agent. The second level of information used is the distance of class-subclass concepts, defined by the user/agent. Even if the user does no such assignment, we still use the subsumption level of concepts in ranking to give precedence and a higher rate to the services that are closer to each other semantically. This second level of semantic-distance information usage will provide a numerical ordering rather than discrete set-based rating and it gives a more accurate result.

In addition, analyzing different semantic web service discovery frameworks and approaches, we propose a general framework. This framework, for storing semantic

metadata at the back-end, offers a hybrid UDDI structure, parallel with the classification of services on higher service ontology and the matchmaking algorithm. Our proposed approach provides efficiency in the discovery process.

We base our framework and algorithm on the mostly accepted technologies that are ontologies, UDDI, OWL-S and other web service technologies. It would also be possible to integrate our proposed architecture to the existing models by making some modifications in the current algorithms.

As a result, we try to provide a simple, extendable, accurate and rank-based semantic web service discovery framework and an algorithm for the solution of semantic web service discovery and selection problem. We use the semantic-distance of ontological concepts to extend the amount of information to be used. Although our proposed approach may not cover all the cases and constraints on semantic web service discovery, it proposes a more accurate matching procedure using the semantic distance information. In this research, we do not cover the quality of services, performance and other related issues.

3.1. Sample Scenarios and Ontologies

To cover the main extension and suitability of our architecture and matching algorithm, we specify the following scenarios and application of our procedure on these cases. We kept the scenarios simple to focus on the main benefits of the algorithm with regard to other approaches.

Scenario I

We have a web site or a service and listing of some books to users. We have some money and we want to give ‘Price’ as input and get ‘Book’ as output (Book covers properties such as book information, ISBN, title, publish date). We want to get the list of books available for that price. Our web site users are mainly technology book users and we want to give preference to technology books. We want to sell books in YTL.

Scenario 2

A user/agent wants to buy a car from the web. He is interested in Sedan cars mostly, but he is open to other offers also, which may not be a Sedan car. He will ask for a 'Car' as an output, and he will specify the production year and color as the properties of the car as input. Therefore, conceptually the Car itself is also the input.

Scenario 3

A user is interested in reading books and he is interested in Technology Books. He also does not want to get any offer for other type of books such as history, political, social etc. Therefore, the main and only focus is on technological books and all the other options should be eliminated for this user. He wants to express this preference to the matching process.

Scenario 4

This scenario assumes that there are may services registered and we want to limit our search on service category of Payment Services and after the matching we do not want evaluate many number of services. We want to give a quick decision and have a short time to evaluate return service. Therefore, we limit the return number of services to five.

We based the example scenarios on user preferences and focused on using subsumption relation with semantic weight calculation with simple inputs and outputs. Constraint searches, such as looking for a car having price less than 1000 dollar or having two doors should be possible. It is possible with the current OWL-S descriptions on constraint and properties. Since our focus is on subsumption relations and semantic distance of concepts, we do not try to cover these cases directly. Nevertheless, with just some extension on matching process, integration of constraint matching can be possible.

We used the following ontologies in our sample scenarios (Figure 3.1, 3.2 and 3.3).

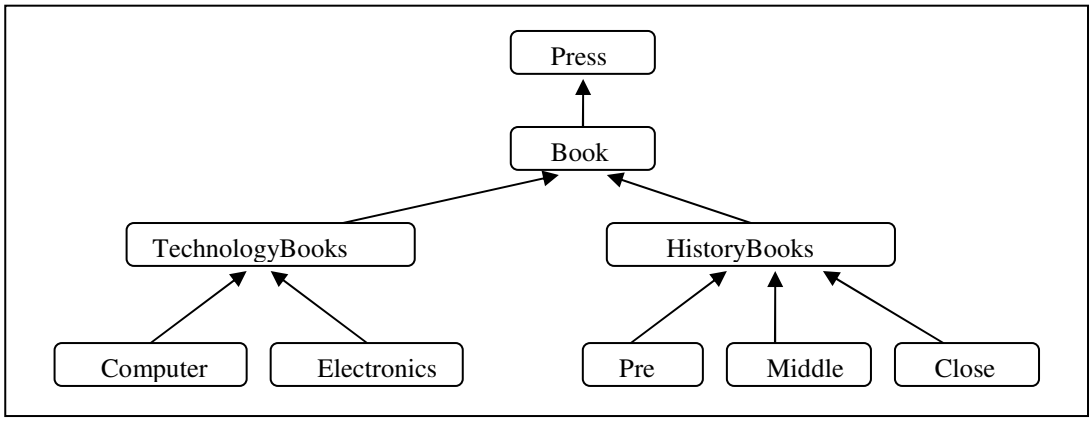

Figure 3.1. Sample Press Ontology

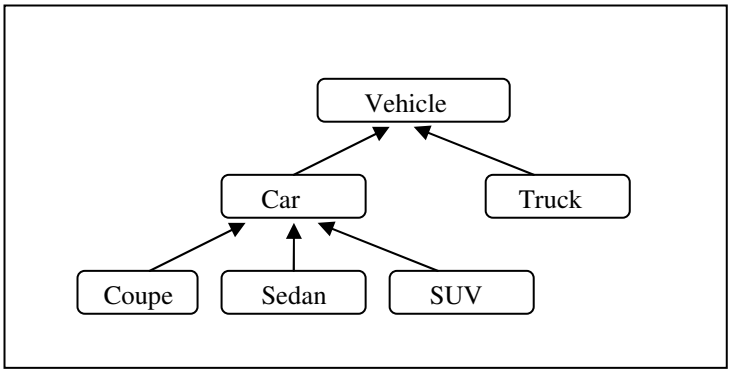

Figure 3.2. Sample Vehicle Ontology

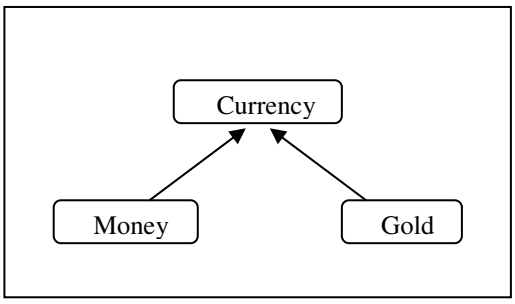

Figure 3.3. Sample Currency Ontology

4. PROPOSED SOLUTION

In this chapter, we give information about the underlying structure/architecture, general concepts, definitions and knowledge base of our proposed framework.

4.1. UDDI Architecture

Repositories form the backend of the service discovery architecture, forming a knowledge base to store semantic service descriptions and ontologies. In storing these semantic metadata of service descriptions, UDDI is widely accepted by researchers and technology users. As mentioned before, there may be different choices for these repository architectures, which are centralized, de-centralized and hybrid approaches which effect the structure of discovery. Having a centralized architecture is simple to implement and easy by a single point of administration but it may cause a single point of failure. De-centralized architectures (P-2-P) may have good performance and scales good, but overhead and administration problems occur because of the distributed nature. We propose a hybrid UDDI architecture focusing on the scalability and service category filtering of the matchmaking algorithm. There are many hybrid architectures [40] for repository structure. Our proposed approach, as given in Figure 4.1, is similar to these solutions.

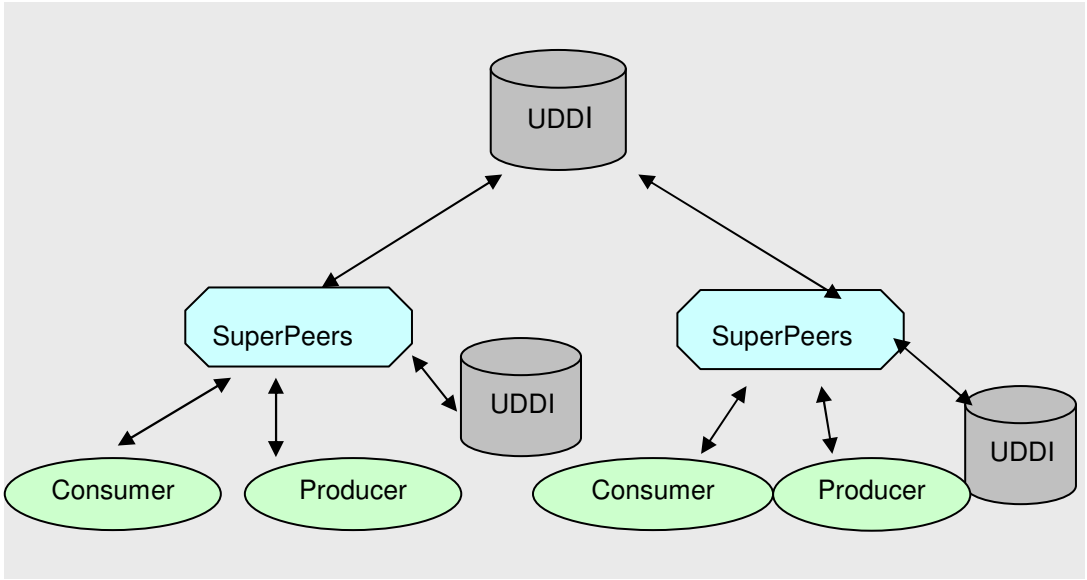

Figure 4.1. Proposed Hybrid UDDI Architecture

A similar architecture to our proposed UDDI architecture is GloServ [40]. The GloServ architecture spans both local and wide area networks. It maps knowledge obtained by the service classification ontology to a structured peer-to-peer network, as given in Figure 4.2, such as a Content Addressable Network (CAN) [40].

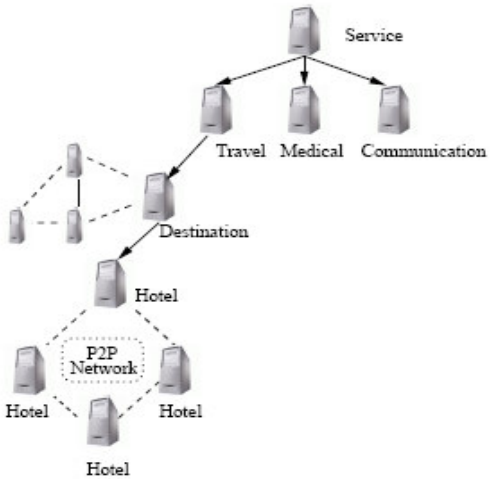

Figure 4.2. GloServ Architecture

In the figure below (Figure 4.3), we show a simple service category based UDDI distribution of web-services from the financial services domain. Here high-level service ontology is defined as financial services, banking and retirement services. Then the services are distributed to repositories based on this categorization.

