SEMANTIC WEB SERVICE DISCOVERY IN A MULTI-ONTOLOGY ENVIRONMENT

by

SWAPNA OUNDHAKAR

(Under the Direction of Amit Sheth)

ABSTRACT

Web services have generated considerable interest in the area of application integration for achieving the enticing promise of "interoperability". Web service discovery is an important aspect of the Web service framework but its existing keyword based search implementation is seriously flawed. This is because the exchanged data is structurally and syntactically verified, but not “understood”. Augmenting Web service descriptions with ontologies can provide this “understanding”. Since ontologies and web services are developed independently the service request and advertisement can be annotated with multiple ontologies. Thus a more pragmatic service discovery algorithm is needed to address the challenging semantic heterogeneity issues in such a multi-ontology environment. This thesis provides an early effort in developing a Web service discovery algorithm for a multi-ontology environment. Preliminary evaluations show that the algorithm is able to find good matches and eliminate false matches by considering the context and the coverage information of the annotated concepts.

INDEX WORDS: Semantic Web services, WSDL, Ontology, semantic Web services discovery, multi-ontology Web service discovery

SEMANTIC WEB SERVICE DISCOVERY IN A MULTI-ONTOLOGY ENVIRONMENT

by

SWAPNA OUNDHAKAR

B.E., University of Mumbai, 1999

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2004

SEMANTIC WEB SERVICE DISCOVERY IN A MULTI-ONTOLOGY ENVIRONMENT

by

SWAPNA OUNDHAKAR

Major Professor:	Amit Sheth
Committee:	Hamid R. Arabnia Liming Cai

DEDICATION

To my Parents and Brother

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Amit P. Sheth for his guidance, foresight, and his assurance in difficult times. Dr. Sheth has been very generous with his time and wisdom. I would like to thank my parents and brother for their continuous encouragement and love. I would also like to thank Dr. Hamid R. Arabnia and Dr. Liming Cai for their valuable suggestions and being a part of my committee. I would like to thank Kunal Verma for his time and guidance. The Large Scale Distributed Information Systems (LSDIS) lab and the METEOR-S project have provided me with the resources and collaborative supporting environment to grow and prepare myself for a career. Special thanks to my friends Abhijit Patil, Maria Chinwala and Saurabh Gupta for their support and encouragement.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS v

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

2 BACKGROUND 4

3 SEMANTIC WEB SERVICE DISCOVERY IN A MULTI-ONTOLOGY ENVIRONMENT 11

4 SERVICE DISCOVERY ALGORITHM 16

5 CONCEPT MATCHING ALGORITHM 23

6 RESULTS AND EMPIRICAL TESTING 37

7 RELATED WORK 45

8 CONCLUSION AND FUTURE WORK 49

REFERENCES 53

APPENDICES

A ONTOLOGIES USED 61

B GLOSSARY OF ACRONYMS 62

C GLOSSARY OF CONCEPTS 64

LIST OF TABLES

Page

Table 1: Search Template 14

Table 2: Candidate Service 16

Table 3: Datatype conversion values for Range Match...26

Table 4: Example for Property Similarity..29

LIST OF FIGURES

Page

Figure 1: Web Service Architecture...5

Figure 2: Four types of Semantics in Web services...8

Figure 3: Web Service Stack and METEOR-S...9

Figure 4: Concept Matching 40

Figure 5: Graph 1 & Graph 2 - Functional Similarity and IO Similarity 42

Figure 6: Graphical view of Ontologies used (Partial) 61

CHAPTER 1

INTRODUCTION

The WWW ushered us into a new era of the information revolution. This technological revolution gave rise to the "new economy" and changed the way businesses work. With the advent of Web services we are on the brink of a new "service revolution". The introduction of new standards such as UDDI [UDDI, 2002], WSDL [Christensen et. al, 2001] and SOAP [Box et.al, 2000] are a step in the right direction. Nevertheless, like any new technology they come with a new set of challenges, some that are unique and some that are of the familiar kind related to technology maturity and adoption. While Web services have already found growing support for intra-enterprise applications, their promise for inter-enterprise and e-commerce applications is yet to be realized. One notable technical problem to be addressed is that of finding the most appropriate Web service from a Web Service registry like UDDI. Consider the keyword based search facility offered by the latest version of UDDI [UDDI, 2003]. Once, a set of services are found containing that keyword, it is entirely left to the user to find the most appropriate one. This involves a painstaking approach to manually analyze the WSDL descriptions. As the number of Web services increase manifold this approach becomes more and more impractical. Recent research [Fensel and Bussler, 2002; Paolucci et.al, 2002; Sivashanmugam et.al, 2003; SWSA 2004] has identified the application of ontologies as one of the solutions to this problem. They seek to demonstrate that if the service descriptions and the user requirements are both annotated with ontologies, then the semantic approach to finding the appropriate service becomes easier, with the discovery resulting in more relevant service identification with better level of automation.

Several papers [Paolucci et.al, 2002; Gonzales et.al, 2001; Colgrave et.al, 2004; Sivashanmugam et.al, 2003] have discussed semantic discovery, when the advertisement and request use terms from the same ontology. An approach based on using a single ontology is not practical as it is highly unlikely that every service provider and requestor will adhere to the same ontology. Very few approaches [Cardoso and Sheth, 2003] consider the case, when the advertisement and request belong to different ontologies from the same domain. This thesis, presents an extension of the algorithm in [Cardoso and Sheth, 2003], which uses property and syntactic similarity. It introduces two new measures-- context and coverage similarity which try to capture additional semantic information about the concepts to be matched by looking at other concepts in their vicinity. This additional semantic information is used to find implied relationships between concepts, which help to understand if the concepts are semantically similar, or semantically disjoint and accordingly improve the match scores

This thesis also presents an evaluation of the algorithm using a prototype implementation. Preliminary results show that using additional measures like context and coverage similarity can lead to higher precision than other approaches which do not consider them. This work was done as part of the METEOR-S project, which aims at creating a comprehensive infrastructure for the complete lifecycle management of Semantic Web processes.

The main contributions of this thesis are -

- A unique Service Discovery algorithm based on functionality for matching services when the advertisement and the request are from different ontologies
- The introduction of two new measures for matching two ontological concepts

- Context Similarity and
- Coverage Similarity

Virtually all data and application interoperability and integration efforts involve matching and mapping steps. While we present our approach to Web service discovery (a form of application interoperability) challenge, it is largely applicable to other situations if a more practical multi-ontology environment is considered.

Chapter 2 discusses the background material for this thesis. A brief overview of the Semantic Web Service Discovery in a Multi-Ontology Environment approach is given in chapter 2. Chapter 3 discusses the discovery algorithm in detail. Concept matching algorithm is detailed in chapter 4. Chapter 5 describes the Results and Empirical Testing. Chapter 6 discusses related work. Finally conclusions and future work are mentioned in Chapter 7.

CHAPTER 2

BACKGROUND

This chapter details some of the background material relevant to this research. In particular it gives an overview of the current Web service framework and ontologies. It also gives a brief introduction to Semantic Web services. Finally it describes the ongoing work on the METEOR-S project at the LSDIS lab.

Web Services

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

The Web service framework follows a Service Oriented Architecture (SOA). It uses three basic XML based standards WSDL, UDDI and SOAP. WSDL i.e. Web Service Description Language is used to describe the inputs, outputs and the invocation information of a Web service. These descriptions are then published or advertised in a registry called UDDI (Universal Description Discovery and Integration). UDDI also provides search mechanisms to find or discover these Web service descriptions. Once the required Web service description is found, the Web service can be invoked using SOAP (Simple Object Access Protocol). Figure 1 shows the SOA for Web services and interactions among its different components. The next section gives brief overview of each of these standards. More emphasis is given on UDDI and WSDL especially WSDL, since they are more relevant to the work described in this thesis.

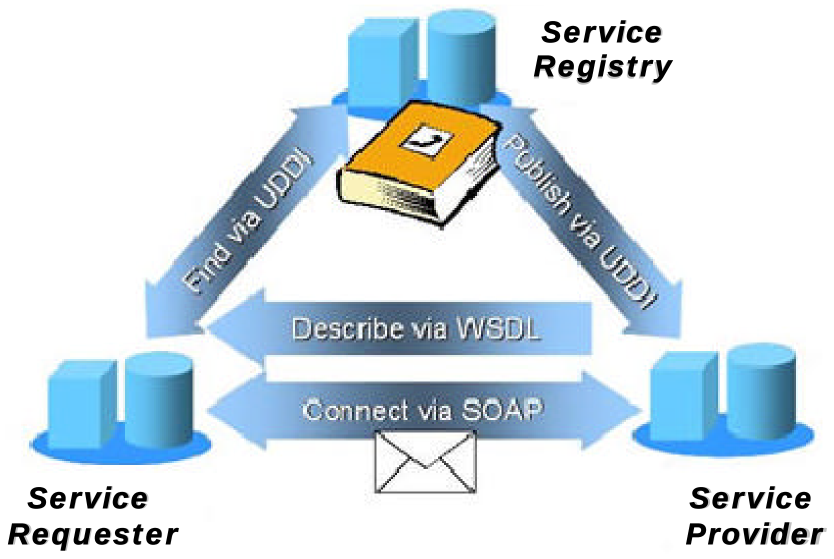


Figure 1. Web Service Architecture

WSDL (Web Service Description Language)

The Web Service Description Language is a standard for describing a Web service. It is an XML grammar for specifying properties of a Web service such as *what* it does, *where* it is located and *how* it is invoked. It can be compared to HTML (Hypertext Markup Language) in the normal WWW which is used to describe a web page.

A WSDL document defines services as collection of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment. This allows the reuse of abstract definitions. A WSDL document uses the following seven elements in the definition of network services.

- Type : A container for data type definitions using some type system
- Message : An abstract, typed definition of the data being communicated
- Operation : An abstract description of an action supported by the service
- Port Type : An abstract set of operations supported by one or more end points
- Binding : A concrete protocol and data format specification for a particular port type.
- Port : A single end point defined as a combination of a binding and a network address.
- Service : A collection of related end points.

UDDI (Universal Description Discovery and Integration)

UDDI is a registry or a directory where Web service descriptions can be advertised and discovered. The person or organization that provides a Web service is called "Service Provider". A service provider publishes the service description i.e. the "Service Advertisement" in the UDDI. A "Service Requester" i.e. a person or an organization can "request" a Web service by using the search mechanisms provided by UDDI. A Web service can be found or discovered in UDDI either by browsing the categories or by using a keyword based query mechanism. Figure 1 summarizes the interaction between service provider, service requester and UDDI registry.

SOAP (Simple Object Access Protocol)

SOAP is an XML based messaging protocol. It defines a set of rules for structuring the messages being exchanged. It can be used for either one way messaging or request response type communication. It does not depend on any specific transport protocol but the standard HTTP (Hypertext Transfer Protocol) is most popular.

Ontologies

The word ontology comes from the Greek *ontos* for being and *logos* for word. Ontologies have been a subject of research for a very long time. In fact the origins of ontologies can be traced back to the times Aristotle. Of late ontologies are receiving more than a fair share of attention, given their potential use in a plethora of applications for achieving interoperability. More recent definitions (and the one most widely used) include the one by [Gruber, 1993] who defines an ontology as "an explicit specification of a conceptualization". [Uschold and Gruninger, 1996] define ontology as a "shared understanding of some domain of interest". To summarize we can say an ontology provides -

- A common vocabulary of terms
- Some specification of the meaning of the terms (semantics)
- An agreement for shared understanding for people and machines.

What makes ontologies more powerful is their ability to abstract the real world to a conceptual hierarchy (class, subclass). Ontologies have evolved from vocabularies, taxonomies etc. to more formal and expressive specifications like OWL (Web Ontology Language) [Pan and Horrocks, 2001]. The feature that makes these formal specifications more expressive and different from general taxonomies is their ability to capture the relationships among the ontological concepts. The ontologies referred to in this thesis are such formal ontologies.

Semantic Web and Semantic Web services

"*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*" [Berners-Lee et.al, 2001]. The primary aim of the semantic web was its extended use in the current web to provide better organization and access to information and for information to be machine processable. This is done by explicitly adding semantics to the existing data using ontologies.

