



Durham E-Theses

Semantic Service Description Framework for Efficient Service Discovery and Composition

DU, XIAOFENG

How to cite:

DU, XIAOFENG (2009) *Semantic Service Description Framework for Efficient Service Discovery and Composition*, Durham theses, Durham University. Available at Durham E-Theses Online:
<http://etheses.dur.ac.uk/111/>

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a [link](#) is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the [full Durham E-Theses policy](#) for further details.

Academic Support Office, Durham University, University Office, Old Elvet, Durham DH1 3HP
e-mail: e-theses.admin@dur.ac.uk Tel: +44 0191 334 6107
<http://etheses.dur.ac.uk>

Semantic Service Description Framework for Efficient Service Discovery and Composition

Xiaofeng Du

Supervisors: Dr. **William Wei Song** and
Prof. **Malcolm Munro**

A thesis presented for the degree of
Doctor of Philosophy

Department of Computer Sciences
University of Durham
United Kingdom



July 2009

Dedicated to

My Mother and Father,
Wife and Daughter

Semantic Service Description Framework for Efficient Service Discovery and Composition

Xiaofeng Du

Submitted for the degree of Doctor of Philosophy
July 2009

Abstract

Web services have been widely adopted as a new distributed system technology by industries in the areas of, enterprise application integration, business process management, and virtual organisation. However, lack of semantics in current Web services standards has been a major barrier in the further improvement of service discovery and composition. For the last decade, Semantic Web Services have become an important research topic to enrich the semantics of Web services. The key objective of Semantic Web Services is to achieve automatic/semi-automatic Web service discovery, invocation, and composition. There are several existing semantic Web service description frameworks, such as, OWL-S, WSDL-S, and WSMF. However, existing frameworks have several issues, such as insufficient service usage context information, precisely specified requirements needed to locate services, lacking information about inter-service relationships, and insufficient/incomplete information handling, make the process of service discovery and composition not as efficient as it should be.

To address these problems, a context-based semantic service description framework is proposed in this thesis. This framework focuses on not only capabilities of Web services, but also the usage context information of Web services, which we consider as an important factor in efficient service discovery and composition. Based on this framework, an enhanced service discovery mechanism is proposed. It gives service users more flexibility to search for services in more natural ways rather than only by technical specifications of required services. The service discovery mechanism also

demonstrates how the features provided by the framework can facilitate the service discovery and composition processes. Together with the framework, a transformation method is provided to transform existing service descriptions into the new framework based descriptions.

The framework is evaluated through a scenario based analysis in comparison with OWL-S and a prototype based performance evaluation in terms of query response time, the precision and recall ratio, and system scalability.

Keywords: Web Services, Semantic Web Services, Service Discovery, Service Composition, Service Composition Patterns, SOA, Service Context, Conceptual Graphs.

Declaration

The work in this thesis is based on the research work carried out at the Department of Computer Sciences, Durham University, U.K. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is entirely the author's work unless referenced to the contrary in the text.

This research has been documented, in part, within the following publications:

Du, X., Song, W., and Munro, M. (2006) Using Common Process Patterns for Semantic Web Services Composition, in Proc. of 15th International Conference on Information System Development (ISD2006), Budapest, Hungary, Aug. 31 - Sept. 2, 2006.

Du, X., Song, W., and Munro, M. (2006) Service Composition in the Context of Grid, in Proc. of UK e-Science Program All Hand Meeting (AHM2006), Nottingham, UK, Sept. 18-21, 2006.

Du, X., Song, W., and Munro, M. (2006) Semantics Recognition in Service Composition Using Conceptual Graph, in the Proc. of International workshop on Semantics in Virtual Organisations and Web services (SVO&WS), held in conjunction with the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI-06) , Hong Kong, China, Dec. 18-22, 2006.

Song, W. and Du, X. (2006) A Case Study for Semantic Description and Discovery of Grid Services, Invited paper at Inaugural Workshop on Dependability in Service-Oriented Grids (WODSOG'06) , held in conjunction with the IEEE Symposium on Reliable Distributed Systems (SRDS 2006) , University of Leeds, UK, Oct. 1, 2006.

Du, X., Song, W., and Munro, M. (2007) Semantic Service Description Framework for Addressing Imprecise Service Requirements, in proceedings of the 16th International Conference on Information Systems Development, Galway, Ireland, Sept. 2007.

Du, X., Song, W., and Zhang, M. (2007) A Context-based Framework and Method for Learning Object Description and Search, in Proc. of 6th International Conference on Web-based Learning (ICWL 2007), LNCS Vol. 4823, Springer, Edinburgh, United Kingdom, Aug. 15-17, 2007.

Du, X., Song, W., and Munro, M. (2008) An Innovative Approach for Service Description and Discovery in the Context of Software as a Service, accepted by the 1st IEEE International Workshop on Barriers towards Internet-Driven Information Services (BINDIS2008), held in conjunction with IEEE COMPSAC 2008, Turku, Finland.

Du, X., Song, W., and Munro, M. (2008) A Method for Transforming Existing Web service Descriptions into an Enhanced Semantic Web service Framework, in Proc. of 17th International Conference on Information System Development (ISD2008), Paphos, Cyprus August 25-27, 2008.

Song, W., Du, X., and Munro, M. (2009) A Concept Graph Approach to Semantic Similarity Computation Method for e-Service Discovery, to appear in International Journal of Knowledge Engineering and Data Mining, Inderscience publishers.

Du, X., Song, W., and Munro, M. (2009) CbSSDF and OWL-S, A Scenario based Solution Analysis and Comparison, to appear in Proc. of 18th International Conference on Information System Development (ISD2009), Nanchang, China, September 16-19, 2009.

Acknowledgements

First and foremost, I would like to thank my primary supervisor Dr. William Wei Song and my secondary supervisor Prof. Malcolm Munro for their full support during my studies. I am very fortunate to have had the opportunity to work under their supervision. They instilled a thirst for excellence in me, taught me how to do scholarly research, and helped me think creatively and independently. Their guidance and patience during my Ph.D. research are greatly appreciated.

I would like to thank Prof. David Budgen and Dr. Shengchao Qin for their helpful suggestions during the first year thesis proposal viva, which have greatly benefited my later studies.

I would like to thank all the friends I have met during my Ph.D. study in Durham for their continuous support and encouragement. I would like to give the special thanks to Dr. Yonghong Xiang, Mrs Yunli Liu, Mrs. Barbara Froner, and Dr. Niko Galiatsatos for their support during the time of my student placement work in British Telecom.

I would like to thank all the colleagues I have worked with in British Telecom during my student placement work, especially Mr. Nader Azarmi, Dr. Ben Azvine, Dr. Detlef Nauck, Dr. Zhan Cui, Dr. Basim Majeed, and Dr. Yang Li, for their supervision and the great inspiration on my Ph.D. research.

Finally, I would like to thank my family and loved ones. Thank my parents for their endless love and support along the path of my studies. I have been a long way from home for a long time and could not make step forward without their understanding and encouragement. I would like to express my deepest gratitude to my lovely wife Motoko Kino. Without her sincere encouragement, great patience and mostly unconditional love, I would not have been able to get through this long journey.

Contents

ABSTRACT.....	II
DECLARATION	IV
ACKNOWLEDGEMENTS	VI
LIST OF FIGURES	XII
LIST OF TABLES	XIV
LIST OF DEFINITIONS	XV
CHAPTER 1: INTRODUCTION	1
1.1 NEW SOA PARADIGM	2
1.2 PROBLEM AREAS.....	5
1.2.1 <i>Service Description Problem</i>	5
1.2.2 <i>Service Discovery Problem</i>	6
1.2.3 <i>Service Composition Problem</i>	8
1.3 CURRENT SOLUTIONS.....	9
1.3.1 <i>Semantic Web Services</i>	10
1.3.2 <i>Web Service Discovery</i>	10
1.3.3 <i>Web Service Composition</i>	11
1.3.4 <i>Remaining Issues</i>	12
1.4 PROPOSED METHOD.....	14
1.4.1 <i>Integrating Context into Service Description</i>	14
1.4.2 <i>Integrating Adequate Semantics into Service Description</i>	15
1.4.3 <i>Addressing Incomplete Information</i>	16
1.4.4 <i>A Flexible Service Discovery Mechanism</i>	16
1.5 CONTRIBUTIONS.....	17
1.6 CRITERIA FOR SUCCESS.....	18

1.7 THESIS OUTLINE	18
CHAPTER 2: CONTEXT OF PROBLEM AREA ANALYSIS AND RESEARCH .	20
2.1 OVERVIEW	21
2.2 WEB SERVICES.....	21
2.2.1 <i>Technology overview</i>	22
2.2.2 <i>Core Standards</i>	23
2.2.3 <i>Drawback of Current Web services Standards</i>	29
2.3 KNOWLEDGE REPRESENTATION	30
2.3.1 <i>Ontology</i>	30
2.3.2 <i>Description Logics</i>	32
2.3.3 <i>Conceptual Graphs</i>	35
2.3.4 <i>Non-monotonic Reasoning – Defeasible Logic</i>	39
2.4 SEMANTIC WEB.....	41
2.4.1 <i>RDF and OWL</i>	42
2.5 SEMANTIC WEB SERVICES	45
2.5.1 <i>OWL-S</i>	46
2.5.2 <i>WSMF and WSMO</i>	48
2.5.3 <i>WSDL-S</i>	51
2.6 SERVICE CONTEXT	52
2.6.1 <i>What is Context?</i>	53
2.6.2 <i>Service Context</i>	54
2.6.3 <i>Significance and Deficits</i>	56
2.7 SERVICE COMPOSITION	57
2.7.1 <i>Manual Service Composition – BPEL4WS</i>	57
2.7.2 <i>Automatic or Semi-Automatic Planning based Service Composition</i>	59
2.8 SEMANTIC SIMILARITY CALCULATION.....	61

2.8.1 <i>Ontology based Methods</i>	61
2.8.2 <i>Vector based Methods</i>	63
2.9 SUMMARY	64
CHAPTER 3: SERVICE USAGE CONTEXT.....	66
3.1 OVERVIEW	67
3.2. GLOSSARY	68
3.3 CONTEXT AND SCHEMATA	69
3.4 SERVICE USAGE CONTEXT	72
3.4.1 <i>Conceptual Level Service Usage Context – T-Context</i>	72
3.4.2 <i>Instance Level Service Usage Context – A-Context</i>	75
3.5 SUMMARY	79
CHAPTER 4: CONTEXT-BASED SEMANTIC SERVICE DESCRIPTION FRAMEWORK.....	80
4.1 OVERVIEW	81
4.2 ATOMIC SERVICE AND COMPOSITE SERVICE.....	83
4.3 CONTEXT-BASED SEMANTIC SERVICE DESCRIPTION FRAMEWORK.....	86
4.3.1 <i>Service Conceptual Graphs</i>	87
4.3.2 <i>Semantic Service Description Model</i>	91
4.3.3 <i>Non-Monotonic Rules</i>	94
4.4 TRANSFORMATION METHOD	97
4.4.1 <i>Step One: Ontology based service classification</i>	98
4.4.2 <i>Step Two: CUPs generation</i>	99
4.4.3 <i>Step Three: S-CGs generation</i>	100
4.4.4 <i>An Example</i>	101
4.4.5 <i>A Note on Information Loss During Transformation</i>	103
4.5 SUMMARY	104

CHAPTER 5: TWO-STEP SERVICE DISCOVERY MECHANISM	106
5.1 OVERVIEW	107
5.2 TWO-STEP SERVICE DISCOVERY MECHANISM	107
5.2.1 Step One: S-CG based Service Retrieval	108
5.2.2 Step Two: SSDM based Service Composition and Ranking	113
5.3 SUMMARY	118
CHAPTER 6: IMPLEMENTATION	119
6.1 OVERVIEW	120
6.2 IMPLEMENTED FEATURES OF CBSSDF	120
6.3 SYSTEM DESIGN AND ARCHITECTURE	122
6.3.1 User Interface	122
6.3.2 System Architecture	126
6.3.3 Implementation Design	127
6.4 IMPLEMENTATION TECHNOLOGIES	129
6.5 SUMMARY	130
CHAPTER 7: EVALUATION	132
7.1 OVERVIEW	133
7.2 EVALUATION STRATEGY	134
7.3 SCENARIO BASED COMPARISON WITH OWL-S	135
7.3.1 Task 1: Locating an Existing Atomic Service	138
7.3.1.1 Solution Comparison	138
7.3.1.2 Summary	139
7.3.2 Task 2: Locating an Existing Composite Service	139
7.3.2.1 Solution Comparison	139
7.3.2.2 Summary	140
7.3.3 Task 3: Dynamically Constructing Composite Service	141

7.3.3.1 Solution Comparison	141
7.3.3.2 Summary	142
7.3.4 <i>Discussion of the Scenario based Comparison</i>	143
7.4 PROTOTYPE BASED PERFORMANCE STUDY	144
7.4.1 <i>Experiment Environment</i>	145
7.4.2 <i>Design of the Experiment</i>	145
7.4.2.1 Analytical Model	145
7.4.2.2 The Independent and Dependent Variables and the Treatment	147
7.4.3 <i>Experiment Results</i>	147
7.4.4 <i>Limitations of the Experiments and Threats to Validity</i>	151
7.5 TRANSFORMATION METHOD APPLICABILITY EVALUATION	153
7.6 SUMMARY	154
CHAPTER 8: CONCLUSION AND FUTURE WORK	158
8.1 OVERVIEW	159
8.2 SUMMARY AND CONTRIBUTIONS	159
8.3 REMAINING PROBLEMS	163
8.4 FUTURE RESEARCH AND DEVELOPMENT DIRECTIONS	164
8.4.1 <i>Service Level Agreement Enhanced Service Registry</i>	164
8.4.2 <i>Business Patterns in SOA</i>	165
8.4.3 <i>Web service Monitoring</i>	166
REFERENCES.....	167

List of Figures

Figure 2.1. Web services Architecture.....	23
Figure 2.2. The technology stack of Web services	24
Figure 2.3. SOAP message structure	25
Figure 2.4. Registry interaction enabled by UDDI 3.0	29
Figure 2.5. An example of inheritance network.....	33
Figure 2.6. A simple conceptual graph	35
Figure 2.7. A layered approach to the semantic web	42
Figure 2.8. A RDF graph example.....	42
Figure 2.9. Top level of the service ontology	46
Figure 2.10. A SOAP message with context information.....	54
Figure 2.11. Context-aware personalised Web services	55
Figure 2.12. The concept similarity measure	61
Figure 3.1. A service ontology example.	68
Figure 3.2. A domain ontology example.....	69
Figure 3.3. A conceptual usage scenario of a service concept – “Money Transfer”. ..	74
Figure 3.4. An instance usage scenario of a service – <i>service</i> ₁	78
Figure. 4.1. An example of an S-CG.....	89
Figure. 4.2. A graphical illustration of SSDM.....	94
Figure 4.3. The generated instance services graph.	102
Figure 4.4. The generated service concepts graph.	103
Figure 4.5. The generated S-CG.	103
Figure 5.1. A graphical representation of part of the vector space <i>V</i>	115
Figure 6.1. The user interface of ServiceComp.	124
Figure 6.2. The first step query interface of ServiceComp.....	124

Figure 6.3. The second step searching interface of ServiceComp	125
Figure 6.4. The system architecture of <i>ServiceComp</i>	126
Figure 6.5. Service repository ER diagram.....	127
Figure 6.6. ServiceComp class diagram.....	128
Figure 7.1. Precision-Recall curves.	149
Figure 7.2. Query response time for performing on a centralised service repository.	150
Figure 7.3. Query response time for performing on both the centralised and decentralised service repositories after increasing the number of services.	151
Figure 7.4. Percentage of acquired information from different service descriptions.	154

List of Tables

Table 2.1. Mapping between OWL and DL.....	45
Table 3.1. Service instances and their basic attributes.....	77
Table 4.1. Acquired information from WSDL and ontology based classification....	102
Table 6.1. The description of tool bar buttons.	122
Table 7.1. Two examples of the CbSSDF based service description.....	137
Table 7.2. The comparison of CbSSDF and OWL-S based solutions for task 1.	138
Table 7.3. The comparison of CbSSDF and OWL-S based solutions for task 2.	140
Table 7.4. The comparison of CbSSDF and OWL-S based solutions for task 3.	141
Table 7.5. Precision-recall table for CbSSDF and OWL-S solutions.	148
Table 7.6. System performance evaluation result – data samples.....	149
Table 7.7. System scalability evaluation result – data samples.	150
Table 7.8. Percentages of the required information in CbSSDF obtained from different service description frameworks.....	153

List of Definitions

Definition 2.1	35
Definition 2.2	36
Definition 2.3	38
Definition 2.4	38
Definition 2.5	40
Definition 3.1	73
Definition 3.2	74
Definition 3.3	76
Definition 3.4	78
Definition 3.5	78
Definition 4.1	85
Definition 4.2	85
Definition 4.3	89
Definition 4.4	91
Definition 4.5	92
Definition 4.6	93
Definition 4.7	99
Definition 5.1	115

Chapter 1: Introduction

In this chapter, we will give an overview of the research background, existing problems, and current solutions of Web services and Semantic Web Services. Based on the remaining problems, we give a compressed view of our solution on service description, discovery, and composition. At the end of this chapter, we will summarise major contributions committed in this thesis.

1.1 New SOA Paradigm

From the time the first networked computer system ARPANET [Abbate, 1999] appeared until now, computer network technologies have developed rapidly and have been adopted widely in all disciplines. Modern network based computing started with the emergence of the distributed computing paradigm [Coulouris et al., 2001]. Distributed computing is all about communications and resource sharing. The success of distributed systems created a new era of network computing, i.e. the emergence of Internet and World Wide Web [Berners-Lee, 1991]. New technologies always bring new challenges. The biggest challenge of distributed computing is how to solve the heterogeneity problem. The Internet enables users to access services and resources through a heterogeneous collection of computing equipments and networks. The heterogeneity over the Internet includes [Coulouris et al., 2001]:

- Networks
- Computer hardware
- Operating systems
- Programming languages

A distributed system has to overcome the heterogeneity issue in order to establish effective communications. Traditional distributed systems have proposed a collection of technologies, such as CORBA, DCOM, and Java RMI, to tackle the issue. These technologies try to provide components that can hide the local heterogeneity with common interfaces for communications. However, these technologies themselves have heterogeneous problems because some of them are platform dependent and some of them are programming languages dependent. For example, if two applications communicate through DCOM technology, they must be both hosted on the Windows operating system and programmed in C, C++, or C# programming languages. If two applications communicate through the Java RMI technology, they must be both programmed in Java. Moreover, these distributed system technologies cannot easily talk to each other without extra system engineering effort. Another issue on traditional distributed system technologies is that binary message based communication and particular communication protocols require specific ports to be opened on the firewall, which brings security risks. Due to these disadvantages, enterprise applications based on traditional distributed system technologies are tightly coupled and federated. If a

company wants to change their suppliers or business partners, more often than not, their applications have to be reengineered. This heavily reduces the flexibility of business to business (B2B) communication and enterprise applications integration (EAI) and consequently, reduces the corresponding speed of an enterprise to new marketing demands. The fundamental problem of traditional distributed systems as discussed above is lack of standards for B2B integration and B2B automation. In order to solve the issues, Web services technology was born.

“Web services are a new breed of web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions that can be anything from simple requests to complicated business processes” [IBM, 2006]. Web services technology is an evolutionary technology based on existing technologies, such as CORBA, DCOM, and Java RMI, rather than a new invention [Sheth & Miller, 2003]. The key point for the success of Web services is the employment of existing standards as fundamental building blocks. A set of XML based standards, i.e. SOAP (Simple Object Access Protocol) [SOAP, 2007], WSDL (Web services Description Language) [WSDL, 2007], and UDDI (Universal Description, Discovery and Integration) [UDDI, 2004], are used to encapsulate data, describe the Web services interfaces, and publish Web services on the web. Web service communication is established upon existing TCP/IP standards, such as HTTP, HTTPS, SMTP, and FTP [SOAP, 2007]. Another key characteristic of Web services is their self-contained and loosely-coupled nature, which makes each Web service autonomic. This means that each Web service is responsible for its own application domain and the business logic encapsulated in a Web service does not need to comply with any other operating systems or technologies. These new features enable Web services to be highly reusable components that can not only solve the application communication and integration issues, but also act as building blocks to rapidly construct new applications, i.e. service composition. The highly reusable, self-contained, and loosely-coupled features of Web services have started a new enterprise application design and development paradigm, the Service Oriented Architecture (SOA). *“A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components – services, that can be reused and combined to address changing business priorities”* [Bieberstein et al., 2005]. In SOA, a service has

been abstracted to “the application of specialised competences (knowledge and skills), through deeds, processes, and performances for the benefit of another entity or the entity itself” [Lusch & Vargo, 2006]. A service in SOA does not necessarily mean a Web service. However, the Web services technology is a well-known implementation of services in SOA.