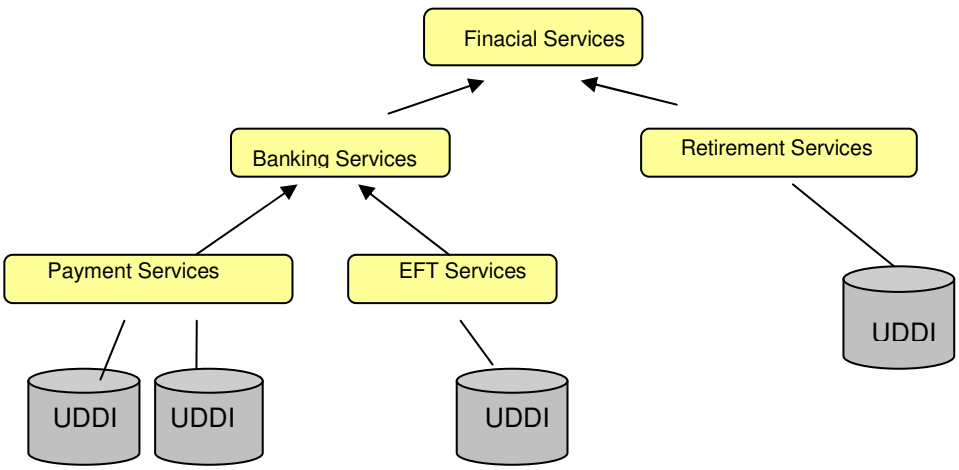

Figure 4.3. Service Categories Based Located UDDI Architecture

4.2. Local to Global Ontology Mapping (Mediation)

Semantic web requirement does not require all agents, users having all the same ontological representation [12]. Some frameworks, such as WSMF, also focus on mediation as one of their main components. Mediation structure in WSMF architecture is given in Figure 4.4. In our proposed framework, we also defined local/global, ontological representations. Since each agent may view the domain differently or it may use only a subset of the global ontology, local ontologies become necessary. As time goes by it is also required that in the predefined ontologies many will be eliminated and the ones largely accepted will remain [58].

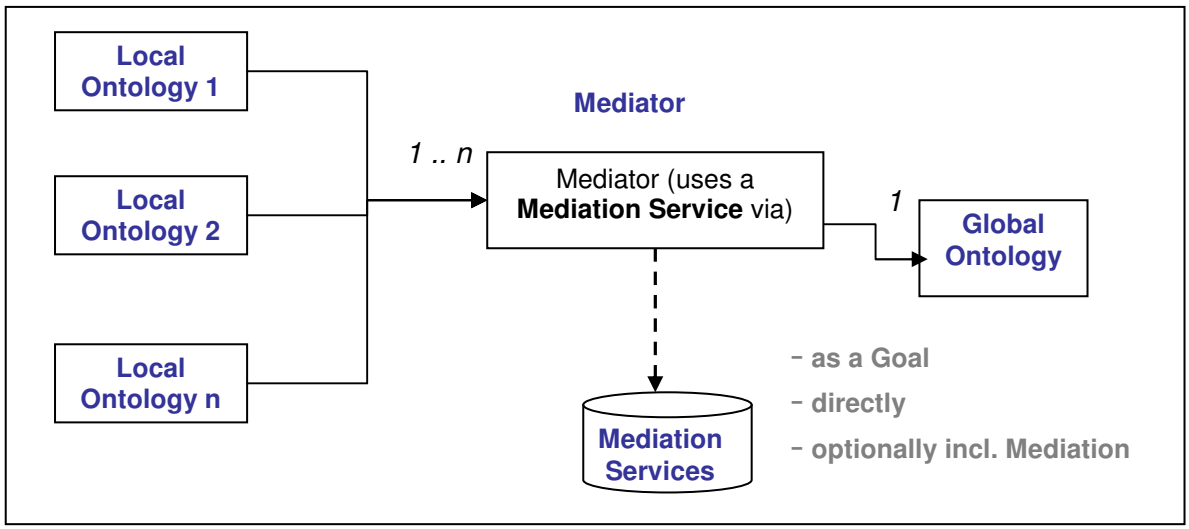

Figure 4.4. WSMO Mediation Architecture

By definition, mediation may be at different levels such as data, protocol, and process [12]. However, in our proposed framework, only ontological mediation is taken into consideration and an ontological mediator is required.

Although conceptually we mention mediation, we assume global ontologies accepted by all users/agents and all the simulations will be run on these ontologies.

4.3. Matching Algorithm

Matching algorithm aims to be a layered, extensible, simple and efficient algorithm. It is applied on two different domains. It applies different filters on the advertised services for discovery and selection of suitable service based on the service request. Firstly, the general OWL-S semantic information of web services, which are service profile, inputs, outputs, pre-conditions and post-conditions (IOPEs) are evaluated. Ranking and selection is done using the semantic distance information defined by the service requestor.

Different from other approaches, a set of partially matching set is also returned to the service requestor. This set will be important forming the required service when none of the existing services provide the needed request but it can be formed by composition of some services. Therefore, this set will be useful for the composition engine that can be designed as part of this framework. The composition of the required services from the set is outside the scope of the research in this thesis.

Layered structure of the algorithm allows the adaptation of user-defined plugins to the matching engine. These plug-ins might be Quality-of-Service plug-in, Location-based service filter plug-in etc.

4.4. Definition of Semantic Distance/Weight

In this research, we define the semantic distance as how the user/agent views relation of concepts and the world of domain from its perspective. Each agent may view the domain differently based on its needs and it may see or change the semantic distance of the concepts for its need in time to satisfy its requirements. Since this view of concepts is defined over the ontological representation of domain, it is easy to maintain and can be held as a separate information.

The semantic distance is a weight assignment on the ontological representation to the nodes having a *subClassOf* relation. Each subclass covers some set/range of the upper class and its range of domain or importance may change with agents view. For example in vehicle ontology, a vehicle class may have three subclasses as Car, Truck, and Van.

Weight assignment is done to these three nodes as how the agents prioritize them and give importance to them in search of web services. If cars are the interest of domain and mostly car operations are important, it may assign a higher semantic-distance (weight) value to the nodes of interest, such as 0.8 to node Car when cars are of interest. Others may have value as 0.15 and 0.05 respectively. In this situation web services having Car as input/output is rated highly in similar situation.

Semantic Weigt /Distance = (parent-class, sub-class, similarity-weight) (4.1)

Semantic distance concept is a research topic in knowledge systems [41] and some of discovery systems. However, it is mainly defined as the similarity of words. Additionally, semantic distance information is not used directly with a mapping and rating criteria in the matchmaking process with agent preferences. In this research, we use the term semantic distance and semantic similarity weight interchangeably, both having a higher value means that the concepts are semantically similar and close.

The accepted definition for this relation is Subsumption. The subsumption pattern is the most important for the proposed matching algorithm. Subsumption can be seen as the determination of subconcept and superconcept relationships between concepts of a given terminology. It means deciding whether one description C is more general than another description D, i.e. if being D logically implies being C. The more general concept C is also called the subsumer and the more specifc concept D the subsumee [42].

Subsumption allows one to structure the terminology in the form of a subsumption hierarchy. This hierarchy will then provide useful information on the relationships among the concepts of the terminology.

4.5. Setting Semantic Weights

In our proposed framework, the local agents/users make the semantic weight assignment on the local ontology. In addition, a case may exist where; the ontology managers do a general concept-similarity weight assignment on the ontological

representation. Based on the usage of global or global&local ontologies, there may be two cases:

Case I : Agents and users define their local ontology and assignment is done on this representation by local ontology managers (local users/agents).

Case II : Assignment is done on the global ontology by the Ontology Designer as shown simply in Figure 4.5. All users and agents use this ontology and similarity weight assignment.

Figure 4.5. Defining Ontology Mapping (Case II)

In our approach, such a weight assignment is not a 'must' condition. User may prefer not to define such an assignment. In this case, we still use the semantic similarity and semantic distance of concepts, but here we use the path-legth value between concepts to evaluate the distance for similar type matches.

4.6. General Rules of Concept Similarity

In the ontological representation, concepts can be in relation with other concepts: they can be placed over or below the other concepts. This relation is defined as subsumption relation and it is an important piece of information [42]. The level of being under or over is important as well. This path-length in tree based graph structure is in one sense the distance of the concepts. On this graph based ontological representation we can define and identify such knowledge. In our proposed framework, we only cover the situations where concepts occur on the path from root node to target node. Other sub concepts relationships that do not occur in this path may also be classified. Nevertheless, this would increase the complexity of the system and it would provide irrelevant classifications to the matchmaking system. We tried to cover basic cases and provide a simple solution for the ranking.

For further explanation of properties in this part, the ontological relations below will simply be assumed (Figure 4.6).

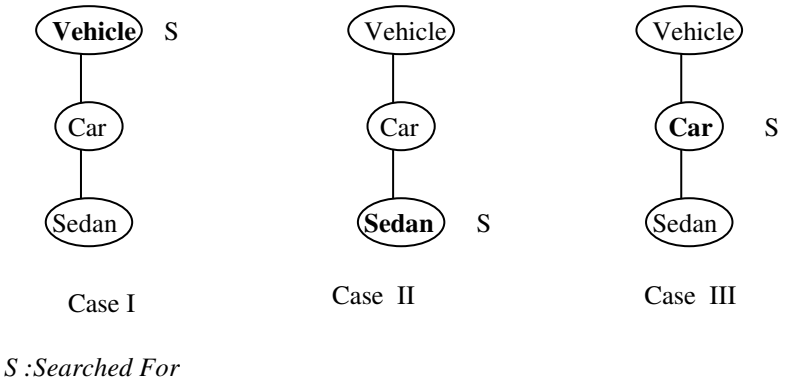

Figure 4.6. Sample Car Ontology for Concept Similarity

Axiom I : *Most strongest match is where advertised concept match with the the requested concept exatly.*

This is the natural criteria that we match with the input/output that is exactly the one searched for. For example while requesting a service providing car, an advetised service that provides car would be the best alternative.

Axiom II : *For the search result concepts under the target concept, the one that is upper in the ontologic representation is preferred.*

This is the situation in case I. While we are searching for a web service with parameter output Vehicle, let two choices be Car and Sedan. The service that provides output as Car is chosen since it is located at the upper level in the hierarchy of the ontology. As it gets near the target concept, its preference increase.

Axiom III : *For the concepts over the target concept, the one that is closer to the searched concept which is in the lower part of the ontologic representation is chosen.*

This is the situation in case II. While we are searching for a web service with parameter output Sedan, let two choices be Car and Vehicle. The service that provides

output Car is chosen since it is located at the lower level in the hierarchy of the ontology and it has a lower semantic distance to the target concept. This is similar to daily life were we usually choose specialized provider close to target domain rather then companies providing all kind of services.