This concept of the semantic web can be extended to web services to envision Semantic web services. Web services as described above provide only syntactic interoperability because although the syntax and structure of the exchanged data is verified, the information content is not "understood". To provide this "understanding" additional semantic information is required. Ontologies can provide this additional information to make web services even more interoperable. This convergence of two technologies "the semantic web" and "web services" is termed as "Semantic Web Services".

METEOR-S

The vision of Semantic Web services can be realized using two different approaches. The first approach is to develop new semantic web based standards for web services [Fensel and Bussler, 2002; OWL-S, 2004]. The second approach is to add semantics to existing Web services standards [Sivashanmugam et.al, 2003]. The METEOR-S project at the LSDIS lab uses the latter approach. Since current web service standards are already widely adopted adding semantics would make their adoption easier rather than shifting to a completely new model or standards.

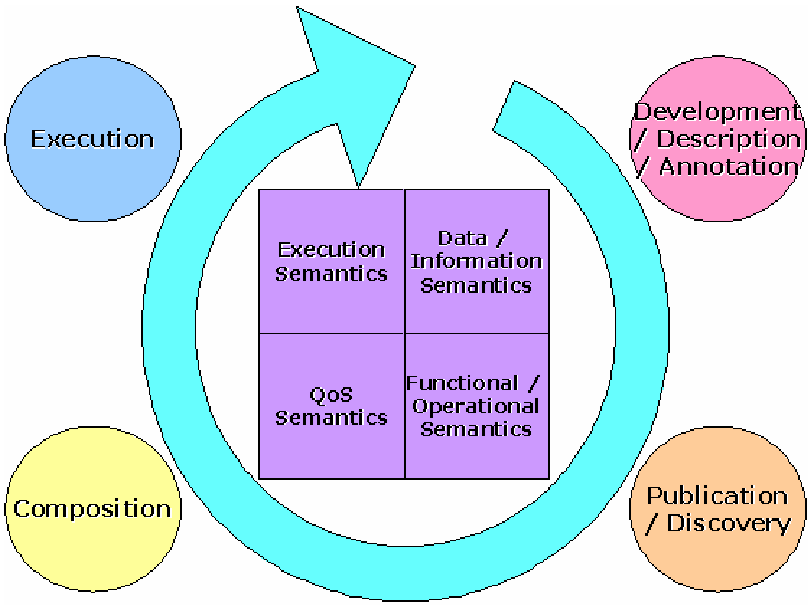

Execution
Development / Description / Annotation
Execution Semantics
Data / Information Semantics
QoS Semantics
Functional / Operational Semantics
Composition
Publication / Discovery

Figure 2. Four types of Semantics in Web services

The METEOR-S project identifies four types of semantics in the Web service domain as

- Data Semantics (semantics of inputs / outputs of Web services)

- Functional Semantics (what does a service do)
- Execution Semantics (correctness and verification of execution)
- QoS (Quality of Service) Semantics (performance / cost parameters associated with service)

A detailed description of these four types of semantics is given by [Cardoso and Sheth, 2005] and can be summarized as shown in Figure 2**Error! Reference source not found.**. These four types of types of semantics can be added to the different layers of the Web services stack. Figure 3 shows the different layers of the Web services stack. At the description layer data, functional and QoS semantics can be added.

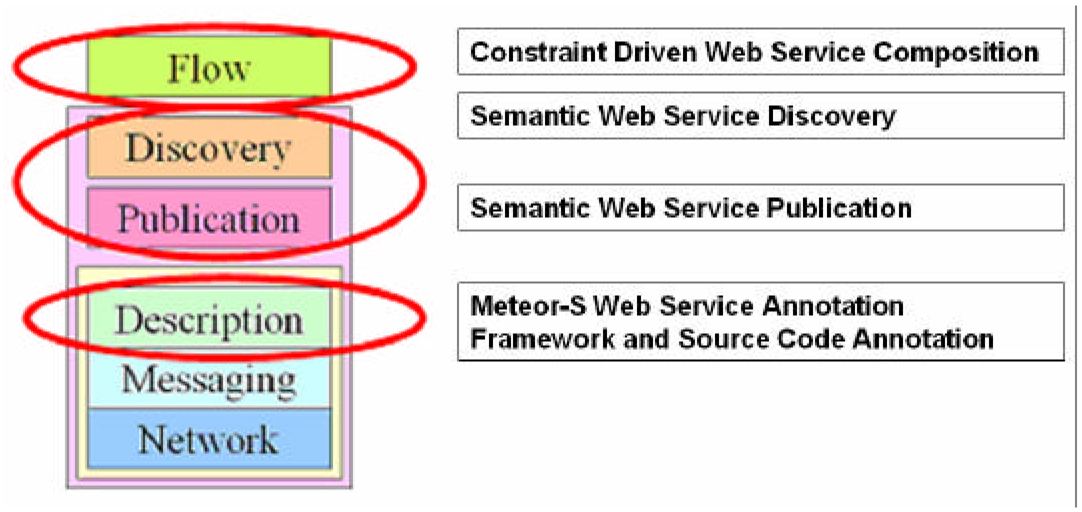

Flow
Discovery
Publication
Description
Messaging
Network
Constraint Driven Web Service Composition
Semantic Web Service Discovery
Semantic Web Service Publication
Meteor-S Web Service Annotation Framework and Source Code Annotation

Figure 3. Web Service Stack and METEOR-S

METEOR-S uses MWSAF [Patil et.al, 2004] and source code annotations [Rajasekaran et.al, 2004] to add data and functional semantics. At the publication and discovery layer data, functional and QOS semantics can be used. MWSDI (METEOR-S Web Service Discovery Infrastructure), the discovery component of METEOR-S provides an infrastructure to leverage data, functional and QoS semantics by enhancing UDDI and/or WSDL [Verma et.al, 2004]. The topmost layer of the web service stack i.e. the flow layer deals primarily with service composition and can utilize Execution and QoS semantics. METEOR-S tries to provide a comprehensive composition framework using MWSCF [Sivashanmugam et.al, 2004c] and Constraint Driven Web service composition [Aggarwal et.al, 2004]. The work presented in this thesis fits at the discovery layer of the web service stack and can be seen as an extension of MWSDI.

CHAPTER 3

SEMANTIC WEB SERVICE DISCOVERY IN A MULTI-ONTOLOGY ENVIRONMENT

The interoperability problem has been the focus of extensive research in the area of application integration [Sheth, 1998; Chawathe et.al, 1995; Papakonstantinou et.al, 1996]. The arrival of XML-based technologies like Web services, hoped to provide a better solution for achieving interoperability at not only the enterprise-level but also at the web level [Curbera et.al, 2001; Stal, 2002]. The XML based standards of Web services however guarantee only syntactic interoperability. Applications can verify the structure of the data sent but they cannot really understand the information captured by that data, and hence the capabilities of that Web service. This poses a big obstacle in finding Web services based on their capabilities. Ontologies have been identified by many researchers as the key ingredient to provide semantic understanding of Web service capabilities [Paolucci et.al, 2002; Fensel and Bussler, 2002; Cardoso et.al, 2002] . Ontologies help Web services to agree on a set of concepts and relationships allowing a shared understanding of the common domain knowledge. This allows matching service requests to service advertisements more efficiently and with less human intervention.

The best possible scenario would be that all the Web services adhere to a single global ontology. Such a single large global ontology is however highly impractical considering the variety of applications and the intent behind their development. A more practical approach would be for Web services to conform to local, domain or application specific ontologies [Paolucci et.al, 2002b]. However, even in a single domain, different domain experts, users, research groups and organizations can conceptualize the same real world entities differently leading to multiple domain ontologies. Besides, the scope of an ontology or for that matter a domain is usually arbitrary and cannot be formally defined. This is because multiple ontologies can model the same function differently. For example, travel syndicators like Travelocity, Expedia etc. partner with different airlines. Each airline may have their own specific ontology. Another reason is because multiple ontologies may exist for an activity. For example, a travel planning service from an organization may require ontologies from the hospitality domain and the travel domain. Yet another reason may be the co-existence of independent organizations and organizational changes. For example, Cisco acquired 40 companies. Each company may adhere to their own ontology. Thus differences and overlaps between the models used for ontologies can exist. Moreover the agreements that are captured in relation to a domain may differ when seen from different perspectives and at different levels of details. Hence, when a person or organization chooses an ontology to annotate he may choose any one of these available ontologies that best aligns to the web service he is trying to annotate. In addition, the Web service itself can span multiple domains. Thus more than one ontology may be required while annotating a single web service. In such an environment where the service requests and advertisements adhere to different ontologies, there arises a need for a matching algorithm which can deal with the domain semantics modeled by multiple ontologies.

This thesis addresses the challenge of Web Service discovery in a Multi-Ontology Environment, in effect addressing some the semantic heterogeneity issues in the presence of multiple ontologies. The approach for finding the appropriate Web service consists of three steps. First the user's service requirement is captured in a semantic Service Template [Sivashanmugam et.al, 2003; Sivashanmugam et.al, 2004a]. This Service Template is then matched against a set of candidate Web services. The result of this operation is then returned to the user as a set of ranked services. This chapter presents an overview of these three steps. The algorithms involved are detailed in the subsequent chapters.

1. Creating Search Templates

A Service Template depicts the user's intent behind searching a Web service. This allows a user to specify the operations, inputs and outputs of the desired Web service using a domain specific ontology. The Service Template does not have a concrete implementation and can be seen as a proxy Web service. Formally a Service Template (ST) can be defined as -

$$ST = < N_{ST}, D_{ST}, Ops_{ST} < N_{OP}, D_{OP}, Os_{ST}, Is_{ST} > >$$

Where, N_{ST} - Name of the Web service to be found

D_{ST} - Textual description of the Web service

OPs_{ST} - Set of operations of the required Web service

The operations in turn are specified as

$OPs_{ST} < N_{OP}, D_{OP}, Os_{ST}, Is_{ST} >$

Where,

N_{OP} - Name of the operation

D_{OP} - Description of the operation

Os_{ST} - Outputs of the operation

Is_{ST} - Inputs of the operation

As an example consider a Service Template which depicts a simple Web service that takes a stock symbol as input and returns the stock quote as shown in Table 1. This Service Template is used throughout the thesis to describe the algorithm.

Table 1. Search Template

	Search Template	***Ontological Concept***
Service Name	StockService	
Operation	getStockQuote	
Input	Ticker	StockTicker
Output	StockQuote	StockQuote
Operation	getCompanyNews	
Input	CompanyName	companyName
Output	CompanyNews	News

2. Finding Set of Candidate Web services

The Service Template is matched against the services in a selected registry using the algorithms detailed in chapter 3 and chapter 4. These services against which the ST is matched are called Candidate Services (CS). A CS is described in the same fashion as the Service Template and is formally defined as -

$CS = < N_{CS}, D_{CS}, OPs_{CS} < N_{OP}, D_{OP}, Os_{CS}, Is_{CS} > >$

Where, the terms correspond to the terms in the Search Template

3. Result aggregation

Each CS when matched against a ST gives a Match Score (MS) which is normalized in the range of [0-1]; 0 being the worst and 1 being the best Match Score. All the Candidate Services with a Match Score above a user provided threshold value are then ranked in ascending order and returned to the user as a set of matched services.

CHAPTER 4

SERVICE DISCOVERY ALGORITHM

The Service Template described in Chapter 2 is matched to the set of candidate Web services in a registry. Match Scores are calculated for each (ST, CS) pair. These match scores are normalized over the interval of 0 to 1. Finally the pairs are ranked in the descending order of the match score. The next few sections will explain the matching algorithm with the help of the following candidate Web service.

Table 2. Candidate Service

	Candidate Service	***Ontological Concept***
Service Name	XQuotes	
Operation	getQuickQuote	
Input	quickQuoteInput	MultipleQuote
Output	quickQuoteResponse	QuickQuote
Operation	getQuote	
Input	quoteInput	MultipleQuote
Output	quoteResponse	StockQuote
Operation	getFundQuote	
Input	fundQuoteInput	FundQuoteQuery
Output	fundQuoteResponse	fundQuote

Due to a lack of domain ontology for stocks, the Service Template (ST) (described in Chapter3) is annotated with a home-grown stock ontology called StocksOnt.owl[1]. The CS used for this example is provided by a Web service company, called xIgnite. In order to capture the domain understanding intended by the concepts used by this Web service, it is annotated with an ontology created from a set of Web services provided by xIgnite. This ontology is called xIgniteStocks.owl[2].

