

國立交通大學

資訊管理研究所

碩士論文

以知識本體建構之網路服務模糊搜尋

Ontology-based Fuzzy Matchmaking for Web Services



研究生：譚道馨

指導教授：羅濟群 博士

趙國銘 博士

中華民國 94 年 6 月

以知識本體建構之網路服務模糊搜尋
Ontology-based Fuzzy Matchmaking for Web Services

研究生：譚道馨

Student: Tao-Hsin Tan

指導教授：羅濟群

Advisor: Chi-Chun Lo

趙國銘

Kuo-Ming Chao

國立交通大學

資訊管理研究所

碩士論文



Submitted to Institute of Information Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Business Administration

in

Information Management

June 2005

Hsinchu, Taiwan, the Republic of China

中華民國 九十四年六月

Ontology-based Fuzzy Matchmaking for Web Services

Student: Tao-Hsin Tan

Advisor: Dr. Chi-Chun Lo

Dr. Kuo-Ming Chao

Institute of Information Management

National Chiao Tung University

Abstract

To fully realize the potential of Web services it is necessary to develop viable dynamic discovery and composition techniques. Matchmaking is considered as one of the crucial factors to ensure dynamic discovery and composition of Web services. Current matchmaking methods such as UDDI or LARKS are inadequate given their inability to abstract and classify Web services on the capability of services, software signatures, and so on. Current mechanisms have lost a hidden dimension for matchmaking. That is, the underlying data of Web services. Otherwise, UDDI doesn't support the imprecise query. Therefore, our research proposes a novel framework which exploits fuzzy logic in order to abstract and classify the underlying data of Web services as fuzzy terms and rules. The aim is to allow vague terms in the search query and to provide more suited services to requesters.

Keywords: Web service, Fuzzy Logic, Matchmaking.

~ Acknowledgement ~

在論文即將付梓之際，歡欣之餘，卻也略帶惆悵。因為這也意味著碩士生涯的結束。回想兩年的研究生涯，首先要非常感謝羅濟群教授和趙國銘教授的指導和教誨，不僅在論文方向及架構上，給予寶貴的意見與建議，在研究的過程中也不時地從討論中獲得精闢的想法及有用的資源。而在生活上，每每跟羅教授出遊，接觸大自然，也從中得到不少人生的啟發。在此，由衷感激兩位教授的栽培與教導。

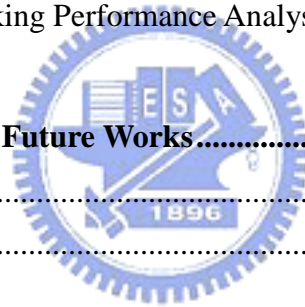
除了研究過程中，羅教授和趙教授的指導外，還要感謝論文口試委員，台大陳文賢教授及本所的陳瑞順教授於口試過程中提出的疑問及寶貴意見，讓自己能有機會修正論文的疏漏及啟發其他思維，論文整體也更趨完整。

而在新竹交大求學的兩年時間，雖覺得好像一眨眼就過去了，但其中的點點滴滴卻也無法一時說盡。每天窩在實驗室研究的日子不再，竟有點不習慣。首先感謝俊龍學長和謂立學長，在論文研究過程中所進行的一次次討論，幫助我釐清許多方向和疑問，尤其是俊龍學長，總能一針見血地提出論文不足的地方，讓我及早改進。論文能夠順利完成，真的非常感謝俊龍學長的從旁指導。當然也感謝同學們，一濤、建智、立群、緒杰及室友惠綾及玉棉，大家的互相打氣也適時地讓快被論文搞瘋的自己恢復動力，繼續努力。而最感謝的是新竹幫的陪伴，呆若、阿粉、小江、寇特妮及上逸，真的很幸運碰到你們，在初來乍到的新竹，偶爾的聚餐聊天、登山出遊，讓在異鄉的自己不再孤獨，你們對我的支持與關心，讓我能堅定地往前走。雖然我們都會遇到許多不同的轉變，但我相信我們可以彼此陪伴，一起面對。謝謝大家!!

~ Outline ~

Abstract	i
Acknowledgement	ii
Outline	iii
Figure List	v
Table List	vi
Chapter 1 Introduction	1
1.1 Introduction.....	1
1.2 Motivation.....	2
1.3 Objective	2
1.4 Thesis Outline	3
Chapter 2 Related Works	4
2.1 Web Service	4
2.1.1 SOAP	5
2.1.2 WSDL	6
2.1.3 UDDI.....	6
2.2 Semantic Web.....	7
2.2.1 Semantic Web.....	7
2.2.2 Ontology	14
2.2.3 OWL	15
2.2.4 OWL-S	15
2.3 Current Matchmaking for Web Service	17
2.3.1 LARKS	17
2.3.2 CBR.....	19
2.3.3 UDDI.....	19
2.4 Fuzzy Set Theory	20
2.4.1 Membership Functions.....	20
2.4.2 Basic Operations	23
2.4.3 Hedges.....	24
2.4.4 PRUF Translation Rules	25

Chapter 3	An Ontology-based Fuzzy Matchmaking Framework.....	27
3.1	Framework	27
3.2	Importing OWL-S in UDDI.....	28
3.3	Web Service Data Representation.....	32
3.4	Fuzzy Classifier	34
3.5	Fuzzy Matchmaking.....	35
3.6	The Constraints for Our Framework.....	36
Chapter 4	Implementation_An Example for Airline Ticket Reservation Service... 37	37
4.1	Implementation Environment	37
4.2	Airline Ticket Reservation Service	40
4.2.1	Fuzzy Terms	41
4.2.2	Building OWL Definition	43
4.2.3	Classifying Web Service Data.....	44
4.2.4	Fuzzy Matchmaking.....	45
4.2.5	Matchmaking Performance Analysis	46
Chapter 5	Conclusion and Future Works.....	50
5.1	Conclusion	50
5.2	Future Works.....	50
Reference.....		51



~ **Figure List** ~

Figure 2-1: Service Oriented Architecture	5
Figure 2-2: Data Model.....	7
Figure 2-3: Original Web Proposal to CERN	8
Figure 2-4: The Smart Data Continuum	9
Figure 2-5: RDF Triple	11
Figure 2-6: Semantic Web Stack.....	12
Figure 2-7: The Components of OWL-S	16
Figure 2-8: Trapezoidal Membership Function	21
Figure 2-9: S Membership Function	22
Figure 2-10: Z Membership Function.....	22
Figure 2-11: Gaussian Membership Function.....	23
Figure 2-12: Hedge	25
Figure 3-1: Ontology-based Fuzzy Matchmaking Framework.....	27
Figure 3-2: TModel Data Structure.....	29
Figure 3-3: TModel Data Structure.....	30
Figure 3-4: UDDI Service Representation.....	31
Figure 3-5: Membership Function for “reasonable” related with Price.....	32
Figure 3-6: Fuzzy Classifier.....	34
Figure 3-7: Fuzzy Matchmaking.....	35
Figure 4-1: The QualityRating of OWL-S Profile.....	40
Figure 4-2: Fuzzy Terms for Price	41
Figure 4-3: “comfortable” for Seat Space.....	42
Figure 4-4: “comfortable” for Air Time	42
Figure 4-5: OWL Definition	43
Figure 4-6: OWL Query in OWLJessKB	44

~ Table List ~

Table 2-1: RDF Classes.....	12
Table 2-2: RDF Properties	13
Table 4-1: The Development Environment.....	37
Table 4-2: Eva Airline’s Database.....	40
Table 4-3: The Classified Result.....	45
Table 4-4: The Result for “cheap airline” Query	46
Table 4-5: The Result for “very cheap airline” Query	46
Table 4-6: The Result for “comfortable airline” Query	46
Table 4-7: The Result for “most tickets are cheap” Query	46
Table 4-8: The Result for “cheap and comfortable airline” Query	46
Table 4-9: The Imprecision Rate for “cheap airline” Query	47
Table 4-10: The Imprecision Rate for “comfortable airline” Query	48
Table 4-11: The Imprecision Rate for “most tickets are cheap” Query	48
Table 4-12: The Imprecision Rate for “cheap and comfortable airline” Query.....	48



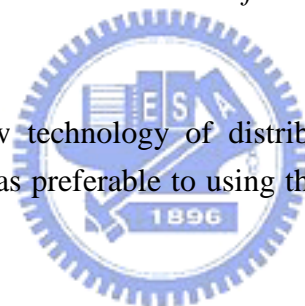
Chapter 1 Introduction

1.1 Introduction

"Web services are Internet-based modular applications that perform a specific business task and conform to a specific technical format. A Web service can be anything from a restaurant review service to a real-time travel advisory to an entire airline ticket reservation process. The modular technical format ensures these self-contained business services (from the same or different companies) will mix and match easily to create a complete business process. Businesses can dynamically publish, discover and aggregate a range of Web services via the Internet; in this way, they can more easily and dynamically create innovative products, business processes, and value chains. Web services can be delivered to any customer device (cell phone, PDA, computer, etc.) and can be created or transformed from existing applications."

--From IBM

Web service is the new technology of distributing computing concepts and another alternative promoted as preferable to using the distributed object middleware such as Java RMI or CORBA.



Using the HTTP protocol and XML, Web service is more user-friendly as compared with Java-RMI or CORBA, allowing people to develop distributed computing technology. The stateless hypertext HTTP permits to access the services without the problem like firewall that happens when people use RMI or CORBA. Without the complex architecture of CORBA, Web service is user-friendly.

The XML-formatted documents enable communication between computers across networks by exchanging and parsing XML documents. Technologies developed with HTTP and XML such as SOAP, WSDL and UDDI improve the prevailing use of Web service.

As the Web progresses, the concept of semantic web provides a common idea that allows knowledge to be shared and reused. Semantic web is the advancement of the current web that knowledge is represented by the given well-defined language and is understood by computers. It is helpful for Web service to use semantic web for different machines to cooperate with Web services automatically.

Web service is the emerging concept, and many enterprises or industries develop the application of Web service. The core developers, such as IBM, Microsoft, and W3C, aim to improve the protocol of Web service. Otherwise, many industries, like Amazon or Airline Service, build business application for Web service.

1.2 Motivation

Today, the number of Web services is increasing. How to discovery the required service effectively and compose the services correctly is critical issue for the use of Web services. Various technologies have been developed to improve the matchmaking of Web services. UDDI is one kind of the discovery mechanisms for Web services. It is a service directory including a list of services which can be registered by providers and can be searched by consumers. Although UDDI facilitates the discovery process, does not allow the vague search for match-making.

Another kind of matchmaking mechanism is using OWL-S, the ontology language that describes Web services with semantic. OWL-S provides non-functional and functional types of semantics for matchmaking. One matchmaking mechanism, LARKS, is to compare the advertisement with the requirement described by ontology and it pays more attention to the comparison of IOPE [6]. However, data or information provided by Web service is not taken into consideration for matchmaking in LARKS system, but it is an important factor for consumers to search the appropriate services.

1.3 Objective

For improving the matchmaking process in UDDI, we proposed the integrated matchmaking framework based on fuzzy logic and ontology. We want to abstract the data of Web services with fuzzy logic and interpret data of Web services with ontology. Hence, we use fuzzy logic for Web services to support the imprecise or vague terms in the query.

1.4 Thesis Outline

In this chapter, we have generally introduced the thesis background, our motivation, proposed solution and our objective. The remainder of the thesis is built as follows. In the next chapter, we provide background knowledge through the description of related technologies, such as the concept of fuzzy logic and semantic web, and the discussion about the current methods for Web service matchmaking. Chapter 3 explains the complete framework we proposed. Chapter 4 will illustrate how we implement the architecture for Airline service as an example and analysis the imprecision rate of searching in UDDI and in our proposed framework. Chapter 5 gives conclusion and the future works.



Chapter 2 Related Works

In chapter 2, we introduce Web service and the concept of semantic web for Web services. Further, we discuss the current mechanisms for matchmaking of Web services. In the long run, we indicate certain concepts of fuzzy set theory.

2.1 Web Service

In past, the way that we utilized functions only provided by the mono computing architecture is called “centralized computing.” With the progress of technology, people hope that they can use resource from other computing systems. This idea was implemented as one technology, RPC (Remote Procedure Call). It is adopted now in widespread use and becomes the main part of the development of system integration.

For RPC technology, the predecessors of Web service are CORBA、RMI、DCOM. Web service is different from its predecessors in that Web service is based on HTTP protocol and is utilized more easily. *“Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (especially WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards”* by [7]. Web service is based on “Service Oriented Architecture (SOA)” which is shown in Figure 2-1. There are three roles, service broker、service provider and service requestor, which are classified according to operation.