Axiom IV : *For the concepts over the target concept and under the target concept, the one that is under the target concept (descendant) is choosen.*

This is a different case from axioms 1-3. Here we compare the concepts in different sides of the target concept. We give presedence to descendant concept intuitively because, specialization rather than generalization is preferred in semantic approach. As in real life situations, we would prefer a "Sedan" rather than any vehicle while requesting a "Car". This case is just idendified by the classification of match-type, being over as type plug-in and being under as type subsume. Other axioms are used in the ranking of same match-type matchings.

Our ranking will be based on the axioms defined in I, II, III for ranking same match types. A similar taxanomic properties is also discussed in Maximilian et. al. [43]. In their work they identified more cases that can be used in matching.

4.7. Semantic Distance Weight Assignment

We define the semantic-distance as the similarity of concepts, which may be the rate of coverage of sub-concepts for each concept in relation with *subClassOf* construct. For each tuple having a *subClassOf* relation a weight is assigned in the range [0-1]. The total weight of all subclasses of a parent concept should be 1. We have assumed to add all values up to one to provide a smoothness in evaluation and to restrict the users setting weight assignments unlogically. This assumption is just a choice of design and restriction on weight setting adding up to 1 may be eliminated if required.

This similarity weight assignment may be done by sub-ontology managers. These people define and assign the weights during the first development of ontologies. However, sometimes there may not be such a similarity rating. Even in this situation the properties

defined is valid and a homogenous weight assignment is assumed. For a concept having 3 subclass relations each may be assigned 0.33 (1/3) rating with the parent concept. This homogeneity will be assumed all over the ontologic representation.

In our proposed model, user/agents make such weight assignment on the ontologies according to their preferences and view-of-world on concepts. In this way, they manage the road of matchmaking and give preference on some services. We also eliminate the misdirections by making ontology homogenity assumption.

We hold the semantic distance information as a tuple relation.

SemanticDistance = (parent-concept subclass-concept, similarity-weight) (4.2)

Examples may be given as:

SemanticDistance 1 = (Vehicle, Car, 0.8)
SemanticDistance 2 = (Vehicle, Truck, 0.2)
SemanticDistance 3 = (Car, Sedan, 0.4)

This preference will be specified by the user/agent as an xml file named as *.sd (named for semantic distance) file and will be submitted to the matchmaker agent for the discovery process. The simple format of the file, for all class-subclass concepts, will be:

```
<Class>Book
        <Subclass>TechnologyBooks                    (4.3)
        <semantic-distance>0.8</semantic-distance>
        </Subclass>
</Class>
```

This weight setting is not hold in the ontologic representation beacuse it is different for each user and may change with time based on user needs. As well, it would require new constructs in ontology formation so we have developed weight assignment information as a seperate information file.

This similarity info will be used during the matching process in rating the input and output matching. Rather than just classifying as exact, subsume match etc, at the second level we use this similarity values to order the matching.

4.8. Partial Result Set

Most of the approaches for matchmaking eliminates the service advertisements when one or more of the inputs/outputs fail to the match with the requested service input/outputs [44]. However this can strictly eliminate the services that could just be the service that satisfies user with just a slight modification in the search. Even more, these services can be used in a compositon process to provide the service that the requestor searches for. So the services that match on some input/output and fail on the other is put into a *Partail Result Set* in our proposed framework. This set of services can be returned to the requestor also in the order of match type and value. Also a composer component in the framework can process these services and form a combination to satisfy the user requirement. In this research, we just mention this partial result set conceptually and form it in our proposed matching algorithm. Forming composed service form these partail services is out of the scope of this research. But it is an important point not to eliminate these services directly.

5. THE MATCHING ALGORITHM

Algorithms in literature are mainly based on matching of inputs and outputs and any mismatches are eliminated directly [20]. Also only a match level scale is defined without computing a result match number value (rank) and sorting of similar services is not clearly specified. Discovery is specified in discrete scale and offering based on a ranking is not provided to the service requester with a numeric value. In the proposed algorithm, these cases are taken into consideration within the layered structure of the algorithm. Steps described below are applied in order in the matchmaking process. In each step, the web services that are filtered pass to the next step and others are eliminated. The main focus is also input/output matching considering the semantic distance evaluation, which provides a second level information to be used in ranking similar match-type of services.

The general architecture of our matching algorithm and matchmaking engine, which we name as MS-Matchmaker, is given below (Figure 5.1). In this chapter, we will describe parts of the matching in detail.

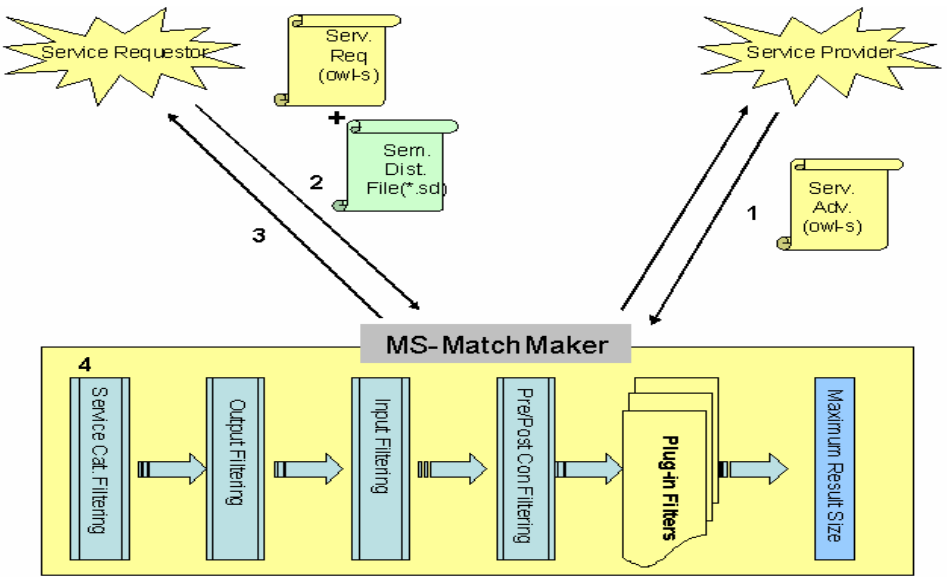

Figure 5.1. MS-Matchmaker Architecture

5.1. Service Category/Domain Matching

The web services will be defined on the high level web service definition ontology and searches will be limited to the specified ontologies. Such a high level service ontology is already worked on by the W3C [7] consortium. Some examples of these service categories may be *E-commerce Service*, *Information Service, Travel Service* etc. This level of filtering is especially important for specifying the range of searches directly and efficiently. For example when searching for book buying services, health services will be eliminated directly, since service categories are different.

In this step, service categories will be filtered according to subsumption relationships. This is just a pre-filtering of number of web services that is to be evaluated. The strictness of this level is adjustable by the agent/user. Here filtering level can be set to level 1, 2 or 3 as defined in Table 5.1 according to the definitions given below. By default, we will use Level=1 for service category filtering where services on same ontological path will be evaluated and pass the filtering.

Table 5.1. Service Category Match

Category	***Definition***	***Level***
Exact Category	A web service that exactly matches as defined in the web service ontology is taken, others eliminated	3
Subsume Category	Web services that exactly matches or subsumed (ones under conceptually the searched one) is filtered	2
Invert-Subsume Category	The ones exact, subsume or invert	1

In some approaches [21], service category is decided according to having the same ontologies used in the header part of the OWL-S documents of the searched web service and advertised web service. This is similar to our approach but our proposion of general service ontology is simpler and more clear to identify.

5.2. Input Matching

Input matching tries to evaluate how good the input of the advertised service matches with the inputs provided by the service requestor. Each input is iterated and tried to be matched with the best-matched input property in the advertised service. The order of inputs is not important. The combination of input mappings is taken with the advertised and requested service and highest matching level of input property is taken as the match type for that input. Here we only consider the concept of matching. In addition, type match may be taken into consideration such as being integer, double, date etc. but these values are assumed to be contained in the ontological description so the concept match will cover these matches.

Input matching is based on subsumption relation. Subsumption as defined in section 4.4 is the determination of subconcept and superconcept relationships between concepts of a given terminology. For being in the same type of subsumption relation, a semantic distance value between the concepts is evaluated according to the weight assignments done by the agent/user. This provides a ranking order on the similar type of matches and to understand the agent/user preferences.

Matching level, signifies the level of match between the client's request and matched advertisement [44]. Based on the subsumption relation of the ontological tree, as generally accepted by similar other approaches and the semantic web studies, there are 4 types of match levels defined as given in Figure 5.2.

Exact : the searched criteria is same as the services definition. For example when searching a web services with providing an input price, finding a service that requires price will match exactly on this constraint.

Plug-in : the provided criteria by the agent/user is a more general concept than the requested criteria of the web service. For example, a web service requiring book as input would be a plug-in one when we provide input Press for the web service. Since Press is on a higher level on ontologic representation.

Subsume : the provided criteria is a smaller concept than the searched criteria; it is a subclass of the searched criteria. For example when we look for press and there exists some service having book as output it is in subsume relation with the search.

Fail : if criteria is not related with the definition of the web service on related concepts it will be set as fail. For example when looking for a web service having a Book as input, a web service defined as having Person as input will be unrelated to the search criteria (not related on the ontologic representation by any way) so that it will be defined as fail.

The matching order of input types are:

Exact > Plug – in > Subsume > Fail

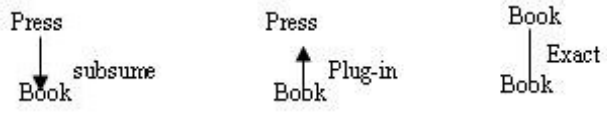

Figure 5.2. Subsumption Relation

Based on this strength we assume to assign the following values for the match types:

Exact : 1

Plug-in : 0.8

Subsume: 0.5

Fail :0

These values will be used in the calculation of match-value. Although they may be changed, their effect should be in the order defined. The level of the match will be the minimum of the set of matches for input/outputs. For example, when 2 input match levels are plug-in and subsume, it will have a final match-degree of subsume for outputs. The minimum match level value for inputs and outputs will be the level of matching for all input and outputs.

Output and input levels and match values will be evaluated separately and outputs are given priority according to the inputs. The reason for this is that the important point for web service discovery is finding web services that satisfy our need. Satisfaction is highly dependent on output. For example we can change the inputs provided by the requestor to be able to use the service but we cannot change the service advisors output easily. Result match number is calculated as an aggregate of all level matches and used specially in ordering of same kind of matched level services. This is especially important for comparing very similar properties web services.