The Service Template (ST) is matched to the candidate Web service based on the required functionality. This is achieved by matching the operations of the ST and CS rather than the inputs and outputs [Cardoso and Sheth, 2003; Paolucci et.al, 2002].The match function matchServices (ST, CS) compares the ST and CS and returns a similarity value in the range of 0 and 1. This similarity value has two dimensions – Syntactic Similarity and Functional Similarity.

Syntactic Similarity – This computation is based only on syntactic considerations and no semantic information is taken into account. It relies on the name and the description of Service Template (ST) and Candidate Service (CS).

Functional Similarity – Considering only the inputs and outputs for matching is not sufficient as they can be grouped differently to give different functionality. Operations are used as the matching unit while matching ST and CS. These operations are in turn matched based on their inputs and outputs.

$$MS(ST,CS) = \frac{w1 * SyntacticSim(ST,CS) + w2 * FunctionalSim(ST,CS)}{w1 + w2} \in [0 \ldots 1]$$

Equation 1. Service Similarity Calculation

[1] StocksOnt.owl can be found at http://lsdis.cs.uga.edu/Projects/METEOR-S/MWSAF/Ontologies/Stock/StocksOnt.owl

[2] xIgniteStocks.owl can be found at http://lsdis.cs.uga.edu/Projects/METEOR-S/MWSAF/Ontologies/Stock/xIgniteStocks.owl

The overall service similarity is the weighted average of Syntactic and Functional similarities and is calculated as shown in Equation 1.

If the Service Template does not have a name or the user does not want to use the name and description similarity while matching services then w1 can be reduced to 0 and the MS will then consist of only the Functional similarity.

4. Syntactic Similarity

The similarity function *SyntacticSim* measures the similarity between the names and descriptions of the Service Template and the Candidate Service. The result of this function i.e. Syntactic Similarity of ST and CS is normalized on a scale of 0 to 1. Equation 2 summarizes the calculation for syntactic similarity.

$$SyntacticSim(ST, CS) = \begin{cases} \dfrac{w3 * NameMatch(ST, CS) + w4 * DescrMatch(ST, CS)}{w3 + w4} \in [0 \ldots 1] & Descr[ST] \neq f \; or \; Descr[CS] \neq f \\ NameMatch(ST, CS) & Descr[ST] = f, Descr[CS] = f \end{cases}$$

Equation 2. Calculation of Syntactic Similarity

It is calculated as the weighted average of the Name similarity (*NameMatch (ST, CS))* and the Description Similarity *(DescrMatch (ST, CS))*. The weights ω_3 and ω_4 are also real values between 0 and 1, and indicate the degree of certainty that a user has of the service name and service description supplied during a request. High weight values indicate that the user is confident that the information he supplied to search a Candidate Service is correct. If both ST and CS do not have a description then the syntactic similarity is the same as name similarity.

The **name similarity** is calculated using various name and string matching algorithms like NGram, synonym matching, abbreviation expansion, stemming, tokenization etc. The *NGram* algorithm calculates the similarity by considering the number of qgrams [Angell et.al., 1983; Salton, 1988; Zamora E., 1981] that the names of two concepts have in common. The *CheckSynonym* algorithm uses WordNet [Miller et.al, 1990] to find synonyms whereas, a custom abbreviation dictionary is used by the *CheckAbbreviations* algorithm. The *TokenMatcher* uses the Porter Stemmer [Porter, 1980] algorithm, tokenization, stop-words removal, and substring matching techniques to find the similarity. It first tokenizes the string based on punctuation and capitalization. Then it removes unnecessary words from the list of tokens, using a stop-word list. If it can not match these individual token then it stems them using porter stemmer algorithm and tries to match them using the NGram technique. If any of these algorithms return a full match, i.e., 1 on scale of 0 to 1, then a match score of 1 for name similarity is returned. If all the match algorithms give a match value of zero, then the name similarity of those concepts is 0. If on the other hand, none of the match algorithms give a match score of 1, i.e., an exact match, then the average of all non-zero match scores is taken. The **description similarity** on the other hand uses only the n-gram algorithm.

5. Functional Similarity

This algorithm uses functional similarity as a measure for calculating the similarity between two services. The functional similarity is given by matching the operations of ST and CS. To understand the semantics of an operation it is not enough to consider just the names of the operation but it is also important to consider the inputs and outputs of that operation. Changing the inputs and outputs can result in entirely different operations. Hence functional similarity is not just the syntactic similarity of the operations but also the similarity between its inputs and outputs. In addition since the web service operations are also annotated with functional ontologies, functional concept similarity is also used as a measure for calculating functional similarity. The functional similarity (FS) is calculated as the average Match Score of the operations of ST and CS and is represented by Equation 3.

$$FS = \frac{\sum_{i=0}^{n} fs_i}{n} \in [0 \dots 1]$$

$$where, \quad fs_i = best\ functional\ similarity\ of\ an\ Operation\ of\ ST$$
$$n = number\ of\ Operations\ of\ ST$$

Equation 3. Functional Similarity for Service

The mappings between operations of ST and CS are calculated. A function *getFM(ST, CS)* returns a best mapping for each ST operation and can be described as in Equation 4.

$$getFM\ (OP_{ST}, OP_{CS}) = best(getfm\ (OP_{ST}, OPi_{CS}))$$
$$where\ , OPi_{CS}\ represents\ individual\ operation\ of\ CS$$

Equation 4. Finding the best mapping for individual operation

In the above equation *getfm* calculates the functional similarity (*fs*) for an individual operation pair. Each operation of ST is mapped against each operation of CS and the best matching CS operation is selected for every ST operation. In the example, getStockQuote operation of ST is mapped to all three operations of CS, and the best matching operation is considered for calculating the final service match.

Operations of ST and CS are annotated with ontological concepts, and these concepts are also considered while calculating the similarity between these operations. Thus *fs* comprises of Syntactic similarity, conceptual similarity and IO similarity and is given by Equation 5.

$$fs = getfm(OP_{ST}, OP_{CS}) = \frac{w5 * SynSim(OP_{ST}, OP_{CS}) + w6 * ConceptSim(OP_{ST}, OP_{CS}) + w7 * IOSim(OP_{ST}, OP_{CS})}{w5 + w6 + w7}$$

Equation 5. Matching operations

Syntactic Similarity of Operations *(SynSim)* – It is the similarity of the names and descriptions of the operations to be matched and is calculated in the same manner as the Syntactic Similarity of services.

Conceptual Similarity of Operations *(ConceptSim)* – It is the similarity of the ontological concepts to which the Web service operations of the ST and CS are annotated. The next chapter details how these concepts are actually matched.

IO Similarity of Operations *(IOSim)* – This is the similarity between the inputs and outputs of operations and is calculated in terms of the similarity of inputs i.e. *InputSim* and similarity of outputs i.e. *OutputSim*. The definition of the IOSim (Equation 6) captures three distinct cases based on the inputs and outputs of ST and CS

$$IOSim(OP_{ST}, OP_{CS}) = \begin{cases} \sqrt{InputSim(OP.Is_{ST}, OP.Is_{CS}) * OutputSim(OP.Os_{ST}, OP.Os_{CS})} & OP.Is_{ST} \neq f, OP.Os_{ST} \neq f \\ InputSim(OP.Is_{ST}, OP.Is_{CS}) & OP.Is_{ST} \neq f, OP.Os_{ST} = f \\ OutputSim(OP.Os_{ST}, OP.Os_{CS}) & OP.Is_{ST} = f, OP.Os_{ST} \neq f \end{cases}$$

Equation 6. IO similarity for Operations

- If an operation of ST has both a set of inputs and a set of outputs the similarity is calculated as the arithmetic mean of *InputSim* and *OutputSim*
- If an operation of ST only specifies a set of outputs and no inputs, then the function only computes the similarity between the outputs of the Service Template and the outputs of the Candidate Service

- The same concept will be applied if the operation of ST includes inputs but not outputs

The function *InputSim(OP.Is$_{ST}$, OP.Is$_{CS}$)*, computes the best set of mappings that can be obtained from matching the inputs of operation of ST with the inputs of operation of CS. This is done using the dynamic programming and can be represented as a recursive relation as shown in Equation 7.

$$\begin{aligned}&InputSim\left(OP.Is_{ST}, OP.Is_{CS}\right)\\&= Max\left(InputSim\left(OP.Is_{ST} - OP.I_{ST}, OP.Is_{CS} - OP.I_{CS}\right) + iSim\left(OP.I_{ST}, OP.I_{CS}\right)\right)\end{aligned}$$

Equation 7. Calculating the best set of mappings for Inputs

For each potential mapping, the function *iSim(OP.I$_{ST}$, OP.I$_{CS}$)* is employed to evaluate the similarity of a single input of a Service Template operation with a single input of Candidate Service operation. As these inputs are annotated with the ontological concepts, *iSim* actually matches two ontological concepts. Chapter 5 describes in more detail how two ontological concepts are mapped. Output similarity is also calculated in the same way as input similarity.

CHAPTER 5

CONCEPT MATCHING ALGORITHM

There has been a significant amount of research in the area of ontology matching. The related work described in Chapter 7 gives a brief insight into some of the matching techniques used for ontology matching. Some approaches [Cardoso and Sheth, 2003], use properties as a measure of finding similarity between two ontological concepts, but these approaches are plagued with the problem of false matching. This is because properties are matched based on their names and ranges, and two entirely different concepts can still have properties with the same names. In the example, the FundQuote concept from xIgniteStocks ontology has property names similar to the StockQuote concept of the StockOnt ontology. Some approaches, consider [Paolucci et.al, 2002; Gonzales et.al, 2001] subsumption relationships to find the similarity, but these are also limited as they match concepts from same ontology.

As explained in Chapter 4, inputs, outputs and functions of the Service Template and Candidate Service are annotated with ontological concepts, and hence it is important to match these concepts while finding the similarity between them. Functional ontologies are an area of active research. Currently taxonomies with no properties are considered. Concept matching for all ontological concepts is first explained in general and later concept matching for the case and when the concepts are from the same ontology is explained.

This approach tries to eliminate the limitations faced by other approaches by introducing two new measures, Coverage Similarity and Context Similarity. In addition property matching is made more efficient by adding a penalty for unmatched properties.

In general there are four different parts of the similarity calculation of two ontological concepts as shown in Equation 8.

$$conceptSim(C_{ST}, C_{CS}) = \frac{w8 * synSim + w9 * propSim + w10 * cvrgSim + w11 * ctxtSim}{w8 + w9 + w10 + w11}$$

Equation 8. Calculating Similarity between two Ontological Concepts

1. Syntactic Similarity *(synSim)*

This is the measure of syntactic similarity of the concepts. This includes matching the name and description of the concepts. The names are matched using techniques described earlier in chapter 3 about String Matching algorithms, whereas descriptions are matched using the n-gram technique.

2. Property Similarity *(propSim)*

Since a concept is defined using its properties, it is important that these properties should be matched while matching two concepts. These properties are matched as one to one mappings and the mappings are calculated in such a way that the average property match is maximized. Equation 9 shows how the property similarity is calculated.

$$PS = \frac{1}{n}\sum_{i=0}^{n} ps_i \in [0 \ldots 1]$$

$$propSim(C_{ST}, C_{CS}) = Max(propSim(C.P_{ST} - p_{ST}, C.P_{CS} - p_{CS}) + propMatch(p_{ST}, p_{CS}))$$

Equation 9. Finding property Similarity

propMatch calculates similarity *(ps)* between two individual properties of the concepts. While matching two properties it is important to remember that the values the properties can take are characterized by their ranges and hence range similarity is an important part of the similarity function. Since cardinality provides the information about how many range values a property can take at a time, it is also considered while calculating the *propSim*. The final part of the *propSim* is the syntactic information of the property i.e. name and description. The property similarity for unmatched properties is also penalized with a penalty of 0.05 for each unmatched property.