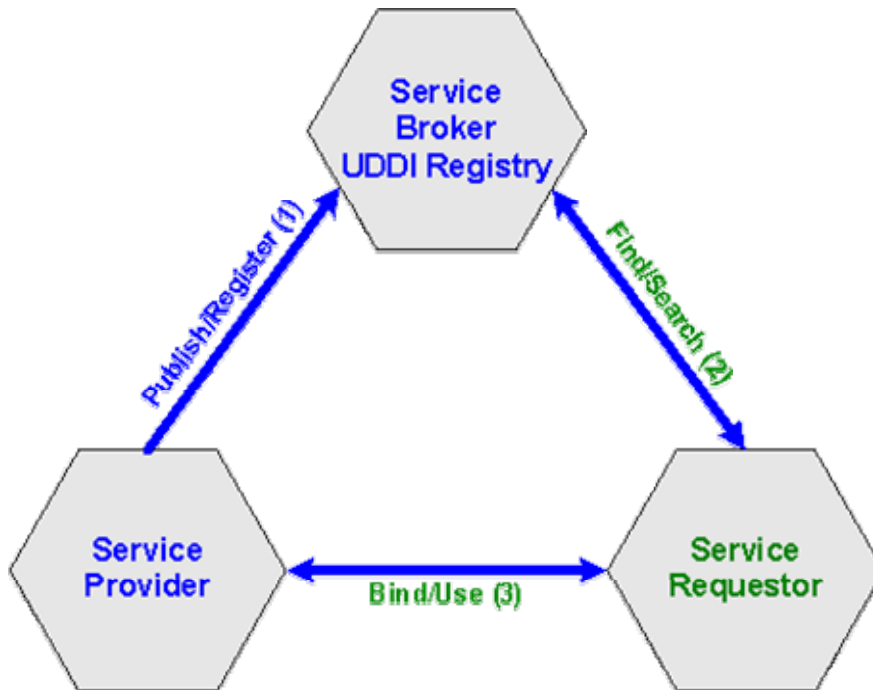


Figure 2-1 Service Oriented Architecture[22]

Service broker acts as the medium of Web services. It provides service providers the registry service to advertise their services. Service providers mainly build Web services and advertise their services with the detail description of Web services. Otherwise, it can deal with the requests from service requestors. Service requestors ask service broker for suited Web services with some conditions. There are many operations, like service finding 、 service publishing and service binding, which are accomplished by related technology standards, SOAP 、 WSDL and UDDI. We later introduce them in detail.

2.1.1 SOAP (Service Oriented Architecture Protocol)

SOAP was developed by Microsoft. It aims to overcome the problem of communication between heterogeneous platforms. To communicate with each other, SOAP uses XML to package and to exchange messages from local system and remote system. In the context of this architecture, SOAP also provides a convenient way for referencing capabilities [7]. With SOAP, applications can transmit messages in text-based way over the internet. SOAP messages can be carried by a variety of network protocols, such as HTTP 、 SMTP 、 FTP or RMI/IIOP, or a proprietary messaging protocol.

2.1.2 WSDL (Web service Description Language)

WSDL is also based on XML technology and is used to define how to describe the details of Web service. WSDL represents Web services with messages that are exchanged between service providers and service requestors. Those messages themselves are described abstractly and then bound to a concrete network protocol and message format [7]. WSDL service definitions provide documentation for distributed systems and are served as recipe for automating the details involved in machine communication [8].

2.1.3 UDDI (Universal Description, Discovery and Integration)

UDDI is a technical specification for finding, publishing, and integrating Web services. UDDI is like directory service, and also like the trader mechanism of CORBA. It provides two services, one is to publish Web services for service providers, and another is to search Web services for service requestors. UDDI is based on SOAP for communication and WSDL for Web service description to mediate the needs between providers and requestors. Figure 2-2 shows the data model of UDDI that is advertised by service providers. There are four data structures in UDDI. It includes basic business information in `businessEntity`, such as company name, contact information and business category. What kind of Web services provided by the specific company is described in `businessService`. `BusinessService` includes information such as service name, service key and service description. In UDDI, one business company could have many `businessServices`. Besides, one `businessService` contains a list of binding `Templates` that in turn contain `tModel`. `BindingTemplate` and `tModel` introduce the technical information about how to access and exploit Web services.

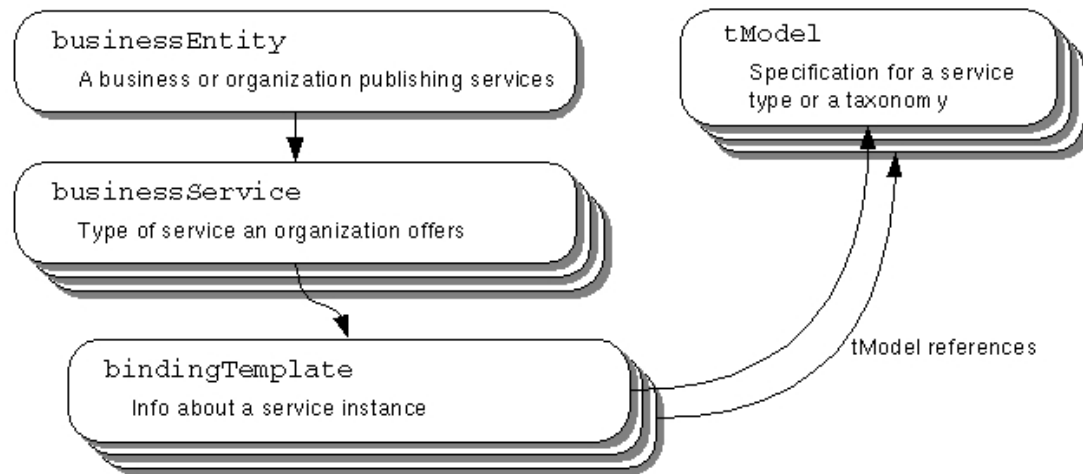


Figure 2-2 Data Model[26]

Business companies benefit from UDDI because its specifications provide interoperability between trading partners and drive companies to cost down. IBM and Microsoft make much effort to develop UDDI specifications to support more complicated business logic and to let UDDI as a public standard.



2.2 Semantic Web

2.2.1 Semantic Web

University of the World Wide Web is its vital property. The concept of hypertext link implements that “anything can link to anything”. The Web has been developed most rapidly as a medium of documents for people rather than for data. People can understand what the document means over internet, but machines can’t. To let data be processed by machines automatically, we must enable machines understanding what documents mean over internet. The Semantic Web aims to make up for this problem [28].

“The first step is putting data on the Web in a form that machines can naturally understand, or converting it to that form. This creates what I call a Semantic Web—a web of data that can be processed directly or indirectly by machines,” by Tim Berners-Lee. The creator of Web, Tim Berners-Lee, has a two-part vision for the future of the web. The first part is to make the Web a more collaborative medium. The second part is to make the Web understandable, and thus processable, for machines.

Figure 2-6 shows Tim Burners-Lee's diagram of his proposed vision to CERN (European Organization for Nuclear Research)[33].

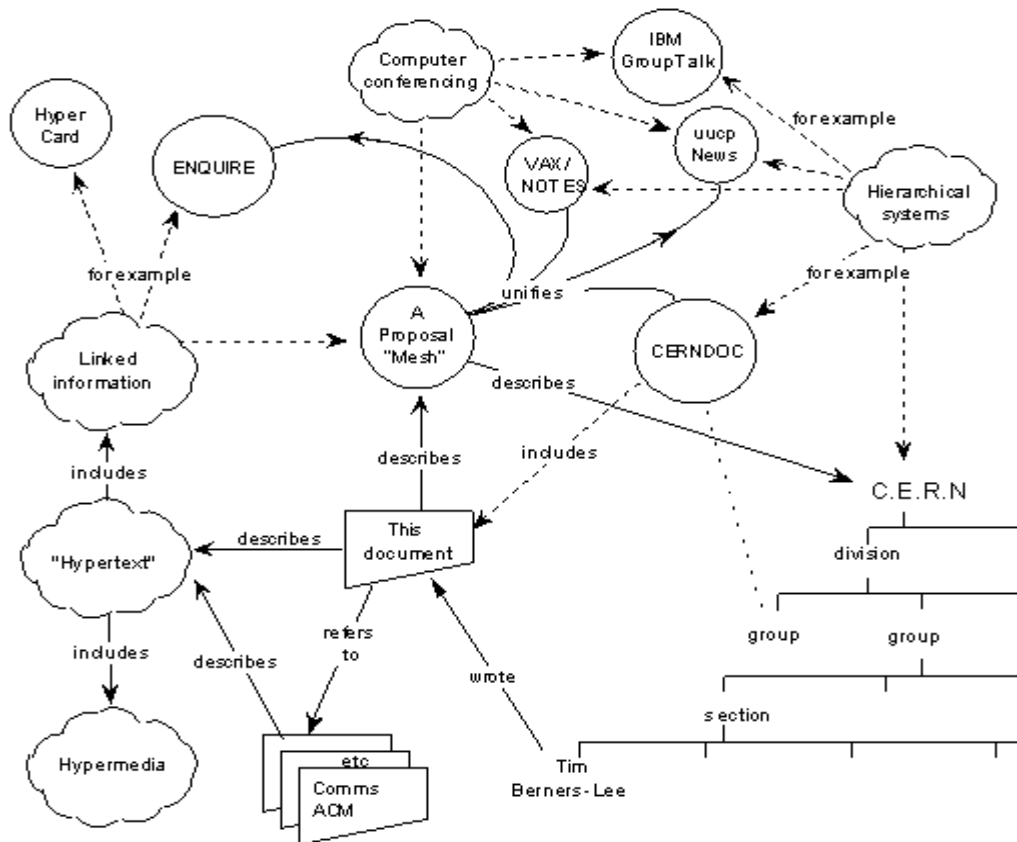


Figure 2-3 Original Web Proposal to CERN[33]

In Figure 2-3, we see the relations like “includes,” “describes,” and “wrote” between information items. Unfortunately, these relationships between resources are not currently defined on the Web. Tim Burners-Lee’s vision is to describe relationships between objects over the internet. Further, he proposed the concept of semantic web. The technology that captures such relationships is called the Resource Description Framework (RDF) described later.

For the consideration of data, usually, software is totally dependent on good data. The computing experts realize that data is important and must be verified and protected. Therefore, with the Web, Extensible Markup Language (XML), and now the emerging Semantic Web, the power shifts from applications to data. This also gives us the key to realize the Semantic Web. The path to machine-processed data is to make the data smarter. The objective of Semantic Web is to enable machines to understand what data means because data is made smarter. Figure 2-4 displays the progression of data along a continuum of increasing intelligence.

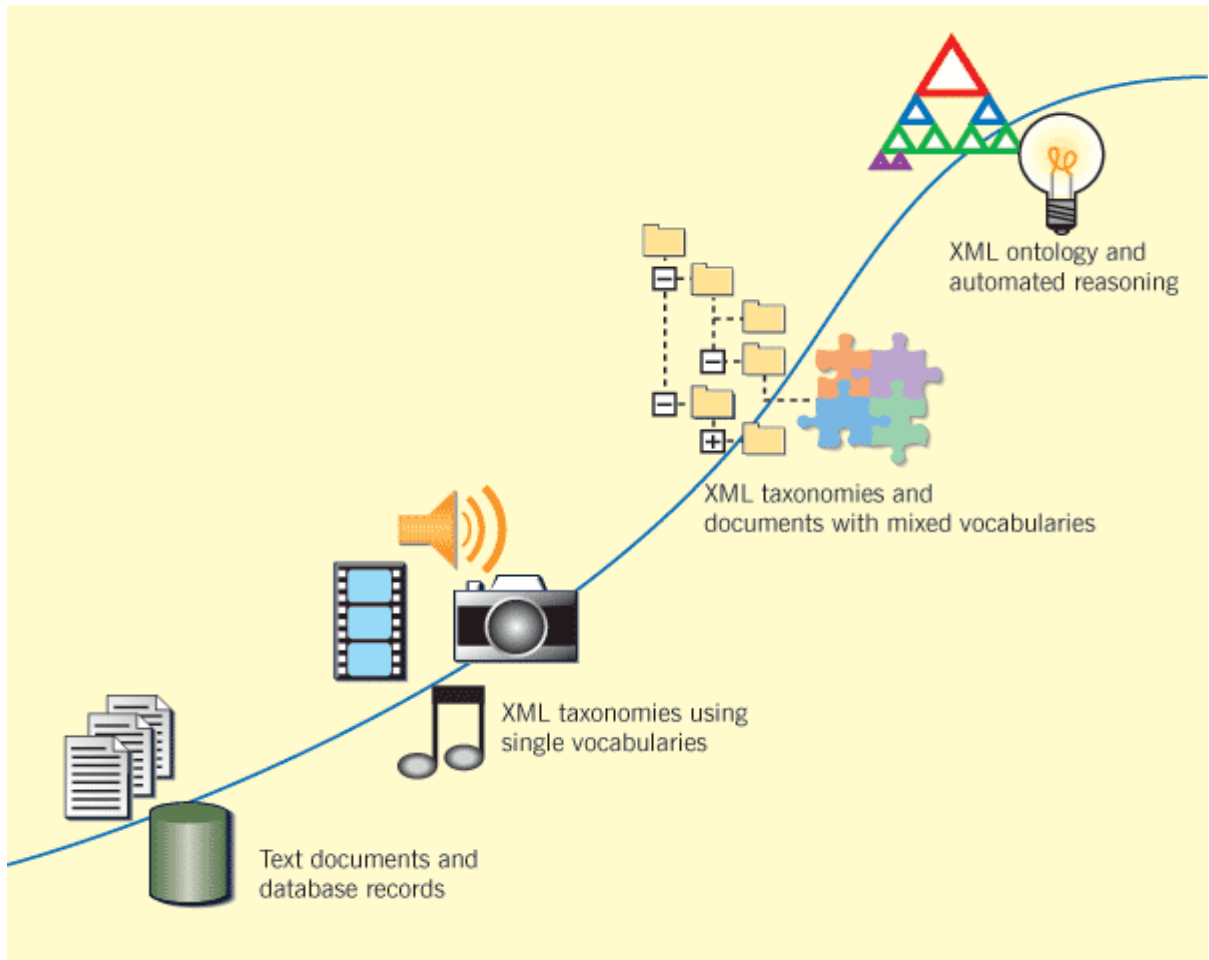


Figure 2-4 The Smart Data Continuum[19]

In the first two stages, data isn't smart in various domains. It is till the stage 3 that the relationship of data can be used to relate and combine data. Hence, data is smart now to be easily discovered and combined with other data. In stage 4, data can be inferred with existing data by following logical rules. Now data can be processed more intelligently and can build other relationship between data [19].