The reason for SOA being so sanctified and widely adopted by many large enterprises, such as IBM, Oracle, and British Telecom, is that it tackles the main challenges that an IT executive is currently facing. The first challenge is the heterogeneity of the legacy systems, i.e. the systems from multiple vendors and different partners and suppliers. An IT executive cannot avoid integrating these heterogeneous systems. The second challenge is the pace of change. The speed of change in Global e-commerce is accelerating. How IT executives steer their enterprise applications to promptly satisfy fast changing market trends is a crucial issue for an enterprise’s survival. However, by adopting SOA, the problems behind these two challenges can be solved. Each component of an enterprise application can be wrapped as a service and therefore a repository of services can be formed. They can be moved around from one application to another, replaced, and modified without affecting other parts of the application. In fact, SOA has turned traditional application developers into “Lego Brick” builders. A service can be used whenever and wherever it is needed by either an enterprise or the enterprise’s partners and customers. The benefits of SOA [Bieberstein et al., 2005] that an enterprise can gain are summarised as follows:

- It saves money, time, and effort over the long term through reuse of “components” because of the flexibility of SOA.
- It eliminates frustrations with IT through flexible solutions and shorter lead time to deployment.
- It justifies IT investments more clearly through the closer association of IT to business services.
- It provides to business executives with a clear understanding of what IT does and its value.

- It allows the creation of and changes to services incrementally rather than leaving a guesstimate of the development costs, thereby eliminating the classic IT 6-6 answer: “The project will take 6 months and cost 6 figures.”
- It provides to a business and competitive differentiator with direct rationalisation and relation to how that competitive advantage is implemented in IT.

However, new opportunities always bring new problems and drive new solutions. SOA has painted a big picture of the future of enterprise applications. Although the Web services technology is a realisation of SOA, it cannot fully achieve what SOA requires due to limitations of the technology. In the next section, we will give a brief discussion on key problem areas that the Web services technology currently encounters, such as service description, composition, and discovery.

1.2 Problem Areas

The concept and technology of Web services significantly improve the enterprise application communication and integration. However, to fully satisfy business requirements and achieve SOA, the current technology needs to be enhanced. In this section, we focus on three major problems of Web services technology that limit the realisation of SOA.

1.2.1 Service Description Problem

Currently, Web service description is based on WSDL, which is an XML based, low level syntactical, and developer oriented service description language. A WSDL document outlines input and output data types of a service, the structure of messages and protocols for communicating with the service, and the URI to locate the service. WSDL supports parsing a message from a web service, verifying whether it is in the expected format, and extracting the information contained in the message. However, WSDL's support is limited to understanding the structure of the message, not the content/semantics of the message and the capability of the service. The consequence of the limitation is that each time when invoking a service, human intervention is required for interpretation of the semantics of the message content and the capability of the service in order to make a correct and appropriate use of the service. Lack of

semantics in describing service capability is a major drawback of current Web service descriptions [Paolucci et al., 2003]. From this perspective, Web services are not actually self-contained because how to use them is relying on human interpretation. Ideally, a Web service should be completely autonomous so that it can be discovered automatically by software agents or other services. To be completely autonomous, a better service description solution is required.

Apart from lack of semantics, another limitation of WSDL is that it does not address abstraction and granularity. The service detail addressed in WSDL is technical information. However, this kind of information never appears in an enterprise service users' service query, as they have little knowledge of the technical detail of the Web services technology. Very likely the service query proposed from an enterprise service user is a general business requirement or a business task description. Here we can see an obvious mismatch between the abstract business requirement and the technical level service description provided in WSDL.

1.2.2 Service Discovery Problem

The current industrial standard for Web service discovery is the Universal, Description, Discovery and Integration (UDDI) [UDDI, 2004]. UDDI provides a set of facilities for service advertising, browsing, and search. The information provided in a UDDI description includes the service provider information, a natural language based service description, and the service binding information etc. Additionally, UDDI description can refer to a description component called T-Model, which is a set of open-bounded service attributes that can represent any type of information that service providers think is relevant to their published services. The T-Model can also classify Web services within a given taxonomy, for example, the North American Industry Classification System¹ (NAICS) and the United Nations Standard Products & Services Code² (UNSPSC). Although UDDI provides many kinds of information for service discovery, its service discovery and advertising mechanism have crucial limitations. The first limitation is the keyword based search. The keyword based service search generates imprecise results with large amount of irrelevant information, even though UDDI has many features to advertise services and the T-Model does

¹ <http://www.census.gov/epcd/www/naics.html>

² <http://www.unspsc.org/Defaults.asp>

allow integration of standard taxonomies. The second limitation is lack of machine understandable semantics. UDDI is designed for developers to search for services manually and therefore, the information provided on UDDI is human readable only. However, as the number of services is getting large, manually searching services become more and more time consuming and inefficient. The demand for automation is increasing, which requires a service registry supporting automatic service discovery. The current UDDI registry does not support automatic service discovery, hence restricts the full realisation of SOA.

If we assume that the semantics have been integrated into both the service description and service registry, matching a service query with service descriptions is still not a trivial task. The problems come from two aspects. The first aspect is about choosing or creating a suitable semantic similarity calculation method. The semantic similarity calculation between a service query and a service description is not as simple as calculating the semantic distance between two concepts in an ontology because the semantics of a service are very complex. When we compute the semantic similarity of services, we have to consider the functional semantics, the non-functional semantics, the data semantics, and the execution semantics of a service [Cardoso & Sheth, 2006]. The complexity of the service semantics requires a dedicated method to precisely and effectively compute the semantic similarity. The second aspect is about building up a mapping between service queries and service descriptions. In real business scenarios, service users usually propose general service queries to describe what they need because they are not aware of the technical detail of services [Du et al., 2007]. However, it is difficult to find a direct mapping from this kind of query to the service capability attributes addressed in a service description based on the current Web service description standards. Therefore, the semantic similarity calculation cannot be carried out straightforwardly using the existing methods [Berry et al., 1999] [Wu & Palmer, 1994] [Tous & Delgado, 2006]. Consequently, an extra step may be needed to gather sufficient information from the service requester in order to perform semantic matching.

1.2.3 Service Composition Problem

No matter how many functionalities the currently published Web services provide, there are always some requirements that cannot be fully satisfied. If a new service is created for each new requirement, it is too costly and a waste of existing resources. It also breaches the principle of Web services development. A sound solution is to provide new services by composing existing services. The self-contained and loosely-coupled features give Web services the ability to be composed to form a new service with new capabilities. Web service composition is important in business process management. A complex business interaction always involves a series of high level business functionalities that contain basic business activities. If we consider each of the basic business activities as a service, the process of creating a composite service is actually a process of creating a business process for fulfilling certain business requests. Through service composition, an enterprise can create new value-added business services from their existing resources with less delivery time at a lower cost. There are many industrial standards that are used to describe how services can be composed to form a composite service, such as WS-BPEL (Web Services Business Process Execution Language) [WS-BPEL, 2007], WSCI (Web Service Choreography Interface) [WSCI, 2002], and WSCL (Web Services Conversation Language) [WSCL, 2002]. These languages (or standards) propose the solutions for manually constructing composite services. There are also some semantic approaches to automatically generate composite services based on the semantic annotation in service description.

Most of the semantic based automatic service composition approaches are using AI planning techniques [Sirin et al., 2004] [Zhang et al., 2004]. A goal of a business interaction can be decomposed into sub-tasks and each sub-task can be further decomposed until each sub-task can be achieved by an existing service. After all the sub-tasks are located related services, a service flow (a plan or a composite service) can be formed. However, the AI planning based approach has exposed some problems of service composition that need to be addressed in future research.

- The situation a planning algorithm faces in a service composition scenario is far more complicated than a traditional AI planning scenario. The reason is that in a traditional AI planning scenario, the action repository is closed and static, whereas, in a service composition scenario, the action repository is open

and dynamic. The size of the action repository and the availability of each action are uncertain.

- A Web service is different from an action in AI planning. It has interrelationships with other services. Therefore, considering each service as an individual action may lower the efficiency of the planning algorithm.
- From the industrial perspective, an explicit goal of a composite service is difficult to identify [Srivastava & Koe, 2003].

Another issue of Web service composition is handling incomplete information [Lu et al., 2006]. The web is a highly dynamic environment. Under one situation, two services may be composable; whereas under the other situation the same two services may not be composable. Most of the information of service execution conditions on the web is incomplete, i.e. when more information becomes available, the situation may change. The incompleteness is especially prominent in business domain. The incompleteness in business domain is not only from the web, but also from the business itself. For example, in designing business applications or services, it is almost impossible to get complete information from customers or business partners and therefore some assumptions have to be made. Business rules and policies can also bring incompleteness. If a service composition approach cannot handle incomplete information, the result is more likely impracticable, especially in a business domain. However, the current Web service composition approaches assume that the available information during service composition is static and complete [Peer, 2005].

1.3 Current Solutions

In the previous section, we present an overview of the problem area of Web services. In this section, we outline the current solutions to the problems discussed above. These solutions are mainly focusing on providing a comprehensive service description framework in order to support automatic or semi-automatic service discovery, invocation, and composition. In Chapter 2, we will give a detailed survey of the literature.

1.3.1 Semantic Web Services

The idea of the semantic web is to extend the current web, in which the information on the web is given a well-defined meaning through semantic annotation so that both human and computer can interpret it and make use of it [Berners-Lee et al., 2001]. One of the important components of the semantic web is ontology. An ontology [Uschold & Grüninger, 1996] is a shared conceptualisation or a common model to formally define the meaning of concepts and their relationships. Through the ontology, concepts can be interpreted by computer programs. An important application of semantic web is Semantic Web Services. As discussed previously, a drawback of the current Web services standards is lack of semantics so that there is no way for a computer program to identify a required service without human intervention. Semantic Web Services technology extracts the data and capability semantics of a Web service, which are essential for making use of the service. This is done through annotating Web services with concepts from a common ontology.

There are two ways of creating semantic annotated Web services. One way is to create an independent semantic Web service description framework and link it to the current Web services standards. The leading research efforts are OWL-S [Martin et al., 2004] and WSMO [Fensel et al., 2007]. The other way is to add semantic annotations into the current Web services standards. The major research work in this way is WSDL-S [Akkiraju et al., 2005]. All of these Semantic Web Services research efforts try to overcome the drawback of lack of semantics of the current Web services standards and support automatic Web service discovery, invocation, and composition. However, semantically annotating functional components, i.e. Web services, is much more complicated than annotating static web information. Therefore, further research efforts are still required in this area.

1.3.2 Web service Discovery

The traditional way of discovering Web services is through the use of a UDDI service registry. The discovery is performed by the keyword based searching and manual selection. As discussed previously, the keyword based searching and manual selection is not suitable anymore for dealing with large amount of Web services and the increasing demand for Web service automation [Paolucci et al., 2003]. The current

solution is to integrate semantics into Web service descriptions and the UDDI registry in order to improve the efficiency of service discovery and support automatic service discovery. Paolucci et al. [Paolucci et al, 2002] propose an algorithm that can match a service request with an OWL-S based service advertisement and rank the matching result according to the semantic similarity between a service request and semantic service descriptions. Their algorithm is based on calculating minimal distance between two concepts in an ontology. In order to improve the keyword based service discovery on UDDI, they also augment the UDDI registry with an extra semantic layer in order to perform semantic based service capability matching. Web services described using OWL-S are also published on the enhanced UDDI registry so that they also can be retrieved by keyword search.

1.3.3 Web service Composition

There are two main streams of Web service composition approaches. One stream is manual service composition based on the current Web services standards. A representational example is WS-BPEL. It is an industrial standard and has been adopted by many large software companies, such as Oracle, Progress Software, and Microsoft. WS-BPEL is based on WSDL and provides a rich syntax for constructing abstract business processes. A WS-BPEL process can be created by either directly writing WS-BPEL code or through business process design and management tools. The other stream is semantic Web service composition. A representational example is OWL-S. OWL-S provides a machine readable semantic description of Web service capabilities and therefore, Web services can be discovered and composed with much less human intervention. An OWL-S description consists of three components called service profile, service model, and service grounding. The service profile provides a semantic description of a service, service model defines atomic and composite process of the service, and the service grounding provides the linking between the semantic description of a service and WSDL. There are also other Web service composition approaches, such as service composition as planning [Wu et al., 2003].

1.3.4 Remaining Issues

In the previous sections, we have given an overview of the problem area of Web services and the current solutions. In order to further investigate what the remaining issues are, we conduct a simple experiment to help us to analyse the underlying problems and the possible reasons. The experiment is described as follows:

- **The aim of the experiment:** is to explore the problems of service discovery.
- **The participant of the experiment:** We invite two groups of postgraduate students, one group from the computer science department with professional computer knowledge and the other group from other departments with general computer operating skills. There are 15 students in each group.
- **The tools used in the experiment:** We provide two types of service discovery interface. One has a text field with a search button and the other one has multiple text fields for gathering the technical detail of the required service, such as the inputs and outputs data types, and a search button.
- **The process of the experiment:** We ask both groups of students to search for a list of required services using the two interfaces. Then we interview the participants how they feel about these two search interfaces. For the usability of the interfaces, we ask them to rate as “Easy to use”, “Normal”, and “Difficult to use”. For how difficult it is to provide the technical information for searching services, we ask them to rate as “No way to provide”, “Difficult to provide”, “Normal”, and “Easy to provide”.
- **The experiment result:** The result for the interface usability shows that 73.3% of the students in the group without the professional computer background rate the single text box interface as “Easy to use” and 100% of them rate the multiple text boxes interface as “Difficult to use”, and 66.7% of the students in the computer science group rate the single text box interface as “Easy to use” and 86.7% of them rate the multiple text boxes interface as “Difficult to use”. The result for how difficult it is to provide the technical information of required service shows that 93.3% of the students in the group without the professional computer background rate it as “No way to provide” and the rest of them rate it as “Difficult to provide”,

and 60% of the students in the computer science group rate it as “Difficult to provide”, 13.3% of them rate it as “No way to provide”, and 26.7% of them rate it as “Normal”.

From the result of the simple experiment, we can see the gap between what kind of information a user can provide to search for services and what kind of information is expected to be matched with the technical description of services, especially for the users who have little domain knowledge. It is even difficult for the computer science students to provide some of the technical information of the required service.

By analysing the result of the experiment and studying the current research in the literature, we identify the following key issues that need to be further investigated in order to improve service discovery and composition. In this thesis, we will discuss in details how we address these issues and provide a research solution.

- **Insufficient usage context information:** Current semantic Web service description frameworks are mostly focusing on ontology based data and capability semantics of Web services. They do not sufficiently address the usage context information of a service. Although there are some research work [Maamar et al., 2005] [Maamar et al., 2007] [Medjahed et al., 2007] on Web services context, they mainly study the runtime environmental context, which does not help to locate required services during the service discovery process. The usage context of a service includes the information about how a service is used and its relationships with other services. This kind of context information can be helpful for service users to locate their required services.
- **Precise service specifications:** In order to locate a required service, the current service discovery requires precisely defined technical specifications for the required service, such as service input and output data types and service capabilities. This kind of information is difficult for a service user to provide at the preliminary stage of service discovery, especially when the service user is not a domain expert in the required service area.
- **Insufficient information about inter-relationship among service:** The current work inadequately addresses the inter-service relationships. A Web service always needs to interact with other Web services to achieve its functionalities.

If we consider each service as an isolated individual and ignore the possible relationships with other Web services, the efficiency of service discovery and composition about this service will be decreased.

- Lack of incomplete information handling: Although some of the existing work support rules in service description and composition [Martin et al., 2004] [Orriens et al., 2003] [Charfi & Mezini, 2004], these rules are based on monotonic logic and reasoning which are not suitable for handling incomplete information.

In the following sections, we will discuss what our solutions are for tackling the issues discussed above.

1.4 Proposed method

In this section, we will give an overview of our solutions to the problems discussed previously. We propose a context based semantic service description framework that provides sufficient usage context information of services, adequate semantics, and non-monotonic rules for handling incomplete information. We also propose an enhanced service discovery mechanism based on our service description framework to illustrate how the proposed framework improves the efficiency and effectiveness of service discovery and composition.

1.4.1 Integrating Context into Service Description

Context as a term has been addressed in many pieces of Web services related research. In most of the literature, context is interpreted as the runtime environment of Web services [Keidl and Kemper, 2004] [Maamar et al., 2005a] or as constraints and the changeability of quality of service (QoS) [Zhou et al., 2008]. However, we interpret the context of Web service from the conceptual and usage perspectives. To achieve a Web service's functionalities, the service needs to interact with other services and entities. Although the self-contained and loosely-coupled features are important characteristics, Web services are not isolated individuals. Given a Web service, there are always some typical usage scenarios in which this service can participate. In other words, given a usage scenario, certain types of Web service are always involved. We consider typical usage scenarios as a kind of service context. This kind of context can

help a service user to identify a service. It is useful because more often than not, when a service user searches for a service, the service user has a usage scenario in mind rather than the technical detail of the required service. We embedded this kind of usage context into Web service description to enhance the service discovery method to locate services not only based on service technical specification, but also based on service users' usage scenarios. The detail of the service usage context and how it is applied in the service description, discovery and composition, is discussed in detail in chapters 3, 4, and 5.

1.4.2 Integrating Adequate Semantics into Service Description

The semantic web proposes the annotation of static information on the web with machine understandable semantics. The annotation is constructed mainly using XML based semantic web languages and common domain ontologies. However, the semantics of a Web service are far more complicated than the semantics of static information because a Web service is a functional unit and its semantics contain many aspects. Nagarajan summarise four types of semantics that should be addressed in Web service descriptions [Nagarajan, 2006].

- **Data Semantics:** Data semantics is the formal definition of the data in the input and output messages of a Web service. It is normally used in service discovery process for matching with service requirements. It is also essential in indicating the interoperability between services.
- **Functional Semantics:** Functional semantics is the formal definition of the capabilities of a Web service. It is normally used in the service discovery and composition process.
- **Non-Functional Semantics:** Non-functional semantics is the formal definition of quantitative or non-quantitative constraints and requirements, such as QoS (Quality of Service), minimum cost and policy requirements, message encryption. It is normally used in the service discovery and composition process. It is also essential in indicating the interoperability between Web services.

- **Execution Semantics:** Execution semantics is the formal definition of the execution or flow of services in a process or of operations within a service. It is normally used in process verification and exception handling.

In our proposed context based semantic service description framework, we extend the four types of semantics in order to better describe Web services and assist service discovery and composition. We extend the functional semantic with usage context so that the capabilities of a Web service can be identified through not only its functional semantics, but also the typical usage scenarios of the service. We also extend the execution semantics so that it can be also used for identifying services in a service discovery process. The detail of the framework is discussed in Chapter 4.

1.4.3 Addressing Incomplete Information

In traditional Artificial Intelligence systems, problem solvers are designed based on complete information. A problem solver assumes a complete knowledge base and its main task is to draw correct conclusions from the knowledge base using a classical reasoning mechanism [Genesereth & Nilsson, 1987]. In this case, a classical monotonic logic is sufficient, such as predicate logic. However, in the Web services case, the situations are more complicated and the ability to handle incomplete information is crucial for the rule system used for service description and composition. In our work, we adopt a non-monotonic reasoning mechanism and use Defeasible Logic [Nute, 1994] as a formalism to describe pre-conditions and the effects of services and rules for service composition and invocation. Defeasible Logic supports different types of rule and has a built-in rule priority handling mechanism. It allows us to draw conflict conclusions and the priority handling mechanism will decide the conclusion with a higher priority in different situations. In Chapter 4, we will discuss the Defeasible Logic based rules in our service description in detail.

1.4.4 A Flexible Service Discovery Mechanism

In order to improve the accuracy and efficiency of service discovery and composition, we propose a two-step service discovery mechanism using the context based semantic service description framework. As discussed previously, in order to locate a service, the service search methods based on current service description frameworks require a service requester to provide a detailed and exhaustive specification of the required

service, including input and output data types, pre- and post-conditions, and service capabilities etc. From our experiment it is observed that providing a detailed service specification at the beginning of the service discovery process is infeasible for most of the service users. Except for domain experts in the required service areas, most of the users would not be able to provide such detailed technical information. To solve this problem, we develop a two-step service discovery mechanism that can guide service users step by step to locate suitable services. The first step is to capture a service user's mind to see what service the user wants and what scenario the required service will be applied in. This is done through matching the service user's requests or usage scenarios with the service usage context in our service description framework. This step ensures that the services that are relevant to the user's request are located. From the preliminary results, the user can get some hints for proposing further detailed requirements. In the second step, based on the user's more detailed requirements, the result from the first step is refined and composite services are generated if existing services cannot fulfil the requirement. Before returning to the service user located services are ranked according to their similarity degree to the user's requirement. However, if the user is familiar with the technical details, the first step can be skipped.

1.5 Contributions

The main contributions of this thesis are outlined as follows:

- **Define usage context of Web services:** We define the service context in an angle that is different from other research work. We consider context as the information that can help a service user to locate required services, i.e. the usage context, rather than the environmental information to regulate a service's behaviour at runtime.
- **A context-based service description framework:** We propose a context-based service description framework that considers not only the semantics of Web services, but also the usage context. Using the components provided in this framework, the service discovery result is more accurate and the process of service composition is more efficient.
- **Handling incomplete information:** Non-monotonic rules are used to represent pre-conditions and effects of services and rules for service

composition. The non-monotonic rules can handle the dynamic and incomplete information.