The algorithm uses the subsumption relation of inputs/outputs similar to the approaches in Paolucci [44] and Horrocks studies [22]. The match-types are defined as given in Table 5.2. However, here the path-length from reached input to the requested output on the ontology tree representation information is used in the calculation of match value. We defined this information as semantic-distance (semantic-similarity) rating as described in the previous chapter.

Table 5.2. Input Match Types

Degree	***Definition***	***Value***
Exact	Property requested is the same property with the advertised property.	1
Plug-in	Property of the advertised input is subsumed by the provided input. Agent provides a more general concept as input.	0.8
Subsume	Property of the advertised input is subsumed by the provided input. Agent provides a smaller concept as input.	0.5
Fail	There exists no matching between requested and provided input. input of the requested service.	0

The value assignments are done to represent the strength of match value and just based on heuristic decrease of match values.

For further explanation the ontology below will be assumed (Figure 5.3). This ontology is mainly composed by the rdf *subClassOf* relation. The sematic-distance weight assignments to nodes are done randomly for test scenarios as example.

< owl : Class rdf : ID = ”Book” >

< rdfs : subClassOf rdf : resource = ”#Press”/ >

< /owl : Class>

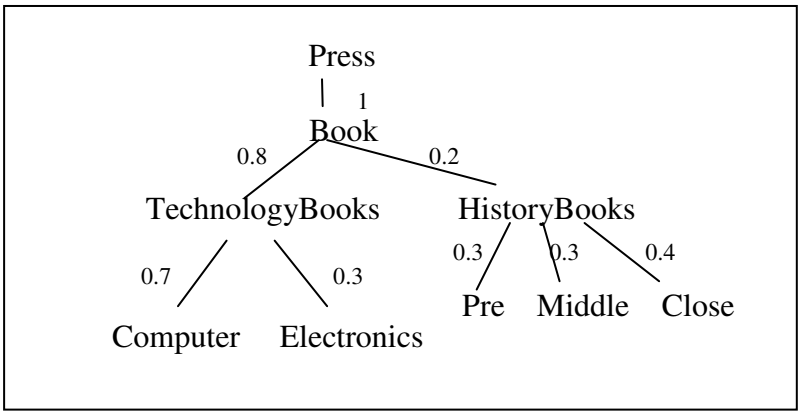

Figure 5.3. Sample Semantic Weight Assigned Ontology For Input/Output Matching

Input match iterates all inputs of the requested service comparing with the advertised services inputs. For each input parameter, a match type is evaluated by the *Reasoner* according to subsumption relation and also according to the semantic-distance information specified between concepts, a numerical similarity measure is evaluated. So for each input paremeter we have a *(match-type, distance-value)* pair. The total result of the input match is minimum match-type of all pairs and sum of match values.

Below is given the pseudo-code and main functions of the input matching algorithm (Figure 5.4.).

```
Public MatchResult  inputMatch (Vector searchInputList, Vector advertisedInputList ) {
MatchResult matchResult = new MatchResult();
İf(advertisedInputList ==null) { // if no input service requires then  exact match
        matchResult .type= EXACT;
        matchResult.rank = 1;
}
int totalRank=0;
for (int i=0; i< advertisedInputList.size();i++ { // for each input of adv. service
Input advInput = (Input) advertisedInputList.elementAt(i);
Int  type =0, bestMatch =0;
double rank = 0, bestRank=0;

İf(searchInputList != null ) { // for each searched input try to match
        for (int j=0; j< searchedInputList.size();j++ {
                Input srcInput = (Input) searchedInputList.elementAt(i);
                type = findMatchType (srcInput, advInput ); // find match type
                rank = rankSemanticWeigth(srcInput, advInput); // find rank value
                İf(type < bestmatch) {
                        bestmatch= type;
                        bestRank = rank;
                        }}
                        totalRank = totalRank +  rank;
}
if(bestMatch == 0)
return FAIL;
if(resultType == FAIL) { // according to final result type set total result
        setToPartiaResultSet(Vector searchInputList, Vector advertisedInputList)
        matchResult.type = FAIL;
        matchResult.rank = totalRank;
} else if(resultType == PLUG-IN) {
        matchResult.type = PLUG-IN;
        matchResult.rank = totalRank;
} else if(resultType == SUBSUME) {
        matchResult.type = SUBSUME;
        matchResult.rank = totalRank;
} else if(resultType == EXACT) {
        matchResult.type = EXACT;
        matchResult.rank = totalRank;
}
```

```
Public SemDistance  rankSemanticWeigth(srcInput, advInput) {

SemDistance  sd = new SemDistance();

// Level distance is how much node below or upperin ontologic representation
        İnt levelDistance = findLevelDistance(srcInput, advInput);

// Here multipy the weights from source  to taget to evaluate the distance such as d = 0.5*0.7
        double semanticDistance = calculateSemDistance(srcInput, advInput, DistanceMap);

sd levelDistance = levelDistance;
sd.semanticDistance = semanticDistance;
return sd;
}
```

Figure 5.4. Pesudo-code for Rank SemanticWeight

```
Public double  calculateSemDistance (srcInput, advInput, DistanceMap) {

Double totalSemDist = 1;
Input[] Concept_list_in_order =findListOfconceptInPath(srcInput, advInput );

// If concepts are not on same path
İf(Concept_list_in_order.size() == 0)
totalSemDist = 0;
else  {
        for(int i=0;i< Concept_list_in_order.size();i++)

        double weight  = DistanceMap[Concept_list_in_order[i]] Concept_list_in_order[i+1]]
        totalSemDist = totalSemDist * weight ;
}

return totalSemDist;
}
```

Figure 5.5. Pseudo-code for Calculating Semantic Distance

5.3. Output Matching

Output matching is similar to input matching just with the meaning of subsumes and plug-in reversed. Comparing with input matching, in output matching a service advertisement that subsumes the requested output is better than the plug-in, because the output provided will be a more general concept covering the requested concept. In output matching we will define the type plug-in as SPECIALIZATION, meaning that the output of the advertised service is a more specialized concept than the requested output concept to provide clarity. The output match types are defined as given in Table 5.3.

Table 5.3. Output Match Types

Degree	***Definition***	***Value***
Exact	Output requested is the same concept with the advertised concept. Output of request and advertisment match exatly. Best case.	1
Subsume	Output of the advertised service subsumes the requested output. Advertised service provides a more generalized concept than the requested output. Second best case.	0.8
Specialization	Output of the advertised service is a subconcept of the requested service output. This case is the reverse of subsume case defined above where outputs of request and advertisements are located in reverse direction on the ontologic representation.	0.5
Fail	There exists no matching between the requested output and advertised service output. They are not on the same path in the ontological representation.	0

As defined with the values, the order of output matches is:

$$Exact > Subsume > Specialization > Fail$$

These definitions are done for each output parameter and combination with all the output parameters of the advertised service is done to find the match type. The result match type for the output parameters is the minimum match-type of all output parameters. For example, let the match types of a 3 output parameter service be (exact, subsume,

specialization) in order. Since the minumum type is specialization, result match type for outputs will be specialization. This rule is the general rule that “the case that forms the bottleneck is the result of all process” [65].

As well as result match-type we also calculate a result semantic distance value which is the sum of all semantic distances of output parameters. Let the semantic distances for (exact, subsume, specialization) matching be (1, 0.8, 0.3). Then the total semantic-distance value of output parameters will be 1+0.8+0.3 =2.1. Total result of match is Result(match-type, semantic-distance) = (specialization , 2.1). The ordering firstly will be done by type of match and secondly by semantic-distance value. For simplicity, to evaluate a single rank value match-type can be assigned the values defined in Table 1.1 and single result can be obtained by multiply this value with the semantic-distance of each parameter. Then sum of all these would give the total match value. However we prefered two level ordering because these numerical assignments is hard to assign and they can cause mismatches in evaluation.

There can be more than one path from source concept to target concept. In case of having more than one parth to the target concept node, we take the maximum similarity value obtained as the semantic distance value. Although both path is discovered, result similarity value is evaluated as the maximum value.

We have used three types of input and output matching in the following order: Exact, Plug-in, Subsume and Fail. For consistency purposes we have also used these types for Input matching.

Table 5.4. Input/Output Match Type Mappings

Input Match Type	***Output Match Type***
Exact	Exact
Plug-in	Subsume
Subsume	Specialization
Fail	Fail

A pseudo-code for output matching is given in Figure 5.6.

```
Public MatchResult  outputMatch (Vector searchOutputList, Vector advertisedOutputList ) {
MatchResult matchResult = new MatchResult();
İf(advertisedOutputList ==null) {
        matchResult .type= EXACT;
        matchResult.rank = 1;
}
int totalRank=0;
for (int i=0; i< advertisedOutputList.size();i++ {
Input advOutput = (Input) advertisedOutputList.elementAt(i);
Int  type =0, bestMatch =0;
double rank = 0, bestRank=0;
İf(searchOutputList!= null ) {
        for (int j=0; j< searchOutputList.size();j++ {
                Output srcOutsput = (Output) searchOutputList.elementAt(i);
                type = findMatchType (srcOutput, advOutput );
                rank = rankSemanticWeigth(srcOutput, advOutput);
                İf(type < bestmatch) {
                        bestmatch= type;
                        bestRank = rank;
                        }}
                        totalRank = totalRank +  rank;
}
if(bestMatch == 0) return FAIL;
if(resultType == FAIL) {
        setToPartiaResultSet(Vector searchInputList, Vector advertisedInputList)
        matchResult.type = FAIL;
        matchResult.rank = totalRank;
} else if(resultType == SPECIALIZATION) {
        matchResult.type = SUBSUME;
        matchResult.rank = totalRank;
} else if(resultType == SUBSUME) {
        matchResult.type = PLUG-IN;
        matchResult.rank = totalRank;
}  else if(resultType == EXACT) {
        matchResult.type = EXACT;
        matchResult.rank = totalRank;}
return matchResult;
```

Figure 5.6. Pseudo-code for Output Match

5.4. Pre/Post Condition Matching

Pre and post condition matching is based on a rule defined setting. In this step of matching elimination is done on the constraints of inputs and outputs. For example web services having output of age, value greater than 18 may be defined as pre condition. Having a valid credit card number for example may be defined as a pre condition of a service. On the other hand to discharge credit card account may be defined as a postcondition of BookSellingService. The matching should be done using some DL based rule languages. SWRL (Semantic Web Rule Language) [45] is currently used for this kind of matching. Also these constraints could be defined on predefined degrees such as must, nice-to-have, must-not. Although pre/post condition matching is not done in detail through implementation, theoretically it may be added to the implementation. However the constructs of the OWL-S descriptions are not mature yet and it forms a bottleneck . But just with a slight modification pre/post conditions can provide a better discovery and selection knowledge. For now we assume to integrate only rule based comparasion of pre/post conditions and elimination of services that do not satisfy these requirements. Pre/Post Condition is not implemented in the system but can be added with some modification in the implementation.