After the introduction of Semantic Web, we may wonder how to make Semantic Web. Semantic Web is generally built on syntaxes which use URIs to represent data, usually in triple-based structures. Many of URI data that is held in databases, or is interchanged on the World Wide Web uses a set of particular syntaxes developed especially for the task. These syntaxes are called "Resource Description Framework" (RDF) syntaxes.

- URI-Uniform Resource Identifier

A URI is simply web identified: like the strings starting with "http:" or "ftp:"

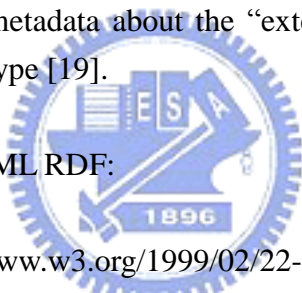
that often are found on the World Wide Web. URI enable people or machines to know where the resource is located. In fact, the World Wide Web is such a thing: anything that has a URI is considered to be “on the Web”.

The syntax of URI is carefully governed by the IETF, who published RFC 2396 as the general URI specification. The W3C maintains a list of URI schemes [4].

- RDF

A triple can simply be described as three URIs. A language which utilizes three URIs in such a way is called RDF [4]. The Resource Description Framework (RDF) is an XML-based language to describe resources. Resource on the Web is usually accessed via a Uniform Resource Locator (URL). While the metadata is attached as the part of XML document, one use of RDF is to build metadata about the document as a standalone entity. In other words, instead of marking up the internals of a document, RDF apprehends metadata about the “externals” of a document, like the author, the creation date, and type [19].

Here is an example of XML RDF:



```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/0.1/foaf/" >
  <rdf:Description rdf:about="">
    <dc:creator rdf:parseType="Resource">
      <foaf:name>Sean B. Palmer</foaf:name>
    </dc:creator>
    <dc:title>The Semantic Web: An Introduction</dc:title>
  </rdf:Description>
</rdf:RDF>
```

This piece of RDF means that this article has the title “The Semantic Web: An Introduction,” and was written by Sean B. Palmer. RDF produces the triples like the following [31]:

```
<> <http://purl.org/dc/elements/1.1/creator> _:x0 .
this <http://purl.org/dc/elements/1.1/title> "The Semantic Web: An Introduction" .
```

_:x0 <http://xmlns.com/0.1/foaf/name> "Sean B. Palmer"

Hence, the RDF model is often called a “triple” because it has three parts. Those three components are described in terms of the grammatical parts of one sentence: subject, predicate, and object. Figure 2-5 displays the elements of triple model.



Figure 2-5 RDF Triple[9]

Subject: Subject of the sentence tells us what the sentence is about. In logic, this is the term about which something is asserted. In RDF, this is the resource that is described by the ensuing predicate and object.

Predicate: In one sentence, predicate tells us something about the subject. In logic, a predicate is a function from individuals to truth-values with an entity based on the number of arguments it has. In RDF, a predicate is like a relation between the subject and the object.

Object: With logic, an object is acted upon by the predicate. In RDF, an object is either a resource referred to by the predicate or a literal value.

- **RDF Schema**

RDF schema is the language layered on top of RDF. The stack of Semantic Web is shown in figure 2-6. The bases of the stack are URI and Unicode. Above those concepts, we layer the XML syntax and namespaces to prevent vocabulary conflicts. On top of XML are the RDF and syntax discussed in the previous section. If we want to use the triple to denote a class, class property, and value, we can build class hierarchies for the classification and description of objects. This is the goal of RDF schema.

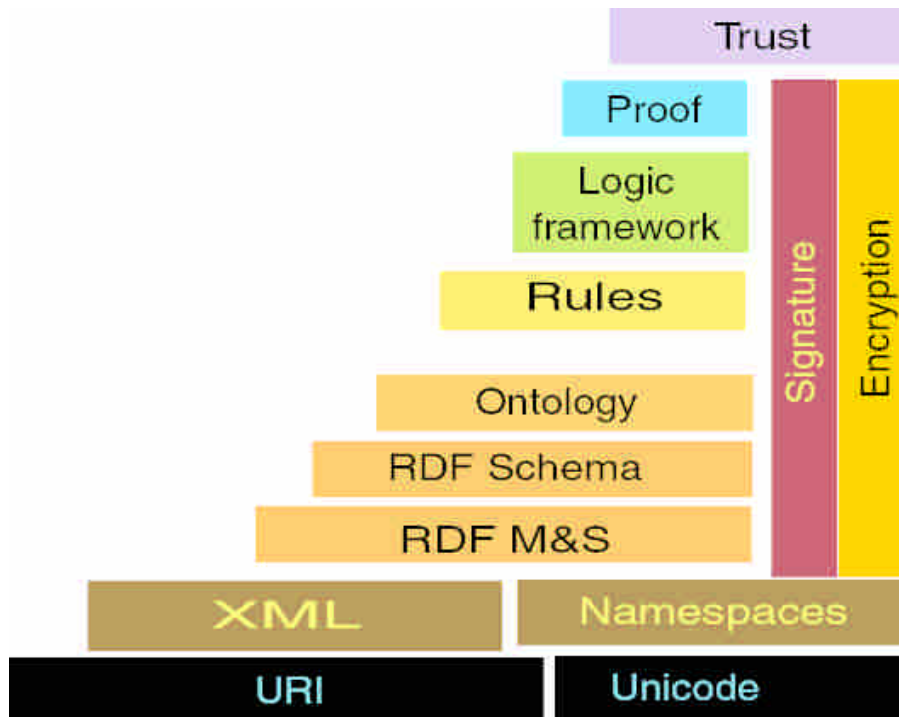


Figure 2-6 Semantic Web Stack[10]

RDF Schema is a simple set of standard RDF resources and properties to enable people to create their own RDF vocabularies. RDF Schema allows you to create classes of data. A class is defined as a group of things with common characteristics. An object is one instance of a class [19]. It also allows classes to inherit characteristics and behaviors from a parent class.

Otherwise, the concept of an RDF property is view as a relation between subject resource and object resource.

Table 2-1 presents an overview of the vocabulary of RDF, drawing together vocabulary originally defined in the RDF Model and Syntax specification with classes and properties that originate with RDF Schema [4].

Table 2-1 RDF Classes

Class name	comment
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:XMLLiteral	The class of XML literals values.
rdfs:Class	The class of classes.

rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements.
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.
rdf:List	The class of RDF Lists.

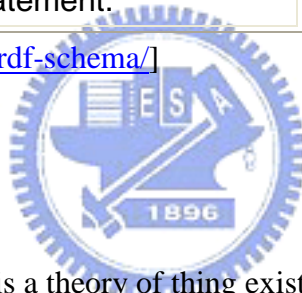
Table 2-2 RDF Properties

Property name	comment	domain	range
rdf:type	The subject is an instance of a class.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF	rdf:List	rdf:List

	list after the first item.		
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values (see the RDF Primer for <u>an example</u> of its usage).	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

[From <http://www.w3.org/TR/rdf-schema/>]

2.2.2 Ontology



In philosophy, ontology is a theory of thing existence [28]. For AI systems, what “exists” is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. In the context of knowledge sharing, ontology means a specification of conceptualization [32].

The Artificial-Intelligence literature contains many definitions of ontology; many of these contradict one another [23]. For the purposes of this guide an **ontology** is a formal explicit description of concepts in a domain of discourse (**classes**), properties of each concept describing various features and attributes of the concept (**slots**), and restrictions on slots (**facets**). An ontology together with a set of individual instances of classes constitutes a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins.

The important feature of ontology is the inference facility. Machines usually

don't realize any of information, but it can deal with the terms through ontology and reason rules. There are many applications to use ontology to enhance the functioning of the Web. More advanced applications use ontology to relate the information on one page and retrieve more knowledge via inference rules.

2.2.3 OWL

The Semantic Web is the future of the Web. It gives data meaning and enables machines to automatically process data. The Semantic Web will build on XML's ability to define customized tagging schemes and RDF's flexible way to presenting data. The first level above RDF is an ontology language that can represent the meaning of terminology in Web document. To let machines to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF schema.

OWL is one language of ontology and is intended to be used when information contained in documents needs to be processed by applications. OWL has more power than XML, RDF, and RDF-S to express meaning and semantics. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL.

2.2.4 OWL-S

OWL-S (formerly DAML-S) is an OWL-based Web service ontology that supplies web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. The latest 0.9 draft release is expected to be the last one built on DAML+OIL, and the later releases will be based on OWL.

OWL-S, the ontology of Web services, is to provide three essential types of knowledge about one web service (Shown in Figure 2-7).

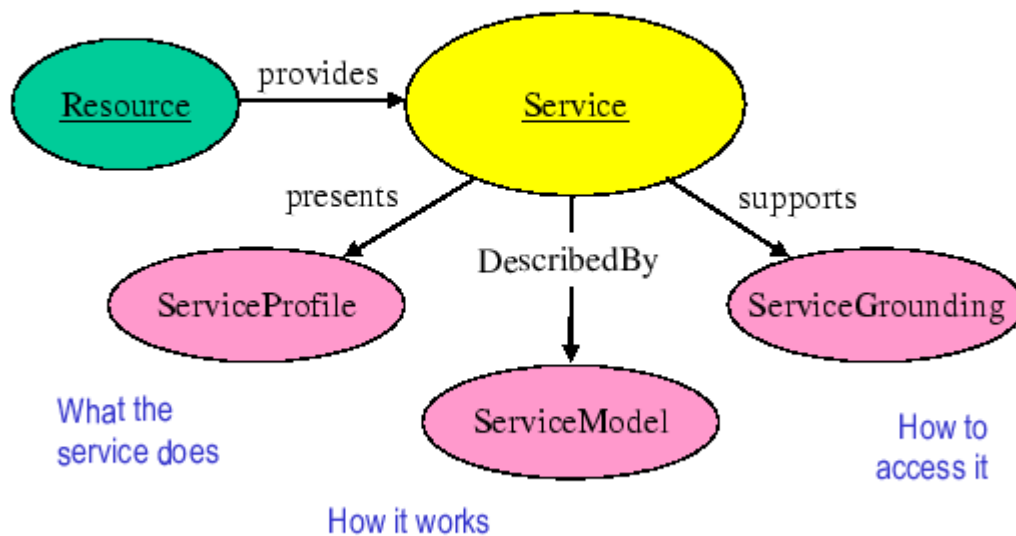


Figure 2-7 The Components of OWL-S[6]

The class “Service” is viewed as an organizational point of reference for declaring Web services; for different published service, one instance of Service will exist. The subclasses of one web service are Service Profile, Service Model, and Service Grounding. Each instance of Web service will *present* a descendant class of Service Profile, be *describedBy* a descendant class of Service Model, and *supports* a descendant class of Service Grounding. The details of profiles, models, and groundings are different each other. But those three components provide the essential type of information about Web service, as described latter.

A service profile tells “what the service does”; it provides types of information that are needed by matchmaking agents to determine which service meets their requirements. Otherwise, service profile also can represent the requirement the matchmaking agent needs. Therefore, service profile is the dual-purpose representation for the matchmaking process.

A service model tells “how the service works”; that is, it describes what happens when the service is carried out. For nontrivial services (those composed of several steps over time), this description may be used by a service-seeking agent in at least four different ways: (1) to perform a more in-depth analysis of whether the service meets its needs; (2) to compose service descriptions from multiple services to perform

a specific task; (3) during the course of the service enactment, to coordinate the activities of the different participants; and (4) to monitor the execution of Web service.

Service grounding defines the details of how to access a service. The definition usually includes a communication protocol, message formats, and other service-specific details like port numbers for contacting the service. Otherwise, the service grounding must specify the precise way of exchanging data elements of the type with the service [6].

In other words, Service Profile contains essential information for matchmaking agent to discover Web services. Service Model and the Service Grounding are associated with a Web service to provide enough information for making use of a service.

2.3 Current Matchmaking for Web service

2.3.1 LARKS



A good work exists in the field of matchmaking for Web service including LARKS [15]. The LARKS matchmaking process contains both syntactic and semantic matching and allows the specification of local ontology. LARKS matchmaking has five filters for matching, such as context matching, profile comparison, similarity matching, signature matching and constraint matching. Different combination of those filters can result in different degrees of partial matching. The following introduces the matchmaking mechanism with these five filters.

First, there is the specification of LARKS as a frame with the following slot structure.

Context	Context of specification
Types	Declaration of used variable types
Input	Declaration of input variables
Output	Declaration of output variables
InConstraints	Constraints on input variables
OutConstraints	Constraints on output variables
ConcDescriptions	Ontological descriptions of used words
TextDescription	Textual description of specification

Based on the specification of LARKS, the following is the matching process with distinct filter.

- Context matching

Context matching chooses suited advertisements compared with the request in the same or similar context. This filter roughly decreases the amount of advertisement which is not related with a given request.

- Profile matching

To compare two specifications depends on a standard technique from the information retrieval area, called term frequency-inverse document frequency weighting (TF-IDF). In accordance with the result of TF-IDF, two specifications are assumed similar or not.

- Similarity matching

The limitation of profile comparison is that it doesn't take the structure of the description into consideration. It means it can't differentiate input and output declarations of a specification. Hence, the similarity matching computes the distance values of pairs of input and output declarations, as well as input and output constraints to know how similar advertisement and request is.

- Signature matching

Both similarity and signature matching compare the request with advertisement selected by the context matching in advance. The signature filter considers the meaning of the logical constraints in LARKS specification.