- **A two-step service discovery mechanism:** We propose a two-step service discovery mechanism based on the proposed framework to demonstrate how the framework can facilitate the service discovery and composition process, especially when service users do not have the sufficient knowledge about their required services.
- **Implementation and evaluation:** We provide a proof-of-concept prototype that implements our approach and conduct a set of experiments to evaluate the result accuracy, the performance, and the scalability of our approach. We also compare our solution with the existing solution, i.e. OWL-S, through a series of tasks in a given scenario to analyse the pros and cons of our solution.

1.6 Criteria for Success

In order to fully evaluate our work, we list a set of criteria for success below:

- **Technological novelty:** Our work must be novel in comparison with the existing work.
- **A Context-based Semantic Service Description Framework:** one of the outcomes of our work is a framework that can better describe Web services in order to improve the efficiency of service discovery and composition.
- **A suitable prototype:** A proof-of-concept prototype needs to be realised in order to show that our service description framework can be actually implemented and the result from the prototype shows that it is beneficial.
- **Acceptable system performance:** The performance of the system has to be at an acceptable level.

1.7 Thesis Outline

The structure of the thesis is as follows:

-
- **Chapter 2: Context of Problem Area Analysis and Research.** In this chapter, we will give a comprehensive survey of existing work and the relevant background knowledge for the problem areas that have been discussed previously.
 - **Chapter 3: Service Usage Context.** In this chapter, we start a detailed discussion of our work. The definition of service usage context and its relevant concepts are discussed in this chapter.
 - **Chapter 4: Context based Semantic Service Description Framework.** In this chapter, we discuss the context based semantic service description framework and its main components
 - **Chapter 5: Two-Step Service Discovery Mechanism.** Chapter 5 proposes the two-step service discovery mechanism and how it benefits service discovery and composition.
 - **Chapter 6: Implementation.** In this chapter, we discuss the implementation detail of the prototype
 - **Chapter 7: Evaluation.** In this chapter, we evaluate our work based on the prototype and scenario based analysis.
 - **Chapter 8: Conclusion and Future Work.** Finally, we summarise our work and point out future research directions for Semantic Web Services.

Chapter 2: Context of Problem Area Analysis and Research

In this chapter, we will give a detailed survey of the related work in relevant research areas, including Web services, semantic web, Semantic Web Services, service context, service composition, and semantic information processing.

2.1 Overview

Our research focuses on how to improve the current service description technologies and standards in order to achieve more efficient service discovery and composition. There are many research areas that are related to our work. In the following sections we will review the literature from two aspects: the work related to our research and the methods that are used in our work for solving problems.

For the work related to our research, we will first give a detailed discussion on the Web services technology and point out what problems it has. Then, we will discuss what the semantic web is and how current research work combines semantic web technology with Web services technology to enhance the service discovery and composition, so called Semantic Web Services. Context is one of the key areas in our work for describing services. Therefore, we review the literature to see how context is defined and used in the service description, discovery and composition. Finally, we will look at the current technologies for service composition and semantic information processing.

For the methods that are used in our work, we will first look at some of the knowledge representation methods, such as ontology, conceptual graphs (CG), and Defeasible Logic. We then discuss semantic similarity calculation methods, such as ontology based methods and vector based methods.

2.2 Web services

The Internet has become a very important tool in daily business. As the Internet becomes faster and more reliable, people are not satisfied with using the Internet only as an information publishing and consuming tool, they want to use it to directly connect their business with partners and customers. Thus, enterprise application integration issues emerge in the research and development of e-commerce. Many solutions have been proposed for solving the application integration problem, such as CORBA, DCOM, and Java RMI [Coulouris et al., 2001]. However, all these solutions have a common weakness, i.e. lack of effective interoperability. The reason is that to achieve these solutions, specific platforms, programming languages, and protocols are

required [Gray 2004]. Only enterprise applications using the same communication technology can be directly integrated. Otherwise, extra development work has to be done to achieve the integration. This problem has drastically reduced flexibility and increased complexity hence the cost of the enterprise application integration. Furthermore, all of these solutions use binary messages and specific network ports for their communication, which are normally not allowed by the firewall. To solve these problems, the Web services technology emerges. In this section, we first give a brief overview of the basic architecture of the Web services technology. We then discuss in detail the core standards of the Web services technology, such as WSDL [WSDL, 2007], SOAP [SOAP, 2007], and UDDI [UDDI, 2004], and the problems exposed by the basic architecture and the core standards that need to be tackled in the future research and development work.

2.2.1 Technology overview

The success of the Web services technology relies on the standard data formatting language XML, which is used to format Web service communication data [SOAP, 2007], describe the Web service interfaces [WSDL, 2007], and advertise on the web [UDDI, 2004]. Communication with Web services is also established via a set of standard TCP/IP protocols such as HTTP, HTTPS, SMTP, and FTP [SOAP, 2007]. The XML based language and the standard TCP/IP based communication protocols make Web services programming language independent, operating platforms independent, and firewall friendly. The loosely coupled and self-contained features give Web services the ability to be invoked at runtime and composed with other services to achieve complex tasks.

The W3C organisation defines the fundamental architecture of the Web services technology. The goal of the Web services architecture is to provide a way for a service provider to publish their services, for a service requestor to find required services, and for the service requestor to invoke the services. To achieve this goal, three components are defined to build up the Web services architecture [Kreger, 2001], see Figure 2.1.

- **Service provider:** Companies, organisations, or individuals that have developed and published Web services.

- **Service requester:** Companies, organisations, or individuals who require the functionalities of the Web services. Once the required services are located, the requesters can directly invoke the services according to the information provided by the service description documents.
- **Global service registry/Service broker:** A global registry acts as a central service catalogue for the service providers to publish services and the service requesters to locate services.

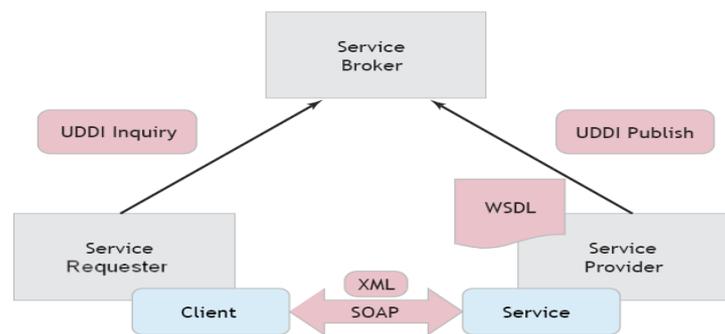


Figure 2.1. Web services Architecture [Kreger, 2001].

2.2.2 Core Standards

A series of XML based standards, such as SOAP, WSDL, and UDDI, have been used to implement the fundamental architecture of Web services. The communication messages between a service requester and a service provider are encoded into SOAP messages which are plain text XML messages rather than binary messages, so that the runtime environment details are not required before establishing the communication as long as the communicators on both sides can interpret SOAP encoded messages. WSDL is used to describe the Web service's invocation details including the service name, the provided operations, and the input and output data types etc. UDDI defines the standards and APIs for publishing and discovering Web services. The Web services technology stack [Gunzer, 2003] is shown in Figure 2.2.

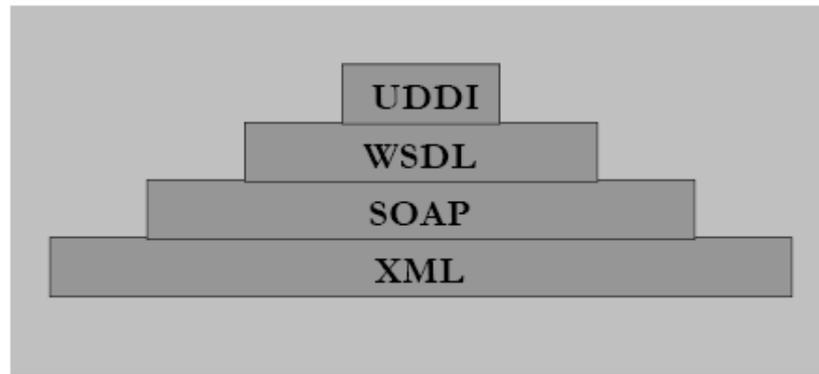


Figure 2.2. The technology stack of Web services [Gunzer, 2003].

SOAP:

Since the main purpose of Web services is to solve the heterogeneous environment integration problem, a runtime environment independent protocol is needed for transferring the communication data between the enterprise applications. SOAP is a protocol that is suitable for this purpose. SOAP is an XML based message formatting protocol. It is a stateless and one way message exchange paradigm [SOAP, 2007]. The root element of a SOAP message is “<Envelope>” and contains two sub-elements: an optional “<Header>” element and a “<Body>” element. The “<Header>” element can contain the authentication or data encoding information. If a “<Envelope>” contains a “<Header>” element, then the “<Body>” element must not be the first element within the “<Envelope>”. A “<Body>” element contains the request/response message and the error message if there is any. The error message only appears once within a “<Body>” element. The structure of a SOAP message [SOAP, 2007] is shown in Figure 2.3.

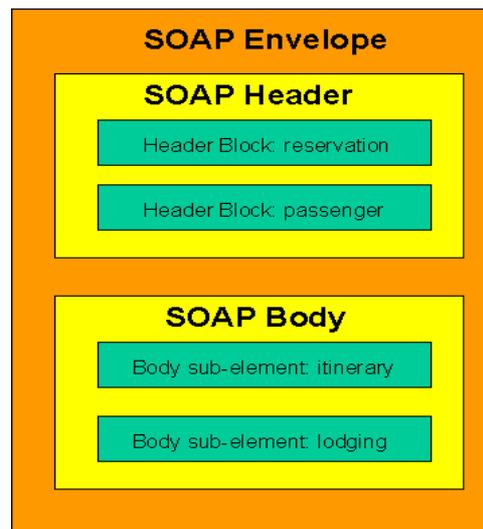


Figure 2.3. SOAP message structure [SOAP, 2007].

WSDL:

WSDL is an XML based standard for describing Web services. Based on WSDL a service requester can know where and how to invoke services. The role of WSDL is similar to the IDL file in CORBA [Sheth and Miller, 2003] or the Remote Interface in Java RMI. The root element of a WSDL document is “<definitions>”. In a typical WSDL document the sub-elements under the “<definition>” element could be: <types>, <message>, <portType>, <binding>, and <service>.

The “<types>” element contains all the data type definitions of the described service for sending and receiving messages. The XML schema namespace (<http://www.w3.org/2001/XMLSchema>) is recommended for the data type definitions. An example of the “<types>” element is given below.

```
<types>
  <s:element name="addition">
    <s:complexType>
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
        <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
      </s:sequence>
    </s:complexType>
  </s:element>
</types>
```

The “<message>” element defines how the data types defined in the “<types>” element are bound with the request and respond messages. The “<message>” element can appear many times depending on how many functions the Web service provides. An example of the “<message>” element is given below.

```
<message name="additionResponse">
  <part name="additionReturn" type="xsd:float" />
</message>

<message name="additionRequest">
  <part name="in0" type="xsd:float" />
  <part name="in1" type="xsd:float" />
</message>
```

The information within the “<portType>” element is a set of operations. “<operation>” sub-elements are used to identify the functions provided by the service and their operation types. Four kinds of operation have been defined in the WSDL specification [WSDL, 2007].

1. **One-way operation:** The endpoint receives a message.
2. **Request-response Operation:** The endpoint receives a message and sends a replay.
3. **Solicit-response Operation:** The endpoint sends a message and receives a reply.
4. **Notification Operation:** The endpoint sends a message.

These four operations are represented by three sub-elements within the “<operation>” element, i.e. the “<input>” element, the “<output>” element, and the “<fault>” element.

- A one-way operation only specifies the “<input>” element.
- A request/response operation specifies the “<input>” element first, followed by the “<output>” element and some optional “<fault>” elements.
- A solicit response operation specifies the “<output>” element first, and then the “<input>” element followed by some optional “<fault>” elements.
- A notification operation only specifies the “<output>” element.

An example of the “<portType>” element is given below.

```

<portType name="Service1Soap">
  <operation name="addition">
    <input message="s0:additionSoapIn" />
    <output message="s0:additionSoapOut" />
  </operation>
  <operation name="subtraction">
    <input message="s0:subtractionSoapIn" />
    <output message="s0:subtractionSoapOut" />
  </operation>
  <operation name="multiplication">
    <input message="s0:multiplicationSoapIn" />
    <output message="s0:multiplicationSoapOut" />
  </operation>
  <operation name="division">
    <input message="s0:divisionSoapIn" />
    <output message="s0:divisionSoapOut" />
  </operation>
</portType>

```

The “<binding>” element defines the message format and the transformation protocol details for each operation defined in the “<portType>” element. A binding must specify exactly one protocol and must not specify any address information. An example for binding an operation to the SOAP protocol is given below.

```

<binding name="Service1Soap" type="s0:Service1Soap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  <operation name="addition">
    <soap:operation soapAction="http://tempuri.org/addition" style="document" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
</binding>

```

The “<service>” element contains a set of ports and each port specifies a single address for binding, so that the service requester can know where the service can be found. An example of the “<service>” element for SOAP binding is given below.

```

<service name="Service1">
  <port name="Service1Soap" binding="s0:Service1Soap">
    <soap:address location="http://localhost/ws/webservice1/Service1.asmx" />
  </port>
</service>

```

UDDI:

UDDI is a standard designed to provide a searchable directory of businesses and their Web services [Gunzer, 2003]. In the Web services architecture, it acts in the service broker role. There are three sets of information provided in an UDDI directory.

- **UDDI white pages:** the basic information about the service providers, such as a company name, address, and phone numbers, as well as other standard business identifiers like Dun & Bradstreet and tax numbers.
- **UDDI yellow pages:** the detailed business data, organised by relevant business classifications.
- **UDDI green pages:** the technical information about Web services that are exposed by the business, e.g., how to communicate with a Web service.

To provide the UDDI services at runtime, UDDI also supplies a set of APIs including the APIs for publishing Web services and the APIs for searching Web services. The main component of the UDDI service is the UDDI business registration which is an XML based document. The business details and the provided Web services are described in this document. There are four key data structures within the UDDI registration to provide information [UDDI, 2004].

- **Business entities:** Describe the basic information about the organisation that publishes the services.
- **Business services:** Describe the provided services in business terms.
- **Binding templates:** Describe the provided services in technical terms.
- **tModels:** Describe the technical specifications of published Web services e.g. wire protocols, interchange formats, or sequencing rules.

In the latest version of UDDI, a main architectural change is the concept of “registry interaction.”[UDDI.org, 2006]. Figure 2.4 illustrates how the private domain service registry can interact with the public domain service registry.

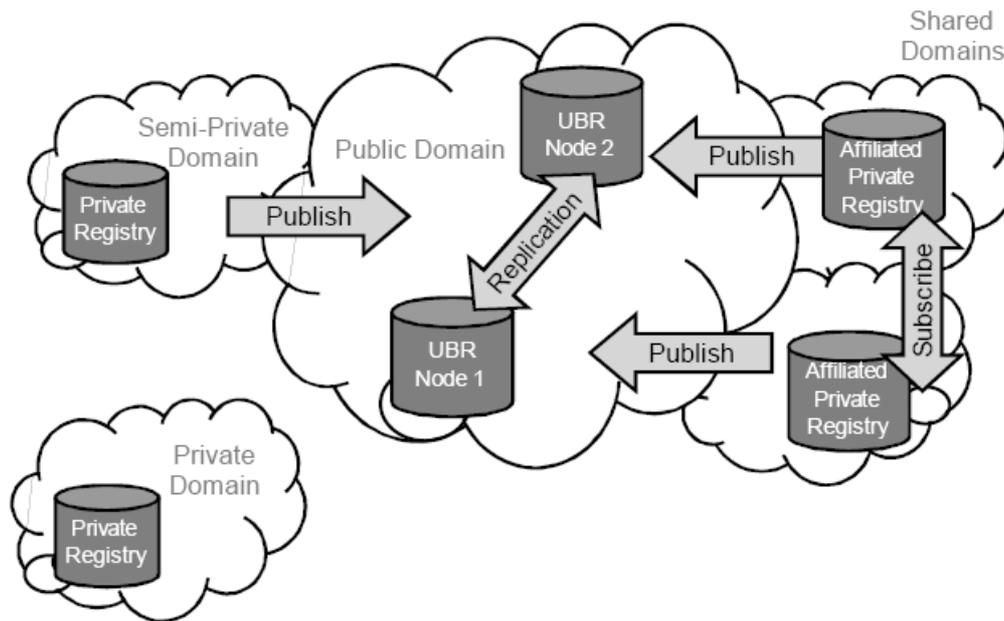


Figure 2.4. Registry interaction enabled by UDDI 3.0 [UDDI.org, 2006].

2.2.3 Drawback of Current Web services Standards

For the last decade, Web services have become an important research topic in the fields of Service-Oriented Architecture (SOA) [Huhns and Singh, 2005] and Grid computing [Foster et al., 2001]. Web services as a new distributed system technology has been widely adopted by industries in areas, such as enterprise application integration, business process management, and virtual organisations. With an exponential increase of services available on the web, it becomes an extremely difficult task to find a service that fits well to a user's requirements without automating the process of service discovery and composition. The problems of automation are twofold: a formal representation for services is required such that service comparison and matching could be done more precisely and efficiently, and a powerful and expressive service description framework is also required such that the richer semantic description for services is introduced with less semantic loss during service construction, organisation, and publishing. Currently, WSDL only syntactically addresses the data structure and message type, the interaction protocols, and the endpoint address of a web service and UDDI only provides a keyword based service discovery mechanism. Lack of semantics in current web service standards has become a barrier in achieving automatic or semi-automatic service discovery, invocation and composition. To overcome this drawback, an emerging research area,

so called Semantic Web Services, has gained considerable attention and seems to be the most promising way towards achieving the automatic service discovery and composition. We will give a detailed survey on the semantic web and Semantic Web Services technologies later.

2.3 Knowledge Representation

Knowledge representation is a subject in cognitive science, artificial intelligence, and knowledge modelling. However, in this section, we only focus on the knowledge representation methods that are relevant to our research work. The knowledge representation methods we are interested in are ontology, description logics, conceptual graphs, and non-monotonic reasoning – a method that is used to deal with incomplete knowledge base.

2.3.1 Ontology

The term ontology is first adopted by the artificial intelligence community from philosophy where it was originally used to describe the nature of existence. The original meaning of ontology in philosophy is “the branch of philosophy that deals with the nature of existence” [Collins, 1995]. So what is Ontology? Guarina [Guarino, 1998] has stated ontology as:

“An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.”

Now, the concept of ontology is widely used in many areas of computer science to describe a certain reality and the intended meaning of the vocabulary words [Guarino, 1998]. What is ontology in computer science? Here we list some definitions from the literature:

“An ontology is an explicit and formal specification of a conceptualization” [Gruber, 1995]

“An ontology is a logical theory accounting for the intended meaning of a formal vocabulary” [Guarino, 1998]

“Ontology is the term used to refer to the shared understanding of some domain interest” [Uschold and Grüninger, 1996]

From the above definitions, we can see that the goal of an ontology is to build a unifying framework through specifying conceptualisation to achieve shared understanding between people or applications. It consists of a taxonomy for concepts and the relationships among concepts [Antoniou, 2004]. The taxonomy is built up by classes and subclasses, and the properties of classes build up the class relationships. Ontology is important concept in computer science field because it clarifies the structure of knowledge, enables knowledge sharing, and let us build specific knowledge bases to describe specific situations [Chandrasekaran et al. 1999]. By applying ontologies, some existing problems, such as poor communication between people within their organisations, lack of a shared understanding, and limited interoperability of software tools, can be addressed [Uschold and Grüninger, 1996].

Fensel [Fensel, 2001] categorises ontologies into five types each fulfilling a different role in the process of building knowledge based systems.

- **Domain ontologies:** capture valid knowledge for a specific domain.
- **Metadata ontologies:** provide vocabulary for describing online content, e.g. Dublin Core [DCMI, 2005].
- **Generic or common sense ontologies:** capture general knowledge about the world, such as time, space, state, and event. Therefore, these ontologies are valid for many domains.
- **Representational ontologies:** do not commit themselves to any particular domain.
- **Method and task ontologies:** provide a reasoning point of view on domain knowledge.

Ontologies can be used in many areas. Uschold and Gruninger [Uschold and Grüninger, 1996] divide ontologies into three spaces of use: communication, interoperability, and software engineering (including specification, reliability, and reusability). More specifically, Guarino [Guarino, 1998] specifies how ontologies can

be used in development of information systems from the temporal dimension, such as development time and runtime, and the structural dimension, such as databases, user interfaces, and application programmes. Based on the different purpose and scope of an ontology, the targets for building ontologies are diverse. However, the important aspects in building ontologies are the same as those which should be considered in order to build reasonable ontologies. Uschold and Gruninger [Uschold and Gruninger, 1996] summarise three aspects which are related to building ontologies.