5.5. Plug-in Matching (Add Value)

This step of matching is just given as a different example which may be used as a plug-in in the architecuture. This step has a different view from other proposed filters and it tries to match the service according to specified add-value of the use of the services that would be gained by the user when the service is used. For example, when a web service for hotel reservation, say webServiceHotelReservA, is used with a web service buying bus ticket together, say webServiceBusTicketBuyA, user will get a %10 discount. Or as another example if this service is used with credit card type A for charging, service requestor will win 1000 bonus from the bank. This matching is also based on semantic-rule languages used by the pre/post condition matching. It is especially important to cleverly choose the best service and to provide more descriptions for sorting similarly defined web services on other constructs. The described plug-ins are not implemented and they are just given to provide an example for its applicability.

This plug-in may be used based on user preferences as a plug-in in the matching engine. A similar plug-in may be Quality of Service plug-in.

5.6. Maximum Result Set Size Filtering

If this criterion is specified by the user, only the specified number of web services, matched with highest level and value will be returned to the user. This step is required when a large number of web services is returned from the matchmaker. The result of matchmaking returned to the service requestor will be (Service information, match level, match value) set. The results will be an ordered set restricted to the predefined and specified criteria. Also some default criteria are applied if not specified.

Also another set named as PartialResult Set will be returned. This set will be the web services that fail on some inputs or outputs but provides some subset of inputs/outputs. This result set is especially important for composing the required web service by the requestor if no matched service is found. Current approaches on matchmaking, eliminate the services that do not match on input/output sets, but this set may be important. When no services is found matching the requirements, it may be composed from the ones in the partial set.

As an example assume a web service request that ”Given Price, return list of Electronics and Pre(Histroy) books”. And two web services that is defined as (WS1: Input : Price Output: ComputerBooks, PreHistory) and (WS2: Input: Price OutPut:ComputerBooks, ElectronicBooks). These two web services would be eliminated if matchmaking is done strict on all input/output matching. In the proposed algorithm, these web services will be returned in the partial result set.

6. SIMULATION

6.1. Simulation Environment and Tools

We implemented the algorithm by using Java. The open-source tools and APIs for semantic web is used for forming the base documents and test cases. Some special internal structures are formed for the appliance of our algorithm. A simple file parser is written for parsing *.sd documents.

Ontologies is defined in Protègè [46] ontology editor simply with *subClassOf* relations of owl construct. Protègè [46] with given user interface in Figure 6.1, 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. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats [47]. Defined ontologies using Protègè are *vehicle.owl, press.owl, currency.owl*

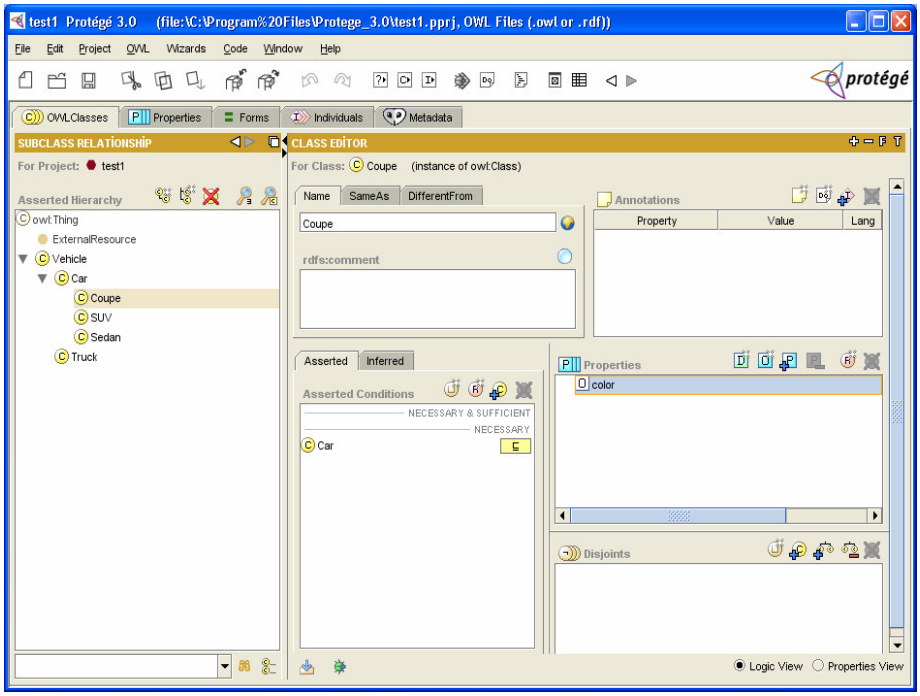

Figure 6.1. Protègè Ontology Editor - Ontology Formation

The OWL-S descriptions of web services are simply defined in OWL-S structure. OWL-S editor [48] is used as defining web services in owls-s format.

For parsing owl-s documents, OWL-S API [45] developed by MINDSWAP [49] laboratory is used. Querying for advertised service is done by owl-s documents also and a semantic-similarity file which we define as *.sd name (sd prefix indicates the file format semantic distance). So the input for the semantic discovery is owl-s document of the required service and semantic-distance assignment document which is an xml file, formed in the format defined in the previous parts.

To parse and evaluate the ontology documents in owl format we ussed the Jena framework. Jena [50] is a Java framework for enabling Semantic Web applications. It is open source software and was developed at the HP Semantic Labs. It provides a programmatic environment for RDF, RDFS and the Web Ontology Language (OWL). In essence, Jena converts RDF files into an RDF model represent by subject-verb-object (SVO) triples and supports information retrieval through the RDQL query language. It also includes a rule-based inference engine for basic reasoning support.

We use Pellet [51], as the reasoner to cover the class-subclass relation of concepts and to reason about the subsumption relations. A simple arcitecture of Pellet is given in Figure 6.2. Pellet is an open-source Java based OWL DL reasoner. Pellet is now a complete and capable OWL-DL reasoner with acceptable to very good performance, extensive middleware, and a number of unique features [51].

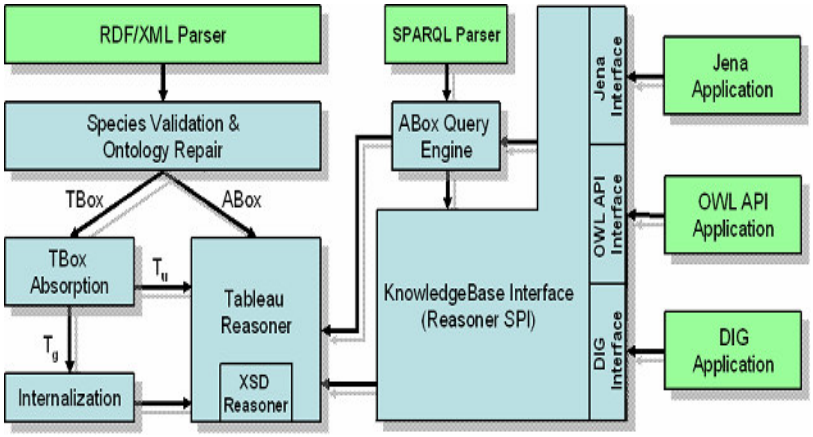

Figure 6.2. Pellet Architecture

Theorotically, UDDI is assumed to be used as a repository for storing the OWL-S documents. However to obtain simplicity in implementation, we just stored the documents and information in simple file structures. Related ontologies are loaded to memory when needed. OWLS-to-UDDI mapping can be done by the work shown in Paolucci et al study [52].

6.2. Simulation Values

We tried to simulate the scenarios described in Section 3 and try to see how user weight assignment can affect the ranking of discovered web services. We try to see applicability of weigth assignment and its usability to define and find the services that matches with requirements best way. We assumed to see the most interested type of services by the user are rated higher by the matchmaker agent.

We ran the simulations on scenarios described in Section 3. For the scenarios, we constructed some sample web services, which are related with the requested service and they can satisfy the requirements in different ranges. Mainly on simulation in this part, we try to show application of our algorithm and getting an ordered result set that is most suitable for the requested web service.

We defined the used sample ontologies in OWL format, as given in the sample below and these documents are stored in a knowledge base (Figure 6.3).

```
<?xml version="1.0"?>
<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:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="NearHistory">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="HistoryBook"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Computer">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="TechnologyBook"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#TechnologyBook">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Book"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MiddleHistory">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#HistoryBook"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PreHistory">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#HistoryBook"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Book">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Press"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#HistoryBook">
    <rdfs:subClassOf rdf:resource="#Book"/>
  </owl:Class>
  <owl:Class rdf:ID="Electronic">
    <rdfs:subClassOf rdf:resource="#TechnologyBook"/>
  </owl:Class>
</rdf:RDF>
```

Figure 6.3. Test Ontology Document - *Press.owl*

For the press ontology, we assume the following .sd document is defined by the user which defines the semantic distance between concepts (Figure 6.4).

```
<Class>Press
        <Subclass>Book
        <semantic-distance>1</semantic-distance></Subclass>
</Class>
<Class>Book
        <Subclass>TechnologyBooks
        <semantic-distance>0.8</semantic-distance></Subclass>
</Class>
<Class>Book
        <Subclass>HistoryBooks
        <semantic-distance>0.2</semantic-distance></Subclass>
</Class>
<Class>TechnologyBooks
        <Subclass>Computer
        <semantic-distance>0.7</semantic-distance></Subclass>
</Class>
<Class>TechnologyBooks
        <Subclass>Electronics
        <semantic-distance>0.3</semantic-distance></Subclass>
</Class>
<Class>HistoryBooks
        <Subclass>Pre
        <semantic-distance>0.3</semantic-distance></Subclass>
</Class>
<Class>HistoryBooks
        <Subclass>Middle
        <semantic-distance>0.3</semantic-distance></Subclass>
</Class>
<Class>HistoryBooks
        <Subclass>Close
        <semantic-distance>0.4</semantic-distance></Subclass>
</Class>
```

Figure 6.4. User Semantic Distance Assignment on Press Ontology - *press.sd*

On the assumed ontological representation each path is assigned a weight based on user preferences as shown in Figure 6.5. and the number of child nodes from the parent. For example under the Book there are 2 child nodes, paths having weights 0.8 and 0.2. For History books there are 3 child nodes which are Pre, Middle, Close; with weights 0.3, 0.3, 0.4. To calculate the combined path weight, for example from Books to Middle (History) books we find the path weights by multiplication of weights on the path as: 0.2* 0.3 = 0.06.