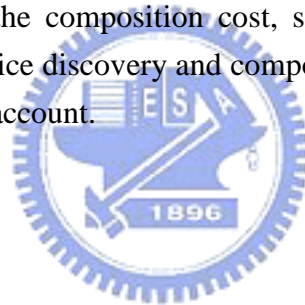
- Constraint matching

This filter mainly deals with the input/output constraints and declaration parts of the specifications. It narrows the search by using semantic information. It investigates if the input/output constraints of request and advertisement logically match.

Those filters are the different dimensions for LARKS matchmaking. However, it focuses on the service process, not the underlying data of Web service.

2.3.2 CBR (Case-Based Reasoning)

Limthanmaphon et al apply a Case-Based Reasoning technique to discover, match, and compose different Web services [2]. The fundamental idea is to solve new problems by comparing them to old problems, which have been solved in the past time. CBR uses the similarity measure to evaluate the similarity between a given query and cases in the case base. It enables proactive and reactive composition of Web services. It helps to reduce the composition cost, service collaboration, and client satisfaction, and efficient service discovery and composition. However, it doesn't take the data of Web services into account.



2.3.3 UDDI

UDDI is like a directory service to provide publishing and finding capabilities for Web services. It is the broker between providers and requesters to match Web services requesters want. UDDI includes much information, such as service description, business metadata and taxonomies of business and services. We have simply introduced UDDI. The following will explain service discovery mechanism in UDDI [29].

The finding capability is achieved by the inquiry API in UDDI. The inquiry API has two types of operations that are related with the UDDI data types: find operation and get_Details operations. The find operation searches the services registered in UDDI in accordance with some criteria, such as “business”, “service”, “binding” or “tModel”. After finding operation, UDDI can retrieve the detail information of services by get_Details operation.

In the process of searching services, keyword search is mainly used in UDDI. However, UDDI can't support the vague query like "cheap" or "comfortable". To improve the service matchmaking, we proposed that fuzzy logic is integrated with UDDI for semantic query.

2.4 Fuzzy Set Theory

The Theory of fuzzy set is not totally different with classical set theory. It is proposed on the basis of the classical set theory. A fuzzy set is a set with the boundary between 0 and 1. Unlike classical set theory, the value of fuzzy set isn't just 0 or 1. It is a smooth boundary for the fuzzy set theory. It generalizes the notion of membership from a black-and-white binary classification in classical set theory into one that allows partial membership. When the value of membership is 0, it means complete non-membership. Otherwise, if the value of membership is 1, it represents a complete membership. Usually, the membership or characteristic function is denoted by the Greek lowercase letter μ [12].

A fuzzy set could be defined in two ways: (1) by calculating membership values of those members in the set, or (2) by defining membership function mathematically.

Generally, the first way is used when the set is discrete, because a continuous fuzzy set has an infinite number of elements. However, a fuzzy set A can be defined through enumeration using the expression.

$$A = \sum \mu_A(x_i) / x_i$$

Where the summation and addition operators refer to the union operation and the notation $\mu_A(x_i) / x_i$ refers to a fuzzy set containing exactly one (partial) element x with a membership degree $\mu_A(x_i)$. In sum, we don't represent those elements x_i whose membership value is zero.

2.4.1 Membership Functions

Whereas there are numerous types of membership functions, the most commonly used in practice are trapezoids, Gaussian, S and Z functions. In the

following we will introduce the four types of membership functions.

- Trapezoidal Membership Function

A trapezoidal membership function is specified by four parameters {a, b, c, d} as follows:

$$\mu_A(a) = \begin{cases} \frac{x - (c - b)}{b}, & c - b \leq a \leq c \\ 1, & c \leq a \leq d \\ \frac{-x + (d + b)}{b}, & d \leq a \leq d + b \\ 0, & a > d + b, a < c - b \end{cases}$$

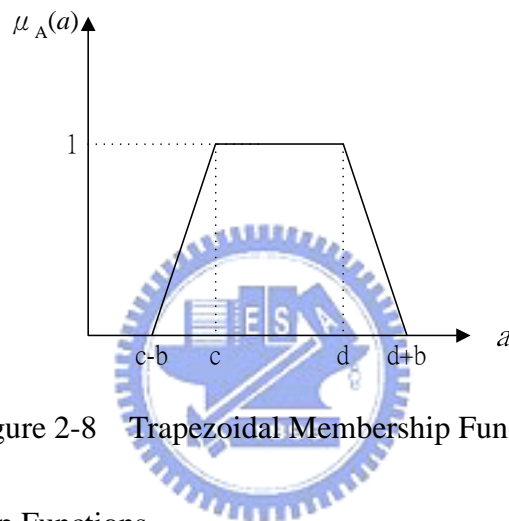


Figure 2-8 Trapezoidal Membership Function

- S Membership Functions

A S membership function is a smooth membership function with three parameters: a, b and c.

$$\mu_A(a) = \begin{cases} 0, & a \leq b \\ 2\left(\frac{a-b}{d-b}\right)^2, & b \leq a \leq c \\ 1 - 2\left(\frac{a-b}{d-b}\right)^2, & c \leq a \leq d \\ 1, & a \geq d \end{cases}$$

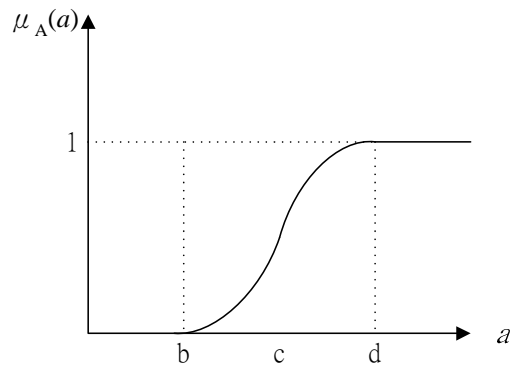


Figure 2-9 S Membership Function

- Z Membership Functions

A Z membership function is a smooth membership function with three parameters: a, b and c.

$$\mu_A(a) = \begin{cases} 0, & a \leq b \\ 2\left(\frac{a-b}{d-b}\right)^2 & b \leq a \leq c \\ 1 - 2\left(\frac{a-b}{d-b}\right)^2 & c \leq a \leq d \\ 1, & a \geq d \end{cases}$$

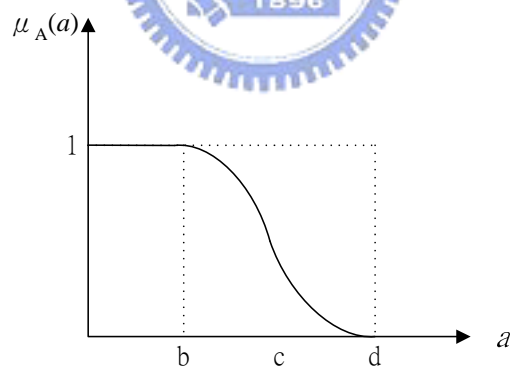


Figure 2-10 Z Membership Function

- Gaussian Membership Functions

A Gaussian membership function is specified by two parameters {m, σ}

As follows:

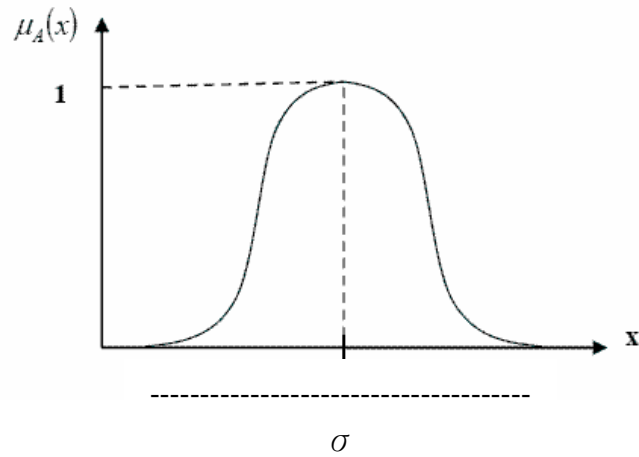


Figure 2-11 Gaussian Membership Function

Where m and σ denote the center and width of the function, respectively. We can control the shape of the function by adjusting the parameter σ .

2.4.2 Basic Operations

Since membership of a fuzzy set is a matter of degree, the operation of fuzzy set should be defined accordingly. There are three basic operations: union, intersection, and complement [1].

Union

The union operation is used in various ways. There isn't unique way to do union operation. Hence, we discussed the definition that is used in most cases. The union of two fuzzy sets \tilde{A} and \tilde{B} with the membership functions $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$ is a fuzzy set \tilde{C} , written as $\tilde{C} = \tilde{A} \cup \tilde{B}$, whose membership function is related to those of \tilde{A} and \tilde{B} as follows:

$$\forall x \in U : \mu_{\tilde{C}} = \max[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$$

Intersection

There isn't the unique way to do intersection operation as well as the union operation. According to the *min-operator* the intersection of two fuzzy sets \tilde{A} and \tilde{B}

with the membership functions $\mu_{\underline{A}}(x)$ and $\mu_{\underline{B}}(x)$, respectively, is a fuzzy set \underline{C} , written as $\underline{C} = \underline{A} \text{ I } \underline{B}$, whose membership function is related to those of \underline{A} and \underline{B} as follows:

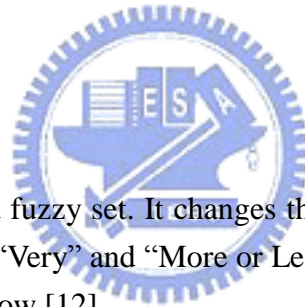
$$\forall x \in U : \mu_{\underline{C}} = \min \left[\mu_{\underline{A}}(x), \mu_{\underline{B}}(x) \right]$$

Complement

The complement of a set \underline{A} , denoted $\overline{\underline{A}}$, is represented as the collection of all elements in the universe which aren't included in the set \underline{A} .

$$\forall x \in U : \mu_{\overline{\underline{A}}} = 1 - \mu_{\underline{A}}(x)$$

2.4.3 Hedges



Hedge is a modifier to a fuzzy set. It changes the meaning of the original set to create a compound fuzzy set. “Very” and “More or Less” are two usually used hedges. Their definitions are listed below [12].

[Very]

$$\mu_{\text{Very}A}(x) = [\mu_A(x)]^2$$

[More or Less]

$$\mu_{\text{MoreOrLess}A}(x) = \sqrt{\mu_A(x)}$$

Figure 2-12 shows the variations of membership functions with hedges.

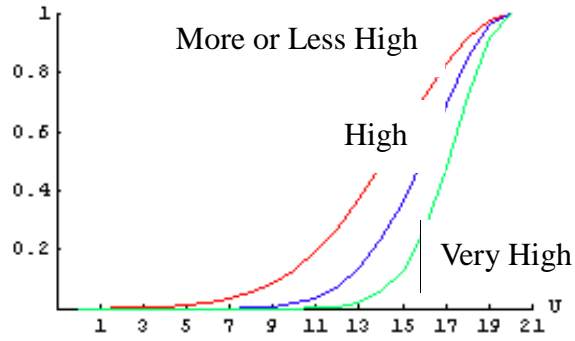


Figure 2-12 Hedge

As shown in figure 2-12, “Very” has the effect of converging the membership function, while “More or Less” broaden the membership function. This is intuitively appealing because the criteria for “Very High” should be more stringent than those for “High”, while the criteria for “More or Less high” should be relaxed. The relationship can be shown:

$$\mu_{VeryA}(x) \leq \mu_A(x) \leq \mu_{MoreOrLessA}(x)$$

Practically, we can use hedge to any fuzzy sets. However, it is used only when the compound term is meaningful.

2.4.4 PRUF Translation Rules

Fuzzy set theory is very useful to deal with imprecision problem. To represent vagueness of natural language queries, Possibility Relational Universal Fuzzy (PRUF), derived from possibility theory, is a representation language for natural language proposed by Zadeh (1978). PRUF is a mean to express imprecise knowledge as well as to make precise fuzzy propositions expressed in natural language. It is able to translate a set of premises expressed in natural language into expression in PRUF. The rules of inference of fuzzy logic can be applied to translate other expressions in PRUF into natural language. The main constituents of PRUF are: a collection of translation rules, and a set of inference rules [18].

The most important categories of translation rules in PRUF are as follows [11]:

Type 1 Rules pertaining to modification:

$$P^+ \hat{=} N \text{ is } m \tilde{F} \text{ that is } \pi_{(x_1, \dots, x_n)} = \tilde{F}^+$$

where \tilde{F}^+ is modification of \tilde{F} by the modifier m . $\pi_{(x_1, \dots, x_n)}$ denotes a possible distribution over x_i . For example: A is very comfortable airline.