- **Capture:** this aspect is about identifying the key concepts and their relationships in a domain of interest, producing text definitions for these concepts and relationships, and identifying terms to refer to these concepts and relationships.
- **Coding:** this aspect is about defining the concepts and relationships in a formal ontology language.
- **Integrating Existing Ontologies:** integrating existing ontologies can save ontology development time. However, it is a hard task to be achieved and there are some issues need to be further studied [Pinto et al., 1999].

To summarise, ontologies are very important in providing shared understanding and a unified framework for people and applications to easily communicate. Based on the shared domain ontology, automatic information processing and retrieval can be achieved. However, there are still many issues that need to be studied further, such as ontology mapping [Ehrig and Sure, 2004] [Hage et al, 2005] and integration [Pinto and Martins, 2001].

2.3.2 Description Logics

Concept

Description logics (DLs) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way [Baader et al., 2002]. The basic elements of DLs are *concept* and *role*. The *concept* denotes the classes of objects and the *role* denotes the binary relationships between classes [Calvanese et al., 2001]. As DLs are a set of languages for knowledge representation, they have sets of symbols and syntax to describe the world and suitable knowledge representation expressions for reasoning. The DLs are derived from a knowledge representation called inheritance networks

[Brachman, 1979], see Figure 2.5. Inheritance networks simply build up “*is-a*” relationships between concepts and properties. DLs are very similar to the model of inheritance networks, but with much richer expressiveness.

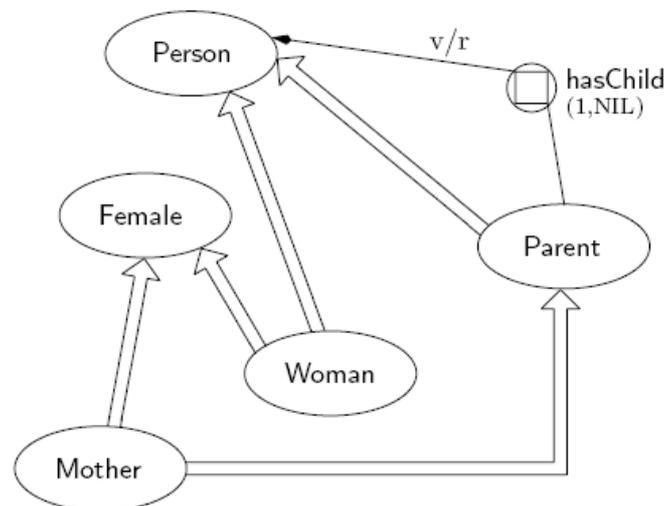


Figure 2.5. An example of inheritance network [Brachman, 1979].

In a typical DL knowledge base, there are two components: T-Box and A-Box, which are used to represent intensional knowledge and extensional knowledge [Nardi and Brachman, 2003].

The T-Box contains terminologies which are intensional knowledge. The basic form of declaration in the T-Box is concept definition. A definition of a new concept is made in terms of previously defined concepts. For example, a definition of woman could be represented as follows:

$$Woman \equiv Person \cap Female$$

This expression means that a woman is a female person. We can also write the expression like:

$$Woman \sqsubseteq Person$$

which means that a *woman* is a person. The later expression is called inclusive axiom. There are some important assumptions of concept definition in the T-Box: 1) Only one definition for a concept name is allowed; 2) Definitions are acyclic which means that defining a concept in terms of itself or in terms of other concepts that indirectly refer to it is not allowed [Nardi and Brachman, 2003]. The main task of the T-Box is classification.

The A-Box contains assertions about individuals, usually called membership assertions, which are the extensional knowledge. There two kinds of assertions, concept assertions and role assertions [Nardi and Brachman, 2003]. A concept assertion defines to which concept an instance belongs. A role assertion defines the relationships between instances. For example, $Male \cap Person(Tom)$ is a concept assertion of Tom and $hasChild(Tom, Mike)$ is a role assertion to describe the relationship between Tom and Mike. The basic reasoning task of an A-Box is instance checking to ensure knowledge consistency, realisation, and instance retrieval [Nardi and Brachman, 2003].

Family of DL

Description logics are a big language family and have many variants. However, all the variants are based on the syntax of very simple DL \mathcal{AL} , and add extra expressive features on it. The syntax rules of basic DLs \mathcal{AL} are listed below [Baader and Nutt, 2003]:

$$\begin{aligned}
 C, D &\rightarrow A \quad (\text{Atomic Concept}) \\
 &\top \quad (\text{Universal Concept}) \\
 &\perp \quad (\text{Bottom Concept}) \\
 &\neg A \quad (\text{Atomic negation}) \\
 &C \cap D \quad (\text{Intersection})
 \end{aligned}$$

Other variants of \mathcal{AL} add extra expressive rules based on the rules above. For example, \mathcal{U} adds union of the concept written as $C \cup D$; \mathcal{E} adds full existential quantification written as $\exists R.C$; \mathcal{N} adds number restriction written as $\geq n R$ and $\leq n R$; and \mathcal{C} adds negation of arbitrary concepts written as $\neg C$. Therefore, a full featured \mathcal{AL} language could be written as $\mathcal{AL}\mathcal{E}\mathcal{U}\mathcal{N}\mathcal{C}$. There is another family of DL called expressive DLs. In this family of DLs, the languages used for building concepts and roles comprise all classical concept forming constructs, plus several role forming constructs such as inverse roles, and reflexive-transitive closure, and no restriction is posed on the axioms in the T-Box [Calvanese and Giacomo, 2003]. Additional letters indicating more extensions are listed below [Horrocks, 2005]:

\mathcal{S} for \mathcal{AL} with transitive roles (R_+)

\mathcal{H} for role hierarchy (e.g., $hasDaughter \sqsubseteq hasChild$)

\mathcal{O} for nominals/singleton classes (e.g., $\{Italy\}$)

\mathcal{I} for inverse roles (e.g., $isChildOf \equiv hasChild^{-}$)

\mathcal{Q} for qualified number restrictions (e.g., $\geq 2 hasChild.Doctor$)

\mathcal{F} for functional number restrictions (e.g., $\leq 1 hasMother$)

$\mathcal{ALC} + R_+ + \text{role hierarchy } (\mathcal{H}) + \text{inverse } (\mathcal{I}) + \text{QNR } (\mathcal{Q}) = \mathcal{SHIQ}$

2.3.3 Conceptual Graphs

Conceptual graphs (CGs) are a system of logic based on Charles Sanders Peirce's existential graphs [Peirce, 1936-58] and the semantic networks [Sowa, 1987] of artificial intelligence. CGs represent knowledge in a way that is logically precise, human readable, and machine computable. As CGs are directly mapped to natural language, they are widely used for transform natural language to and from computational formalism.

A CG is a finite, connected, bipartite graph with nodes of one type called concepts and nodes of the other type called conceptual relations [Sowa, 1976]. Bipartite graph means that there are no arcs between a concept and another concept, and no arcs between a relation and another relation. All arcs either go from a concept to a relation or from a relation to a concept. A simple conceptual graph is shown in Figure 2.6, where the square boxes are concepts and oval boxes are relations.

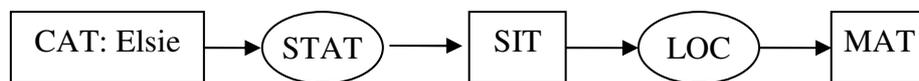


Figure 2.6. A simple conceptual graph [Sowa, 1984].

CGs are defined over a set of vocabularies, which we call a support [Chein and Mugnier, 1992]. A support is formally defined as follows:

Definition 2.1. A support is a 4-tuple $\mathcal{F} = (T_C, T_R, \phi, \tau)$, where,

- T_C and T_R : Two partially ordered finite sets, respectively of concept types and relation types.
- \mathcal{I} : A set of individual markers.
- T_C , T_R , and \mathcal{I} are pairwise disjoint.
- τ : A mapping from \mathcal{I} to T_C .
- The generic marker is denoted as $*$, where $*$ $\notin \mathcal{I}$. The set $\mathcal{I} \cup \{*\}$ is partially ordered in the way that $*$ is the greatest element.

A CG is defined over a support \mathcal{I} , which has two kinds of nodes, the conceptual nodes and the relation nodes. Formally a CG is defined as follow:

Definition 2.2. A CG g , defined over a support \mathcal{I} , is a 4-tuple (C_g, R_g, E_g, l_g) , where,

- $(C_g \cup R_g, E_g)$ is a bipartite graph, where, C_g and R_g are the node sets, respectively of concept nodes and of relation nodes, and E_g is a set of edges.
- l_g is a labelling function of nodes and edges. A concept node c is labelled by an ordered pair $(type(c), marker(c))$, where $type(c) \in T_C$, $marker(c) \in \mathcal{I} \cup \{*\}$. A relation node r is labelled by $type(r)$, where $type(r) \in T_R$. Edges around each relation node are labelled with number from 1 to the number of edges that are connected to that relation node. No edge is labelled in the CG representation.

A verbal explanation of CG and some examples are given as follows. The concept nodes in a CG represent entities, actions, and attributes in a given application domain. The label of a concept node consists of two fields: *type* and *referent*, separated by a colon, [type: referent]. The *type* represents the class of a concept. The *referent* represents an instance of the class. The functions *type()* and *referent()* can be used to get a concept node's type and referent. If the value of *referent(c)* is an individual marker (an identification of an instance, such as name or id), e.g. [Cat: Tom], then the concept c is an individual concept. If the value of *referent(c)* is "*", e.g. [Cat: *], then the concept c is a generic concept. A concept having only a type label is equivalent to a generic concept, e.g. [Cat] = [Cat:*]. The relation nodes in a CG represent the relationships between concept nodes. $type(r)$ is the type of a relation node r .

There are two ways to represent a CG: the display form and the linear form [Sowa, 1999]. In the display form representation, a CG is displayed as a graph, see Figure 2.6. The concept nodes are represented by rectangles and the relation nodes are represented by ovals or circles. In the linear form representation, a CG is represented using linear text. Square brackets represent the concept nodes and round brackets represent the relation nodes. Therefore, the graph shown in Figure 2.6 can also be represented as:

$$[\text{CAT: Elsie}] \rightarrow (\text{STAT}) \rightarrow [\text{SIT}] \rightarrow (\text{LOC}) \rightarrow [\text{MAT}]$$

Before we discuss some of the important definitions and rules of CG, we first introduce the concept of canonical graph and canonical formation rules. Theoretically, to construct a CG, we can combine any concept nodes and relation nodes as long as they satisfy the CG definition. However, some of the CGs may not make any sense. Meaningful graphs that represent the real world are called canonical graphs [Sowa, 1984]. A set of formation rules are used to derive new canonical graphs from existing canonical graphs. If u and v are canonical CGs, then a canonical CG w can be derived from u and v by applying the following rules [Sowa, 1984]:

- **Copy:** w is an exact copy of u .
- **Restrict:** for any concept c in u , $type(c)$ can be replaced by a subtype; if c is generic, its reference may be changed from “*” to an individual marker. These changes are only permitted if $referent(c)$ conforms to $type(c)$ before and after the changes.
- **Join:** if a concept c in u is identical to a concept d in v , then w can be obtained by deleting d from v and linking to c in u all the arcs of the conceptual relations that had been linked to d .
- **Simplify:** if conceptual relations r and s in the graph u are duplicates, then one of them may be deleted from u together with all its arcs. If two conceptual relations are duplicated, it means that the direction of their edges and the types of concepts that they can link to are identical.

By having the canonical derivation rules, we can introduce some important definitions and properties of CG, such as specialisation, generalisation, and projection, which will be applied in our research work.

Definition 2.3. For any CGs u and v , u is called a specialisation of v (or v is called a generalisation of u), denoted as $u \leq v$, if u is canonically derivable from v .

Definition 2.4. Let u_1 , u_2 , v , and w be CGs. If $u_1 \leq v$ and $u_2 \leq v$, then v is called a common generalisation of u_1 and u_2 . If $w \leq u_1$ and $w \leq u_2$, then w is called a common specialisation of u_1 and u_2 .

From the definitions above and the canonical derivation rules, Sowa [Sowa, 1984] has proved that any CG is a generalisation of itself and any sub-graph is a generalisation of its original. The graph consisting of the single concept $[\top]$ is a generalisation of every other CG. Therefore, Sowa derives the generalisation hierarchy properties, which are for any CG u , v , and w , the following properties hold:

- **Reflexive:** $u \leq u$.
- **Transitive:** if $u \leq v$ and $v \leq w$, then $u \leq w$.
- **Antisymmetric:** if $u \leq v$ and $v \leq u$, then $u = v$.
- **Sub-graph:** if v is a sub-graph of u , then $u \leq v$.
- **Sub-type:** if u is identical to v except that one or more type labels of v are restricted to subtypes in u , then $u \leq v$.
- **Individuals:** if u is identical to v except that one or more generic concepts of v are restricted to individual concepts of the same type, then $u \leq v$.
- **Top:** the graph $[\top]$ is the generalisation of all other conceptual graphs.

For a CG u , if there is a CG v that is a specialisation of u , i.e. $u \leq v$, then there must be a sub-graph v' embedded in v that represents u . However, the form of v' may differ from u because some of the concepts are restricted to their sub-types or some of the generic concepts are restricted to individual concepts or some of the concepts or relations are removed due to duplication. The sub-graph v' is called a projection of u in v , which is formally defined in the following theorem.

Theorem 2.1. For any CGs u and v , where $u \leq v$, there must exist a mapping $\pi: v \rightarrow u$, where $\pi_u v$ is a sub-graph of u called a projection of v in u . The projection operator π has the following properties:

- For each concept c in v , $\pi_u c$ is a concept in $\pi_u v$ such that $type(\pi_u c) \leq type(c)$, “ \leq ” here represents the sub-type relationship between concepts. If c is an individual concept, then $referent(\pi_u c) = referent(c)$.

- For each relation r in v , $\pi_u r$ is a conceptual relation in $\pi_u v$ such that $type(\pi_u r) = type(r)$. If the i -th arc of r is linked to a concept c in v then the i -th arc of $\pi_u r$ must be linked to $\pi_u c$ in $\pi_u v$.

Proof. See [Sowa, 1984] (p. 99) for detail.

If two CGs have generalisation and specialisation relationships between them, the next theorem provides the properties of their logical formulas. The operator “ ϕ ” can translate a CG into its equivalent logical formula.

Theorem 2.2. For any CG u and v , if $u \leq v$, then $\phi u \Rightarrow \phi v$.

Proof. See [Sowa, 1984] (p. 98) for detail.

2.3.4 Non-monotonic Reasoning – Defeasible Logic

In traditional knowledge representation approaches, such as first order logic and the methods introduced in the previous sections, a complete knowledge world is assumed. Under this assumption, once a logic statement is concluded to be true, then it will remain true even more information is added unless the conclusions that it was based on are rejected. This sort of logic and reasoning is termed as monotonic [Brachman and Levesque, 2004]. However, in a real world situation, it is seldom the case. In many situations, there is only incomplete information available, it may be because the required information is unavailable at the time or because the necessary response time means there is no time to find all the information. In this case, some conclusions drawn true may become false in the future when new information becomes available to the system. Sometimes the available information conflicts and leads to completely different conclusions. To deal with these situations, a more flexible knowledge representation and reasoning system is introduced, i.e. non-monotonic logic and reasoning [Brewka, 1991]. In this section we review a non-monotonic formalism introduced by Nute [Nute, 1994], called Defeasible Logic, which will be applied in our work later.

Defeasible Logic belongs to a class of non-monotonic approaches and was first developed by Nute (Nute, 1987).

Definition 2.5. A *defeasible theory* DT is a triple, $DT = (F, R, >)$, where:

- F : a set of facts;
- R : a finite set of rules;
- $>$: a superiority relation on R .

There are three kinds of rules in Defeasible Logic:

Strict rules: rules that are always true, denoted by $A \rightarrow q$, which reads as “if A , then definitely q .”

Defeasible rules: rules that can be defeated by contrary evidence, denoted by $A \Rightarrow q$, which reads as “if A , then typically q .”

Defeaters: rules that are used to defeat some defeasible rules, but not for drawing conclusions, denoted by $A \rightsquigarrow q$, which reads as “if A , then perhaps not q .”

The superiority relation “ $>$ ” defines which rule has a higher priority in the case where several rules have contrary conclusions. For example, a strict rule can be “Hens are birds”, formally written as:

$$hen(X) \rightarrow bird(X)$$

An example of the defeasible rule is “a bird normally can fly”, formally written as:

$$bird(X) \Rightarrow flies(X)$$

An example of the defeater is “a heavy bird may not be able to fly”, formally written as:

$$heavy(X) \wedge bird(X) \rightsquigarrow \neg flies(X)$$

If we have a set of rules with superiority relations, for example:

$$r_1: bird(X) \Rightarrow flies(X)$$

$$r_2: brokenWing(X) \Rightarrow \neg flies(X)$$

$$r_2 > r_1$$

then, we can conclude that although a bird normally can fly, but if it has a broken wing, normally it cannot fly, because the superiority relation between r_1 and r_2 , i.e. $r_2 > r_1$, decides that if r_1 and r_2 's conditions both are true, then r_2 's conclusion is considered.

Defeasible Logic is historically the first of a family of approaches based on the idea of *logic programming without negation as failure* [Antoniou et al., 2001]. The built in superiority handling mechanism and the computational efficiency make Defeasible Logic distinct from other non-monotonic approaches [Brewka, 2001]. In our work, Defeasible Logic based rules are used to describe service pre-conditions, effects and the conditions for service composition.

2.4 Semantic Web

The semantic web is first described by Tim Berners-Lee [Berners-Lee et al., 2001] (who is the person invented World Wide Web in late 1980s). The semantic web is not a separate web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation [Berners-Lee et al., 2001]. Traditional web documents are written in HTML which is designed to present information in a human readable way [HTML, 1999]. However, today's web contains an incredibly large amount of data. It is time-consuming if not impossible for a human to read through this large amount of information and locate required data. Therefore, the computational power of computers is needed to help people to retrieve information from the web. Unfortunately, the information representation on the web currently is only machine-processable, not machine-understandable. The consequence of this is that when searching for certain information, a large amount of irrelevant information is also returned. Therefore, an alternative way to represent the information is required, this is where the semantic web comes in. In the semantic web, information and knowledge are represented in both human and machine understandable ways so that the information can be processed automatically by machine in order to enhance the efficiency and accuracy of information retrieval. The technologies used in the semantic web to help to represent and retrieve knowledge are metadata, ontology, logic, and agents [Antoniou, 2004]. The semantic web is not a new invention. It is built upon existing technologies. Figure 2.7 illustrates the layered approaches used to make the semantic web happen [Antoniou, 2004].

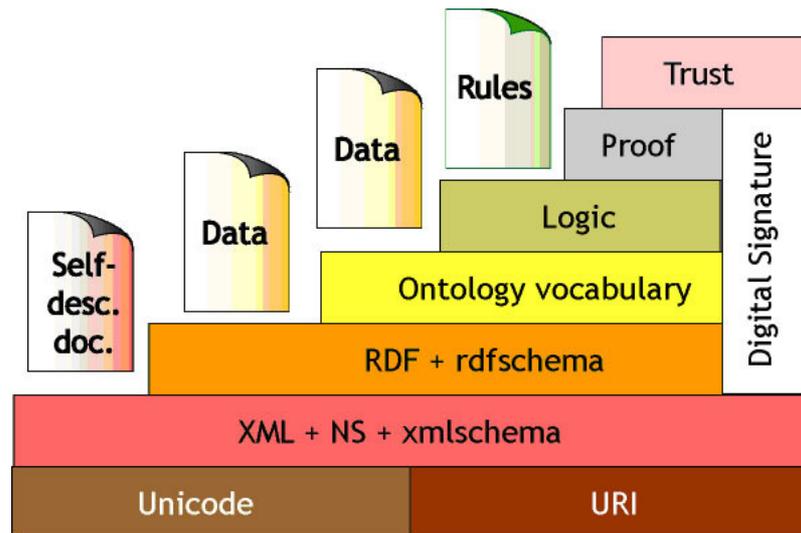


Figure 2.7. A layered approach to the semantic web [Antoniou, 2004].

In order to represent machine-understandable data on the web, the semantic web uses XML based languages rather than HTML to describe information and knowledge. RDF is an XML based semantic web language to describe resources on the web, and RDFS and OWL are ontology languages to describe complex ontologies.

2.4.1 RDF and OWL

RDF and RDFS

The Resource Description Framework (RDF) is a language for representing information about resources on the web [Manola and Miller, 2004]. It is an XML based language and each expression in RDF is a collection of triples. Each triple consists of a subject, an object and a predicate, this is also called a RDF graph [Klyne and Carroll, 2004], see the diagram below.

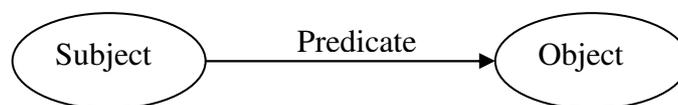


Figure 2.8. A RDF graph example [Klyne and Carroll, 2004].