Equation 10 and Table 4 summarize this information and shows the calculations for *propSim*.

$$propSim = c * \sqrt[3]{rangeSim * crdnSim * synSim} - 0.05 * |unmatched\ properties|$$

Equation 10. Property Similarity

propSim also has another contributor *c* which is a constant decided based on Equation 11.

$$c = \begin{cases} 1 & p_{ST}, p_{CS}\ are\ inverse\ functional\ properties \\ 1 & p_{ST}, p_{CS}\ are\ not\ inverse\ functional\ properties \\ 1 & cardinality(p_{ST}) \geq 1\ and\ p_{CS}\ is\ inverse\ functional\ property \\ 0.8 & cardinality(p_{CS}) \geq 1\ and\ p_{ST}\ is\ inverse\ functional\ property \end{cases}$$

Equation 11. Calculating the Constant factor of Property Similarity

This constant is calculated based on if the properties being mapped are inverse functional properties or not. Inverse Functional property is an OWL construct which tells that if a property is inverse functional then that property value is unique for every instance of that concept. Consider the following example for an inverse functional property. SSN is unique for a person and Stock Symbol is unique for every Stock. No two stocks can have same stock symbol and no two persons can have same SSN. This information gives some more insight into the real world entity that is being captured by the concept being matched and hence is an integral part of the similarity calculations. For non-owl ontologies the second case of Equation 11 is considered.

2.1 Syntactic Match *(syntacticMatch)*

This provides the syntactic similarity (syntacticSim) component of the propSim. It is calculated as the similarity of names and descriptions of the properties. This again uses the same algorithms mentioned in the Chapter 3 and is the weighted average of name and description similarity.

2.2 Range Match *(rangeMatch)*

The similarity component provided by the range match is called *rangeSim*. Range of a property can either be a primitive data type or another ontological concept. Hence based on the ranges of the properties being matched, three different cases are possible.

The first case is when both the properties have primitive data types as their ranges. In such a case data type conversion is required. Table 3 gives the match scores based on the information loss involved in such conversions.

Table 3. Datatype conversion values for Range Match

Source Datatype	Target Datatype	Match Score
Decimal	String	1
Decimal	Float	1
Decimal	Double	1
Long	Decimal	3/4
Long	Integer	1
Integer	String	1
Integer	Long	2/3
Integer	Decimal	1/3
Integer	Float	1/3
Integer	Double	1/3
String	Integer	½
String	Decimal	½

The second case is that both the property ranges are Ontological concepts. In such case a **Shallow Concept Match** is done to match these range concepts. This shallow concept match involves matching names and descriptions of these concepts and matching the properties of these concepts syntactically as in Equation 12.

$$rangeSim = rangeMatch\ (p_{ST}, p_{CS}) = \frac{w12 * synSim + w13 * propSynSim}{w12 + w13}$$

Equation 12. Shallow Concept Match

propSynSim is calculated as the fraction of properties that the range concept for ST property has in common with the range concept of CS property and is called as **Shallow Property Match**. Equation 13 describes this Shallow Property Match.

$$propSynSim(p.Range_{ST}, p.Range_{CS}) = \frac{|p(p.Range_{ST}) \cap p(p.Range_{CS})|}{|p(p.Range_{ST})|}$$

Equation 13. Shallow Property Match

The third and the simplest case is when one property has a primitive data type as range and the other has the range as an ontological concept. In such a case as the ranges are incompatible the range match is 0. However making the range similarity 0 will also make the property similarity 0. This is not desirable, as ontologies model concepts with different levels of details. In order to avoid the property similarity to become entirely 0, additional conditions like the value of syntactic similarity can be checked and the range similarity can be set to a default value. For example, if both the properties are inverse-functional properties[3] and their syntactic

[3] Inverse-functional property has a unique value for each instance of the Concept, e.g. StockSymbol is unique for every stock

similarity is more than 0.5 and their cardinality is also the same then we assign a default value of 0.5 to the range similarity.

2.3 Cardinality Match *(cardinalityMatch)*

Cardinality also plays an important part when two properties are matched. For example, when the property from ST is expecting more than one value and the CS property has only one value, the match would be less as the ST requirement and is not satisfied completely. Equation 14 summarizes different cases for cardinalities. The value returned by this function contributes as *crdnSim* for the *propSim* calculations.

$$crdnSim(p_{ST}, p_{CS}) = \begin{cases} 1 & cardinality(p_{ST}) = cardinality(p_{CS}) \\ 1 & p_{ST} \text{ and } p_{CS} \text{ are functional properties} \\ 0.9 & cardinality(p_{ST}) < cardinality(p_{CS}) \\ 0.7 & cardinality(p_{ST}) > cardinality(p_{CS}) \end{cases}$$

Equation 14. Cardinality Similarity for properties

2.4 Property Penalty

When properties of two concepts are matched, different scenarios arise based on the number of properties both the concepts have and the actual number of properties matched. The candidate concept can have the same number of properties as the requirement concept, or it can have less number of properties or can have more number of properties.

For the cases where number of properties of candidate concept, are same or more than number of properties of the requirement concept, a one to one mapping for all the properties of requirement concept can be found. But when the candidate concept has less number of properties then only some of the property requirements of the ST concept can be satisfied giving a many to one mapping. In such cases the candidate concept is penalized for not satisfying all the requirements by reducing the match score by a value proportional to the number of unmatched properties. With experiments, a reduction of 0.05 for every unmatched property is found to be fair enough for getting proper match scores.

2.5 Example of PropSim

Table 4 shows an example of the property similarity for two *StockQuote* concepts from different ontologies.

Table 4. Example for Property Similarity

Property Pair		Syntactic Similarity	Range Similarity	Cardinality Similarity	c	Total Similarity
StockOnt	xIgniteStocks					
bestBidSize	Bid_Size	0.72	1.00	0.90	1.00	0.87
bestAsk	Ask	0.66	1.00	0.90	1.00	0.84
bestBid	Bid	0.53	1.00	0.90	1.00	0.78
bestAskSize	Ask_Size	0.70	1.00	0.90	1.00	0.86
lastSale	Last	0.79	1.00	0.70	1.00	0.82
Open	Open	1.00	1.00	1.00	1.00	1.00
dayLow	Low	0.54	1.00	1.00	1.00	0.81
fiftyTwoWeekLow	Low_52_Weeks	0.54	1.00	0.90	0.80	0.63
previousClose	Previous_Close	0.16	1.00	1.00	1.00	0.54
quoteTime	Time	0.56	0.50	1.00	1.00	0.65
netChange	Change	0.80	1.00	1.00	1.00	0.93
dayHigh	High	0.66	1.00	1.00	1.00	0.87
Date	Date	1.00	0.50	1.00	1.00	0.79
tradingVolume	Volume	0.56	1.00	1.00	1.00	0.82
tickerSymbol	Symbol	0.60	0.50	1.00	1.00	0.67
fiftyTwoWeekHigh	High_52_Weeks	0.56	1.00	0.90	0.80	0.64

It can be seen that for *tickerSymbol* and *Symbol* pair has range similarity as 0.5. The reason is, *StockOnt* models range of *tickerSymbol* as concept *StockTicker* whereas, *xIgniteStocks* models range for *symbol* as a string. By human perception one can deduce that they both represent the Stock Symbol for the *StockQuote* concept but the range match will be zero as one is a data-type property while other is an object property. To avoid this, extra conditions are checked. As syntactic similarity is more than 0.5, the cardinality is also the same (crdnSim = 1) and both the properties are inverse-functional properties, a value of 0.5 is assigned to the range similarity.

fiftyTwoWeekLow and *fiftyTwoWeekHigh* are modeled as inverse-functional properties in *StockOnt* ontology but their counterparts in the *xIgniteStocks* are not. However, they have a cardinality of 1 giving the value for c as 0.8. Similarly, *bestBidSize* from *StockOnt* has cardinality of 1 whereas, *Bid-Size* from *xIgniteStocks* has cardinality greater than 1 and hence a value of 0.9 for the caridianlity similarity is used.

3. Coverage Similarity *(cvrgSim)*

This is another dimension of the concept similarity which measures the extent of knowledge that a concept covers. A concept in ontology is the abstract representation of a real world entity and it has sub-concepts which specialize this concept to an even more specialized real world entity. For example, an ontological concept Automobile can have sub-concepts as TwoWheeler and FourWheeler. Hence, when the requirement is the concept automobile, it can be satisfied either with TwoWheeler or FourWheeler. Considering this, another measure "coverage similarity" is added while measuring concept similarity. We will encounter different cases in this scenario as shown in Equation 15.

$$cvrgSim(C_{ST}, C_{CS}) = \begin{cases} 1 & MS(C_{ST}, C_{CS}) \geq 0.8 \\ 1 - 0.1 * 2^{x-1} & MS(C_{ST}, parent_x(C_{CS})) \geq 0.8 \; and \; x \; is \; the \; level \; of \; parent \; above \; ST \; concept \\ 1 - 0.05 * y & MS(C_{ST}, child_y(C_{CS})) \geq 0.8 \; and \; x \; is \; the \; level \; of \; child \; below \; ST \; concept \\ 0 & otherwise \end{cases}$$

Equation 15. Coverage Similarity

The coverage similarity is reduced exponentially for super-concepts based on the level of ancestry. For example, if an immediate parent of the requirement concept is being matched then the coverage similarity is reduced by 0.1, for a grandparent by 0.2 and so on till it reduces to zero. Similarly, the reduction by a multiple of 0.05 is employed if the candidate concept is a sub-concept. Here the multiples are the levels one has to go below the required concept to reach the candidate concept. The reason behind a lesser penalty for the sub-concept is that the sub-concept satisfies the properties of the requirements completely, but it still needs to be distinguished from a complete match.

Concepts are matched with a shallow concept match while calculating Coverage Similarity. This shallow concept match differs from the one described in range similarity in terms of how it calculates the shallow property match. In this case the number of properties of the candidate is also considered and hence the shallow property match is calculated by the Equation 16.

$$propSim(Concept_{ST}, Concept_{CS}) = \frac{|\, p(Concept_{ST}) \cap p(Concept_{CS}) |}{|\, p(Concept_{ST}) \cup p(Concept_{CS}) |}$$

Equation 16. Shallow property match for Coverage Similarity

A detailed example of the coverage similarity with the Service Template is explained in the testing chapter.

4. Context Similarity *(ctxtSim)*

Most of the approaches for matching ontological concepts consider only the information that describes those concepts i.e. name, description and properties of these concepts. But many times this information is insufficient and confusing, and leads to false matches.

Considering this shortcoming, another dimension called Context Similarity is added to the concept similarity. It is an attempt to understand more about a concept by considering the semantic similarity and semantic disjointness of the concepts in the vicinity of that concept. For example, a Bond means a Fixed Income Fund and not Stocks; Bond is also a different concept from the concept Investment Company. Such information is captured in two different sets.