Type 2 Rules pertaining to composition:

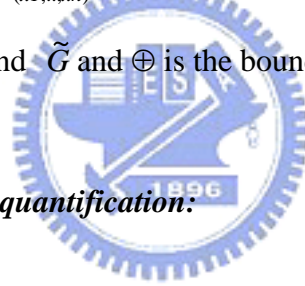
The compound statement can be translated as $p=q*r$ where $*$ denotes a logical connective (e.g. and, or).

$$\begin{aligned} q &\hat{=} M \text{ is } \tilde{F} \rightarrow \pi_{(x_1, \dots, x_n)} = \tilde{F} \\ r &\hat{=} N \text{ is } \tilde{G} \rightarrow \pi_{(y_1, \dots, y_n)} = \tilde{G} \end{aligned}$$

Statement p gets the following:

$$\mu_{\tilde{F}'_L \oplus \tilde{G}'_L}(u, v) = \min\{1, \mu_{\tilde{F}}(u) + \mu_{\tilde{G}}(v)\}$$

If M is \tilde{F} then N is \tilde{G}' $\rightarrow \pi_{(x_1, \dots, x_n)} = \tilde{F}' \oplus \tilde{G}'$ where \tilde{F}'_L and \tilde{G}'_L are the cylindrical extension of \tilde{F} and \tilde{G} and \oplus is the bounded sum. For example: A is cheap and B is good



Type 3 Rules to pertaining to quantification:

$\tilde{Q} = \{[\text{prop}(\tilde{F} / \tilde{G}), \mu_{\text{most}}(u, v)] \mid u \in \tilde{F}, v \in \tilde{G}\}$, where the proportion of \tilde{F} in \tilde{G} is denoted as $\text{prop}(\tilde{F} / \tilde{G})$ and

$$\text{prop}(\tilde{F} / \tilde{G}) = \frac{|\tilde{F} \cap \tilde{G}|}{|\tilde{G}|}$$

For example: Airline X has flights to most European cities

Type 4 Rules pertaining to qualification:

$$\text{cons}\{N \text{ is } F \mid N \text{ is } G\} \hat{=} \text{poss}\{N \text{ is } F \mid N \text{ is } G\} = \sup_{u \in U} \{\min(\mu_{\tilde{F}}(u), \mu_{\tilde{G}}(u))\}$$

where poss is possibility operator and sup is support operator. For example: “X is cheap” is very possible.

Chapter 3 An Ontology-based Fuzzy Matchmaking Framework

In this chapter, we introduce our novel matchmaking framework, an ontology-based fuzzy matchmaking for Web services. There are details about how to import OWL-S in UDDI and the representation for the underlying data of Web services. Further, we explain how to classify Web services according to data of Web service by fuzzy theory and then to match suited services with the classified result.

3.1 Framework

For improving the current matchmaking mechanisms, we proposed the framework that exploits fuzzy logic and ontology to explain and to represent the data of Web services. Otherwise, we use the fuzzy description in UDDI to allow consumers to search with vague queries. It enhances the matchmaking of UDDI and explores the hidden dimension to describe Web services for searching.

Figure 3-1 shows our proposed framework, and the following is our detail of our framework.

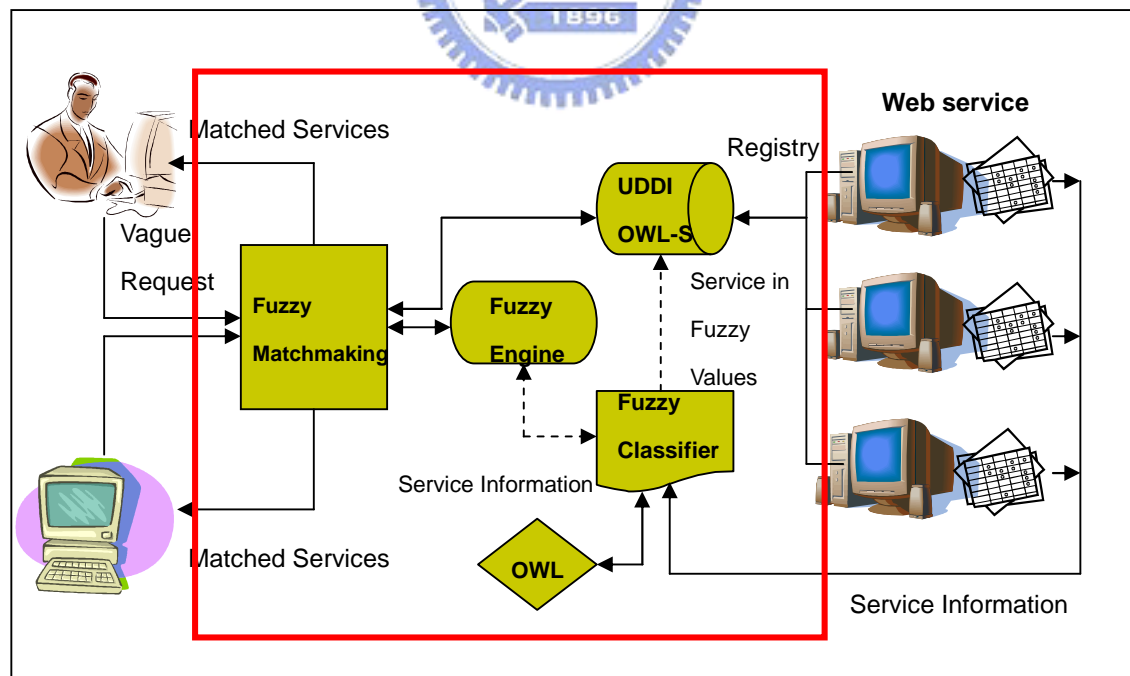


Figure 3-1 Ontology-based Fuzzy Matchmaking Framework

Our proposed framework is shown in figure 3-1. There are six steps in our

framework.

- step 1 First, all Web services described with OWL-S will be registered in UDDI by their service providers.
- step 2 Fuzzy classifier asks for the databases of Web services that were registered in UDDI.
- step 3 Fuzzy Classifier uses OWL to explain the heterogeneous database of each Web service.
- step 4 In fuzzy engine, it defines fuzzy terms and related membership functions that are based on the data schema of the database of Web services.
- step 5 Based on the OWL interpretation, fuzzy classifier asks fuzzy engine to calculate the fuzzy value of Web services for different fuzzy terms.
- step 6 Fuzzy classifier explains the heterogeneous database by OWL and represents the abstract description of Web services by fuzzy logic. Fuzzy classifier classifies Web services as the representation of different fuzzy terms. Hence, Fuzzy Classifier records fuzzy values of each Web service for specific fuzzy terms in UDDI for searching.
- step 7 When the query is addressed, fuzzy matchmaking parses query sentence and request UDDI and fuzzy engine to return suited Web services to requesters.

Those steps are the progress of our framework. The following will explain how to import OWL-S in UDDI, and how to represent the data of Web services. Furthermore, we will introduce the capability of fuzzy classifier and fuzzy matchmaking.

3.2 Importing OWL-S in UDDI

The aim of semantic web is to locate services automatically based on the functionalities Web services provide. It is helpful to discovery Web services with semantic web. The integration of semantic web and UDDI performs the semantic matchmaking.

OWL-S is the ontology for Web service, it has three components: service profile, service model, and service grounding. In our proposed architecture, the quality rating parameter of the service profile is the parameter we use. We intend to focus on the quality rating parameter. The detail illustration about how to import the OWL-S profile file to the UDDI registry is mention in [20].

UDDI represents a business as a BusinessEntity object that contains information like name of the business, contact ways such as the physical address, telephone number or fax number, the URL of the company site. A BusinessEntity is related with one or more Business Services that describe the specific service business provides. In turn a business service is also related with one or more BindingTemplate that point out how to access the service. In addition to describe businesses, services and binding templates, UDDI provide a data structure called TModel that allows the specification of extra attributes of the entities in the UDDI repository. The data structure and its diagram are show in figure 3-2[30] and figure 3-3[17].

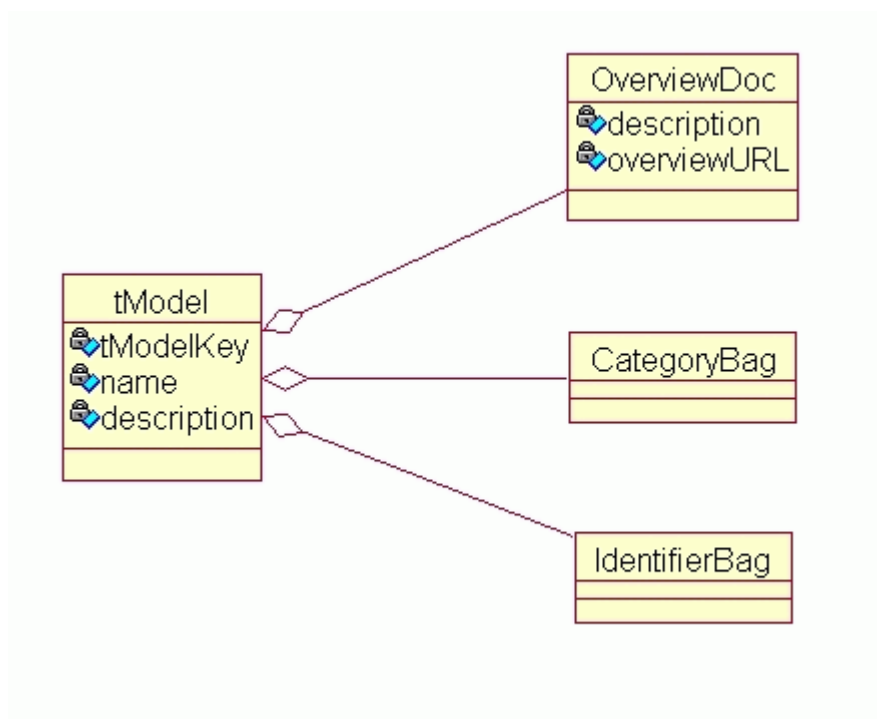


Figure 3-2 TModel Data Structure

```

<tModel tModelKey="uuid:cd153..">
  <name>unspsc-org:unspsc</name> ← Name of tModel
  <description xml:lang="en">
    Product Taxonomy: UNSPSC (Version 7.3) ← Explanation of tModel
  </description>
  <overviewDoc>
    <description xml:lang="en">
      This tModel defines Version 7.3 of the UNSPSC product taxonomy.
    </description>
    <overviewURL>http://www.uddi.org/taxonomies/...</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf2..." ← Information that shows
      keyName="types" this tModel shows identification code
      keyValue="categorization" />
  </categoryBag>
</tModel>

```

Figure 3-3 TModel Data Structure

Services in UDDI can be searched by name, by location, by business, by bindings or by TModels. However, UDDI doesn't support any inference based on the taxonomies referred to by the TModels. Integration of semantic web (OWL-S) and UDDI will solve this problem.

The mapping of OWL-S profiles into UDDI representations is shown in figure 3-4 [20].



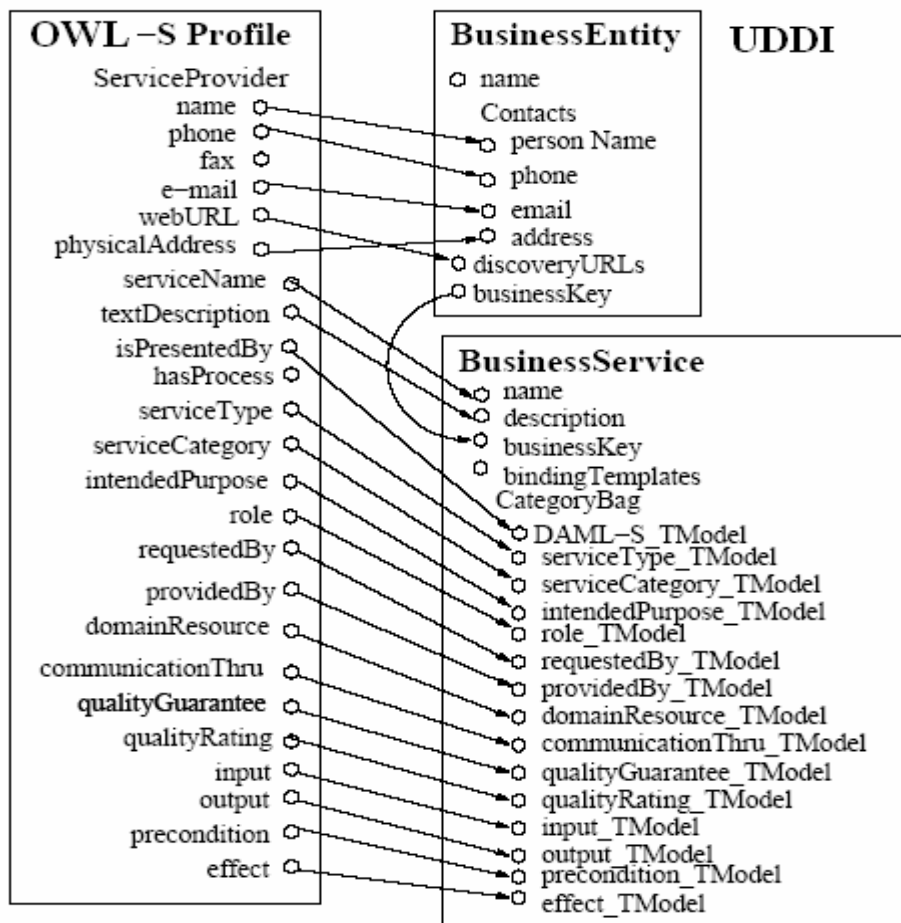


Figure 3-4 UDDI Service Representation

There are some parameters mapped directly from OWL-S Profile to UDDI. Besides, OWL-S specific attributes such as input, output, geographicRadius, and qualityrating are instead represented by the TModel structure.

After mapping OWL-S to UDDI, UDDI can support the semantic inference with OWL-S through TModel query operation. There are finding operations with TModel, such as find_tModel operation and get_tModelDetail operation. We could obtain more detail information for the specific Tmodel with those finding operations.