Subject is the resource that the RDF wants to describe, predicate is a property used to describe a relationship between resources, and object is a value or another resource. For example, if we want to describe “a web site <http://www.example.org/index.html> as being created on 16 Aug, 1999”, then the RDF statement could be:

```
<rdf:Description rdf:about="http://www.example.org/index.html">
  <exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>
```

The vocabulary for the RDF statement, such as creation-date, is defined in a simple ontology language called RDF Schema (RDFS) [Brickley and Guha, 2004]. As RDF is a universal language to describe any resource in any domain by any user defined vocabularies [Antoniou and Harmelen, 2004], RDFS is needed to define the semantic meaning in a specific domain of the vocabulary used in a RDF. RDFS can build up a class and property hierarchy to define the meaning of concepts and their relationships. Furthermore, RDFS supports class and property inheritance. An example of an RDFS class and property is shown below to illustrate how to build up the class and property hierarchy.

```
<rdfs:Class rdf:ID="vehicle"/>

<rdfs:Class rdf:ID="car">
  <rdfs:subClassOf
rdf:resource="#vehicle">
</rdfs:Class>

<rdf:Property rdf:ID="hasWheels">
  <rdfs:domain rdf:resource="vehicle"/>
  <rdfs:range
rdf:resource="&rdf;Literal"/>
</rdf:Property>
```

OWL

From the above example we can see that RDFS can build up an ontology of concepts, but its semantic grammar is insufficient to describe complex relationships between concepts, such as disjointness or cardinality restriction. Therefore, a richer ontology language - web ontology language (OWL) [McGuinness and van Harmelen, 2004] is recommended by W3C for building up fully described ontologies. An OWL ontology includes the descriptions of classes, properties and their instances. It describes an ontology with a much richer syntax than RDFS. OWL has three sublanguages: OWL Lite, OWL DL, and OWL Full [Smith et al., 2004].

- **OWL Lite** supports users primarily needing a classification hierarchy and simple constraint features
- **OWL DL** supports users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be

computed) and decidability (all computations will finish in finite time) of reasoning systems.

- **OWL Full** is meant for the users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

The relationships among these three sublanguages [McGuinness and van Harmelen, 2004] are:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL provides a richer set of language features than RDFS for describing ontologies. The extra features includes *equivalentClass*, *equivalentProperty*, *allValuesFrom*, *maxCardinality*, *disjointWith*, *unionOf*, etc. [McGuinness and van Harmelen, 2004]. With these new features, OWL can describe an ontology with complicated relationships among the concepts defined in the ontology. An example of an OWL code fragment is shown below:

```
<owl:Class rdf:ID="Vintage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#vintageOf"/>
      <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Relationship between OWL and DL

The design of OWL is based on the expressive DL SHIQ. The sub languages OWL-Lit and OWL-DL can be viewed as expressive DLs, with an ontology being equivalent to a DL knowledge base [Horrocks et al., 2003]. Most of the properties in OWL can be translated into a DL expression and the reasoning of OWL uses DL reasoner, which means that OWL and DL are logically equivalent. The table below shows the corresponding DL expressions for some OWL properties [Horrocks, 2005].

Table 2.1. Mapping between OWL and DL

OWL Syntax	DL Syntax
<i>subClassOf</i>	$C_1 \sqsubseteq C_2$
<i>equivalentClass</i>	$C_1 \equiv C_2$
<i>subPropertyOf</i>	$P_1 \sqsubseteq P_2$
<i>equivalentProperty</i>	$P_1 \equiv P_2$
<i>transitiveProperty</i>	$P^+ \sqsubseteq P$

2.5 Semantic Web Services

As the number of available Web services increases exponentially, automatic service discovery and composition tools become essential for efficiently using Web services. Furthermore, the loosely coupled and self-contained features give Web services the ability to be dynamically invoked at runtime by other applications or other Web services. Achieving dynamic service invocation at runtime is also an issue of automation. To achieve automatic Web service processing, service capabilities and service requirements need to be described in such a way that the description can be processed by computer without or with minimum human intervention. Due to the lack of formally defined semantics in WSDL based service descriptions, enormous research efforts are being put into the development of a semantic rich semantic framework on describing Web services. An emerging research area, i.e. Semantic Web Services, has gained considerable attention and has become the most promising way of achieving automatic service discovery, invocation, and composition. Semantic Web Services combine the semantic web and Web services technology to make a service description understandable not only by the humans, but also by computer software [McIlraith et al. 2001]. Lara et al. [Lara et al., 2003] have stated a set of criteria that should be addressed in a Semantic Web Services description in order to fully describe a service's capabilities, such as its pre-condition, post-condition, textual description, and identifier. Several major Semantic Web Services frameworks have been proposed, such as OWL-S [Martin et al., 2004], WSMF [Fensel & Bussler, 2002], WSMO [Fensel et al., 2007], and WSDL-S [Akkiraju et al., 2005]. The main idea of the existing frameworks is to build a semantic layer either on the top of WSDL or to be integrated into WSDL to semantically describe the capabilities of Web services so that a software agent or other services can find out about a Web service's capabilities and how it can be used.

2.5.1 OWL-S

In order to overcome the lack of semantics in WSDL, Martin et al. [Martin et al., 2004a] have developed an OWL based semantic language called OWL-S to add a semantic layer on the top of the WSDL to describe a Web service. The aim of OWL-S is to enable automatic Web service discovery, invocation, and composition and interoperation [Martin et al., 2004b]. The upper ontology of OWL-S consists of four classes for describing Web services. One class called **Service** provides an organisational point of reference for a declared Web service. It has three properties, known as *presents*, *describedBy*, and *supports*. The other three classes **ServiceProfile**, **ServiceModel**, and **ServiceGrounding** are the ranges of the Service class's three properties, see Figure 2.9 for the relationship among these classes [Martin et al., 2004b].

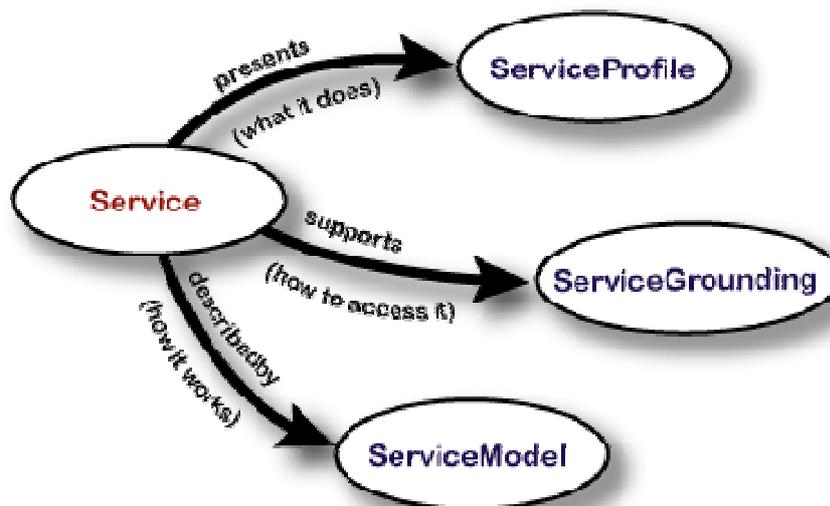


Figure 2.9. Top level of the service ontology [Martin et al., 2004b].

These classes provide three essential types of knowledge for a Web service, these are:

1. What does the service provide for the service requester? The answer is given in the **ServiceProfile** class.
2. How is it used? The answer is given in the **ServiceModel** class.
3. How does the service requester interact with the service? The answer is given in the **ServiceGrounding** class.

All the vocabularies used in the classes **Service**, **ServiceProfile**, **ServiceModel**, and **ServiceGrounding** are defined in another set of ontologies, **Service**, **Profile**, **Process**, and **Grounding** [OWL-S Upper Ontology, 2006].

ServiceProfile provides three types of information about a service [Martin et al., 2004b]. The first type of information consists of service name, service description, and service provider's contact details. The second type of information is the service's functional description which has been expressed in terms of the transformation produced by the service. Also the pre-conditions for invoking the service and the results after the service is executed are provided. The third type of information is the description of the properties that are used to describe features of the service which includes the category of the service, the quality rating of the service, and the parameters for response time or geographic availability of the service etc.

In OWL-S, how a service requester can interact with a web service is represented by a process. This process information is captured in the **ServiceModel** class. A process is the building block of a web service. It is not the actual implementation of the web service, but a specification of the way a service requester can interact with the service. The effects of a process are twofold: on one hand, it provides some new information based on its own information, i.e. the inputs and outputs of a process; on the other hand, it changes the state of the world, e.g. the credit card has been debited. This state change can be described by the pre-conditions and effects of a process. Three types of process have been modelled to represent the interaction between a service requester and a web service. *Atomic* processes represent the processes where service requesters can invoke the service and get the result in a single interaction. *Composite* processes are collections of *atomic* processes and *composite* processes. The sub-processes of a *composite* process are composed together according to business logic. How information is passed between the sub-processes is based on the control structures and conditions proposed in a *composite* process. Ten control structures are introduced in the service process model [Martin et al., 2004a], including Sequence, Split, Split + Join, Choice, Any – Order, Condition, If-Then-Else, Iterate, Repeat -While, and Repeat – Until.

Simple processes are abstract processes with no instances and no associated grounding and therefore cannot be invoked. A *simple* process is conceived as an

atomic process and the purpose of the *simple* process is to give an abstract view of an *atomic* process or a simplified view of a *composite* process. It can be realised by (the *realizedBy* property) an *atomic* process or expanded to (the *expandsTo* property) a *composite* process.

ServiceGrounding provides the information about how a service requester can actually invoke a Web service. The provided information includes communication protocol, message format, serialisation, transport, and addressing [Martin et al., 2004a]. In the Web services infrastructure, the communication information unit is the SOAP message. OWL-S uses the input and output properties to represent the communication, rather than directly giving the specifications for SOAP messages. Therefore, the role of service grounding is to give the mapping information from the process inputs and outputs to the concrete messages. By having this mapping information, a service requester can invoke the service. The service grounding only provides the grounding details for atomic processes. For a composite process the service grounding gives a list of referencing links to the atomic processes that have been composed into the composite process.

2.5.2 WSMF and WSMO

The goal of the Web service Modelling Framework (WSMF) is to provide a full-fledged description framework for Web services to enable the required automation [Bruijn et al., 2005]. The key principles of WSMF are strong decoupling and strong mediation [Fensel and Bussler, 2002]. Strong decoupling is to decouple the various components used to build an e-commerce application [Fensel and Bussler, 2002]. Strong mediation is to build up mediation services to the heterogeneity so that anybody can talk to anybody [Fensel and Bussler, 2002]. Based on these two principles the components of the framework are built. There are four main components in this framework, which are Ontologies, Goal Repositories, Web service, and Mediator.

Ontologies

In the WSMF, ontologies are used to define terminology so that other components of the WSMF can communicate based on shared or linked terminology [Fensel and Bussler, 2002].

Goal Repositories

Goal repositories in the WSMF give the descriptions of the goals that clients may have when they consult Web services. This framework separates goal descriptions from Web service in order to achieve the many to many relationships between Web services and goals; one goal can be achieved by many services and vice versa [Fensel and Bussler, 2002]. There are two elements in a goal description: the pre-condition and the post-condition. The pre-condition states what a Web service requires to provide its service. The post-condition states what the response to a Web service's inputs will be.

Web service

The WSMF developers state that many Web service description languages have distinguished between elementary and complex Web services in an incorrect way [Fensel and Bussler, 2002]. In their opinion, the thing that complicates Web services is not the Web services themselves, but their external visible descriptions or interfaces. Therefore, in this framework a web service is described by a black box description which contains thirteen aspects as follows.

- **Name** is assigned to a Web service as a unique identifier.
- **Goal reference** is a reference linked to the goal in the goal repositories component.
- A Web service also has **pre-condition** and **post-condition** and they could be a stronger pre-condition or a weaker post-condition of a goal.
- **Input data** and **output data** define the data structure and types required by a Web service. The input data and output data are passed through the components called input ports and output ports.
- **Error data** can be returned through the error ports to indicate the problems or the error states of a service invocation.
- In order to hide the details of how a service can achieve its result by invoking other services, the WSMF defines the **invoked Web service proxy**. A proxy can be a goal definition or a specification of the required service so that the service can be located at runtime.
- **Data flow** describes how a service's input and output ports are connected to the invoked Web service proxy's input and output ports.

- **Control flow** describes the connection sequence of the invoked Web service proxies.
- Invoked Web services can fail and return an error or exception code. In this case, depending on the error code, **exception handling** must take place in order to deal with the error situation.
- An **acknowledgement of message understanding** is sometimes required in a business integration process to indicate that a message has been understood or not.
- A Web service description needs to relate to **the message exchange protocol**. The message exchange protocol indicates how the messages are exchanged over reliable and unreliable networks.
- Finally the Web service description also needs to provide appropriate **non-functional properties**, such as geographical location, price, or average/maximum performance time.

Mediator

In order to overcome the heterogeneity of data structures, business logic, and message exchange protocols, and achieve dynamic service invocation, WSMF introduces a mediator component. There are three kinds of mediators: **data structure mediator**, **business logic mediator**, and the **exchange protocols mediator**. The idea of the mediation is to resolve the standard heterogeneous problems described earlier, but it is difficult to fully implement. This framework has not provided details on how to create or design these mediators.

Web service Modelling Ontology (WSMO) [Roman et al., 2005] is based on the WSMF and refines and extends WSMF to provide a formal ontology and a set of languages to semantically describe Web services. The aim of WSMO is to provide the conceptual and technical means to realise Semantic Web Services and improve the cost-effectiveness, scalability and robustness of the current solutions [Roman et al., 2005]. It defines the ontological specifications for the core components of the WSMF. The top-level elements in WSMO are similar to the core components in WSMF which are **Ontologies**, **Web services**, **Goals**, and **Mediators**. WSMO uses class and properties to describe an element, for example, the Web service class is described by

five properties, and the range of the properties is stated by **type**. See the code listed below:

```
Class webService
  hasNonFunctionalProperty type nonFunctionalProperty
  importsOntology type ontology
  usesMediator type {ooMediator, wwMediator}
  hasCapability type capability multiplicity = single-valued
  hasInterface type interface
```

In WSMO four kinds of mediators are concretely specified:

- **OO (ontology to ontology) Mediators** are used to resolve mismatches between ontologies.
- **GG (goal to goal) Mediators** are used to resolve terminology mismatches between goals.
- **WG (Web service to goal) Mediators** are used to resolve terminology mismatches between Web services and goals.
- **WW (Web service to Web service) Mediators** are used to establish interoperability between Web services.

In each element's description, which kind of mediators is required is clearly specified. For example, in the Web service element description, the required mediators are the OO Mediator and the WW Mediator. However, similarly to WSMF, the concepts of mediators are still underspecified and it is not clear whether the mediators should be a set of rules or a service [Paolucci et al., 2004]. It is also unclear whether the mediators should be implemented on the client side or the server side.

2.5.3 WSDL-S

WSDL-S is proposed by the LSDIS (Large Scale Distributed Information Systems) Lab, University of Georgia, USA. The way it integrates semantics into Web service descriptions is different from the frameworks discussed previously. The design principles of the WSDL-S are listed below:

- Build on the existing Web services standards.
- The mechanism for annotating Web services with semantics should be independent of the semantic representation language.

- The mechanism for annotating Web services with semantics should allow the association of multiple annotations written in different semantic representation languages.
- Support semantic annotation of Web services whose data types are described in an XML schema.
- Provide support for rich mapping mechanisms between Web service schema types and ontologies.

The first design principle of WSDL-S is “*Build on existing Web services standards*”. The designers of WSDL-S consider a very important requirement for Semantic Web Services is alignment with the existing Web services standards. Therefore, their solution is to design the semantic model separately, which can be used to annotate the information provided in WSDL using the WSDL’s extendible elements. The semantic model can be defined in any languages, for example RDF, OWL, XSLT, or UML.

The elements provided by WSDL-S, which can be plugged into the standard WSDL documents, are:

- **modelReference**: an extension element that is used to handle one to one mapping between the schema elements and the concepts in a semantic model.
- **precondition** and **effect**: two new elements specified as the sub-elements of the *operation* element in WSDL. These two elements are used to describe pre-conditions and effects of an operation.
- **category**: an extension attribute of the *interface* element in WSDL. It contains the category information for publishing services to a UDDI registry.

WSDL-S also supports directly publishing to UDDI and enables dynamic discovery of services. However, as WSDL-S is tightly bound with WSDL, its expressiveness is limited. For example, it cannot be used to describe a composite service with logical control structures.

2.6 Service Context

A service is not an independent function unit. To realise its functionality, a service must interact with its outside environment. A service could behave differently in different situations and environments. Therefore, context awareness is a very

important issue that needs to be considered in service-oriented computing. In this section, we will discuss the context concept in general and how it is applied in the service-oriented paradigm.

2.6.1 What is Context?

The term “context” is often used in relevant computer science literature. However, its meaning is mostly dependent upon the understanding of the author and its usage is often implicit. Many research works use the term “context” or “contextual situation”. Here we give a summary of the typical definitions that have been proposed in the literature for “context”:

Dey and Abowd [Dey & Abowd, 2000] define context as *“any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”*;

Brown et al. [Brown et al., 1997] from the user’s perspective define context as the users’ location, who the users are with, and what the time of day is etc.

Ryan et al. [Ryan et al., 1997] consider the information about the environment of the computers as context, such as location, time, temperature, or user identity etc.

The various definitions demonstrate that “context” is a very complex term and may be interpreted in many different ways. From the above definitions we can summarise that context is mainly understood as the environment of either computer software or a computational device, for example:

- **Computing environment:** available processors, devices accessible for user input and display, network capacity, connectivity, and costs of computing.
- **User environment location:** collection of nearby people, and social situation
- **Physical environment:** time, temperature, etc.

However, context could mean not only the environment, but also the conceptual relationships between one object and the other objects in a certain situation. Mineau and Gerbe [Mineau and Gerbe1997] from the semantics and natural language angle define context as a set of assertions whose existence depends upon some other

assertions which describe the premises of its existence. For example, the statement “Peter loves Mary” may not be universally true. However, if the context is added to this statement “Mary thinks that Peter loves Mary”, then it becomes universally true.

2.6.2 Service Context

Current research efforts to add context to the SOA paradigm are mainly concentrating on the following three aspects: context aware service execution, context aware service discovery/provision, and context aware service composition. In the following, we discuss each aspect in detail.

Keidl and Kemper [Keidl and Kemper, 2004] propose a framework for development and deployment of context-aware adaptable Web services. They add contextual information into SOAP messages so that Web services can behave differently based on different contextual information without modifying the actual infrastructure or implementation of the Web services. An illustration of the modified SOAP message is shown in Figure 2.10. The contextual information they consider in their solution is the service clients’ information that can be utilised by Web services to adjust their executions to perform personalised behaviours.

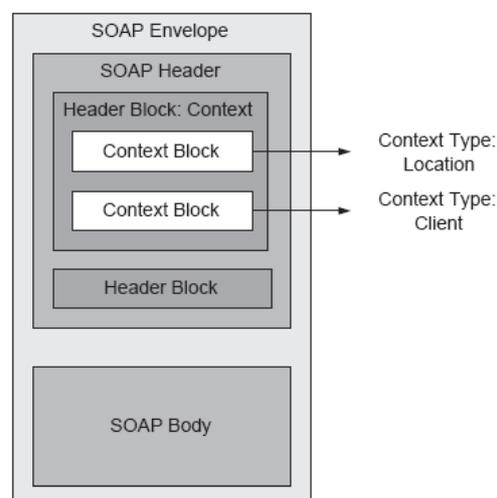


Figure 2.10. A SOAP message with context information [Keidl and Kemper, 2004].

The proposed framework also has a set of context components to process the contextual information. The contextual information is processed by either context services, a context plug-in, or the actual invoked Web service. They also define the precedence rules used to determine which component should be used for processing. The context processing can be done either locally or remotely.

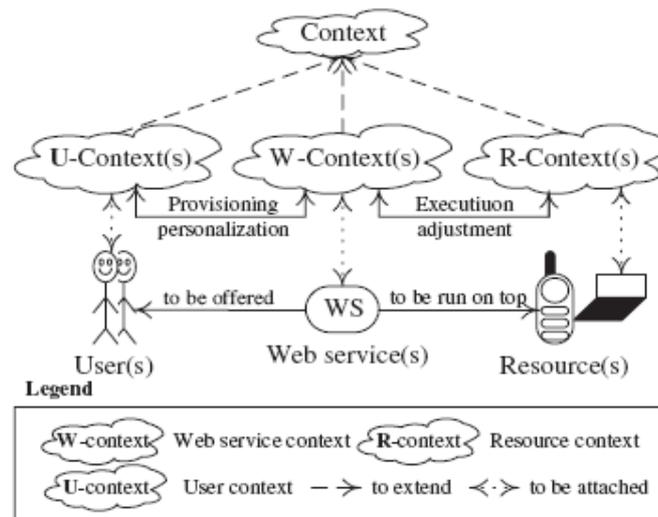


Figure 2.11. Context-aware personalised Web services [Maamar et al., 2005a].

Maamar et al. [Maamar et al., 2005a] categorise service context into *U*-context (User context), *W*-context (Web Service context), and *R*-context (Resource context). Based on these three types of context, Web services can be deployed and supplied to satisfy the user's personalised requirements, see Figure 2.11. The *U*-context keeps tracking the user's preferences in terms of execution time or location. The *W*-context looks into the execution constraints of Web services based on the service user's preferences. The *R*-context tracks current resource information. They also define a set of policies to ensure that a Web service still does what it is supposed to do after personalisation. The *Consistency* policy checks the status of the Web service after personalisation. The *Feasibility* policy checks if a personalised Web service can find a resource on which it will be executed according to the constraints of time and location. Finally, the *Inspection* policy ensures that a personalised Web service is being deployed according to the adjusted specification. In their earlier work [Maamar et al., 2005], they apply agent technology on context-aware service composition, in which they introduce two new types of context, the *I*-context (Web service instance context) and the *C*-context (composite service context). When a composite service request comes in, the different kinds of agent that manage different contextual information work cooperatively together to ensure that a composite service is created correctly.

Zhou et al. [Zhou et al., 2008] view context as the constraints and changeability of QoS (Quality of Service) values. In the real business world, the QoS values of Web services are heavily affected by service providers and the combination of the sub-

services in a composite service. For example, one service provider may announce that their Web services are only compatible with some specified vendor's Web services, in order to exclude other candidates; another service provider may claim that if you buy service *A* from them, you will get service *B* cheaper. To tackle this problem, they proposed a context-sensitive QoS model that considers the effect of the QoS values from the service providers. In the context-sensitive model, QoS objective functions are proposed with a context effect indicator function $g()$. For example, the objective function for the cost of a service is proposed as follows:

$$\sum_{i=1}^n (c_i + g(c_i, \rho_k))$$

where c_i is the cost of each sub-service of a composite service and $g(c_i, \rho_k)$ represents how c_i is affected under a certain combination of services ρ_k . Therefore, under the different user QoS requirements and service providers' constraints, different composite services can be generated.

2.6.3 Significance and Deficits

As a functional unit, a Web service cannot achieve its functionality without interacting with the external environment and this interaction can be greatly affected by the Web service's surrounding context, either the physical context or the conceptual context. By reviewing the literature in the previous sections, we can see that the context has already been considered as an important issue in Web service provision, composition, and execution. Current research work in the Web services domain mainly addresses context as environmental factors, such as physical environment, user profile, and computational environment. Although as mentioned by Mineau and Gerbe [Mineau and Gerbe1997] context can also be conceptual relationships between concepts or assertions, only few researchers have applied this notion of context in the Web services domain. As mentioned by Du et al. [Du et al., 2007a], this notion of context is important in fully describing a Web service because to comprehensively understand a Web service we cannot ignore its relationships with other services or entities. In our work, we consider context as the relationships between a service and other services or entities at either the conceptual level or the instance level. We will discuss this in detail in Chapter 3.

2.7 Service Composition

Web service composition has become a major research area [Dustdar and Schreiner, 2005] whose outcome can support efficient and effective business integration so that enterprises can rapidly deliver better business services to their customers with low cost. Service composition is also crucial in achieving SOA because it enables new business values to be created from the existing resources [Huhns and Singh, 2005]. From the business angle, the enterprises need to work cooperatively with their business partners and customers [Sayah and Zhang, 2005]. The on-demand model proposed by SOA provides more freedom for an enterprise to dynamically interact with their business partners to provide their own services. Therefore, a service provided by an enterprise can be achieved by many services from different service providers therefore the enterprise can concentrate on their core businesses. This is the way that service composition technology can help enterprises deliver their business services efficiently and achieve their business objectives.

In this section, we review some common techniques for Web service composition, such as manual service composition, automatic or semi-automatic service compositing, and workflow based service composition.

2.7.1 Manual Service Composition – BPEL4WS

Manual service composition is the way that the composite service or workflow designers manually analyse the task of a composite service and decide what sub-services need to be involved in the composite service to achieve its desired goals and how those sub-services should be linked. BPEL4WS [Andrews et al., 2003] is a manual service composition model and language that can be used to build composite services.

Web service technology is a platform for enterprise application integration. However, it is too simplistic to achieve complex business interaction because it is essentially a stateless model of synchronous or uncorrelated asynchronous interactions [Andrews et al., 2003]. A business interaction model typically needs peer-to-peer message exchanges, both synchronously and asynchronously, within stateful, long-running interactions involving two or more parties [Andrews et al., 2003]. Transaction management needs to be supported because errors may happen during the interaction.

Therefore, Microsoft, IBM, and other IT companies cooperatively developed a language – BPEL4WS [Andrews et al., 2003] to support constructing stateful composite business processes. This language is based on two existing process modelling language WSFL [Leymann, 2001] and XLANG [Thatte, 2001] and includes all the features from these two languages. BPEL4WS is an XML based language and is used to describe the following aspects of a business process.

- **Partners:** a set of Web services used to achieve the business process.
- **Containers (called Variables in V1.1):** providing the means for holding messages that constitute the state of a business process.
- **faultHandlers:** routine for exception handling.
- **CompensationHandler:** compensations to be performed when a transaction rollback happened.
- **EventHandlers:** routines to handle external asynchronous events.
- **CorrelationSets** precedence and correlations that cannot be expressed in the main interaction logic.
- **Main process logic:** a set control flow structures to form primitive activities into a complex algorithm. It includes the following control structures:
 - **Sequence**
 - **While**
 - **Switch**
 - **Pick**
 - **Flow**
- **Control structures related to atomic actions:** invoke, receive, reply, wait, assign, throw, terminate, and empty.

A simple example shown below illustrates the structure of a BPEL4WS document