The first set i.e. SameConceptSet is the set of the concepts which describes the same real world entities as described by the concept in question. The second set i.e. DifferentConceptSet is the set of concepts which are semantically disjoint from the concept in question. Considering the above example for the concept Bond the two sets will be as follows

SameConceptSet(Bond) = {FixedIncomeFund, Bond}

DifferentConceptSet(Bond) = {Stocks, InvestmentCompany}

Sets are created as follows

- SameConceptSet
    - Concepts are added to SameConceptSet, if an ontology specifies them as same or equivalent concepts e.g. OWL language has constructs like sameClassAs or equivalentClass to describe that two concepts are similar to each other
    - The SameConceptSet also contains the member concepts of the main concept i.e. concepts which are used to define main concept e.g. OWL has collection classes, which are described in terms of other classes

    - The concept itself is also added in the SameConceptSet

- DifferentConceptSet
    - Concepts which are explicitly modeled as disjoint concepts of the main concept e.g. in OWL concepts which are related with disjointWith or complementOf relationships
    - Concepts appearing as ranges of properties of main concept except if the range is concept itself e.g. in the stocks ontology, Company concept has a property with Stocks concept as range and hence Company and Stocks do not represent same the concept and Stocks will go in DifferentConceptSet of Company
    - Concepts which has properties with main concept as range e.g. Stocks appears as range of investsIn property of MutualFund concept and hence MutualFund goes in the DifferentConceptSet of Stocks
    - Siblings of the main concept are also added to the DifferentConceptSet as they depict an entirely different specialization than the main concept e.g. EquityFund and FixedIncomeFund are siblings and can not replace each other in a request

Once these sets are established, the context match can be calculated. The context match is given by the equation 17.

$$contextSim = \begin{cases} 1.0 & \exists\ SC_{ST}\left(shallowConceptMatch\left(SC_{ST},\ SC_{CS}\right) > 0.8\right) \\ & where, \\ & SC_{ST} \in sameConceptSet\left(C_{ST}\right),\ SC_{CS} \in sameConceptSet\left(C_{CS}\right) \\ -1.0 & \exists\ SC_{ST}\left(shallowConceptMatch\left(SC_{ST},\ DC_{CS}\right) = 1\right) \\ & where, \\ & SC_{ST} \in sameConceptSet\left(C_{ST}\right),\ DC_{CS} \in differentConceptSet\left(C_{CS}\right) \\ -1.0 & \exists\ DC_{ST}\left(shallowConceptMatch\left(DC_{ST},\ SC_{CS}\right) = 1\right) \\ & where, \\ & DC_{ST} \in differentConceptSet\left(C_{ST}\right),\ SC_{CS} \in sameConceptSet\left(C_{CS}\right) \\ -1.0 & avgConceptMatch\left(SC_{ST},\ SC_{CS}\right) < 0.5\ and \\ & avgConceptMatch\left(SC_{ST},\ DC_{CS}\right) > 0.8\ or \\ & avgConceptMatch\left(DC_{ST},\ SC_{CS}\right) > 0.8 \\ \sqrt{ms_1 * ms_2} & where,\ ms_1 = avgConceptMatch\left(SC_{ST},\ SC_{CS}\right) \\ & and\ ms_2 = avgConceptMatch\left(DC_{ST},\ DC_{CS}\right) \end{cases}$$

Equation 17. Context Similarity

The equation represents the following steps –

- The SameConceptSet and DifferentConceptSet of the ST concept are matched against SameConceptSet and DifferentConceptSet of the CS concept. This match is a shallow concept match, as seen in the range match calculations in Equation 12. Each of these four combinations gives a set of mappings.
- Once these mappings are found, the following conditions are checked
  - If any concept mapping from (SameConceptSet$_{ST}$, SameConceptSet$_{CS}$) has a match score greater than 0.8, it signifies that the candidate concept and service

template concept resemble each other and hence a context match score of 1.0 is returned.

- If any concept mapping from (SameConceptSet$_{ST}$, DifferentConceptSet$_{CS}$) or (SameConceptSet$_{CS}$, DifferentConceptSet$_{ST}$) has match score equal to 1.0, it clearly signifies that the candidate concept and the Service Template concept model entirely different real world entities and hence the Context Match is -1.
- If the average match score of (SameConceptSet$_{ST}$, SameConceptSet$_{CS}$) mappings is less than 0.5 and average concept match score for either (SameConceptSet$_{ST}$, DifferentConceptSet$_{CS}$) or (SameConceptSet$_{CS}$, DifferentConceptSet$_{ST}$) mappings is greater than 0.8, it can be assumed that the candidate concept and service template concept do not model the same real world entity and hence the Context Match is -1.
- If none of the above cases occur then average match score of (SameConceptSet$_{ST}$, SameConceptSet$_{CS}$) mappings is calculated. Similarly average match score of (DifferentConceptSet$_{ST}$, DifferentConceptSet$_{CS}$) mappings is also calculated and then overall context similarity is given by the geometric mean of these two match scores.

5. Matching Concepts from Same Ontology

While matching concepts from the same ontology, first the coverage similarity is calculated. If the coverage similarity is not zero then that means that the concepts are linked to each other by a subsumption relationship. A Shallow Property Match is then calculated as shown in Equation 13. The context match is also defaulted to 1.0 in such case and accordingly the overall concept match is calculated. In the case where, the concepts are from the same ontology but are not connected to each other through any ancestors, the match is calculated in the same manner as matching concepts from different ontology.

The testing chapter describes a graph and an explanation to depict all the above dimensions of the Concept similarity with respect to the Service Template defined in chapter 3.

CHAPTER 6

RESULTS AND EMPIRICAL TESTING

The service discovery algorithm is tested on a set of 24 real world Web services from the Stocks domain to see how functional similarity and concept similarity, help to eliminate false matches. This set contains real world Web services that were gathered from categorized Web service indexes of salcentral.com and bindingpoint.com. A few more services derived from these services were also added to this set. Then these Web services are annotated using the MWSAF tool [Patil et.al, 2004] with two different ontologies. The first ontology named StocksOnt.owl[4] is based on nasdaq's web site and the online stock tutorial from investopodia whereas the second ontology xIgniteStocks.owl[4] was obtained from concepts used by xIgnite, a Web service company. This selection is adequate and sufficient for preliminary testing purposes as this set represents 14 different operations modeling scenarios with respect to a Service Template (detailed in Figure 5) and contains 8 distinct cases that can arise while matching two ontological concepts (detailed in Figure 4). Most of the annotated Web services will adhere to one of these 14 cases. For comprehensive testing, would require rich, real world ontologies and a vast set of well written existing Web services. Due to lack of either, ontologies rich enough to show the benefits of this approach were specifically developed for testing purposes.

Figure 4, shows how the context and the coverage similarity used by the concept matching algorithm help to get more accurate match scores between concepts. These improved

[4] A partial view of these ontologies is shown in appendix A

match scores help to eliminate false matches between service advertisements and templates as shown in the Figure 5.

The better performance of the Concept matching algorithm is shown with the help of 8 distinct cases. The *StockQuote* concept from the example ST is considered for this testing. This *StockQuote* concept is from the *StockOnt* ontology. While matching this concept with other concepts, eight distinct cases need to be considered. Four of these cases occur when the candidate concept is from the same ontology i.e. it is a direct match, a super-concept match, a sub-concept match or a completely different concept. The remaining four cases occur when the candidate concept is from a different ontology; again it could be a direct match, a super-concept match, a sub-concept match or a completely different concept. Figure 4 shows a graph of similarity values for each of these eight cases. This graph shows seven bars for each concept pair. The very first bar among these seven bars depicts the similarity value using only property similarity and syntactic similarity [Cardoso and Sheth, 2003]. The second bar shows the property similarity without considering the property penalty. The third bar shows the syntactic similarity. The fourth bar shows the context similarity. The fifth bar shows the coverage similarity. The sixth bar shows the property penalty by considering the property penalty. The second and the sixth bar can be compared to see the improvement using the property penalty. The last bar shows the similarity values calculated by the concept matching algorithm detailed in chapter 4. Each of these conditions are described in detail further with a comparison to the match scores calculated considering only property and syntactic similarity (the first and the last bar).

Candidate Concepts from Same Ontology

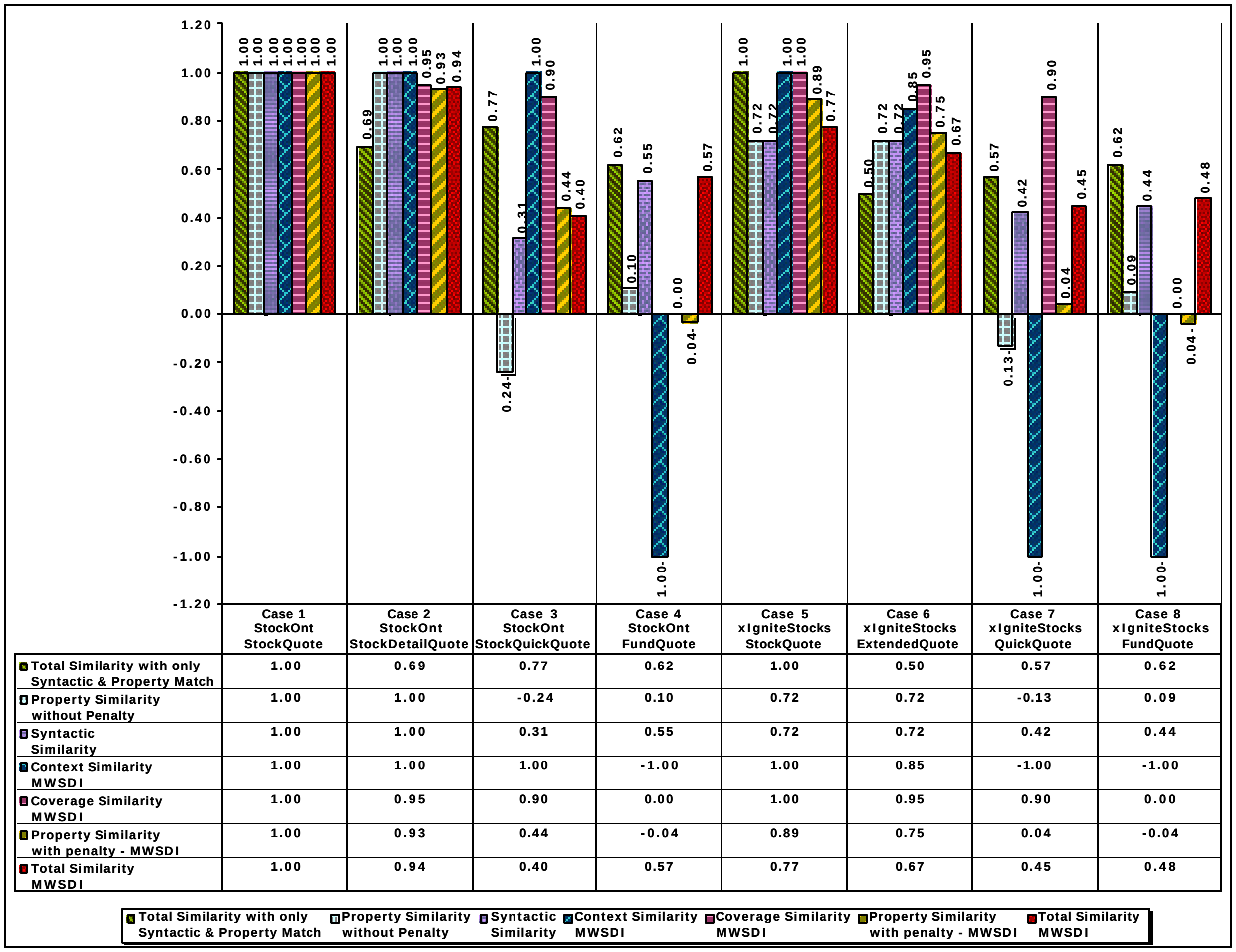
Case 1. The simplest and best case is when the candidate concept is the same as the ST concept. In this case all the four dimensions give a similarity of 1 and the overall match score is also 1. Here, considering only property and syntactic similarity will also give a match score of 1.

Case 2. The second case is when the candidate concept is a sub-concept of the ST concept. In such a case, even if the requirement is satisfied completely, it is still not the exact concept that is required and hence, for *StockDetailQuote*, the coverage similarity is reduced by 0.05 as it is the *immediate* child of *StockQuote*. As coverage similarity has a non-zero value and the concepts are from the same ontology, the context similarity is defaulted to 1. This boosts the overall similarity even more to give a better match score compared to the similarity calculated by using only property and syntactic similarity.

Case 3. When the candidate concept is a super-concept of the requirement, it does not satisfy all the properties of the requirement. Here a penalty of 0.05 is applied for each unmatched property. This reduces the property similarity for *StockQuickQuote* to a negative value, which signifies that most of the properties of the requirement are not satisfied. A deduction of 0.1 is also applied to the coverage match, as the candidate concept is the *immediate* parent of the requirement. Again since the coverage similarity has a non-zero value, and as the concepts are from the same ontology, the context similarity is defaulted to 1. With the help of the negative property similarity the overall concept similarity is also reduced but the context similarity of 1 gives an advantage to this match as concepts are from same ontology. This avoids too much reduction in the final match score.