3.3 Web Service Data Representation

Fuzzy classifier classifies the information resided in Web services, such as the room information of the hotel accommodation services or the seat information of the airline booking services. Fuzzy classifier represents Web services as fuzzy terms, like “cheap ticket” or “nice view” as primitive terms or “comfortable” or “convenient” as composite term for airline ticket service. A primitive term can be derived from a fuzzy set A which is defined as $A = \{(x, \mu_A(x)) | x \in X\}$. The following explains a membership function to define the concept “reasonable” for price.

$$\mu_{\text{reasonable-price}}(x) = \left\{ \begin{array}{l} 1, \text{ if } x=50 \\ 50-x/40, \text{ if } 10 < x < 50, \\ 0, \text{ if } x < 10 \text{ or } x > 200, \\ 100-x/50, \text{ if } 50 < x < 100 \\ \end{array} \right\}$$

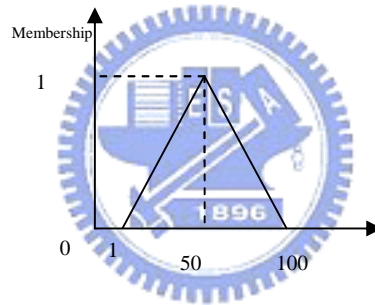


Figure 3-5 Membership Function for “reasonable” related with Price

Composite terms are generated through the combination of compound statements and fuzzy rules. For example, a composite term “comfortable” may contain sub-attributes such as “spacious”, and “quickly to arrive to destinations”. A composite term could be derived from other composite terms. The composite terms can be described using expression as follows.

$$\mu_{\tilde{C}}(x) = \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\}, x \in X$$

$$\mu_{\tilde{D}}(x) = \min\{\mu_{\tilde{C}}(x), \mu_{\tilde{D}}(x)\}, x \in X$$

where $\mu_{\tilde{A}}(x)$, $\mu_{\tilde{B}}(x)$, and $\mu_{\tilde{D}}(x)$ represent three different primitive terms. $\mu_{\tilde{C}}(x)$ is a

transitive term and $\mu_{\tilde{D}}(x)$ is a composite term.

Otherwise, by the definition of hedge, modified fuzzy terms such as “very cheap” or “more or less comfortable” can be defined below.

$$\text{very } \tilde{A} = \text{con}(\tilde{A})$$

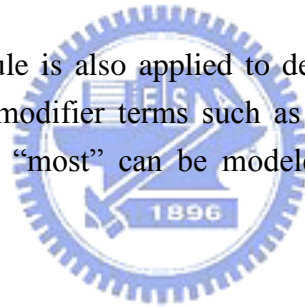
where $\mu_{\text{con}(\tilde{A})} = (\mu_{\tilde{A}}(u))^2$ and con stands for concentration operator.

$$\text{more or less } \tilde{A} = \text{dil}(\tilde{A})$$

where $\mu_{\text{dil}(\tilde{A})} = (\mu_{\tilde{A}}(u))^{1/2}$ and dil stands for dilution operator.

On the other hand, the complement membership function is also powerful expressive operator. If user or software wants to search “not expensive ticket”, a complement operation can exclude unnecessary search space such as “cheap” and expand the search possibility.

Otherwise, the PRUF rule is also applied to define composite terms, and also could be used to model the modifier terms such as “very cheap ticket” and “most tickets are cheap”. The term “most” can be modeled using quantification type of PRUF rule.



3.4 Fuzzy Classifier

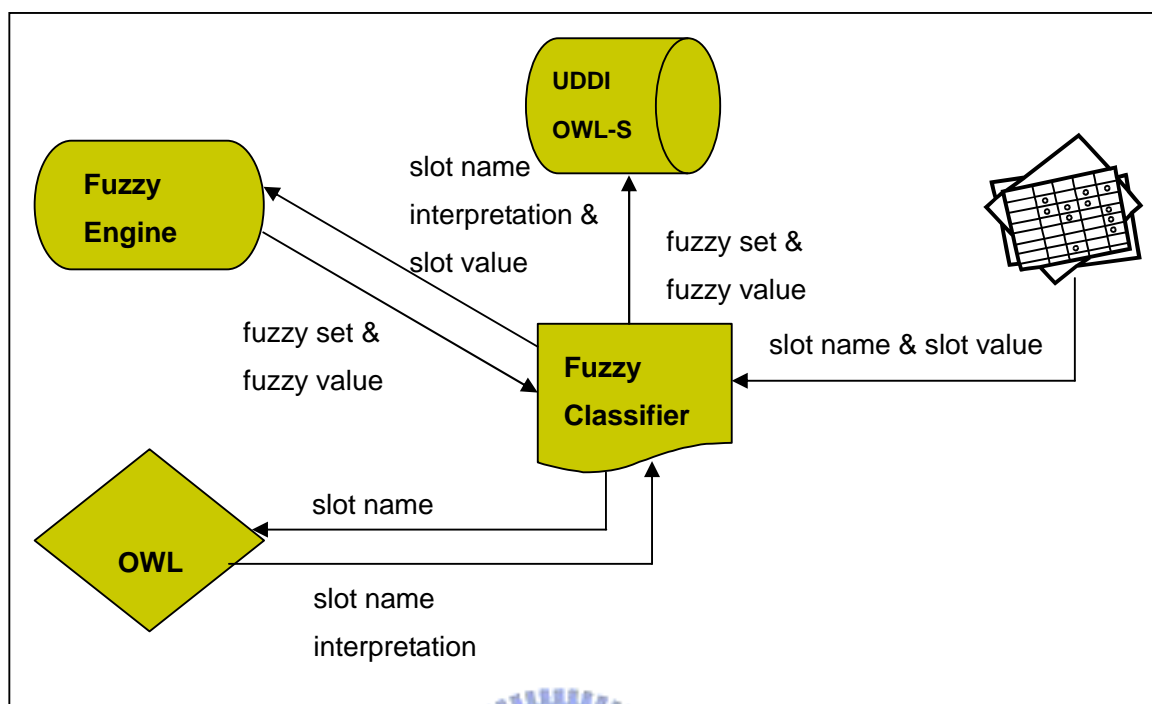


Figure 3-6 Fuzzy Classifier

The objective of fuzzy classifier is to represent Web services as fuzzy terms that are related with different fuzzy sets. Figure 3-6 illustrates how fuzzy classifier works. First, it would obtain the database of Web service such as the room information of the hotel accommodation or the seat information of airline ticket reservation service. Web service is classified by fuzzy classifier according to data of Web service such as the fare of airline ticket to represent Web services with fuzzy description. Though current matchmaking for Web services has taken the input, output and other context as filters, it doesn't take the data of Web services into account. In our proposed framework, we first take the data of Web services as a matching criterion.

There are different slot names in the heterogeneous database of each Web service, but they all have the same meaning. For an example, the fare of the ticket recorded in the database of Eva Airline is represented as slot name "fare", and that of East Airline is represented as different slot name "value". There would be the public interpretation existing for interpreting different slot names inside the heterogeneous databases to help machines deal with these databases automatically. In our framework, we define the explanation about slot names by OWL.

We use OWLJessKB as a reasoning tool to parse OWL document that interprets

the relationship among heterogeneous databases. After realizing the meaning of each slot, we can calculate the slot value with related membership functions defined by fuzzy rules. If one slot name has some meaning defined by OWL, there will have one or more fuzzy terms related with the slot. For examples, the slot “ticket price” which means how much to buy one airline ticket in OWL is defined to be related with three fuzzy terms, “cheap” , ”medium” and “expensive” because those three fuzzy terms are usually people’s perception for price. After mapping slots with related fuzzy terms, classifier uses fuzzy engine to calculate the value of each slot as fuzzy value that belongs to some fuzzy term. Fuzzy engine returns the fuzzy value to the classifier. Fuzzy classifier then records the fuzzy values in UDDI for matchmaking. The fuzzy values for one Web service are recorded in the description of TModel as follows.

```

<tModel tModelKey=“uuid:cd153..”>
<name>Cheap</name>
<description xml:lang= “en”
    0.5    <!--fuzzy value for “cheap” as fuzzy term -!>
</description>

```

Fuzzy Classifier records the different fuzzy values of each Web service according to different fuzzy terms in UDDI. It integrates UDDI with fuzzy logic, hence, it allows consumers to search Web services with vague queries in UDDI.

3.5 Fuzzy Matchmaking

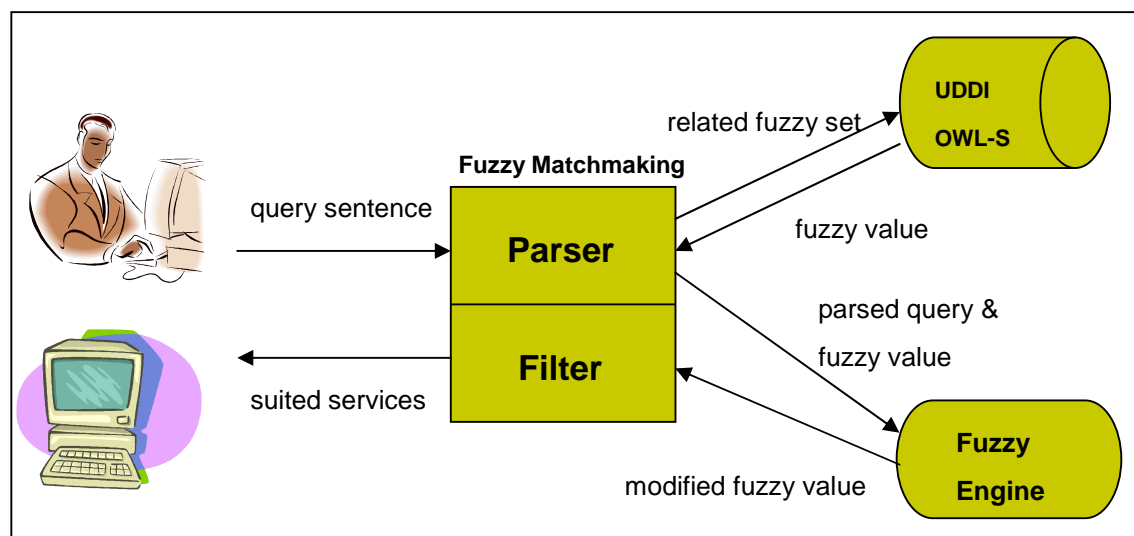


Figure 3-7 Fuzzy Matchmaking

Figure 3-7 illustrates how fuzzy matchmaking works. There are three parts in the fuzzy matchmaking. First, fuzzy matchmaking should understand what the vague query means. In addition to realize the vague query, fuzzy matchmaking should parse input query from consumers. Based on fuzzy operations, it may parse “AND” “OR” “NOT” as basic fuzzy operation from the query. They are called operators. Otherwise, fuzzy matchmaking also has to parse the query to retrieve some describable fuzzy terms. They are called operands. When the operators and operands are parsed by fuzzy matchmaking, the first part is done and it continues to the second part of fuzzy matchmaking.

The second part is to retrieve fuzzy description for Web service from UDDI. Fuzzy matchmaking finds services in UDDI and then asks for the fuzzy description of TModel for related fuzzy terms. After fuzzy matchmaking found Web services and related descriptions of TModel as the fuzzy values, it requests the fuzzy engine to do fuzzy operations by the parsed result from the first part. Fuzzy operation means the intersection, or union, or complement among fuzzy terms retrieved from the vague query. The objective of fuzzy matchmaking is to obtain the suited Web Services by vague query. With the fuzzy matchmaking mechanism, it can obtain target services from UDDI and filter these services with the description of TModel through the fuzzy operations. Fuzzy matchmaking is the interface for communication between UDDI, fuzzy engine, and consumers.

3.6 The Constraints for Our Framework

There are some constraints in our proposed framework. When the new Web service is registered in UDDI, our framework should detect the new Web service and classify it with fuzzy terms. However, there are many new Web services are registered in UDDI at the same time. Our proposed architecture doesn't support the dynamic classifying mechanism.

Another constraint is the qualityRating definition of OWL-S. There are only the two properties, the data property “ratingName” and object property “rating”. If the quality definition of OWL-S can add the other data property “ratingValue”, it isn't necessary to record fuzzy value in TModel of UDDI. Therefore, it can be searched only by OWL-S without UDDI. Therefore, it truly achieves the automatic searching by machines.

Chapter 4 Implementation

An Example for Airline Ticket Reservation Service

Our implementation for our ontology-based fuzzy matchmaking framework is described in this chapter. First, we introduce the implementation environment and airline ticket service. Then, we classify data of Web services with fuzzy terms defined by our framework. Finally, we match Web services based on the classified data and then compare the performance between our framework and UDDI.

4.1 Implementation Environment

Our environment for implementing our framework is based on Windows XP. We run Tomcat 5.5 for our web server as the container for the OWL document. For integrating OWL and Jess, we had tried to use SWRL to transform rules to Jess. We also had tried to use the JessTab of protégé to enable Jess to interpret the OWL definition. However, OWLJessKB is most useful for Jess to integrate OWL with java expert system, Jess. Table 4-1 shows the detail of our implementation environment. Otherwise, we will briefly introduce tools we used for implementation.

Table 4-1 The Development Environment

Hardware	Pentium-4 2.0G 512K L2 cache 512MB DDR DRAM
OS	Windows XP with Service Pack 2
Language	JDK1.5
Expert System	Jess
OWL Editor	Protégé 3-1-Beta
OWL Parser	OWLJessKB
Server	Tomcat 5.5

Jess

The Java Expert Shell System (JESS) is a ruled-based programming environment written in Java. Jess was originally inspired by the CLIPS expert system shell, but was developed as a complete, distinct, dynamic environment of its own [34].

Jess Facts

The data stored in Jess is called “facts”. It expresses a ground atom. A jess fact can be the one sentence or the member of the group of data. Facts can be formed as the relational database by defining the named template which specifying the name and number of slots in facts. For example, the following statement defines a template named “car” with slots such as the color, the name and the brand of cars to store the information about “car” in Jess.

```
(deftemplate car (slot brand) (slot name) (slot color))
```

It help us to create some working memory for the fuzzy computing.