```

<!ENTITY BPEL http://schemas.xmlsoap.org/ws/2002/07/business-process
<process name="simple" targetNamespace="urn:simple:stockQuoteService"
      xmlns:tns="urn:simple:stockQuoteService"
      xmlns:sqp=http://tempuri.org/services/stockquote
      xmlns=&BPEL; />
  <containers>
    <container name="request" messageType="tns:request"/>
    <container name="response" messageType="tns:response"/>
    <container name="invocationRequest" messageType="sqp:GetQInput"/>
    <container name="invocationResponse" messageType="sqp:GetQOutput"/>
  </containers>
  <partners>
    <partner name="caller" serviceLinkType:"tns:StockQuoteSLT"/>
    <partner name="provider" serviceLinkType:"tns:StockQuoteSLT"/>
  </partners>
  <sequence name="sequence">
    <receive name="receive" partner="caller" portType="tns:StockQuotePT"
      operation="wantQuote" container="request" createInstance="yes"/>
    <assign>
      <copy>
        <from container="request" part="symbol"/>
        <to container="invocationRequest" part="symbol"/>
      </copy>
    </assign>
    <invoke name="invoke" partner="provider" portType="sqp:StockQuotePT"
      operation="getQuote" inputContainer="invocationRequest"
      outputContainer="invocationResponse"/>
    <assign>
      <copy>
        <from container="invocationResponse" part="quote"/>
        <to container="response" part="quote"/>
      </copy>
    </assign>
    <reply name="reply" partner="caller" portType="tns:StockQuotePT"
      operation="wantQuote" container="response"/>
  </sequence>
</process>

```

However, BPEL4WS is heavily based on WSDL and models static service composition only. If a business process needs to dynamically invoke required services at runtime, BPEL4WS is not a suitable solution. BPEL4WS is a communication oriented language and the communication between activities is established through sending and receiving messages [Wohed et al., 2003]. BPEL4WS is a low level business process model or service composition language because the messages used to communicate are technical information and no semantics involved.

2.7.2 Automatic or Semi-Automatic Planning based Service Composition

Agarwal et al. [Agarwal, 2004] propose a semi-automatic service composition framework, called OntoMat-Service. In the framework, a service browser is provided

with a function that can convert a service's WSDL document and its relevant advertisement into a human readable HTML web page. The users can then use their own ontology to annotate the relevant information in the WSDL. An annotator provides the functionality to translate the terminologies used to describe the service into the client side ontology. If inconsistency exists between the client side terminologies defined by the user ontology and the service terminologies defined by the service provider ontology, some mapping rules will be automatically generated by the annotator to map the equivalent terminologies. This is called deep annotation and the mapping rules are defined using F-Logic. The mapping rules enable third parties to invoke the Web service on the basis of the client side ontology. After the semantic annotation of the Web services, the users can drag and drop services or operations into the composition panel. The Web service planner will automatically generate possible service flows based on the Web service's pre and post conditions, input and output data types, and the goal of each participated Web service and the composite service. The OntoMat-Service framework has achieved some automation such as terminology mapping and service flow planning, but is still a semi-automatic service composition framework because the selection of the services and annotation of the services have to be done manually.

In order to achieve automatic service composition, first the services have to be described semantically so that the description can be processed by the machine. Secondly, automatic planning techniques have to be used to generate service flows based on goals [Rao and Su, 2004]. Wu et al. [Wu et al., 2003] propose an automatic Web service composition solution. Their solution combines the semantic Web service description model DAML-S (the previous version of OWL-S) and SHOP2 [Nau et al., 2003], an HTN (Hierarchical Task Network) [Sirin et al., 2004] planning system. DAML-S is the only Web service language that claims a direct connection with AI planning [Rao and Su, 2004]. HTN is an AI planning methodology that creates plans by task decomposition [Sirin et al., 2004]. SHOP2 is a domain independent planning system, so before starting planning, a domain has to be specified to the system. In their work, the domain for the SHOP2 system can be automatically translated from the DAML-S model. Therefore, a DAML-S Web service composition problem can be encoded as a SHOP2 planning problem. There are four components used to achieve the automatic service composition. A DAML-S to SHOP2 translator takes the

DAML-S process definition file as an input and generates a SHOP2 domain file as output. A monitor is used to handle SHOP2's calls for external information on provided services during planning. A SHOP2 to DAML-S converter converts the plans produced by the SHOP2 system into a DAML-S composite process model that can be directly executed by the DAML-S executor. However, a limitation of this system is that all the Web services have to be well described in DAML-S. Otherwise automation is not possible, in addition a shared ontology is required between service providers and consumers.

2.8 Semantic Similarity Calculation

Semantic similarity calculation is an important technique in automatic Web service discovery and composition. It helps a search engine locate the most suitable services for a service request. In this section, we will review some of the common methods and algorithms that are used for calculating semantic similarity and ranking.

2.8.1 Ontology based Methods

Wu and Palmer [Wu and Palmer, 1994] address how to calculate the semantic distance between two concepts within a domain ontology. The similarity of two concepts is defined by how closely they are related in the hierarchy, i.e., their structural relations.

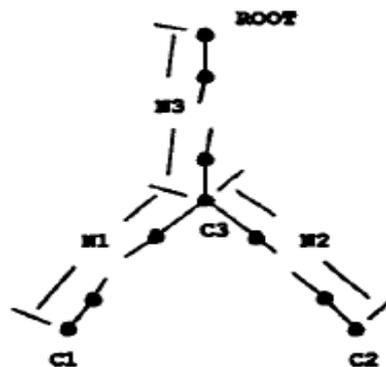


Figure 2.12. The concept similarity measure [Wu and Palmer, 1994].

Suppose we have the ontology illustrated in Figure 2.12, then the similarity between the concepts C_1 and C_2 is calculated as follows:

$$\text{sim}(C_1, C_2) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3}$$

where, C_3 is the least common super-concept of C_1 and C_2 ; N_1 is the number of nodes on the path from C_1 to C_3 ; N_2 is the number of nodes on the path from C_2 to C_3 ; and N_3 is the number of nodes on the path from C_3 to root.

Paolucci et al. [Paolucci et al., 2002] propose some semantic matching and ranking algorithms for Web service capabilities. Their solution is based on the OWL-S framework. The main rationale behind their algorithms is a service advertisement matches with a service request if the provided service can be of some use to the requester. It means that in their solution a service is not necessarily exactly matched with a request, a partial match, e.g. where one of the outputs is matched or outputs partially satisfy a requirement, is also acceptable, however, the degree of the similarity will be lower than for a full match. They provide a ranking algorithm to rank services according to their semantic similarity to the service request. The ranking algorithm is illustrated as below:

```

degreeOfMatch(request, advertisement)
  if request equal advertisement return exact
  if request subClassOf advertisement return exact
  if advertisement subsumes request return plugin
  if request subsumes advertisement return subsumes
  otherwise return fail

```

The degree of similarity is determined by the minimal distance between concepts in an ontology. There are four degree assignments:

- **Exact:** if a request and a service advertisement are matched or if a request is a subclass of the service advertisement, the service is assigned with “exact” matching degree.
- **Plug in:** if a service advertisement subsumes a request, then the service advertisement is a set includes the request. However, this relationship between the service advertisement and the request is not as strong as subClassOf. The service may not provide exactly what the request wants. In this case the service is assigned with the “Plug in” matching degree.

- **Subsumes:** if a request subsumes a service advertisement, then the service does not completely fulfil the request. The request uses this service to achieve its goal, but it certainly needs other complement services or resources.
- **Fail:** if there is no subsumption relationship between a service advertisement and a request, then the service fails to fulfil the request.

The solution proposed by Paolucci et al. has made a good attempt at applying semantic web technology into Web services and matchmaking. However, the algorithm they proposed is too simplistic because it only considers the input and output semantics of a service. In a realistic case, a service selection decision should be made based on many factors, such as pre- and post-conditions, quality of service (QoS), business rules, and service providers.

2.8.2 Vector based Methods

When comparing two objects, each of which has more than one property that needs to be compared in order to measure the similarity between these two objects, e.g. a service requirement and a Web service or between two documents, vector based similarity measurement methods are more appropriate. The idea of the vector based similarity measurement is to convert the properties of the objects into vectors and apply well known mathematical operations on vectors to compute the similarity between the vectors and accordingly measure the similarity between the objects.

Suppose we have two objects O_1 and O_2 and their property vectors $V_1=(t_1, t_2, \dots, t_j)$ and $V_2=(t_1, t_2, \dots, t_j)$ then we can apply the following methods proposed in [Frakes and Baeza-Yates, 1992] [Berry et al., 1999] [Agosti and Smeaton, 1996] to measure the similarity between O_1 and O_2 .

Dice's Coefficient:

$$Dice(O_1, O_2) = \frac{2|V_1 \cap V_2|}{|V_1| + |V_2|}$$

Jaccard Coefficient:

$$Jaccard(O_1, O_2) = \frac{|V_1 \cap V_2|}{|V_1 \cup V_2|}$$

Overlap Coefficient:

$$Overlap(O_1, O_2) = \frac{|V_1 \cap V_2|}{\min(|V_1|, |V_2|)}$$

Cosine Similarity:

$$Cos\theta_{V_1, V_2} = \frac{V_1 \cdot V_2}{\sqrt{V_1^T} \sqrt{V_2^T}}$$

Term Weighted Cosine Coefficient:

$$Cos\theta_{V_1, V_2} = \frac{\sum_{k=1}^j w_{1k} \cdot w_{2k}}{\sqrt{\sum_{k=1}^j w_{1k}^2} \cdot \sqrt{\sum_{k=1}^j w_{2k}^2}}$$

where, the variable w_{ik} represents the vector V_i 's k -th term's weighted value. It is normally calculated based on the importance of the properties of an object during matchmaking, e.g. a word that appears many times in a document may have higher weight than the words that seldom appear in a document.

These methods have been widely adopted in the areas, such as information retrieval. In our work, we will use some of these methods in Web service discovery.

2.9 Summary

In this chapter, we have reviewed the research areas that are most relevant to our work, such as Web services, Semantic Web Services, service context, and service composition. We have also reviewed the technologies that can help us to develop our solutions, such as CG, Defeasible Logic, and semantic similarity measurement methods.

From the literature survey we observed that there are many problems in the Web services domain, especially in the area of service description, that still need to be solved, especially in service description. We have summarised two major gaps between the existing work and what is needed:

- **Inter-service relationships are ignored.** According to the literature we have reviewed, all the existing works view Web services as isolated individuals without considering their relationships in business domains or business scenarios. In our opinion, the relationship between services is an important characteristic of the services involved in them. It enhances the semantic description of the service capabilities.
- **Service context regarding the use of services is not addressed.** The context of services discussed in the literature mostly considers the runtime environment of services. None of the context in the existing works takes into account how services are used, i.e. the usage context. In our opinion, the usage context of a service is a crucial part of the service description. On one hand, it can help a service search engine to better match services with user requirements. On the other hand, it bridges the gap between the user requirements and the service descriptions based on technical specifications.

In this thesis, we analyse and bridge these gaps through a Context based Semantic Service Description Framework (CbSSDF). CbSSDF integrates service usage context into service descriptions and takes into account inter-service relationships, in order to improve the efficiency and effectiveness of service discovery and composition. In the following three chapters, we will discuss the framework and its related concepts in detail.

Chapter 3: Service Usage Context

In this chapter, we introduce the concept “Service Usage Context”, which is the foundation of the service description framework that will be discussed in chapter 4.

3.1 Overview

In this chapter, we discuss the important concept of “Service Usage Context”, which is the fundamental concept of the service description framework that we will discuss in the next chapter. Web services can be considered as remote function units that provide certain capabilities. The loose coupling feature of Web services gives them the ability to be invoked independently without taking into account the heterogeneity of deployment platforms and programming languages. However, this does not mean that a Web service can be invoked in any situation where its capabilities are required. For example, a motor vehicle repair service is not suitable for a broken toy car and a customer data provision service can only be used within an enterprise, otherwise the data may not be relevant or the data protection policy of that enterprise may be breached. When a Web service is created, based on its functionality there are some situations or scenarios in which the service can best participate, i.e. under which context the service should be used. The other elements in these scenarios for the service can be other services, service users, and software agents. Such 'best-fit' scenarios for a service form the *Service Usage Context* (SUC) of the service. We model the SUC at two levels: the conceptual level and the instance level. In our opinion, to fully describe a service, we need to address not only the capabilities of the service, but also how the service is related to its SUC. The SUC is mainly about how a service is connected to other services and entities. This is not in conflict with the loosely coupled feature of Web services because the loosely coupled feature only deals with technology independency, not the conceptual relationship or usage dependency. In the rest of this chapter, we will discuss the SUC and its relevant concepts in detail.

The content of this chapter is organised as follows: we first introduce some basic terms to be used in this and following chapters in order to make the later discussion clearer, and then discuss how the idea of the SUC is developed; finally we give the definitions of SUC and its relevant concepts.

3.2. Glossary

Several terms are frequently used in the later discussion in the thesis. Many of them have been widely discussed in the literature and have been given slightly different meanings. It may cause confusion without a clear clarification. In this section, we list them here and explain what these terms mean in this thesis.

Service: Service is a more general term than Web services. A service can be any business component that provides functionalities over the network through standardised interfaces. A service's functionalities can be provided by the service itself or by a combination of other services through the service's interfaces. Although this thesis mainly uses Web services as examples, the applicability of the work proposed in the thesis is not limited to Web services. Therefore, sometimes the term "service" is used instead of Web services.

Service Ontology: Based on the capabilities of a service in a given business domain, a data model that represents a set of concepts about the classification of the service's capabilities and the relationships among those concepts can be created, that data model is called the service ontology. An example is shown in Figure 3.1.

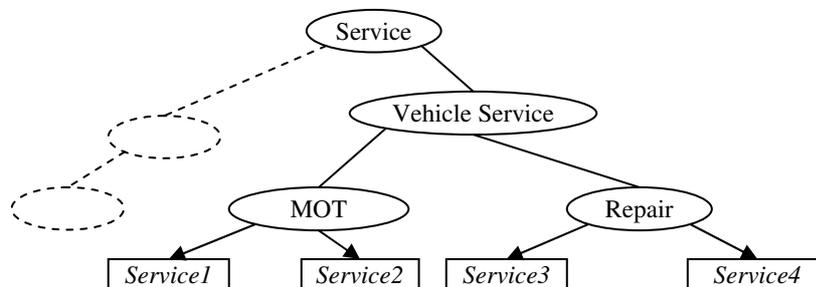


Figure 3.1. A service ontology example.

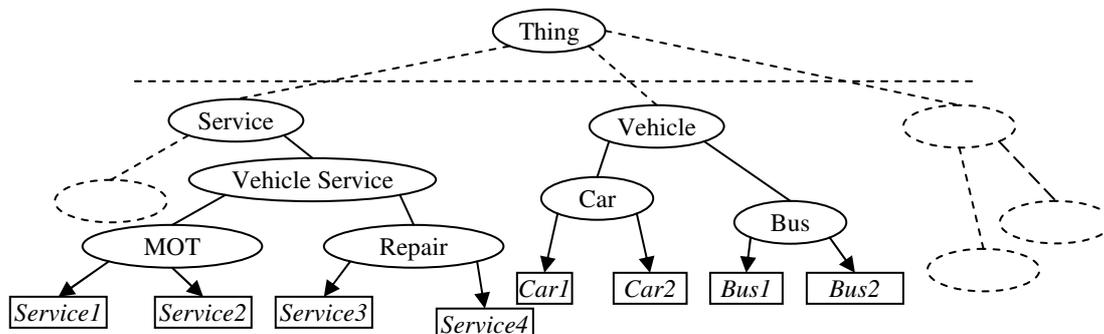


Figure 3.2. A domain ontology example.

Domain Ontology: Domain ontology is a comprehensive ontology that includes the service ontology of a given business domain and other concepts that are relevant to the business domain. An example is shown in Figure 3.2.

Service Concept and Instance Service: A service concept is a concept in the service ontology. If a service concept has corresponding services, these services are called the **instance services** of the service concept. A service concept can have direct instance services, indirect instance services, or both. The indirect instance services are the instance services of the service concept's sub-concepts in the service ontology. The reason for introducing service concept is that by having a service concept, we can discuss the usage of a service at a conceptual level without worrying about the technical details.

Domain Concept: a domain concept is a concept in the domain ontology, but not in the service ontology.

User/Service User: the term user or service user represents the actual end users, i.e. the human users, who require the functionality provided by a service to achieve their goals.

Service Requester: Service requester is a broader term than the service user. It can be a service user, a software agent, or a service, which requires the functionality provided by a service.

3.3 Context and Schemata

The term “context” is often used in the computer science literature. Its meaning is mainly based on each individual researcher's understanding (and its usage is implicit). As we discussed in the previous chapter, there are many research efforts using or focusing on the term “context” or “contextual situation” and providing various definitions, which we discuss as follows:

Dey and Abowd [Dey & Abowd, 2000] define context as “*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application,*

including the user and applications themselves"; Brown et al. [Brown et al., 1997] define context from the user's perspective as users' location, who the users are with, what the time of day is, etc.; Ryan et al. [Ryan et al., 1997] consider the information about the computers environment as context, such as location, time, temperature, or user identity etc.

These definitions demonstrate that the "context" is a very complex term and can be interpreted from many different perspectives. However, in our work, context does not focus on the physical environment as most of the researchers do. The context that we take into account is the usage context of Web services and it is the information that can help service users to understand and use Web services, i.e. the *Service Usage Context* (SUC).

The SUC is proposed in the belief that the identification of meaning of a concept mainly stems from its contexts, i.e. its relationships to other concepts [Guha et al., 2004]. Before discussing the SUC in detail, we first look at how the meaning of a concept is defined. There are two ways to define a concept: type definition and schemata [Sowa, 1984]. The type definition defines a concept through genus and differentiae. It states all the obligatory or essential properties that must be hold for that type [Sowa, 1984]. For example, if we define a hammer using the type definition, it would be "a hand tool with a heavy rigid head and a handle". The definition demonstrates that a new concept "Hammer" is introduced by an existing generic concept "Tool" with some essential features or properties, i.e. "a heavy rigid head and a handle". Through the type definition, concepts are organised into a hierarchical structure, e.g. an ontology. In the above example, the concept "Tool" is a super type of the concept "Hammer". A schema presents a perspective on one way that a concept may be used [Sowa, 1984]. For example, if we define the hammer using schemata, it would be "a hammer can strike nails into wood" and "a hammer can smash stones". The schemata definition of a concept defines the meaning of the concept collectively using several typical schemata of the concept. Through schemata, concepts are organised into a network. From the difference between the type definition and the schemata we can see that to fully express the meaning of a concept, especially when the concept is not an abstract concept, it is insufficient to only label it with what it is, i.e. the type definition. In some cases, it is difficult to learn the genus of a concept and

it is even more difficult to find properties to differentiate this concept from other concepts with the same genus. In this situation, we need to describe how the concept can be applied in instance usage scenarios, i.e. the schemata, in order to make its type definition understandable. In the “hammer” example, if a person is told that “a hammer is a type of tool with heavy rigid head and a handle”, he may or may not be able to infer what the tool is for. However, if we say “a hammer can strike nails into wood”, at least now he knows one way to use a hammer despite not knowing clearly exactly what a hammer is. Sometimes the type definition of a concept is not important when only the concept’s functionalities are of interest. For a hammer’s user, as long as the hammer can achieve the user’s targets/requirements, e.g. knocking nails into wood, it is the right thing for the user.

Web services encapsulate discrete functionalities to achieve certain desired goals. The following reasons suggest that only using type definition, i.e. an ontology, to describe Web services is insufficient. The descriptions need to comprise type definitions and schemata, i.e. the combination of service ontology and SUC.

- On the one hand, it is difficult for a type definition to clearly describe the capabilities of a service and its usage. For example, if a service is categorised as a “Credit Card Payment” service, this does not indicate any information about how and under which circumstances the service can be used because the type definition for “Credit Card Payment” does not provide that information. On the other hand, some service’s genus and differentiae are not clear and therefore it is difficult to select suitable type definitions for those services.
- It is very difficult to tell what the differences are between services that belong to the same service concept, even though these services have been given suitable type definitions.
- How to use a service is often considered as knowing how to interact with the service or how the service interacts with other services. This cannot be clearly represented in a type definition hierarchy.

Therefore, to describe a Web service, we need to address not only the type definition of the service, but also how the service is used in various usage contexts, i.e. SUC. SUC can help service users to better understand services and better match services

with their own requirements. SUC emphasises that parts can only make sense in the context of a greater whole [Fensel et al., 2007].

3.4 Service Usage Context

We define SUC at two levels: the conceptual level SUC and the instance level SUC. In this section, we will discuss the two levels of SUC in detail and give the definitions of the related concepts.

3.4.1 Conceptual Level Service Usage Context – \mathcal{T} -Context

As a function unit, a Web service needs to interact with other services and entities in order to perform its function. Entities that interact with the Web service are service users, service requesters, or other business components. In a given business domain, we can categorise a Web service's interactions into different scenarios in which certain business tasks are achieved. Each scenario of the service demonstrates a way of using the service.

If we abstract a scenario into the concepts and relations that represent the service, other services, entities, and their interactions, then we can construct a conceptual usage scenario of the Web service's service concept in the business domain. Each conceptual usage scenario describes the conceptual relationships between the Web service, other services and entities required to achieve a business task or collaboratively provide a business service. If we have a collection of conceptual usage scenarios in which a Web service can participate, i.e. the schemata of its service concept, then we can tell conceptually how the service should be used in the business domain, i.e. the usage context, and how it is related to other services and entities. We use conceptual level SUC (\mathcal{T} -Context) to represent the usage context of a service and the conceptual relationships between the service and other services and entities.

Concepts in a \mathcal{T} -Context are service concepts and type definitions that are defined in a domain ontology. Conceptual relations between concepts in the \mathcal{T} -Context can be either hierarchical relations, such as $\text{Bus} \sqsubseteq \text{Travel_Service}$ ³, or horizontal relations,

³ Here we borrow the symbol from Description Logic [Baader et al., 2002] to represent sub-type relations.

such as *requires*(Payment_Service, Card_No.). They represent the relationships between the service, other services and entities that are collaboratively providing business services.

If a service is applicable in an open domain rather than a specific business domain, it may be able to participate in a large or even infinite number of scenarios, i.e. its service concept may theoretically have a large or infinite number of relations with other concepts. In this case, the \mathcal{T} -Context of the service could be infinitely large. However, in reality the \mathcal{T} -Context does not necessarily to be infinite to describe the service. As we mentioned in Section 3.3, to define the meaning of a concept it is not necessary to enumerate all the schemata of the concept, the typical schemata are sufficient to define the meaning. For example, although people can use a hammer as a paperweight, that is not a hammer's major function and without enumerating that usage the meaning of the hammer concept is still clear. In the Web services case, as long as a service concept's \mathcal{T} -Context covers the most popular usages of the service, it is sufficient. In an enterprise, services are normally sitting in a closed business domain and their capabilities are manageable by the enterprise [Strang, 2005]. In this case, a service's \mathcal{T} -Context will not have the infinite size problem. However, \mathcal{T} -Context is extendable, therefore when a service can participate in a new business service or scenario, its \mathcal{T} -Context can be extended to enclose it.

As the \mathcal{T} -Context of a service consists of a collection of conceptual usage scenarios, before formally defining \mathcal{T} -Context, we first give the definition of a conceptual usage scenario. Given a business domain, let \mathcal{O} be a domain ontology, $\mathcal{O}_c \subseteq \mathcal{O}$ be the whole collection of concepts from \mathcal{O} , \mathcal{R} be a set of labels representing the conceptual relationships among services and entities in the business domain, a conceptual usage scenario is defined as following:

Definition 3.1. Given a service concept $c_s \in \mathcal{O}_c$, a conceptual usage scenario $\varphi(c_s)$ of c_s is a pair (G, l) where:

- $G = (V, E)$ is a directed labelled graph.

– $l : V \cup E \rightarrow \mathcal{O}_c \cup \mathcal{R}$ is a labelling function that labels each vertex in V with concept name, so that $V = \{v_i \mid v_i \in \mathcal{O}_c, 0 < i\}$, and each edge in E with predefined relations to indicate the relationship between concepts.

An example of a conceptual usage scenario is given in Figure 3.3. From the diagram we can see how the service concept “Money Transfer” is related to other concepts in the domain, such as “Financial Service”, “Currency”, “Currency Conversion”, and “Bank”. This conceptual usage scenario example tells the story about how the “Money Transfer” service is performed in the given business domain.

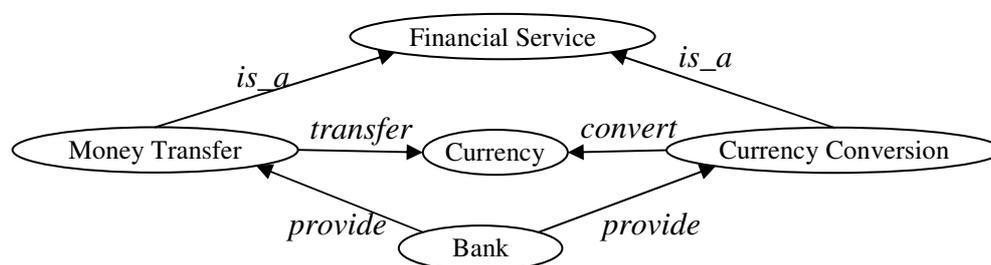


Figure 3.3. A conceptual usage scenario of a service concept – “Money Transfer”.

The \mathcal{T} -Context of a service is a collection of conceptual usage scenarios for the service’s service concept. Formally, it is defined as follows:

Definition 3.2. Given a service concept $c_s \in \mathcal{O}_c$, its \mathcal{T} -Context is a set of conceptual usage scenarios $\mathcal{T}(c_s) = \{\varphi(c_s)_1, \varphi(c_s)_2, \varphi(c_s)_3, \dots, \varphi(c_s)_n\}$, whose element satisfies the following conditions:

- $c_s \in (\varphi(c_s)_1.V_1 \cap \varphi(c_s)_2.V_2 \cap \varphi(c_s)_3.V_3 \cap \dots \cap \varphi(c_s)_n.V_n)$
- For any two elements $\varphi(c_s)_i, \varphi(c_s)_j \in \mathcal{T}(c_s)$, $\varphi(c_s)_i \not\subseteq \varphi(c_s)_j \wedge \varphi(c_s)_j \not\subseteq \varphi(c_s)_i$.

We call c_s the *owner service concept* of $\mathcal{T}(c_s)$ and the instance services of c_s the *owner services* of $\mathcal{T}(c_s)$.

In a service repository, if we consider each Web service’s service concept and the relationships between the service concepts as a knowledge base, the \mathcal{T} -Context can be seen as a mechanism for partitioning the knowledge base and assigning partitions to their relevant services. In an enterprise, the knowledge base partitioning is the process

of creating business services based on the enterprise's existing capabilities. In other words, once a new business service is created, certain Web service's \mathcal{T} -Context will be extended (or created if the services have not been used in the past).

However, \mathcal{T} -Context only provides a conceptual view of the relationships between services and other business entities. It does not consider any technical details, such as service interfaces and pre- and post-conditions of each service, which means that although two services are conceptually related in certain business tasks or scenarios, they may not be able to interact at runtime. These constraints come from data semantic differences, data type incompatibility, and pre- and post-conditions dissatisfaction. Therefore, we introduce another level of service usage context, the instance level service usage context (\mathcal{A} -Context), to model the usage context of services at the instance level.

3.4.2 Instance Level Service Usage Context – \mathcal{A} -Context

The \mathcal{T} -Context defines conceptually how each service in a business domain participates in different business services and tasks through the conceptual relations between its service concepts and other concepts. When tasks and business services are executed, the actual instance services of each service concept in a \mathcal{T} -Context need to be allocated and composed in order to achieve the desired goals of the tasks and business services. However, \mathcal{T} -Context does not determine whether each service concept's corresponding instance services are compatible with the required composition. To execute and compose services at runtime, a service execution or composition system needs to know the composability between services. Although this can be achieved through analysing the service interfaces, it could be much more efficient if the information was provided in the service description. To address instance level service interactions, we introduce instance level service usage context (\mathcal{A} -Context). \mathcal{A} -Context can be considered as the instantiation of the \mathcal{T} -Context. It models interactions between instance services at runtime. In \mathcal{A} -Context we only consider interactions between instance services. Interactions between services and service requesters are considered during the query processing stage.

Similar to \mathcal{T} -Context, an instance service in an open service repository, e.g. the Internet, could theoretically interact with infinite number of instance services, which

means the size of its \mathcal{A} -Context can be infinitely large. However, for the same reason as \mathcal{T} -Context, we do not need to enumerate all the possible interactions between an instance service and other instance services. Additionally, the interactions in the \mathcal{A} -Context of a service are limited by its service concept's \mathcal{T} -Context. If the semantics of an interaction is not defined in the service's \mathcal{T} -Context, then that interaction is not included in the service's \mathcal{A} -Context. The \mathcal{A} -Context can also be extended to enclose new interactions of an instance service.

The minimum requirements for an interaction happening between two instance services according to Paolucci et al. [Paolucci et al, 2002] are compatible inputs and outputs and satisfiable pre-conditions and post-conditions/effects. Input and output compatibility means that if a service s_1 's input data is provided by another service s_2 's output, the data generated by s_2 must be both data type compatible and semantically compatible with s_1 's input data requirements. For example, if s_1 requires a "telephone number" in "String" data type as input, then neither a "telephone number" in "Integer" data type nor an "Address" in "String" data type is compatible with s_1 's input. Having satisfiable pre-conditions and post-conditions/effects means that after the execution of s_2 , the effects of s_2 are not violating the pre-conditions for s_1 's execution. For example, if s_1 requires a "telephone number" that must exist in a local telephone number registry, then s_2 will not be the right service for s_1 if its output telephone number does not exist in that registry.

In \mathcal{A} -Context, the composability between two instance services is represented as a service link. A service link is a directed relation. Let in represent input parameters of a service, out represent output parameters a service, p represent pre-conditions of a service, and e represent effects of a service, a service link is defined as follows:

Definition 3.3. Let s_i, s_j be two services. A service link between s_i to s_j is a directed relation that links the m -th output of s_i , i.e. $s_i.out_m$, to the k -th input of s_j , i.e. $s_j.in_k$, denoted as $l(s_i, s_j) \mid s_i.out_m \gg s_j.in_k$, and it satisfies the following conditions:

$-Sim(s_i.out_m, s_j.in_k) > 0$, where $Sim()$ is a semantic similarity function to compute the semantic distance between $s_i.out_m$ and $s_j.in_k$. The returned value is within range $[0, 1]$ to indicate the similarity degree of from no similarity to full match.

– $Comp(s_i.out_m, s_j.in_k) > 0$, where $Comp()$ is a compatibility function to determine whether $s_i.out_m$ and $s_j.in_k$ are data type compatible. The returned value is within range $[0, 1]$ to indicate the compatibility degree from not compatible to fully compatible.

– $Satisfy(s_i.e, s_j.p) = true$, where $Satisfy()$ is a Boolean function that determines whether $s_j.p$ can be satisfied or not be violated by $s_i.e$.

Considering the four services listed in Table 3.1, according to their inputs, outputs, pre-conditions, and effects, we can create three service links among these four services:

– $l(service_3, service_1) | service_3.out_1 \gg service_1.in_1$: the meaning of this service link is that after an amount of money is converted from Dollar to Pound by $service_3$, it is transferred from one account to another by $service_1$.

– $l(service_4, service_2) | service_4.out_1 \gg service_2.in_1$: the meaning of this service link is that after an amount of money is converted from Euro to Dollar by $service_4$, it is transferred from one account to another by $service_2$.

– $l(service_4, service_3) | service_4.out_1 \gg service_3.in_1$: the meaning of this service link is that it converts an amount of money from Euro to Pound (via the conversions Euro to Dollar and Dollar to Pound).

Table 3.1. Service instances and their basic attributes.

Service	Service Concept	Input	Output	Pre-condition	Effect
$service_1$	Money Transfer	in_1 : Pound in_2 : account in_3 : account	out_1 : transfer status	p : valid account detail	e : money transfer succeeded or failed
$service_2$	Money Transfer	in_1 : Dollar in_2 : account in_3 : account	out_1 : transfer status	p : valid account detail	e : money transfer succeeded or failed
$service_3$	Currency Conversion	in_1 : Dollar	out_1 : Pound	p : correct currency type	e : Dollar converted into Pound
$service_4$	Currency Conversion	in_1 : Euro	out_1 : Dollar	p : correct currency type	e : Euro converted into Dollar

Similar to the conceptual usage scenarios in \mathcal{T} -Context, the composable instance services can be grouped into scenarios. We name these scenarios instance usage scenarios.