Case 4. If the candidate concept is from the same ontology but not related to the requirement concept, then it is treated in the same way as concepts from two different ontologies. Here *FundQuote* gives a context match of -1 as it is modeled as a sibling of

StockQuote. The Coverage similarity is 0 as *StockQuote* and *FundQuote* do not have a subsumption relationship between them and the property similarity is also reduced by using the penalty for unmatched properties. This gives an overall match score below zero, making it easier to discard this match. In contrast, without using the coverage and context similarities and the property penalty, decent match score is obtained because the property similarity without penalty is above 0.5.



	Case 1 StockOnt StockQuote	Case 2 StockOnt StockDetailQuote	Case 3 StockOnt StockQuickQuote	Case 4 StockOnt FundQuote	Case 5 xIgniteStocks StockQuote	Case 6 xIgniteStocks ExtendedQuote	Case 7 xIgniteStocks QuickQuote	Case 8 xIgniteStocks FundQuote
▨ Total Similarity with only Syntactic & Property Match	1.00	0.69	0.77	0.62	1.00	0.50	0.57	0.62
▨ Property Similarity without Penalty	1.00	1.00	-0.24	0.10	0.72	0.72	-0.13	0.09
▨ Syntactic Similarity	1.00	1.00	0.31	0.55	0.72	0.72	0.42	0.44
▨ Context Similarity MWSDI	1.00	1.00	1.00	-1.00	1.00	0.85	-1.00	-1.00
▨ Coverage Similarity MWSDI	1.00	0.95	0.90	0.00	1.00	0.95	0.90	0.00
▨ Property Similarity with penalty - MWSDI	1.00	0.93	0.44	-0.04	0.89	0.75	0.04	-0.04
▨ Total Similarity MWSDI	1.00	0.94	0.40	0.57	0.77	0.67	0.45	0.48

▨ Total Similarity with only Syntactic & Property Match
▨ Property Similarity without Penalty
▨ Syntactic Similarity
▨ Context Similarity MWSDI
▨ Coverage Similarity MWSDI
▨ Property Similarity with penalty - MWSDI
▨ Total Similarity MWSDI

Figure 4. Concept Matching

Candidate Concepts from different Ontology

Case 5. A different ontology can model the same *StockQuote* concept with a bit of variety. The same is the case for the *StockQuote* concept from *xIgniteStocks* ontology. It can be seen that all the properties are matched as the property similarity values with and without penalty are the same. Since, both have the same name, the syntactic similarity is 1. As all the properties for both the concepts match each other with a value of above 0.7, and the context and the coverage similarit ies are also 1, the overall match score is boosted.

Case 6. The next case is when the candidate concept from a different ontology matches better with a sub-concept of the requirement. In this example, *ExtendedQuote* from *xIgniteStocks* matches better with *StockDetailQuote* of *StockOnt*. As *StockDetailQuote* is the immediate child of the requirement concept, the coverage match has a value of 0.95. Here, all the properties of the requirement are satisfied and a good context similarity of 0.85 is obtained which boosts the overall match score.

Case 7. The next case is when a concept from different ontology matches better with a super concept of the requirement concept. Here in case of *QuickQuote* as it matches to *StockQuickQuote* which is an *immediate* super concept of the requirement, the coverage similarity is reduced by 0.1. *QuickQuote* is modeled as the sibling of StockQuote in its ontology and the requirement concept matches better with that *StockQuote* concept from *xIgniteStocks*, a value of -1 is assigned for context similarity. In addition to this the penalty for unmatched properties is also applied. This, results in a very low overall match score for *StockQuote*-*QuickQuote* pair. It can be noted here that by considering only syntactic and property similarity gives a good match score for this pair.

Case 8. The last case is when an unrelated concept from a different ontology is matched. In this example, the *FundQuote* concept which has a decent property similarity without penalty is considered. If only syntactic and properties similarities are considered then this concept gives a match score near to 0.5 with the requirement concept. With human perception however, it is evident that this match should be discarded as it is surely not the requirement. The algorithm described in this thesis excels here. The property penalty helps here to reduce the property similarity to a low value. In addition since this concept matches better (MS > 0.8) with the *FundQuote* concept from the ST ontology, which happens to be a sibling of the requirement, a score of -1 can be assigned to context similarity. The coverage match is 0 as both the concepts do not have any indirect subsumption relationship. Overall, a negative score for this pair makes it possible to discard this match.

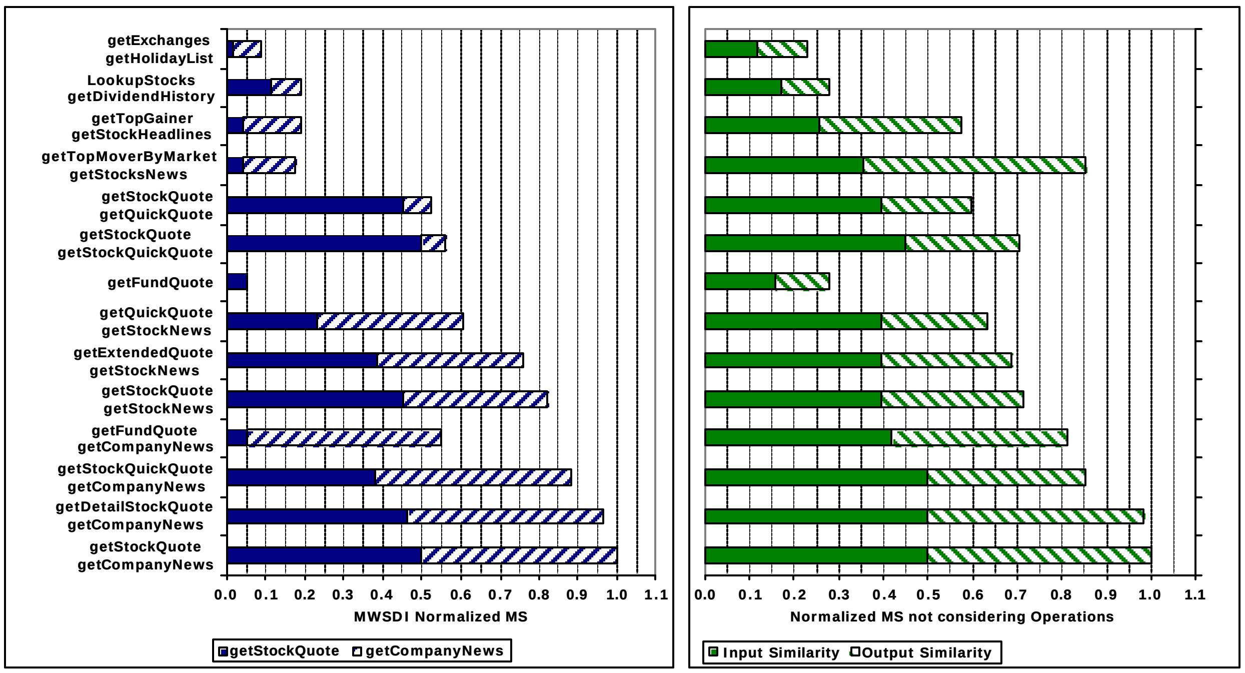


Figure 5. Graph 1 & Graph 2 - Functional Similarity and IO Similarity

Figure 5 illustrate the advantages of matching services based on their functionality instead of only inputs and outputs. Graph 1 shows the functional similarity values for 14 Web services calculated using the discovery algorithm detailed in chapter 3. Each of these 14 Candidate Services depicts a distinct case of how operations may be modeled in a Candidate

Service with respect to the ST described in chapter 2. The horizontal bars shows the decomposition of the overall functional similarity into two parts, each representing the similarity for the individual operations of ST. Graph 2 describes the similarity values calculated using the inputs and outputs of these operations together i.e. all the inputs of ST matched with all the inputs of CS and all the outputs of ST to all the outputs of CS. This graph also shows the decomposition of each bar into input and output similarities.

The first eight cases (from the bottom) in these graphs depict the services based on the eight cases described in Figure 4. In Candidate Services 1 to 4 the second operation is the same as the second operation of ST. In Candidate Services 5 to 8 the second operation is almost similar to the second operation of ST. One can see that for services 4 and 8, the similarity values from Graph 1 are very low as compared to Graph 2, thus avoiding false matches. The algorithm described in this thesis also boosts the values for Candidate Services 6 and 7 as shown in Graph 1. Candidate Services 9 and 10 contain one operation matching very well to the first ST operation and a second operation entirely different from the second ST operation. Matching based on functional similarity can detect this difference, whereas similarity based on inputs and outputs (Graph 2) fails to detect this. Candidate Service 11 has two operations, the first one takes the name of a Stock exchange and returns the top mover's stock quote of the day in that exchange, and the second operation returns the news about stocks based on the StockSymbol. This service is annotated with the same ontology as ST. One can see that even if the inputs and outputs match individually, they don't when operations are considered. Here, not considering operations will give a false match as in Graph 2, but the algorithm described in this thesis avoids this false match. Candidate Service 12 is the counterpart of Candidate Service 11, but from a different ontology. This algorithm avoids this false match too. Candidate Services 13 and 14 have both the operations entirely different from the requirement. One can see that for the algorithm described in this thesis the match scores are lowered even further because it uses functional concepts to identify what the operations are.

The context and coverage similarities are able to boost the match scores for good matches from different ontologies by almost 15-20 %. The bad matches where some or all operations are different from the requirement are also tempered down by 10-20 %.

Using a threshold of 0.7, one can see that the Graph 1 returned 5 matches and the algorithm considering just inputs and outputs returned 7 results. Of the seven, two cases (cases 4 and 11) lead to false matches, which are eliminated in by this approach. The only input and output based algorithm also leaves out a good match i.e. case 6 and returns a bad match i.e. case 9. The bad match here means a partial match where all the operations of request are not satisfied. The algorithm detailed in previous chapters is able to eliminate both these problems. One of the false matches arises because the property names of concepts they are annotated with are same e.g. StockQuote and FundQuote outputs for getStockQuote and getFundQuote operations. The other false match scenario is a result of neglecting the input / output combination in a specific operation. When a threshold of 0.55 was used the algorithm considering just inputs and outputs returned a set of 11 services out of which three cases i.e. 4, 11 and 12 were false matches and two cases, case 9 and case 10 are bad matches. With the same threshold value the thesis approach returns 8 matches out of which only one case i.e. case 4 is a false match. It also reduces the number of bad matches to 1.

Thus at different threshold values both the algorithms return different number of matches, but in all the cases the algorithm described in this thesis worked better in terms of eliminating false matches.

CHAPTER 7

RELATED WORK

This chapter discusses some existing approaches for discovering Web services. As the approach described in this thesis involves matching two ontological concepts, some prominent schema matching and ontology matching efforts are also discussed here.

According to current standards all the Web services are advertised in a central UDDI registry which provides categorized browsing and a keyword based search technique to find Web services. Many researchers however find this approach to be unscalable as it involves replication of all the advertised Web services [Thaden et al., 2003]. Many peer-to peer solutions have been proposed to solve the problem of scalability [Schmidt and Parashkar, 2003; Paolucci et.al, 2003; Maedche and Staab, 2003]. Previous work from the LSDIS lab more specifically the MWSDI system [Verma et.al, 2004] also deals with this issue.