Jess Rules

A Jess rule is similar to “if...then” statement in a programming language, in that it includes the conditional expression and a set of commands to execute when the conditional expression defined in rules is satisfied. An “if” part as the conditional expression on the left-hand-side (LHS) of a rule is known as the antecedent. A “then” part as the commands to execute is known as the consequent. A Jess rule definition begins with the “defrule” followed by a rule name and its LHS and RHS part. The following rule defines that if we have the car from BMW and the color of the car is black, then we can print this kind of information out.

```
(defrule car-rule  
(cat (brand BMW) (name BMW) (color black))  
=>  
(printout t “Database has the data of BMW car” crlf))
```

It helps us to drive some actions. For examples, we can retrieve some data by limiting their values or we can build other facts when the specific facts exist.

Jena

Jena is a Java framework for building application related with the semantic web. It can interpret many ontology languages such as RDF, OWL or RDQL to apply

semantic web to applications. The framework of Jena includes [3]:

- An RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- RDQL-a query language for RDF

Therefore, Jena can interpret those languages above for applications.

OWLJessKB

OWLJessKB is a description logic reasoner implemented inside the expert system. It helps Jess to manipulate with semantic web ontology. In OWLJessKB, OWL documents in RDF-XML form are retrieved from the World Wide Web and converted into triples. Triple is the template in Jess to store the OWL data. Each triple asserts a relation between a subject and an object through a predicate [3]. The example follows

```
(triple (predicate http://www.w3.org/2000/01/rdf-schema#subClassOf)
        (subject http://localhost:8080/tests/definition.owl#value)
        (object http://localhost:8080/tests/definition.owl#price)
)
```

OWLJessKB uses ARP (“Another RDF Parser”) to load and parse XML documents containing RDF [14]. ARP is part of the Jena toolkit for parsing RDF documents. OWLJessKB uses the expert system, Jess, enables the inference of semantic web to improve the machine learning in the network environment. In our proposed framework, we use OWLJessKB to interpret the meaning of slots of databases from Web services with OWL.

Protégé

Protégé is a tool that allows users to construct domain ontology, to customize data entry forms, and enter data. It is also a platform that can easily be extended to include graphical components such as graphs and tables, media such as sound, images, and video, and various storage formats such as OWL, RDF, XML, and HTML. It is very useful to build domain-based ontology easily and quickly [21].

4.2 Airline Ticket Reservation Service

The airline ticket reservation as one kind of Web services is provided by various airline companies on the internet. This service helps travelers find suited airline tickets for their traveling plans. Generally, users want to find airline ticket reservation service through UDDI. The current matchmaking for Web services is the comparison of input, output and other constraints of services processes defined by OWL-S. In our proposed framework, we first take the underlying data of Web services into account. Here we have the ten databases for ticket information of ten air companies. Table 4-2 is the ticket information about the airline ticket reservation for Eva Airline. The slot “Seat Name” means the name of the seat. The slot “Value” means the fare for the round-trip ticket from Taipei to Shanghai. The slot “Seat Size” means the space one seat has. The slot “Air Period” means the time passengers take to arrive to the destinations. The slot “Seat Type” classifies the tickets with the yearly ticket and the traveling ticket.

Table 4-2 Eva Airline’s Database

Seat Name	Value(dollar)	Seat Size(m ³)	Air Period(hour)	Seat Type
A01	14150	2.1	3	Yearly
A02	19500	2.5	3	Traveling
B01	15950	2.5	2.6	Yearly
B02	20200	2.5	2.6	Traveling
C01	16250	2	2.2	Yearly
C02	17050	2	2.2	Traveling

Otherwise, we need the OWL-S file of Eva Airline for registry in UDDI. There is the OWL-S profile from [25]. The “qualityRating” part of OWL-S profile is only the part what we want to make use of. The tag <profile:ratingName> records the different fuzzy terms we use to classify data of Web services.

```

<profile:qualityRating>
  <profile:QualityRating rdf:ID="BravoAir-goodRating">
    <profile:ratingName>
      QoS
    </profile:ratingName>
    <profile:rating rdf:resource="&concepts;#NullRating"/>
  </profile:QualityRating>
</profile:qualityRating>

```

Figure 4-1 The QualityRating of OWL-S Profile

Those ten airline companies have their ticket information and their own OWL-S documents. We classify those ticket data of all airline companies by fuzzy classifier.

4.2.1 Fuzzy Terms

Based on airline ticket data, Table 4-2, we define four fuzzy terms to classify Web services. The three fuzzy terms, “cheap” 、”medium” 、”expensive”, are related with ticket fare. The other fuzzy term “comfortable” is related with the space of seat and the time passengers take on air. Generally, people evaluate ticket fare with three different levels, the cheap fare, or the medium fare, or the expensive fare. Therefore, we use the three degrees for ticket fare as fuzzy terms. The membership functions related with three fuzzy terms mentioned previously are shown in figure 4-2.

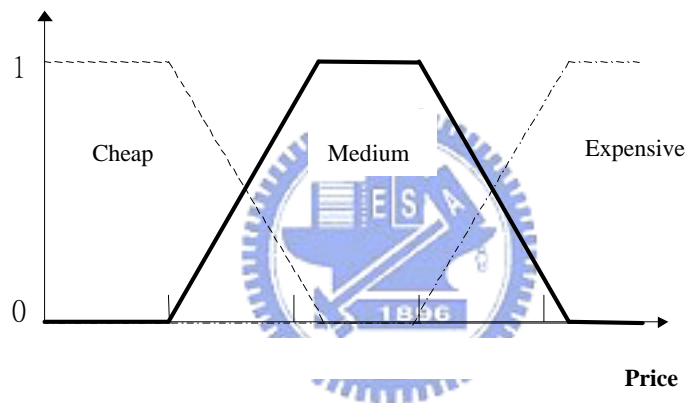


Figure 4-2 Fuzzy Terms for Price

Otherwise, the other fuzzy term is “comfortable”. Consumers select the specific airline company not only for the price, but also for the consideration of comfort. In [35], there are many factors to fulfill the consumers’ satisfaction. We choose two factors to measure the comfort degree. Consumers usually think that the bigger the space of one seat is, the more comfortable the airline service is. Otherwise, consumers also think the less time to arrive to the destination, the more comfortable the airline service is. Therefore, we take “comfortable” as fuzzy term based on those two factors, “seat space” and “air time”. The membership function of “seat space” for “comfortable” is shown in figure 4-3. The membership function of “air time” for “comfortable” is shown in figure 4-4.

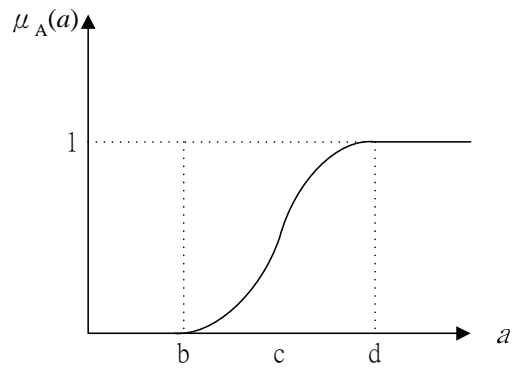


Figure 4-3 “comfortable” for Seat Space

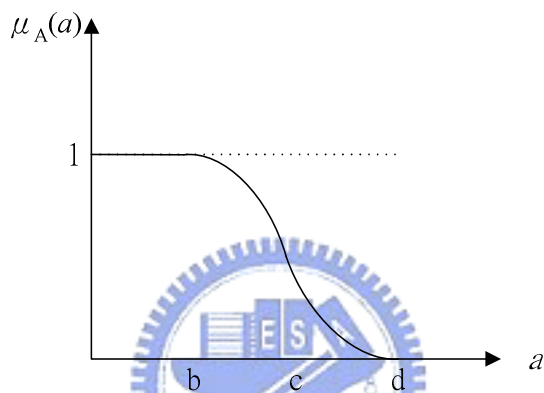


Figure 4-4 “comfortable” for Air Time

For computing the fuzzy value of “comfortable”, we give two factors the same weight as 0.5. When we want to know what the fuzzy value of “comfortable” for one company is, we can average the value of “seat space” for “comfortable” and the value of “air time” for “comfortable”. That is to say, “comfortable” is the composite term, and “cheap” or “medium” or “expensive” is the primitive terms.

4.2.2 Building OWL Definition

For interpreting heterogeneous databases of airline companies, we use ontology to explain the meaning of slot name in the databases. We make use of protégé_3_1_beta as the tool for editing ontology. The meaning of each slot name resided in airline database is defined by OWL. The different slot names that have the same meaning are defined with the relationship “subclassOf” related with one class. For example, the slots such as “fare”、 “value”、 “cost” mean the money you need to pay for one ticket, therefore, they are the subclass of “price”. In other words, if two slot names are the subclass of price, they have the same meaning in the database. It has other interpretations for other slot names in one database. Figure 4-5 explains the hierarchy of OWL definition for airline database that is based on table 4-2.

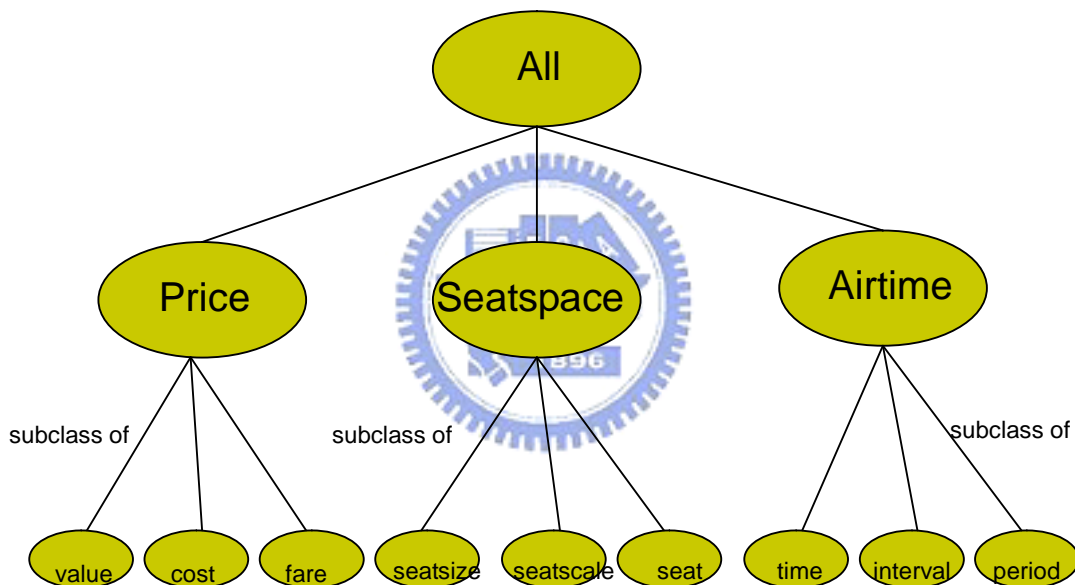


Figure 4-5 OWL Definition

There are many ways to inference the relationship between objects by axioms or properties of classes for OWL. OWL is more useful than the way we use to inference. Here we just use the relationship “SubclassOf” to explain the meaning of different databases.

4.2.3 Classifying Web Service Data

After building OWL document for interpreting the database of airline service, we use the following query in OWLJessKB to tell if the input slot name is the subclass of one class in OWL document. For example, based on the OWL definition, the input slot name “value” is the subclass of “price”, and price is related with three fuzzy terms, “cheap”, “medium” and “expensive”. Therefore we can obtain the three fuzzy values for “cheap” “medium” and ”expensive” as fuzzy terms to describe Web service. Figure 4-6 shows the query with OWL.

```
(defquery query-for-superclasses
  (declare (variables ?x))
  (triple (predicate "http://www.w3.org/2000/01/rdf-schema#subClassOf")
    (subject ?x)
    (object ?y))
)
```

Figure 4-6 OWL Query in OWLJessKB

This query wants to find the super class of subject. For example, in our OWL definition, the “interval” is the subclass of “airtime”, and is also the subclass of “all”. With this query, we can realize that “airtime” and “all” are both the super class of “interval”. Hence, we can know if slot names in heterogeneous databases have the same super class.

By the OWL definition, we classify each slot in database with the related fuzzy terms. With different type of tickets, “Yearly” or “Traveling”, there are different ways to calculate fuzzy value for specific fuzzy term. For one company, we sum up the value of each tuple for one fuzzy term in one database. We finally can conclude the each value of each airline company for different fuzzy terms.

Based on the ticket information from table 4-2 and the OWL definition from figure 4-5, we can obtain the fuzzy values of ten airline companies according to different fuzzy terms shown in table 4-3.

Table 4-3 The Classified Result

	Cheap	Medium	Expensive	Comfortable
Eva Airline	0.24	0.52	0.4	0.62
China Airline	0.17	0.22	0.63	0.33
Cathay Airline	0.0	0.08	0.9	0.99
Japan Asia Airline	0.0	0.0	1.0	0.91
Far Eastern Airline	0.71	0.5	0.02	0.48
Shanghai Airline	0.3	0.49	0.42	0.74
China Eastern Airline	0.37	0.65	0.31	0.46
Trans Asia Airline	0.92	0.62	0.0	0.73
Macau Airline	0.42	0.39	0.5	0.26
Dragon Airline	0.0	0.18	0.8	0.98

All the fuzzy values will be record in TModel of UDDI by different fuzzy terms. By recording the fuzzy value in the description tag of TModel, we can achieve the goal to match Web services with vague requests in UDDI.

4.2.4 Fuzzy Matchmaking

When one vague query like “cheap airline” is addressed, the fuzzy matchmaking will ask UDDI for airline information. After the fuzzy matchmaking obtains the airline information including TModels, it further retrieves the description values of fuzzy terms such as “cheap” or “comfortable” recorded in TModel. With various queries, consumers can obtain the suited Web services.