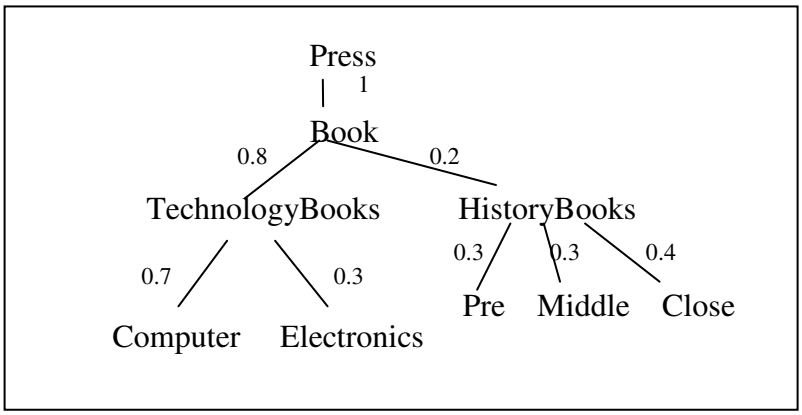

Figure 6.5. Semantic Weight (distance) Assigned on Press Ontology

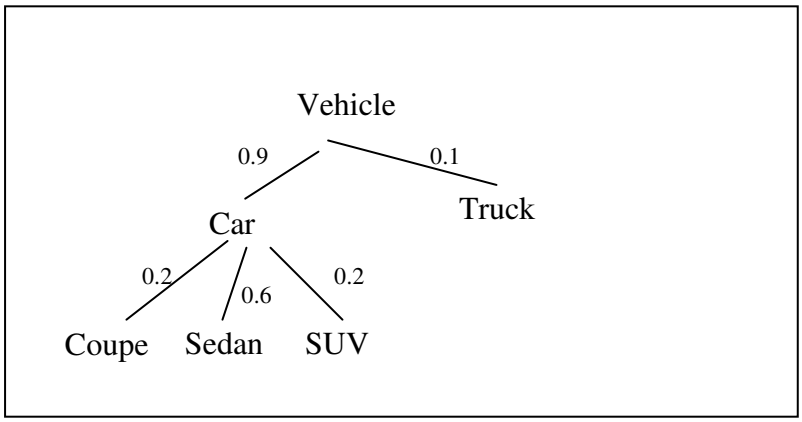

Figure 6.6. Semantic Weight (distance) Assigned on Vehicle Ontology

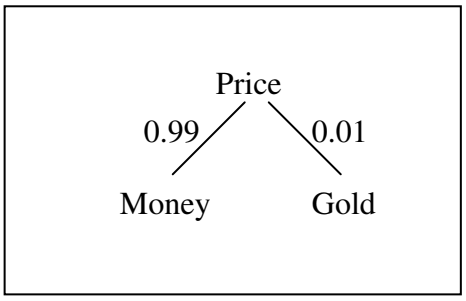

Figure 6.7. Semantic Weight (distance) Assigned on Currency Ontology

total weight = match_degree_value * path_length_value

The idea behind this multiplication and taking 1/n value is to decrease the similarity value as the nodes go further away from each other and as more branches are added to the tree. Because the similarity of concepts decrease as they get far from each other. Although numbers might be modified or predefined semantic similarity weights may be used, the result will remain valid because it is based to decrease relative to path-length and path count.

An important point that had to be used in evalaution is path length from source concept to the target concept. Because by just simple multiplication, we can have two same semantic distance value as the result, but the paths closer to the target concepts should have a higher effect on the resultant semantic weight calculation. For example two paths as 0.9 - 0.1 and 0.1 - 0.9 would have the same semantic result value as 0.09. But the first one, where the first choice of direction is highly rated as 0.9 should have a higher effect on the final result. So to decrease the semantic similarity weight as nodes go further away from source node, we multiply the semantic value with 1/n where n is the path length. By this calculation, as nodes get further away from the target concept their effect will decrease. Different numerical metrics such as logarithmic, division, square root may be used to calculate the distance. This would require complex numerical analysis and different complex test cases to evaluate. For simplicity here we assumed and used *(1/path-length)* to calculate the distance.

*Semantic Distance of input1 = (weight1/ path_len1) * (weight2/ path_len2)*....*

where path_len goes as 1, 2, 3...,n.

Scenario I

Assume as a service requestor, we want to give the price as input and get a list of available books for that price, as output. But as a computer engineering student, I am mostly interested in computer books. However it is not a strict rule given by me and I am open to other types of books and I have some preferences on these kind of books. For example, in history books, I am a little bit more interested in Close history book types. So for my web service search I define my view of world and preferences on press ontology as given in Figure 6.5 and on currency ontology as given in Figure 6.7. I may search with different concepts of input and output values. But for general case I intialize my service search as

Requested ws => Input : Money
Output : Book

Since I want some book, I search for a book and specifiy my priority on Computer books on the semantic assignment. I do not specify any pre/post condition for the search.

Based on this service request assume there are simple Service advertisements as listed below with inputs and outputs (Table 6.1).

Table 6.1. Service Input/Outputs – Scenario I

Web Service	***Inputs***	***Outputs***
WS1	Price	Book
WS2	-	HistoryBooks
WS3	Money	ComputerBooks
WS4	Money	Middle History Books, Price
WS5	Money	Press, Price
WS6	Money	Car

Applying the algorithm on input/output match, based on the subsumption type and semantic-distance information specified we get a result type and value of services as follows (Table 6.2):

Table 6.2. Service Match Results – Scenario I

Web Service	***Input Match Type***	***Input Semantic Dist Value***	***Output Match Type***	***Output Semantic Dist Value***
WS1	Plug-in	0.99	Exact	1
WS2	Exact	1	Subsume	0.2
WS3	Exact	1	Subsume	0.8*(0.7*1/2) = 0.28
WS4	Exact	1	Subsume	0.2*(0.3*1/2) = 0.03
WS5	Exact	1	Plug-in	1
WS6	Exact	1	Fail	0

So based on match-type and match value, as an ordered set we get the following list (Table 6.3):

Table 6.3. Ordered Discovered Results – Scenario I

Order	***Service,Match-type,Value***
1	WS5, Plug-in, 2
2	WS1, Plug-in, 1.99
3	WS3, Subsume, 1.28
4	WS2, Subsume, 1.2
5	WS4, Subsume, 1.03

WS6 is eliminated because it is in fail category and it is not related with the searched criteria on output parameters.

Scenario II

This scenario is similar to the previous one. A user/agent wants to buy a car from the web. He is interested in Sedan cars mostly, but he is open to other offers also, which may not be a Sedan car. His preferences are assigned as shown in Figure 6.6. He will ask for a 'Car' as an output, and will give year of production and color as the properties of the car as input. Therefore, conceptually the Car is the input also.

Requested ws => Input : Car
Output : Car

Based on this service request assume there are simple Service advertisements as listed below with inputs and outputs (Table 6.4).

Table 6.4. Service Input/Outputs – Scenario II

Web Service	***Inputs***	***Outputs***
WS1	Car	City
WS2	-	Vehicle
WS3	-	Sedan
WS4	Money	Sedan
WS5	SUV	SUV
WS6	Car	Sedan

Applying the algorithm on input/output match, based on the subsumption type and semantic-distance information specified we get a result type and value of services as (Table 6.5):

Table 6.5. Service Match Results – Scenario II

Web Service	*Input Match Type*	*Input Semantic Dist Value*	*Output Match Type*	*Output Semantic Dist Value*
WS1	Exact	1	Fail	0
WS2	Exact	1	Plug-in	1
WS3	Exact	1	Subsume	0.2
WS4	Fail	0	Subsume	0.6
WS5	Subsume	0.2	Subsume	0.2
WS6	Exact	1	Subsume	0.6

WS1 and WS4 is eliminated because of fail. So based on match-type and match value, we get ordered set as sorted list of Table 6.6:

Table 6.6. Ordered Discovered Results – Scenario II

Order	*Service,Match-type,Value*
1	WS2, Plug-in, 2
2	WS6, Subsume, 0.6
3	WS3, Subsume, 0.2
4	WS5, Subsume, 0.1

Scenario III

In this scenario a user was interested in getting only Technology books and he is not interested in other books. Therefore he wants to eliminate other type of books. So he gives no input and gets list of avaliable technology books as output.

Requested ws => Input : -

Output : TechologyBooks

For this preference, user makes the following semantic-distance on the ontologic representation; just gives a slight preference on Computer Books (Figure 6.8).

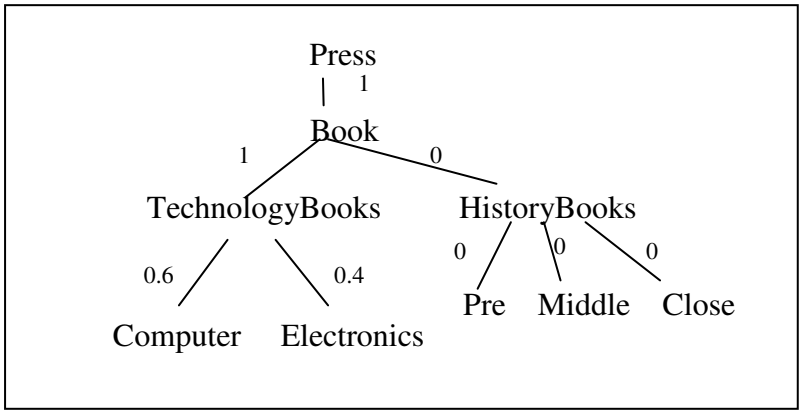

Figure 6.8. Semantic Weight (distance) Assigned on Press Ontology

Now after the filtarion of the service category, assume we have identified the following services as related (Table 6.7).