While distributed registries are instrumental in facilitating efficient service discovery, it is equally important to correctly match the users request to the advertisements once a registry, either centralized or distributed, is selected. A lot of papers discuss the ontology based approach for service matching and discovery. [Klein and Bernstein, 2001] use a process taxonomy based approach for discovery. The matching algorithm uses the semantic relationships encoded in the process ontology to match the service process model against queries. [Gonzales et.al, 2001; Paolucci et.al, 2002] use subsumption relationships to find matches between two concepts. The former uses a DL reasoner whereas the latter determines the degree of similarity using the vertical distance between concepts. Both these approaches are based on DAML+OIL descriptions and are limited to services mapped with a single ontology. [Trastour et.al., 2001] convert services to RDF graphs and two nodes match if all their sub-nodes match. This approach is also limited for a single ontology based service description. [Sycara et.al., 1999] describes the matchmaking process using an agent capability description language, called LARKS (Language for Advertisement and Request for Knowledge Sharing). The matching engine of the matchmaker agent contains five different filters: Context matching, Profile comparison, Similarity matching, Signature matching, and Constraint matching. Although this approach uses a context filter, it differs from the approach described in this thesis in terms of what it means by the context. In LARKS, the context is defined by words that are used to define the overall domain of the service, whereas this algorithm described in this thesis uses context information for every concept. In addition LARKS assumes that even if there are different domain ontologies, they are derived from a common basic vocabulary. Another approach by [Cardoso and Sheth, 2003] specifies an algorithm for matching services annotated with different ontologies. However, this approach also uses DAML+OIL and is plagued with problem of false matching. The Semantic Web Service Architecture [SWSA, 2004] also recognizes the fact that multiple ontologies conceptualizing the same real world knowledge can exist and that Web services can subscribe to such different and only partially comparable ontologies. They recognize the need to capture the user's goal and service capabilities in a way that can facilitate matchmaking. They also propose the need to compare the Web service requirements and advertised descriptions when they are both represented with different ontologies by using "translation, mediation, or matching integrated with ontology mapping rules" [SWSA, 2004]. [Paolucci et.al, 2002a] talk about using UDDI with OWL-S. They propose to import semantics from OWL-S service description into UDDI which can be used while searching for services. A parallel effort by the WSMO group is the implementation of the WSMF [Fensel and Bussler, 2002] framework. It concentrates more on how Web services should be described in order to implement a better service discovery approach.

Matching these service descriptions involves matching the ontological concepts that they are annotated with. The area of ontology matching is a very popular research area in the semantic web since most semantic web applications involve the matching of two concepts in one way or the other. While matching concepts it is important to consider the conflicts that can arise between different ontologies. [Klein, 2001] gives a classification of mismatches between ontologies based on representation languages and intent of conceptualization. Past research in ontology matching includes statistical techniques [Sekine et.al, 1999], linguistic algorithms [Magnini et.al, 2002] that use NLP techniques to find equivalence between labels or names of the concepts, and machine learning approaches like LSD and GLUE [Doan et.al, 2003; Doan et.al, 2004] which match concepts based on probabilistic distribution of instances of the concepts. [Maedche and Staab, 2002] proposes a two level approach to find mappings between ontologies. They use name equivalence and hierarchical equivalence together in order to find matches between two concepts. Since a comprehensive survey is out of the scope of this thesis, one can refer [Kalfoglou and Schorlemmer, 2003] for a complete discussion of ontology matching approaches.

As ontologies are natural extensions of schemas, ontology mapping techniques borrow heavily from schema matching techniques. Extensive research has been done in the area of schema matching and integration. Schema matching approaches vary from matching structure of schemas to linguistic matching of the individual elements in a schema. Most of the schema matching approaches like COMA [Do and Rahm, 2002], Cupid [Madhavan et.al, 2001], and

Similarity Flooding [Melnik et.al, 2002], however use a combination of these techniques. Machine learning techniques like [Berlin and Motro, 2001; Doan et.al, 2001; Doan et.al, 2003] use a training set to deduce matching rules which are used to guess new matches. A comprehensive survey of schema matching techniques can be found in [Do et.al, 2002].

CHAPTER 8

CONCLUSION AND FUTURE WORK

It is unlikely that real-world Web-scale semantic information systems will be based on a single ontology, but there have been few studies to date to develop techniques supporting multiple ontologies. The few multi-ontology systems that have been talked about before [Mena et.al, 1996] have been in the context of data/information systems and in the areas of ontology mapping/alignment/merging [Kalfoglou and Schorlemmer, 2003]. In the context of services only single ontology environments have been considered [Paolucci et.al, 2002]. [Cardoso and Sheth, 2003] were the first to present preliminary work on service discovery in a multi-ontology environment. This thesis is an extension of that work and differs in the following ways:

1. [Cardoso and Sheth, 2003] algorithm considered DAML-S as the the service description language. Although the algorithm was not tied to any specific implementation it depends on an ontology based service description language. The algorithm described in this thesis uses a top down approach where the service description language used is the existing WSDL standard but annotated with ontologies. The approach in this thesis assumes that we are provided with these annotated service descriptions and focuses on the matching algorithm.
2. In [Cardoso and Sheth, 2003] the similarity calculation was based on only the inputs and outputs of the web service of all operations considered together. The algorithm described

in this thesis does not match the inputs and outputs directly but matches them based on the operations.

3. The property similarity calculation in [Cardoso and Sheth, 2003] did not consider a penalty for unmatched properties. It also did not consider cardinality information while matching properties. This thesis adds property penalty for unmatched properties and also considers the cardinality information while matching properties.
4. [Cardoso and Sheth, 2003] does not consider context, while context is considered in this thesis.
5. There is no explicit consideration for coverage in [Cardoso and Sheth, 2003]. The algorithm described in this thesis considers context explicitly even for concepts from different ontologies.
6. [Cardoso and Sheth, 2003] considers QoS (Quality of Service) similarity. This thesis does not consider QoS similarity.

As mentioned earlier previous works in the area of service discovery have considered only single ontologies. This section highlights some of the similarities and differences as compared to a multi-ontology approach.

Similarity Measures

In a single ontology environment, syntactic and property similarity can give enough information about a concept to give good matches. However, in a multi-ontology environment, concepts can be modeled with different levels of abstraction. Hence considering only syntactic and property information is not sufficient since it does not provide enough information about the concept. Measures used in this approach like Context and Coverage similarity provide this extra information.

Linguistic Issues

In a single ontology environment, matching properties and concepts based on names is enough since properties are inherited for the related concepts. In a multi-ontology environment, names of properties and concepts can be synonyms, hypernyms, hyponyms, homonyms of each other. Hence matching them syntactically can return bad match scores. This approach uses WordNet, custom abbreviation dictionary etc. to tackle this problem

Model level issues

In a single ontology environment, a single ontology model does not pose any structure and model level issues. In a multi-ontology environment, ontologies can have different modeling techniques which need to be tackled before matching two concepts e.g. XML Schema models Collection concepts as complexTypes or simpleTypes where as OWL models them as collection classes. A common representation format helps to bridge this gap.

This thesis described an algorithm for semantic Web service discovery in a multi-ontology environment. The investigations presented in this thesis may also be applicable to other situations involving multiple ontologies, such as multi-ontology query processing. Empirical testing and preliminary evaluation of the discovery algorithm showed that considering the context and coverage of the annotated ontological concept helped to better understand the user's service requirement and thus led to better quality matches. The testing in this thesis was hindered due to lack of annotated Web services and ontologies. However the experiments contained real world Web services. In the future, when there are a lot of domain ontologies and different web Services subscribe to different ontologies, this approach will be very valuable.

The service discovery algorithm can be extended to incorporate the following:

- Support for fault matching.

- QoS measures.
- Better heuristic measures for similarity calculations (multiplicative vs additive)

The algorithm should also be tested on a larger set of real world data by building a larger testbed of annotated services.

REFERENCES

[Aggarwal et.al, 2004] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint Driven Web Service Composition in METEOR-S", Proceedings of IEEE International Conference on Services Computing, 2004

[Angell et.al, 1983] R. Angell, G. Freund, et.al, "Automatic spelling correction using a trigram similarity measure", Information Processing and Management 19(4): 255-161, 1983

[Berlin and Motro, 2001] J. Berlin, and A. Motro, "Autoplex, Automated Discovery of Content for Virtual Databases", CoopIS 2001, 108–122

[Berners-Lee et.al, 2001] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web", Scientific American (2001)

[Box et.al, 2000] D. Box et.al, "Simple Object Access Protocol (SOAP) 1.1", May 2000. Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[Cardoso et.al, 2002] J. Cardoso, C. Bussler, A. Sheth and D. Fensel (2002), 'Semantic Web Services and Processes: Semantic Composition and Quality of Service", On the Move to Meaningful Internet Computing and Ubiquitous Computer, Irvine CA, October 2002

[Cardoso and Sheth, 2003] J. Cardoso and A. Sheth (2003), "Semantic e-Workflow Composition", Journal of Intelligent Information Systems (JIIS)

[Cardoso and Sheth, 2005] J. Cardoso and A. Sheth (2005), "Semantic Web Process: powering next generation of processes with Semantics and Web Services", Lecture Notes in Computer Science, Springer, 2005

[Chawathe et.al, 1995] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, “The TSIMMIS project: Integration of heterogeneous information sources”, In Proceedings of the Information Processing Society of Japan Conference, pages 7-18, Tokyo, Japan, October 1995

[Christensen et. al, 2001] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana, “Web Services Description Language (WSDL) 1.1”, W3C Note, March 2001, Available at http://www.w3.org/TR/wsdl

[Colgrave et.al, 2004] J. Colgrave, R. Akkiraju, R. Goodwin, “External Matching in UDDI”, ICWS 2004

[Curbera et.al, 2001] F. Curbera, W. Nagy, and S. Weerawarana, “Web services: Why and how”, In Workshop on Obejcet Orientation and Web Services OOWS2001, 2001

[Do et.al, 2002] H. Do, S. Melnik, and E. Rahm. “Comparison of schema matching evaluations.” In Proceedings of the 2nd Int. Workshop on Web Databases (German Informatics Society), 2002

[Do and Rahm, 2002] H. Do and E. Rahm, “COMA - A System for Flexible Combination of Schema Matching Approaches”, In Proceedings of the 28th International Conference on Very Large Databases (VLDB), 2002

[Doan et.al, 2001] A. H. Doan, P. Domingos, and A. Halevy, Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. SIGMOD 2001

[Doan et.al, 2003] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy, “Learning to Match Ontologies on the Semantic Web”, VLDB Journal, Special Issue on the Semantic Web, 2003

[Doan et.al, 2004] A. Doan, J. Madhavan and A. Halevy, “Ontology Matching: A Machine Learning Approach”, in S. Staab and R. Studer (eds.), Handbook on Ontologies in Information Systems (pp. 385-403), 2004, New York: Springer

[Fensel and Bussler, 2002] D. Fensel and C. Bussler, “The Web Service Modeling Framework WSMF”, Electronic Commerce: Research and Applications, 1 (2002) 113-137

[Gonzales et.al, 2001] J. Gonzales-Castillo, D. Trastour, and C. Bartolini, “Description logics for matchmaking of services”, In Proc. of Workshop on Application of Description Logics, 2001

[Gruber, 1993] T. Gruber, “A Translation Approach to Portable Ontology Specifications”, Knowledge Acquisition, 5(2), 199-220, (1993)

[JXTA] http://www.jxta.org

[Kalfoglou and Schorlemmer, 2003] Y. Kalfoglou and M. Schorlemmer, “Ontology mapping: the state of the art”, The Knowledge Engineering Review, Volume 18 Issue 1, January 2003

[Klein, 2001] M. Klein, “Combining and relating ontologies: an analysis of problems and solutions”, In A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, editors, Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA, Aug. 4--5, 2001

[Klein and Bernstein, 2001] M. Klein and A. Bernstein, “Searching for Services on the Semantic Web using Process Ontologies”, in The First Semantic Web Working Symposium (SWWS-1). 2001. Stanford, CA USA

[Madhavan et.al, 2001] J. Madhavan, P. Bernstein, and E. Rahm, “Generic Schema Matching with Cupid”. In Proceedings of the International Conference on Very Large Databases (VLDB), 2001

[Magnini et.al, 2002] B. Magnini, L. Serafini, and M. Speranza, "Linguistic Based Matching of Local Ontologies", in Working notes of MeaN-02 (Workshop held in conjunction with AAAI-2002) , Edmonton, Alberta, Canada, July 28 - August 1, 2002

[Maedche and Staab, 2002] A. Maedche and S. Staab, "Measuring similarity between ontologies", In Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW). Springer, 2002

[Maedche and Staab, 2003] A. Maedche, S. Staab, "Services on the Move - Towards P2P-Enabled Semantic Web Services", in Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki, Finland, 29th-31st January 2003

[Melnik at.al, 2002] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm", ICDE 2002

[Mena et.al, 1996] E. Mena, V. Kashyap, A. Sheth and A. Illarramendi, OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies, Conference on Cooperative Information Systems, Brussels, Belgium, IEEE Computer Society Press, (1996)