There are queries about the primitive term like “cheap airline” and the composite term like “comfortable airline”. Also, we address queries related with hedge and PRUF rules, such as “very cheap airline” and “most tickets are cheap”. Otherwise, we have the query “comfortable and cheap airline” to do the basic fuzzy intersection operation. The following tables are the different results for the different vague queries. Those

tables show the suited airline companies with fuzzy values for different queries.

Table 4-4 The Result for “cheap airline” Query

Air Company	Fuzzy value
Far Eastern Airline	0.71
Trans Asia Airline	0.92

Table 4-5 The Result for “very cheap airline” Query

Air Company	Fuzzy value
Far Eastern Airline	0.5
Trans Asia Airline	0.84

Table 4-6 The Result for “comfortable airline” Query

Air Company	Fuzzy value
Eva Airline	0.51
Cathay Airline	0.99
Japan Asia Airline	0.91
Shanghai Airline	0.74
Trans Asia Airline	0.73
Dragon Airline	0.98

Table 4-7 The Result for “most tickets are cheap” Query

Air Company	Fuzzy value
Far Eastern Airline	1.0
Trans Asia Airline	1.0

Table 4-8 The Result for “cheap and comfortable airline” Query

Air Company	Fuzzy value
Trans Asia Airline	0.73

4.2.5 Matchmaking Performance Analysis

In the current matchmaking of UDDI, if user requests with imprecise query like “cheap airline service”, UDDI just returns the result services for “airline service”. The amount of result services may be more than those services user really want. With our

proposed matchmaking, we still can't confirm that our result can satisfy the need of all the requester. But we can provide more precise suited result to requesters.

The measure to investigate the performance between UDDI and our mechanism is the imprecision rate. The imprecision rate means that the rate of the amount of wrong services is compared with the total search services.

$$\text{Imprecision Rate} = |S_{\text{user}} - S_{\text{matched}}| / \text{TS}$$

S_{user} : the amount of services user really wants

S_{matched} : the amount of matched services

TS: the amount of total services

The value of $|R_{\text{user}} - R_{\text{matched}}|$ is bigger than zero when (1) the amount of the matched services is more than the amount of services user wants or (2) the amount of the matched services is less than the amount of services user wants or (3) the matched services are different from the services user wants. TS mean the total services with which we match the suited services user requests.

Imprecision rate helps us to investigate what kind of matchmaking mechanism can provide more precise result to users.

The following tables show the imprecision rates of UDDI and our ontology-based fuzzy matchmaking framework under the different situations. In our framework, we classify tickets into the tickets for traveling and the yearly tickets. In table 4-9, the slot "Price Range" means the different expectation of price for cheapness. In table 4-10, the slots "Seat Space" and "Air Time" mean the expectation of the space of one seat and the expectation time passengers take to arrive to their destinations for comfort. In table 4-11, except for "Price Range", we have the expectation ratio for "Price Range". It is used to estimate the quantification type of PRUF rules, "most tickets are cheap". In table 4-12, we evaluate the imprecision rate for fuzzy intersection operation "and".

Table 4-9 The Imprecision Rate for "cheap airline" Query

Price Range	UDDI	OFMWS
Tra<=15500orYear<=19500	0.3	0.5
Tra<=14500orYear<=18500	0.5	0.3

Tra <=13500orYear<=17500	0.8	0
Tra <=12500orYear<=16500	1.0	0.2
Tra <=11500orYear<=15500	1.0	0.2
Average	0.72	0.24

Table 4-10 The Imprecision Rate for “comfortable airline” Query

Comfortable factors		UDDI	OFMWS
Seat Space	Air Time		
Tra>=2.5orYear>=3	Tra<=2orYear <=2	0.4	0.2
Tra>=2orYear >=2.5	Tra<=2.5orYear<=2.5	0.4	0.2
Tra>=1.5orYear >=2	Tra<=3orYear <=3	0.4	0.4
Average		0.4	0.27

Table 4-11 The Imprecision Rate for “most tickets are cheap” Query

Price Range	ratio	UDDI	OFMWS
Tra<=15500orYear<=19500	>=0.5	0.4	0.4
Tra<=15500orYear<=19500	>=0.8	0.7	0.1
Tra<=14500orYear<=18500	>=0.5	0.7	0.1
Tra<=14500orYear<=18500	>=0.8	0.8	0
Tra <=13500orYear<=17500	>=0.5	0.8	0
Tra <=13500orYear<=17500	>=0.8	1	0.2
Tra <=12500orYear<=16500	>=0.5	0.9	0.1
Tra <=12500orYear<=16500	>=0.8	1	0.2
Tra <=11500orYear<=15500	>=0.5	1	0.2
Tra <=11500orYear<=15500	>=0.8	1	0.2
Average		0.73	0.15

Table 4-12 The Imprecision Rate for “cheap and comfortable airline” Query

Price Range	Seat Space	Air Time	UDDI	OFMWS
Tra<=15500orYear<=19500	Tra>=2.5orYear>=3	Tra<=2orYear <=2	1	0.1
Tra<=15500orYear<=19500	Tra>=2orYear >=2.5	Tra<=2.5orYear<=2.5	0.7	0.2
Tra<=15500orYear<=19500	Tra>=1.5orYear >=2	Tra<=3orYear <=3	0.3	0.6
Tra<=14500orYear<=18500	Tra>=2.5orYear>=3	Tra<=2orYear <=2	1	0.1
Tra<=14500orYear<=18500	Tra>=2orYear >=2.5	Tra<=2.5orYear<=2.5	0.8	0.1
Tra<=14500orYear<=18500	Tra>=1.5orYear >=2	Tra<=3orYear <=3	0.7	0.2
Tra <=13500orYear<=17500	Tra>=2.5orYear>=3	Tra<=2orYear <=2	1	0.1

Tra <=13500orYear<=17500	Tra>=2orYear >=2.5	Tra<=2.5orYear<=2.5	0.9	0
Tra <=13500orYear<=17500	Tra>=1.5orYear >=2	Tra<=3orYear <=3	0.9	0
Tra <=12500orYear<=16500	Tra>=2.5orYear>=3	Tra<=2orYear <=2	1	0.1
Tra <=12500orYear<=16500	Tra>=2orYear >=2.5	Tra<=2.5orYear<=2.5	0.9	0
Tra <=12500orYear<=16500	Tra>=1.5orYear >=2	Tra<=3orYear <=3	0.9	0
Tra <=11500orYear<=15500	Tra>=2.5orYear>=3	Tra<=2orYear <=2	1	0.1
Tra <=11500orYear<=15500	Tra>=2orYear >=2.5	Tra<=2.5orYear<=2.5	1	0.1
Tra <=11500orYear<=15500	Tra>=1.5orYear >=2	Tra<=3orYear <=3	1	0.1
Average			0.87	0.12

With table 4-9, 4-10, 4-11, 4-12, we can observe that the imprecision rate of our proposed framework, ontology-based fuzzy matchmaking for Web services (OFMWS), is lower than that of UDDI. That is, our framework can support vague query and return more correct results to consumers. UDDI sometimes can fulfill the demands of consumers, but it can't support vague query to provide precise results. Hence, our framework always returns more suited results to satisfied user's request. Otherwise, consumers don't need to browse each data of different services to find suited services. We reduce the search space by classifying Web services with fuzzy logic.

In sum, though it has some constraints such as the limitation of the parameter of OWL-S and the limitation of dynamic classifying Web services, our proposed framework explores the hidden dimension of matchmaking of Web services, and allows users to search with vague sentences. Otherwise, it reduces the search space for users and provides more fitting Web services to users.

Chapter 5 Conclusion and Future Works

5.1 Conclusion

This work investigates the issue of matchmaking for Web services that is characterized by multiple dimensions such as software signatures, location of services, syntax and semantics of services, and the underlying data of services. We exploit fuzzy logic in order to classify and to abstractly represent the underlying data of Web services. The main contribution of this work are (1) the identification of a hidden dimension, i.e., the Web services data (2) the abstract representation of Web services data in a fuzzy set theory, and (3) representation of fuzzy sets and rules in OWLJessKB.

With the comparison of UDDI and our proposed framework, our framework can provide the more precise result to consumers than the result of UDDI when consumers deliver vague queries.



5.2 Future Works

In the future, we can modify our framework to support the dynamic classifying and searching. Otherwise, we can include the concept of adaptive ranking in our proposed framework. The flexibility in the design will help increase the flexibility of the framework, allowing matched result to be more adaptive to fulfill users' satisfaction in the business world. It is here in the fluidity of the design that is the big advantage compared to other matchmaking mechanisms.

Reference

- [1] “APPENDIX A – Fuzzy Sets, Logic and Control”
http://vlab.unm.edu/documents/GA_Book_Appendix_A_Go.doc
- [2] B. Limthanmaphon, Y. Zhang, “Web service Composition with Case-Based Reasoning”, 14th Australasian Database Conference (ADC 2003), Adelaide, South Australia, February 2003
- [3] Benjamin N. Grosz, Mahesh D. Grandjean, and Timothy W. Finin, “SweetJess: Inferencing in Situated Courteous RuleML via Translation to and from Jess Rules”, 2 May 2003
- [4] Dan Brickley, Brian McBride, “RDF Vocabulary Description Language 1.0: RDF Schema”, 10 February, 2004
<http://www.w3.org/TR/rdf-schema>
- [5] Deborah L. McGuinness, Frank van Harmelen, “OWL Web Ontology Language Overview”, 10 February 2004
<http://www.w3.org/TR/2004/REC-owl-features-20040210#s1.3>
- [6] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Sheila McIlraith, Srinivasan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, Katia Sycara, “OWL-S: Semantic Markup for Web services”, 22 November 2004
<http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122#3>
- [7] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard, “Web services Architecture,” 11 February 2004
- [8] Eric Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, “Web services Description Language (WSDL) 1.1”, 15 March 2001
- [9] Graham Klyne, Jeremy J. Carroll, “Resource Description Framework (RDF): Concepts and Abstract Syntax”, W3C, 10 February 2004
- [10] <http://www.w3.org/DesignIssues/diagrams/sw-stack-2002.png>
- [11] H-J Zimmermann, “Fuzzy Set Theory, and its Applications,” Kluwer Academic Publishers, 1994
- [12] John Yen, Reza Langari, “Fuzzy Logic: Intelligence, Control, and Information”, Prehall
- [13] Joe Kopena, William C. Regli*, “DAMLJessKB: A Tool for Reasoning with the Semantic Web”, October 28, 2002
- [14] Joseph B. Kopena, Christopher D. Cera, William C. Regli*, “Engineering Design Knowledge Management for Conceptual Design”, 2004
- [15] K. Sycara, S. Widoff “LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace”, Journal of AAMAS, May 2002,

Kluwer Academic Publishers

- [16] Ken Shall, "Mark Colan (IBM): SOAP + UDDI + WSDL = Web services", 11 December 2000
<http://wdvl.internet.com/Authoring/Languages/XML/Conferences/XML2000/colan.html>
- [17] Keisuke Kibakura, "From the XML Front Line-UDDI", August 30, 2002
http://www.fujitsu.com/global/services/solutions/xml/frontline/XML_uddi4.html
- [18] Kuo-Ming Chao, Muhammad Younas, Chi-Chun Lo, Tao-Hsin Tan: Fuzzy Matchmaking for Web services. The 19th IEEE International Conference on Advanced Information Networking and Applications (AINA 2005), 28-30 March 2005, Taipei, Taiwan. IEEE Computer Society 2005: 721-726
- [19] Michael C. Daconta, Leo J. Obrst and Kevin T. Smith, "The Semantic Web: A Guide to the Future of XML, Web services, and Knowledge Management", 2003
- [20] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara, "Importing the Semantic Web in UDDI", 2002
- [21] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe, "A Practical Guide to Building OWL Ontologies : Using the Protégé-OWL Plugin and CO-ODE Tools Edition 1.0", 27 August 2004
- [22] "MCI UDDI Registry", 2002
http://uddi.mci.com/about_uddi.jsp
- [23] Natalya F. Noy and Deborah L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology", March 2001
- [24] Naveen Srinivasan, Massimo Paolucci, Katia Sycara, "Adding OWL-S to UDDI, Implementation and throughput", (*SWSWPC 2004*) 6-9, 2004
- [25] "OWL-S 1.0 Release: Examples"
<http://www.daml.org/services/owl-s/1.0/examples.html>
- [26] "Publish and find UDDI TModels with JAXR and WSDL"
<http://www.javaworld.com/javaworld/jw-12-2002/jw-1213-webservices.html>
- [27] R.A.Orchard, "NRC FuzzyJ Toolkit for the Java Platform User's Guide", May 23, 2003
- [28] Scientific American, "The Semantic Web", 17 May 2001
- [29] Steve Graham, Simeon Simeonov, Toufic Boubez, Doug Davis, Glen Daniels, Yuichi Nakamura, Ryo Neyama, "Building Web services with Java: Making Sense of XML, SOAP, WSDL, and UDDI", SAMS, 12 December 2001
- [30] Satya Komatineni, "Understandin UDDI and JAXR", 27 Feb 2002
<http://www.onjava.com/pub/a/onjava/2002/02/27/uddi.html?page=1>
- [31] "The Semantic Web: An Introduction", 2001
<http://infomesh.net/2001/swintro/#simpleData>

- [32] Tom Gruber, “What is an Ontology?”
<http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>
- [33] Tim Berners-Lee, “Information Management: A Proposal”, CERN, March 1989
- [34] Ernest J. Friedman-Hill, “Jess, The Rule Engine for the Java Platform”, 23
March 2005
- [35] 蘇民, “員工與顧客滿意度影響國內航空服務業經營績效之研究”, May 1998