Table 6.7. Service Input/Outputs - Scenario III

Web Service	***Inputs***	***Outputs***
WS1	Price	HistoryBooks
WS2	-	TechnologyBooks
WS3	-	Computer
WS4	Money	TechnologyBooks
WS5	-	Electronics

Then ordering and ranking on these services will be done. We get the following result (Table 6.8):

Table 6.8. Service Match Results – Scenario III

Web Service	*Input Match Type*	*Input Semantic Dist Value*	*Output Match Type*	*Output Semantic Dist Value*
WS1	Fail	0	Fail	0
WS2	Exact	1	Exact	1
WS3	Exact	1	Subsume	0.6
WS4	Fail	0	Exact	1
WS5	Exact	1	Subsume	0.4

Based on these match types and values WS1 and WS4 will fail, and others will be ordered as follows (Table 6.9) :

Table 6.9. Ordered Discovered Results – Scenario III

Order	*Service, Match-type,Value*
1	WS2, Exact, 2
2	WS3, Subsume, 1.6
3	WS5, Subsume, 1.4

Scenario IV

In this scenario, we will focus on service category filtration and maximum number of filtering of the matching algorithm. For the upper service ontology assume a service ontology given below is accepted. Assume there are 100 web services registered in the matchmaker and 10 of them is related with the context of *Payment Services* is some level and others are unrelated. They may be health services, financial services etc. (Figure 6.9).

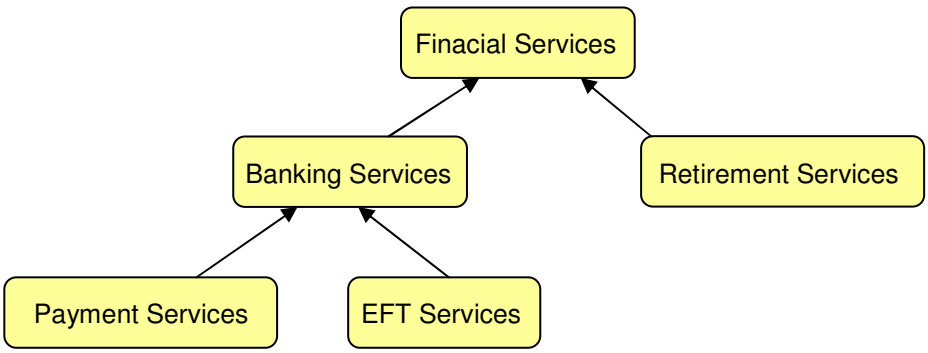

Figure 6.9. Sample Service Category Ontology

For example , service requestor specify the search context as *Payment Service* at the highest level. He wants to apply the service category filtering to default value of matchmaker and set the maximum result set size to 5. Also let us assume that simply input and output parameters are given and no other constraint is defined. In context level matching, 90 of the 100 web services will be eliminated directly since they do not share the same service category of search. After input/output/pre/postcondition and if exist any other plug-in match, only web services that match on these values in some level will remain. Assume 8 web services passed the input/output/pre-condition/post-condition/plug-in filtering of 10 service category related services. As the last step, we have restricted the size of maximum result set as 5, so top 5 services will be returned to the user as a result. Depending on user preferences matching engine may also return a partial result set to user/agent.

7. EVALUATION

We ran the sample scenarios on simple ontologies and simply assumed web services with focusing different parts of the matching algorithm to show its applicability and success in ordering results. We observed that, we can provide information to rank the services discovered using semantic-distance information assigned by the user. By this ontological weight assignment, user can also specify his/her preferences and he can direct/limit the search space of services. Using this approach, rather than getting a category based result set of services, service requestor can obtain an ordered set of web services and then he can decide on services selection with more information.

We have observed that result ordered set of services is obtained and it matches with the ordering of requested services, intuitively. When we evaluate from users' perspective his/her preferences and request will mostly be satisfied. In addition, a search which can be realized by more than a single search, can be obtained by a single search within the proposed framework. This observation is intuitive from the result ranked services obtained.

To focus on ontological concepts distance and assignment, we just used concepts in the search level defining. In addition, property of concepts may be used in the matching procedure. These properties are defined by the *owl:ObjectProperty* construct of the OWL-S. Through this evaluation only concept matching is focused as main concern.

In addition, UDDI mapping is done not in implementation but it is shown in related semantic researches. Efficieny of this repository architecture is proven with researches on the topic. Therefore, repository architecture and other repository related issues such as mapping documents are not covered in the evaluation.

For the evaluation, performance, security and other service properties such as service process model are not used. Basic service profile information with simple web services is used. Since the ontologies and services are simply formed, the cases such as having similar concepts in different global ontologies, having more than one super class of a concept in the ontology are not evaluated. In addition, local to global ontology mediation is not

covered. These would bring some complexity to the test cases but the idea and approach will be applicable for these cases also.

8. COMPARASION WITH OTHER MATCHMAKERS

Compared with other matchmakers, our proposed matching algorithm provides a new angle to the matchmaking: using user defined semantic distance information. Different from most other matchmakers, user is able to obtain a ranked set of services as a result. User is able to specify his/her preferences and view-of world. A new concept, named partial result set, is formed which may be used further for service composition rather than elimination of some simple mismatches in the matching process.

If the current sample scenarios were ran with the Paolucci et. al.'s OWL-S Matchmaker the resulting set of service will be only category based and an order will not be available. In addition, as specified in Scenario III, user will not be able to specify his/her preferences on technology books and especially in computer books. To obtain the same result, more than one search, with different output parameters will need to be made to get theresult.

Compared with LARK architecture, our proposed framework is based on the most recent and currently accepted technologies such as OWL-S. Although semantic similarity of concepts is used in our proposed approach and LARKS, their context of usage and specification are different. In addition, LARKS will not give an ordered set on these scenarios. However, as an advantage LARKS is able to apply constraints on inputs and outputs.

If test scenarios were run on Ian Horrocks and Lei Lui's proposed framework, the results would be similar to OWL-S matchmaker. Here only match service categories will be different and we will not be able to get a rank based service discovery list.

In Table 8.1 we give a comparison of topics in each framework.

Table 8.1. Comparasion Table of Frameworks

Framework	***LARKS***	***OWL-S Matchmaker***	***Lei Lui's Framework***	***MS-Matchmaker***
Language	ITL	OWL-S	DAML-S	OWL-S
Repository	Local KB	UDDI	UDDI	UDDI
Service Category Filter	✓	✓		✓
Input Filter	✓	✓	✓	✓
Output Filter	✓	✓	✓	✓
Pre/post Condition Filter	✓	✓	✓	✓
Plug-in Filter		✓		✓
Semantic Dist. Usage	partial			✓
Ranked List				✓
Type Based List	✓	✓	✓	✓

9. CONCLUSION

In this research, we proposed an extendable and efficient matchmaking algorithm and architecture. Our main concern was to fill the missing points in the systems that are already proposed. By increasing the information using semantic-distance information we tried to get more accurate matchmaking results and a better ordered result set.

Rather than eliminating them directly, this set is proposed for composition architectures. A layered algorithm and architecture is designed to be efficient and extendable. Add-value definition is added for search criteria to match different web services and to assist the benefits of users on behalf.

9.1. Overview

In Chapter 1, we have stated our motivation for studying the topic of semantic web service discovery and matchmaking. Then we briefly overviewed the related works in the field of matchmaking and semantic web service discovery. We pointed the missing points and weaknesses in these approaches.

In Chapter 2, we have provided brief background information on enabling technologies for semantic web services. Starting with web service discovery architectures, be have shortly defined and give information about concepts of XML, SOAP WSDL, UDDI, RDF, OWL-S. Shortly we have mentioned ontologies, since our approach is an ontology-based service discovery approach. Finally, we summarized all the concepts in the view of semantic web.

In Chapter 3, we have clearly state our problem as semantic web service discovery and obtaining a ranked service result using semantic distance information on ontological concepts. We have outlined the deficiencies of current approaches and matchmakers in semantic web service discovery and stated some problems that can occur using these. Some cases that cannot be handled and need improvement in current approaches is

described. Finally, we concluded the chapter by giving sample scenarios we want to handle efficiently and better than other approaches.

In Chapter 4, we have proposed our model, the semantic service discovery framework. We have provided in depth information on general architecture of the framework, main concepts and the basic definitions accepted to be used in the further parts of the framework. We have introduced the *Semantic Distance* concept and outlined its benefits. We clearly defined semantic distance concept and we identified some rules of concept similarity. We have also defined a new concept named as partial result set, which is an extension to other approaches also.

In Chapter 5, we have presented the matchmaking algorithm, which forms the main component of the proposed matchmaking architecture. The layered structure of the algorithm is detail described giving samples and some pseu-code. Some sample ontologies are given to identify the cases clearly. Service category match, input match, output match, pre/post condition match, plug-in match and maximum result size filtering steps of the algorithm is described detail.

In Chapter 6, we have presented the simulation environment and simulation cases. We have simply described the semantic tools, their properties and API s used for the simulation. In the second part of the chapter, we ran the sample scenarios on our proposed framework systematically with the assumed test web services and get order service lists as a result of the discovery process.

In Chapter 7, we have evaluated our framework and outlined the main difference from other frameworks results. We made the evaluation based on sample test scenarios and showed the benefits of using semantic distance information in matchmaking process.

In Chapter 8, we have compared the results and structure of our proposed framework with previously proposed frameworks in semantic web service discovery. We made a comparison basing how other frameworks would give a result for the sample test cases. These evaluations were not based on numerical and real running of test cases, but running the scenarios intuitively on paper.

Finally, in this chapter we present a summary of the research, its contributions and conclude the thesis.

9.2. Contributions

The main contribution of this research is the use of information on the ontological representation of parent-child relations with weigth assigment values which we define as semantic-distance information. Usage of this information defined by the service requesting user, provides an extra information to identify his/her preferences and view-of-world concepts. Based on this information, the proposed framework returns a ranked result of web services.

A new *Partial Result Set* is defined and services that only satisfy some subset of input/output parameters is directed to this list rather than directly eliminating them.

All these matching procedure is presented in a general framework and based on newest accepted technologies. Therefore we proposed a novel framework for semantic web service discovery.

9.3. Future Work

Although the research in this thesis proposes a novel and conceptual architecture for web service discovery, there are some future work to be done. First, in matching services quality of services is not used as criteria. Such a layer can be added to the architecture. In addition, similarity matching is done on the similarity of concepts. It can be widened on the similarity of properties, constraints, etc.

Secondly, mediaton is conceptually proposed but neither implemented nor analyzed in detail. However, scalable and large set of mediation principles of WSMF architecture can also be applied to our proposed framework. Such a mediaton component can form an important part of the architecture and it can show the problems that occur through weight assignments on local and global ontologies.

In addition, to evaluate validity of semantic distance usage, scenarios and ontologies are kept simple and limited number of services is used for the discovery process. The scenarios and ontologies can be made more complex. Therefore, efficiency of the algorithm can be evaluated with increasing the number of advertised services.