[Miller et.al, 1990] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller, "Wordnet: An on-line lexical database", International Journal of Lexicography, 3(4):235--312, 1990

[OWL-S, 2004] OWL-S- Semantic Markup for Web Services, available at http://www.w3.org/Submission/OWL-S/

[Pan and Horrocks, 2001] J. Pan and I. Horrocks, "Metamodeling architecture of web ontology languages", In Proc. of the 2001 Int. Semantic Web Working Symposium (SWWS 2001), pages 131-149. CEUR, 2001

[Paolucci et.al, 2002] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of Web Services capabilities", in Proc. of the 1st International Semantic Web Conference (ISWC), 2002

[Paolucci et.al, 2002a] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Importing the Semantic Web in UDDI", in Web Services, E-Business and Semantic Web Workshop, 2002

[Paolucci et.al, 2002b] M. Paolucci, K. Sycara, and T. Kawamura, "Delivering semantic web services", Tech. Rep. CMU-RI-TR-02-28, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (December 2002). 31

[Paolucci et.al, 2003] M. Paolucci, K. Sycara, T. Nishimura, and N. Srinivasan, "Using DAML-S for P2P Discovery," in Proceedings of the First International Conference on Web Services (ICWS'03), Las Vegas, Nevada, USA, June 2003, pp 203- 207

[Papakonstantinou, 1996] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman, "Medmaker: A Mediation System Based on Declarative Specifications", In Proc. ICDE Conference, 1996

[Patil et.al, 2004] A. Patil, S. Oundhakar, A. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework", in proceedings of the 13th international conference on World Wide Web, 2004

[Porter, 1980] M. Porter, "An algorithm for Suffix Stripping", Program – Automated Library and Information Systems, 14(3):130-137, 1980

[Rajasekaran et.al, 2004] P. Rajasekaran, J. Miller, K. Verma, A. Sheth, Enhancing Web Services Description and Discovery to Facilitate Composition, International Workshop on Semantic Web Services and Web Process Composition, 2004 (Proceedings of SWSWPC 2004)

[Salton, 1988] G. Salton, “Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer”, Massachusetts, Addison-Wesley, 1988

[Schmidt and Parashkar, 2003] C. Schmidt and M. Parashar, “A Peer-to-Peer Approach to Web Service Discovery”, World Wide Web, Internet and Web Information Systems, Kluwer Academic Publishers, August 2003.

[Sekine et.al, 1999] S. Sekine, K. Sudo, and T. Ogino, “Statistical Matching of Two Ontologies”, in the Proceedings of the SIGLEX99: Standardizing Lexical Resources 1999; Maryland USA

[Sheth, 1998] A. Sheth, “Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics”, in Interoperating Geographic Information Systems. M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.), Kluwer, Academic Publishers, 1998, pp. 5-30

[Sivashanmugam et.al, 2003] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller, “Adding Semantics to Web Services Standards”, in proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada (June 2003) pp. 395-401

[Sivashanmugam et al., 2004a] K. Sivashanmugam, J. Miller , A. Sheth and K. Verma, “METEOR-S Web Service Composition Framework”, International Journal of E-commerce 9(2), pp. 71-106

[Sivashanmugam et al., 2004b] K. Sivashanmugam, K. Verma, and A. Sheth, “Discovery of Web Services in a Federated Registry Environment”, Proceedings of IEEE Second International Conference on Web Services, June, 2004, pp. 270-278

[Sivashanmugam et.al, 2004c] K. Sivashanmugam, J. Miller, A. Sheth, and K. Verma, Framework for Semantic Web Process Composition, Semantic Web Services and Their Role in

Enterprise Application Integration and E-Commerce, Special Issue of the International Journal of Electronic Commerce (IJEC), Eds: Christoph Bussler, Dieter Fensel, Norman Sadeh, Feb 2004

[Stal, 2002] M. Stal, "Web services: beyond component-based computing", Communications of the ACM, 45(10):71-76, 2002

[SWSA, 2004] Semantic Web Services Initiative requirement document available at http://www.daml.org/services/swsa/swsa-requirements.html

[Sycara et.al., 1999] K. Sycara, J. Lu, M. Klusch, and S. Widom, "Dynamic Service Matchmaking among Agents in Open Information Environments", Journal ACM SIGMOD Record, Special Issue on Semantic Interoperability in Global Information Systems, Ouksel, A., Sheth, A. (ed.) (1999)

[Thaden et al, 2003] U. Thaden, W. Siberski, and W. Nejdl, "A Semantic Web based Peer-to-Peer Service Registry Network", Technical Report, Learning Lab Lower Saxony.

[Trastour et.al., 2001] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo, "A Semantic Web Approach to Service Description for Matchmaking of Services", Proc. 1st Semantic Web Working Symposium, CA, (2001)

[UDDI, 2002] UDDI "The Evolution of UDDI", UDDI.org White Paper, Available at http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf

[UDDI, 2003] UDDI Version 3.0.1, Available at http://uddi.org/pubs/uddi-v3.0.1-20031014.htm

[Uschold and Gruninger, 1996] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods and Applications", The Knowledge Engineering Review, 1996

[Verma et.al, 2004] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar and J. Miller, "METEOR–S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services”, Journal of Information Technology and Management (to appear, 2004)

[Visser et.al, 1999] P. Visser, D. Jones, M. Beer, T. Bench-Capon, B. Diaz, and M. Shave, “Resolving ontological heterogeneity in the KRAFT project”, In 10th International Conference and Workshop on Database and Expert Systems Applications DEXA'99. University of Florence, Italy, August 1999

[WSDL-S, 2004] WSDL-S: A Proposal to W3C WSDL 2.0 Committee, “Adding Semantics to WSDL”, Available at http://lsdis.cs.uga.edu/projects/WSDL-S/wsdl-s.pdf

[Zamora E., 1981] E. Zamora, J. Pollock, et al. "The Use of Trigram Analysis for Spelling Error Detection", Information Processing and Management 17(6): 305-316, 1981

APPENDIX A – ONTOLOGIES USED

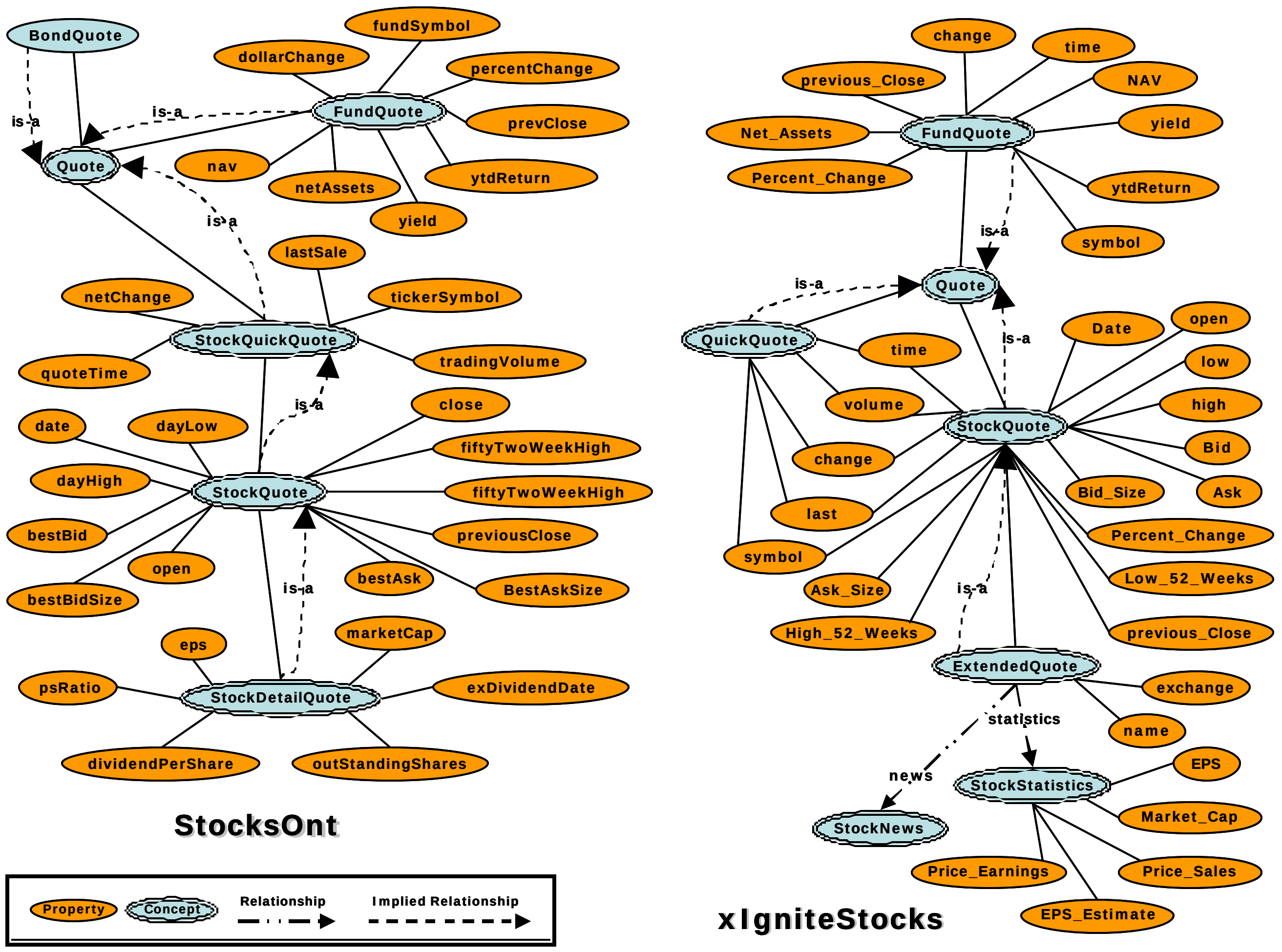


Figure 6. Graphical view of Ontologies used (Partial)

APPENDIX B – GLOSSARY OF ACRONYMS

AI:	Artificial Intelligence
HTTP:	Hyper Text Markup Language
LSDIS:	Large Scale Distributed Information Systems
METEOR-S:	Managing End-To-End OpeRations: for Web Services
OWL:	Web Ontology Language
OWL-S:	OWL-based Web Service ontology
QoS:	Quality of Service
SOA:	Service Oriented Architecture
SOAP:	Simple Object Access Protocol
ST:	Search Template
SWRL:	Semantic Web Rule Language
SWSA:	Semantic Web Services Initiative Architecture
SWSL:	Semantic Web Services Language Committee
UDDI:	Universal Description, Discovery and Integration
W3C:	World Wide Web Committee
WSDL:	Web Service Description Language

WSMF:	Web Service Modeling Framework
WSMO:	Web Service Modeling Ontology
XML:	Extensible Markup Language
XSD:	XML Schema Definition

APPENDIX C – GLOSSARY OF CONCEPTS

Service:	A service in the context of Web services is an application or software system that can be identified with a Uniform Resource Locator (URI) and uses XML for send and receive messages. OR In WSDL, a collection of ports define a service. A port is associated with a network address and a binding, where binding specifies the protocol and data specifications for a specific port type and port type specifies the set of operations supported by the service.
Protocol:	A protocol is an agreed-upon formal specification of rules and message formats that two entities should follow for communication or message exchange.
Syntax:	The structural or grammatical rules that define how symbols in a language are to be combined to form words, phrases, expressions, and other allowable constructs. (http://www.fda.gov/ora/inspect_ref/igs/gloss.html)
Semantics:	It refers to specification of meanings. It can also be defined as an agreed upon meaning of messages and other vocabulary.
Ontology:	"An ontology is a specification of a conceptualization." Tom Gruber. An Ontology is a representation of knowledge in a particular domain.

Logic:	The branch of mathematics that investigates the relationships between premises and conclusions of arguments. (http://wotug.ukc.ac.uk/parallel/acronyms/hpccgloss/all.html)
Description Logic:	Description logics belong to a family of Knowledge Representation languages with formal semantics based on First-Order Logics. Description Logics use inferencing to discover implicit knowledge.