APPENDIX A: TEST ONTOLOGIES

A.1 Press.owl

```
<?xml version="1.0"?>
<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:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
 xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="NearHistory">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="HistoryBook"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="Computer">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="TechnologyBook"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#TechnologyBook">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Book"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="MiddleHistory">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#HistoryBook"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="PreHistory">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#HistoryBook"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#Book">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Press"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#HistoryBook">
  <rdfs:subClassOf rdf:resource="#Book"/>
 </owl:Class>
 <owl:Class rdf:ID="Electronic">
  <rdfs:subClassOf rdf:resource="#TechnologyBook"/>
 </owl:Class>
</rdf:RDF>
```

A.2 Vehicle.owl

```
<?xml version="1.0"?>
<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:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="SUV">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Car"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Sedan">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Car"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Vehicle"/>
  <owl:Class rdf:ID="Coupe">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Car"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Truck">
    <rdfs:subClassOf rdf:resource="#Vehicle"/>
  </owl:Class>
  <owl:Class rdf:about="#Car">
    <rdfs:subClassOf rdf:resource="#Vehicle"/>
  </owl:Class>
</rdf:RDF>
```

A.3 Currency.owl

```
<?xml version="1.0"?>
<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:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:about="#Money">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Currency"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Gold">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Currency"/>
    </rdfs:subClassOf>
  </owl:Class>
</rdf:RDF>
```

REFERENCES

1. *Extensible Markup Language (xml) 1.0 (second edition)*, http://www.w3.org/TR/2000/REC-xml-20001006, 2000.

2. *WSDL, Web Service Description Language*, http://www.w3.org/TR/wsdl, 2001.

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

4. W3C, "Web Services Architecture", *Working Draft 2002*, http://dev.w3.org/cvsweb/2002/ws/arch/wsa/wd-wsa-arch.html?rev=1.28, 2002.

5. *UDDI, Universal Discovery Description and Integration Protocol,* http://www.uddi.org, 2006.

6. *Semantic Web, W3C*, http://www.w3.org/2001/sw/, 2006.

7. *W3C, World Wide Web Consortium*, http://www.w3.org/, 2006.

8. Berners-Lee T., "Semantic Web Road Map", http://www.w3.org/DesignIssues/Semantic.html, 1998.

9. Berners-Lee, T., J. Hendler and O. Lassila, "The Semantic Web", *Scientific American*, Vol. 284, No.5, pp. 34-43, 2001.

10. Motta, E., J. Domingue, L. Cabral and M. Gaspari, "IRS-II: A Framework and Infrastructure for Semantic Web Services", *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, Vol. 2870 of LNAI., Springer, pp.306-318, Florida, USA, 2003.

11. *OWL-S Submission*, http://www.w3.org/Submission/OWL-S, 2004.

12. Fensel, D. and C. Bussler, "The Web Service Modeling Framework: WSMF", *Electronic Commerce: Research and Applications*, Vol. 1, No. 2, pp. 113-137, 2002.

13. Klein, M., B. Konig-Ries and M. Muussig, "What is needed for semantic service descriptions? A proposal for suitable language constructs", *Proceedings of Inernational Journal of Web and Grid Services*, Vol. 1, No. 3/4, pp. 328-364, 2005.

14. Voorhees, E., "Using WordNet for Text Retrieval", C. Fellbaum(Editor),"*WordNet: An Electronic Lexical Database*", pp. 285-303, The MIT Press, Cambridge,1998.

15. Ginsberg, A., "A Unified Approach to Automatic Indexing and Information Retrieval", *IEEE Expert* , Vol. 8, No. 5, pp 46-56, 1993.

16. Lee, J., M. Kim and Y. Lee, "Information Retrieval Based on Conceptual Distance in IS-A Hierarchies", *Journal of Documentation,* Vol. 49, No. 2, pp. 188-207, 1993.

17. Agirre, E. and G. Rigau, "Word Sense Disambiguation Using Conceptual Density", *Proceedings of the 16th Conference on ComputationalLlinguistics*, Vol.1, pp. 16-22, 1996.

18. Hovy, E., "Combining and Standardizing Large-scale, Practical Ontologies for Machine Translation and Other Uses", *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain, 1998.

19. Wang, Y. and E. Stroulia, "Semantic Structure Matching for Assessing Web-Service Similarity", *Proceedings of the 1st International Conference on Service Oriented Computing*, Trento, Italy, 2003.

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

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

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

23. Verma, K., K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar and J. Miller, "METEOR–S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services", *Journal of Information Technology and Management*, 2004

24. *DAML-S*, http://www.daml.org/services/daml-s/0.9/, 2001.

25. *OWL-S, Semantic Mark-up For Web Services,* http://www.daml.org/services/owl-s/, 2006 .

26. *Carnegie Mellon University, Intelligent Agents Software Group,* http://www.cs.cmu.edu/~softagents/, 2006.

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

28. Roman, D., U. Keller and H. Lausen, J. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler and D. Fensel, "Web Service Modeling Ontology", *Applied Ontology*, Vol. 1, No. 1, pp. 77 -106, 2005.

29. Sycara, K., K. Decker and M. Williamson, "Middle-Agents for the Internet" *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, 1997.

30. Phillipa, O., H. M. A. ter Hofstede and D. Edmond, "Capabilities: Describing What Services Can Do", *Proceedings of the 1st International Conference on Service-Oriented Computing*, pp. 1-16, 2003.

31. Jennings, N. R., K. Sycara and M. J. Wooldridge, “A Roadmap of Agent Research and Development", *Autonomous Agents and Multi-Agent Systems*, Vol. 1, pp. 275-306, 1998.

32. *Resource Description Framework, W3C,* http://www.w3.org/RDF/, 2006.

33. Suchanek, F. M., "Summary Intelligent Information Agents", Saarland University, http://www.mpiinf.mpg.de/~suchanek/personal/texts/summaries/iia.txt, 2004.

34. Trastour, D., C. Bartolini and J. Gonzalez-Castillo, "A Semantic Web Approach to Service Description for Matchmaking of Services", *HP Labs white paper*, 2001.

35. OWL-S API, http://www.mindswap.org/2004/owl-s/api/, 2006.

36. Gruber, T. R, "A Translation Approach to Portable Ontologies", *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220, 1993.

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

38. Brulin, J. and P. Axel, "Towards an Ontology Mapping Specification Language for the Semantic Web", *DERI Technical Report 2004-06-30*, 2004.

39. *Review of the State-of-Art- Semantic Web and Web Service Semantics Deliverable D3.1.1.2. (07 April 2004)*, http://www.srdc.metu.edu.tr/webpage/projects/artemis/documents/D3.1.1.2-SOAv1.2_SemanticWeb.doc, 2004.

40. Arabshian, K. and H. Schulzrinne, "A Hybrid Hierarchical and Peer-to-Peer Ontology-based Global Service Discovery System", *Techinal Report*, 2005.

41. Dong, X., A. Halevy, J. Madhavan, E. Nemes and J. Zhan, "Similarity Search For Web Services", *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004.

42. Stefan, T., "Matching of Web Service Specifications Using DAML-S Descriptions", Master Thesis, Technic University Berlin, 2004

43. Maximilien, E. and M. Singh, "A Framework and Ontology for Dynamic Web Services Selection", *IEEE Internet Computing*, Vol. 8, No. 5, pp.84–93, 2004.

44. Paolucci, M., N. Srinivasan, K. Sycara, and T. Nishimura, "Towards a Semantic Web Ecommerce", *Proceedings of 6th Conference on Business Information Systems (BIS2003),* Colorado Springs, Co, USA, pp. 153-161, 2003.

45. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, http://www.w3.org/Submission/SWRL/, 2006.

46. *Protègè Ontology Editor*, http://protege.stanford.edu/, 2006.

47. Gennari, J., M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy and S. W. Tu, "The Evolution of Protégé: An Environment for Knowledge-Based Systems Development", *Proceedings of International Journal of Human-Computer Studies*, 2002.

48. *The OWL-S Editor*, http://owlseditor.semwebcentral.org/, 2006.

49. *Maryland Information and Network Dynamics Lab (Mindswap)*, http://www.mindswap.org/, 2006.

50. *Jena Semantic Framework*, http://jena.sourceforge.net, 2006.

51. Şirin, E., B. Parsia, B. C. Grau, A. Kalyanpur and Y. Katz, "Pellet: A practical owl-dl reasoner", *Submitted for publication to Journal of Web Semantics*, 2006.

52. Paolucci, M., T. Kawamura, T. R., Payne and K. Sycara, ”Importing the Semantic Web in UDDI”, *Proceedings of Web Services, E-business and Semantic Web Workshop*, Toronto, Canada, pp. 225-236, 2002.

53. *OWL-S 1.1 Release*, http://www.daml.org/services/owl-s/1.1/, 2004.

54. Hendler, J. and D. L. McGuinness, "DARPA Agent Markup Language", *IEEE Intelligent Systems*, Vol. 15, No. 6, pp. 72-73, 2001.

55. Trastour, D., C. Bartolini and J. Gonzalez-Castillo, "A Semantic Web Approach to Service Description for Matchmaking of Services", *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001.

56. Srinivasan, N., M. Paolucci and K. Sycara, "An Efficient Algorithm for OWL-S Based Semantic Search in UDDI", *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, Vol. 3387 of LNCS., 96-110, 2004.

57. Jacobs, I., "About the World Wide Web Consortium (W3C)", Technical Report, W3C, http://www.w3.org/2003/01/Consortium.pdf, 2000.

58. The SemanticWeb - ISWC 2003. Lecture Notes in Computer Science, Vol. 2870. Springer-Verlag, pp. 306–318, 2003.

59. *OWL-S Coalition: OWL-S 1.2 Release.*, http://www.ai.sri.com/daml/services/owl-s/1.2/, 2006.

60. Srinivasan, N., M. Paolucci and K. Sycara, "Adding OWL-S to UDDI, implementation and throughput", *Proceedings of First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004*), 2004, San Diego, California, USA, pp. 6-9, 2004.

61. Udupi, Y. B. and M. P. Singh, “Multidimensional Service Matching and Selection”, AAMAS’ 06, Japan, 2006.

62. Broens, T., "Context-aware, Ontology based, Semantic Service Discovery", Master Thesis, University of Twente, the Netherlands, 2004.

63. Munindar, P., P. Singh, M. N. Huhns, *"Service-Oriented Computing: Semantics, Processes, Agents",* John Wiley & Sons, January 28, 2005.

64. *SematicWeb Services Initiative,* http://www.swsi.org/, 2006.

65. Nielson, J. E., C.M Woodside, D.C.Petriu and S. Majumdar, "Software Bottlenecking In client-server Systems and Rendezvous Networks", *IEEE Trans. Softw. Eng.*, Vol. 21, No. 9, pp.776-782, 1995.