An instance usage scenario describes how a group of instance services can interact with each other to provide a composite service in order to achieve desired goals. Let S be a set of instance services, $\mathcal{L}=\{l(s_i, s_j) \mid s_i.out_m \gg s_j.in_k \mid s_i, s_j \in S\}$ be a set of service links, an instance usage scenario is defined as following:

Definition 3.4. Given an instance service $s \in S$, an instance usage scenario $\rho(s)$ of s is a pair (G, l) where:

- $G = (V, E)$ is a directed labelled graph.
- $l : V \cup E \rightarrow S \cup \mathcal{L}$ is a mapping that maps each vertex in V to an instance service in S and each edge in E to a service link in \mathcal{L} .

In Figure 3.4 the example of an instance usage scenario is illustrated using the services from Table 3.1. This scenario describes how two instance services are linked to achieve a money transfer service that transfers money in Dollars from one account to another account that only accepts Pounds.

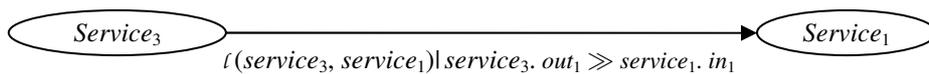


Figure 3.4. An instance usage scenario of a service – $service_1$.

The \mathcal{A} -Context of a service is a collection of instance usage scenarios for the service. \mathcal{A} -Context describes how an instance service (the context owner service) can interact with other instance services (the surrounding services) by means of consuming or providing data from or to other services to complete tasks or provide services. Formally, it is defined as follows:

Definition 3.5. Given an instance service s , its \mathcal{A} -Context is a set of instance usage scenarios $\mathcal{A}(s)=\{\rho(s)_1, \rho(s)_2, \rho(s)_3, \dots, \rho(s)_n\}$, whose elements satisfy the following conditions:

- $s \in (\rho(s)_1.V_1 \cap \rho(s)_2.V_2 \cap \rho(s)_3.V_3 \cap \dots \cap \rho(s)_n.V_n)$
- For any two elements $\rho(s)_i, \rho(s)_j \in \mathcal{A}(s)$, $\rho(s)_i \not\subseteq \rho(s)_j \wedge \rho(s)_j \not\subseteq \rho(s)_i$.
- Let $\mathcal{T}(c_s)$ be the \mathcal{T} -Context of s , where c_s is the service concept of s , then $\forall \rho(s) \in \mathcal{A}(s) (\exists \varphi(c_s) \in \mathcal{T}(s) (\varphi(c_s) \mapsto \rho(s)))$. The expression “ $\varphi(c_s) \mapsto \rho(s)$ ” is read as “ $\rho(s)$ complies with $\varphi(c_s)$ ”, which means that for all the instance services in $\rho(s)$, their service concepts or service concepts’ super concepts must appear in $\varphi(c_s)$.

3.5 Summary

In this chapter, we introduced an important concept, the Service Usage Context (SUC). SUC is defined at two levels: the \mathcal{T} -Context and the \mathcal{A} -Context. The \mathcal{T} -Context of a service defines its conceptual relationship with other services and entities in potential business scenarios. The \mathcal{A} -Context of a service defines how an instance service can interact with other instance services at runtime. In other words, \mathcal{A} -Context defines the composability between instance services.

SUC provides information that can help the understanding and use Web services, thus assist service discovery and composition processes. Both the \mathcal{T} -Context and the \mathcal{A} -Context emphasise that services are not isolated individual components. They always work together in certain patterns to achieve business tasks. Therefore, to fully understand and describe a service’s functionality, we should know its usage context. However, according to the literature that we have reviewed, there is little existing research that formally addresses service usage context in service description.

The concept of SUC is the foundation of the service description framework that we will define in the next chapter. This framework integrates SUC into service description, which aims to make service discovery and composition more effective and efficient.

Chapter 4: Context-based Semantic Service Description Framework

In this chapter, we will first give an overview of the Context-based Semantic Service Description Framework. Then, we will discuss each component of the framework in detail. Finally, we will propose a method that can transform existing Web service descriptions efficiently into Context-based Semantic Service Description Framework based descriptions.

4.1 Overview

As we reviewed in Chapter 2, an enormous number of research efforts on Semantic Web Services have been focusing on semantic Web service description, discovery [Ludwig & Reyhani, 2005] [Paolucci et al., 2003] [Song & Li, 2005], and composition [Agarwal et al., 2005] [Du et al., 2006a]. In order to effectively and efficiently perform Web service discovery and composition, a comprehensive service description framework is essential. There are several semantic service description frameworks that are proposed to provide richer service descriptions to address the semantic issue of Web services, such as OWL-S [Martin et al., 2004], WSDL-S [Akkiraju et al., 2005], and WSMF [Fensel & Bussler, 2002]. The main idea of the existing work is to build a semantic layer either on the top of WSDL or integrated into WSDL to semantically describe capabilities of Web services so that a software agent or other services can reason about a Web service's capabilities and how to interact with it. However, the following problems still exist in the current semantic service description and discovery:

- **Insufficient usage context information:** The current work is focusing on ontology based data type semantics. They do not sufficiently address how a service fits its usage context, i.e. a service's applicability/usability/composability, which we believe is a more effective and more natural way of describing Web services.
- **Precisely specified requirements to locate services:** In order to locate a service, the current work requires a precise specification of the required service. For a service user who is not familiar with the technical side of the required service, providing precise specification can be a difficult task. Furthermore, the service matching mechanisms only consider the limited set of attributes that are defined in the service description. They merely compare text based service description, operation signature (inputs and outputs), and constraints on inputs and outputs. Inter-service relationships, service internal structure, and service composition patterns have not historically been used to discover services, we believe these should be considered very important factors when identifying services.

- Insufficient information about inter-service relationships: although Web services are functional units, more often than not their own functionality is not sufficient to achieve some complex task without interaction with other services. Therefore the best way to understand a service is to know how it should interact with other services, at both the conceptual level and the instance/technical level. Once the inter-service relationships are addressed in the service description, for a given task, we could locate the relevant services all together, rather than individually. The current work considers services as isolated components, which makes service discovery and composition less effective and less efficient.
- Insufficient incomplete information handling: The incomplete information caused by the dynamic nature of the Internet and Web services has not been addressed adequately in current service description frameworks and the service composition process. The monotonic logic based rules in the current service description frameworks are not suitable for handling incomplete information and conflicting conditions.

To address the above problems, we propose a Context-based Semantic Service Description Framework (CbSSDF), which is a comprehensive framework for Web services. Apart from service capabilities, CbSSDF also takes into consideration the SUC. The aim of the framework is to improve the semantic capability of service discovery and simplify the service composition process. It contains three main components: a set of Service Conceptual Graphs (S-CGs), a Semantic Service Description Model (SSDM), and a set of non-monotonic rules that are represented in Defeasible Logic [Nute, 1994]. The set of S-CGs gives an abstract description of the relationships between the services and concepts. The S-CGs are the implementation of the \mathcal{T} -Context. The formalism behind S-CG is conceptual graphs [Sowa, 1984]. The SSDM gives a comprehensive semantic description of a service through different semantic aspects. It also addresses the \mathcal{A} -Context of services. A set of non-monotonic rules are used to represent the pre-conditions and effects of services and the conditions and constraints for service composition. These non-monotonic rules can also handle conflict conditions caused by incomplete information.

The content of this chapter is organised as follows: before giving the details of CbSSDF, we firstly define what the atomic service and the composite service are; then we walk through each main component in CbSSDF; finally, we propose a transformation method that can convert existing service descriptions into CbSSDF based descriptions efficiently.

4.2 Atomic Service and Composite Service

The term “service” in SOA is an important concept and has multiple definitions. Different researchers or organisations understand services differently. In this section, we first examine some of the service definitions in the literature to see what other people think what services are. From these definitions, we summarise some common characteristics of services and give our own definitions. The followings are a list of service definitions that have been mentioned in the literature:

“From a business perspective, services are IT assets that correspond to real-world business activities or recognizable business functions and that can be accessed according to the service policies that have been established for the services.

From a technical perspective, services are coarse-grained, reusable IT assets that have well-defined interfaces (a.k.a. service contracts) that clearly separate the services' externally accessible interface from the services' technical implementation.”

[Newcomer and Lomow, 2004]

“Contemporary SOA represents an open, extensible, federated, composable architecture that promotes service-orientation and is comprised of autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable services, implemented as Web services.” [Erl, 2005]

“A service is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.” [MacKenzie et al., 2006]

“A service in the context of SOA is the IT realization of some self-contained business functionality.” [Josuttis, 2007]

“Services are exposed functionalities which are clearly defined, self contained, and which will not depend on the state and context of peer services.” [Binildas, 2008]

From above definitions we can see that a service in the context of SOA should have the following characteristics:

- **Interface:** a service is for exchanging messages or changing the state of a backend entity that is associated with the service. It must have a well defined interface for doing this.
- **Self-contained:** a service should be independent and autonomous.
- **Searchable:** a service should be formally described and published, so that it can be discovered by service requesters.
- **Pre- and post-conditions/effects:** pre- and post-conditions/effects are part of a service’s capability specification. Pre-conditions are the conditions that must be met before a service requester invokes a service. Post-conditions/effects define the effect caused by the execution of the service.
- **Composable:** a service’s functionalities can be accomplished by combining other services. This does not contradict the self-contained characteristic because from the service requester perspective, the service is till functioning as a whole.
- **Implemented as Web services:** currently, Web services is the most popular technology to implement services in context of SOA.

In the following, we give our own definition of services. We distinguish two types of services, the atomic service and the composite service. Although for a service requester, there is no difference between an atomic service and a composite service, it is necessary to distinguish these two kinds of services from the service discovery and composition perspectives. The reasons are discussed as follows:

- Atomic services are the basic building blocks of composite services. They are the most basic services that are able to accomplish one of the simplest business functions that a service provider can provide to service users. A service is an atomic service when its internal structure is not (necessarily) visible to the service provider, e.g. provided by another service provider, or it has no internal structure, i.e. its functionalities do not rely on any other services, or its internal structure can be ignored, e.g. the service’s functionalities rely on other services that cannot be

used as independent services. The most important characteristic of atomic services is that they cannot be decomposed further, so any decomposition process operating of them can end here, which is very important during the service planning and composition process; they also enable users to see which are the building blocks of the more complex services.

- Composite services are normally created by service providers and they have total control over the internal structures of these services. Service providers can alter the structure of composite services or reuse their sub-components if it is necessary to fulfil service requirements. Distinguishing composite services from atomic services gives us richer semantic capabilities because if we recognise the internal structure of a composite service, then the semantics of its sub-services and their relationships can be taken into account when identifying the service during a service discovery and composition process.

An atomic service is an independent non-decomposable service that provides its functions without relying on other services. Formally it is defined as following:

Definition 4.1. An atomic service s_A is a 7-tuple, $s_A=(id, In, Out, P, E, T, B)$ where,

- id : A service ID.
- $In = \{in_1, in_2, \dots, in_n\}$: A set of inputs of s_A .
- $Out = \{out_1, out_2, \dots, out_n\}$: A set of outputs of s_A .
- $P = \{p_1, p_2, \dots, p_n\}$: A set of pre-conditions to trigger a service execution of s_A .
- $E = \{e_1, e_2, \dots, e_n\}$: A set of effects received after the execution of s_A .
- T : The type of the service, i.e. s_A 's service concept in a service ontology.
- B : The grounding information for interacting with s_A .

A composite service is a service that provides its capabilities by coordinating and assembling other service's capabilities. It has a clearly defined internal data flow and control structure.

Definition 4.2. A composite service s_C is a 10-tuple, $s_C=(id, In, Out, P, E, T, S, DS, CtrlS, B)$ where:

- id : A service ID.
- $In = \{in_1, in_2, \dots, in_n\}$: The set of inputs of s_C .
- $Out = \{out_1, out_2, \dots, out_n\}$: The set of outputs of s_C .

- $P = \{p_1, p_2, \dots, p_n\}$: The set of pre-conditions to trigger a service execution of s_C .
- $E = \{e_1, e_2, \dots, e_n\}$: The set of effects received after the execution of s_C .
- T : The type of the service, i.e. s_C 's service concept in a service ontology.
- $S = \{s_1, s_2, \dots, s_n\}$: The set of services that are used to compose s_C .
- $Df = \{\ell(s_i, s_j) \mid s_i.out_p \gg s_j.in_q, s_i, s_j \in S\}$: The set of service links that represents the data flow of the service.
- $CtrlS = \langle Stmt, Cdt \rangle$: A 2-tuple that represents the control structure of the service, where $Stmt = \{stmt_1, stmt_2, \dots, stmt_m\}$ is a set of control statements and $Cdt = \{cdt_1, cdt_2, \dots, cdt_k\}$ is a set of conditions.
- B : The grounding information for interacting with s_C .

The above service definitions are general and therefore applicable to an entity or a software component that possesses the characteristics of a service. However, the grounding information of a service may vary depending on the service's implementation technology. For example, in the Web services case, the grounding information provides the location of the WSDL and what protocols can be used to transfer SOAP messages, whereas, in the software service case, the grounding information may provide the RPC URL and how the objects transferred through RPC are to be serialised.

4.3 Context-based Semantic Service Description Framework

The Context-based Semantic Service Description Framework (CbSSDF) aims to give a comprehensive description of the aspects that are related to identifying services, such as technical specifications, semantics, and context, especially in identifying services by their usage context. It provides a richer semantic model for services and brings them to their full potential. The CbSSDF's design follows the notion of SUC, and complies with design principles listed below:

- A service description that addresses services in an inter-relational fashion in order to ease the process of service composition. Describing services together with their usage context, i.e. relationships with other services, provides a richer semantic way to identify services.

- A tolerant service description intended to deal with imprecise service requirements from end users by linking syntactical service description, such as WSDL, to a higher level conceptual description, such as service semantics and the SUC.
- A precise service description intended to ensure the executability of discovered services. Conditions relevant to a service, such as pre-conditions, effects, and domain constraints, need to be precisely specified by a formalism that can deal with incomplete information and ensure that a discovered service is actually executable at runtime.

The CbSSDF consists of three main components: a set of *service conceptual graphs* (S-CGs) that gives an abstract conceptual description of the relationships between services and business entities, a *semantic service description model* (SSDM) that gives a concrete semantic description of the services, and a set of non-monotonic rules that represent the pre-conditions and effects of services and constraints for service composition. Among these components, the set of S-CGs is a realisation of the \mathcal{T} -Context and the SSDM implements the \mathcal{A} -Context. In the following sections, we will discuss each component of CbSSDF in detail.

4.3.1 Service Conceptual Graphs

As discussed previously in Chapter 3, to comprehensively describe a service, only using the technical description is insufficient, the SUC information is crucial. If services are technically compatible, this does not necessarily mean that these services can work together because the combination may be logically incorrect. The SUC information, especially the \mathcal{T} -Context, of a service tells how the service is related to other services and entities in a business domain and under which context the service should be used. In CbSSDF, the \mathcal{T} -Context is implemented using Service Conceptual Graphs (S-CG). Each S-CG can be considered as a conceptual usage scenario. It describes a way in which the service can be used. The whole collection of S-CGs represents the \mathcal{T} -Context of the service.

The key point of having S-CGs in the service description is to bridge the gap between the technical detail of services and the conceptual explanation of the service user's needs. By having S-CGs, when the service users search for services, they can first

express their needs in a conceptual way, such as a natural language query, without worrying about any technical detail. These requirements then will be converted into conceptual graphs and matched with S-CGs in the service repository to check which services are the most relevant. Then, based on the relevant services, service users can input technical specifications to refine the result. We call this two-step service discovery, which will be discussed in detail in the next chapter.

The S-CG representation is based on the conceptual graph (CG) formalism and it is the implementation of the \mathcal{T} -Context. As discussed in [Sowa, 1984], the perception process in a person's mind is the process of associating percepts with concepts and assembling concepts with conceptual relations. The result of the perception process is a map or graph of concepts linked by conceptual relations that can be formally represented as CGs [Sowa, 1984]. In other words, CGs can closely represent people's mind and perceptions. This is the main reason why we chose the CG based formalism to represent the \mathcal{T} -Context. Other reasons that we chose CG as the representation formalism are: a) a clear separation is made in CG between ontological knowledge (the type definition) and factual knowledge (the schemata) [Mugnier, 2000]; b) CG is proven equivalent to first order logic (FOL), which means that it has the full expressiveness of FOL [Kerdiles & Salvat, 1997]; c) CG supports more direct mapping from and to natural language and has a direct translation to both natural language and symbolic logic [Sowa, 1984], which is important when dealing with natural language based service queries. Furthermore, graph based modelling is easily understandable by either end users or reasoning systems, and from the computational view point, a graph homomorphism problem has less complexity than logic deduction [Mugnier, 2000].

For the details of the CG formalism, refer to Section 2.2.3 in Chapter 2. Here we directly give the definition of S-CG and its basic properties, which are based on the CG formalism. An S-CG is a simple CG, which means that it does not contain co-reference links and nested context [Sowa, 1984]. Similar to CG, an S-CG is also defined over a support $\mathcal{J} = (T_C, T_R, \phi, \tau)$. For how \mathcal{J} is defined, refer to definition 2.1 in Chapter 2. In the S-CG case, T_C includes domain concepts and service concepts; ϕ includes the individuals of domain concepts and the instance services of service concepts.

Formally, an S-CG is defined as below:

Definition 4.3. An S-CG g_s , defined over a support \mathcal{J} , is a binary $((C_{g_s} \cup R_{g_s}, E_{g_s}), l_{g_s})$,

where,

– $(C_{g_s} \cup R_{g_s}, E_{g_s})$ is a bipartite graph, where, C_{g_s} and R_{g_s} are node sets, respectively of concept nodes and of relation nodes, and E_{g_s} is a set of edges.

– l_{g_s} is a labelling function of nodes and edges. A concept node $c_{g_s} \in C_{g_s}$ is labelled by an ordered pair $(type(c_{g_s}), marker(c_{g_s}))$, where $type(c_{g_s}) \in T_c$, $marker(c_{g_s}) \in \mathcal{G} \cup \{*\}$.

A relation node $r_{g_s} \in R_{g_s}$ is labelled by $type(r_{g_s})$, where $type(r_{g_s}) \in T_R$. The edge labelling is omitted in S-CG.

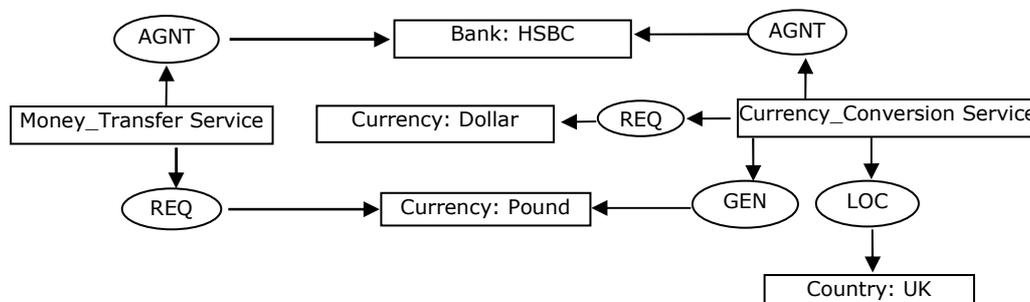


Figure 4.1. An example of an S-CG.

Figure 4.1 demonstrates an example of an S-CG. The example illustrates the usage scenario for a money transfer service. In this S-CG, two service concepts have been addressed: the Money_Transfer Service and the Currency_Conversion Service. The other concepts, such as “Currency”, “Bank”, and “Country”, in the S-CG are related to these two service concepts in this scenario. The relation nodes, such as “AGNT”, “REQ”, “GEN”, and “LOC”, describe relationships between these concepts. The example S-CG illustrates a simple scenario. A larger S-CG that describes a complicated scenario can be either created based on a complex business service or generated dynamically by joining simple S-CGs.

In Chapter 3 we mentioned that each \mathcal{T} -Context has an owner service concept and the owner service concept’s corresponding instance services are owner services. However, in its actual implementation, i.e. S-CGs, all the service concepts addressed in an S-CG are the owner of the S-CG depending on which instance service the S-CG has been

assigned to. This is to avoid the situation where duplicate S-CGs are created for each service and to also reduce the redundancy and complexity of the \mathcal{T} -Context implementation.

As S-CGs are based on the CG formalism, all the properties, rules, and operations that are applicable to CGs are also applicable to S-CGs. In the following, we will summarise some of the important operations on CGs and basic properties of S-CGs.

Specialisation and *generalisation* are two important CG operations that are essential to CG matching and reasoning [Mugnier, 2000], see Section 2.3.3 for relevant definitions. Rules for specialisation and generalisation operations are defined below:

Specialisation operation: Let u be a CG, then a specialisation CG v of u can be obtained from u by:

- **Copy:** v is an exact copy of u ;
- **Relation simplify:** Remove the duplicated relation nodes from u ;
- **Restrict:** Decrease the label of a concept node (its type and/or its marker) or the label of a relation node (its type);
- **Join:** Let w be a CG disjoint from u , if a concept c in u is identical to a concept d in w , then v can be obtained by deleting d and linking to c all arcs of conceptual relations that has been linked to d , then perform the above *relation simplify* operation.

Generalisation operation: Let u be a CG, then a generalisation CG v of u can be obtained from u by:

- **Copy:** v is an exact copy of u ;
- **Relation duplicate:** Duplicate a relation node of u ;
- **Un-restrict:** Increase the label of a concept node (its type and/or its marker) or the label of a relation node (its type);
- **Sub-graph:** If v is a sub-graph of u , then v is a generalisation of u .

As S-CGs are used to match with CGs generated from a service user's query to locate relevant services, we now make a simple analysis of the relationship between an S-CG and a CG generated from a query through the following theorems and definitions.

Theorem 4.1. Let u be an S-CG, v a generated CG from a service query, and ϕ an operator that can convert a CG into its equivalent logic formulas. If $u \leq v$, then $\phi u \Rightarrow \phi v$.

Proof. As S-CGs are simple CGs, the proof proposed by Sowa is applicable here, see [Sowa, 1984] (p. 98).

Definition 4.4. Let u be an S-CG and v a generated CG from a service query. If $u \leq v$ or $\phi u \Rightarrow \phi v$, then the service query is called *conceptually satisfiable* by u .

Theorem 4.2. Let u be an S-CG and v a generated CG from a service query. If v has a projection in u , i.e. $\pi: v \rightarrow u$, then the service query must be conceptually satisfiable by u .

Proof. According to theorem 2.1 (Chapter 2, Section 2.3.3), if $\pi: v \rightarrow u$, u must be identical to or a specialisation of v , i.e. $u \leq v$. By definition 4.4, if $u \leq v$, the service query is *conceptually satisfiable* by u .

□

Theorems 4.1 and 4.2, and Definition 4.4 together describe how to check whether a service query is conceptually satisfiable. Conceptual satisfiability is a very important concept that indicates whether a service request is within or beyond a service provider's business domain. If a service request is not conceptually satisfiable, there is no need to search for instance services. The S-CG u in the above theorems and definitions does not necessarily exist in a service repository. It can however be dynamically generated by joining existing S-CGs. If a service request is conceptually satisfiable this does not necessarily mean that the instance services associated with the S-CG can actually achieve the service request. It only suggests that these instance services are relevant to the request and may possibly propose a full or partial solution. In other words, conceptual satisfiability is important evidence for judging whether an instance service is relevant to a service query/requirement.

4.3.2 Semantic Service Description Model

More often than not, a service query cannot be satisfied by a single service, but can be through the composition of several services. How to correctly and efficiently

construct these composite services is the major task in service discovery and composition. Service discovery and composition techniques based on current semantic service description frameworks [Paolucci et al., 2002; Wu et al., 2003] search for and compose services in an isolated manner. The current techniques locate services individually, without considering the inter-service relationships, thus for each participating service in a composite service, the same searching and planning procedure has to be carried out repetitively. In order to improve the efficiency of service composition and provide service discovery with more accurate results, we propose Semantic Service Description Model (SSDM) as a component of the CbSSDF. First, SSDM is a semantic description model and it addresses four types of semantics [Cardoso & Sheth, 2006] associated with a service, i.e. the data semantics, the functional semantics, the non-functional semantics, and the execution semantics. In SSDM, the data semantics deal with semantically annotated input and output of a service. The functional semantics are captured by a service ontology and pre-conditions and effects. The non-functional semantics are represented through the service metadata. The execution semantics are addressed through the internal structure of a service. Second, SSDM implements the \mathcal{A} -Context to improve the efficiency of service composition.

In the SSDM, the \mathcal{A} -Context of a service is implemented through a set of Common Usage Patterns (CUPs). A CUP describes how an instance service (the owner service of an \mathcal{A} -Context) can be composed with other instance services in a scenario or a part of a scenario. It complies with the owner service's \mathcal{T} -Context, which means that for the owner service and the other services in a CUP, their service concepts must appear together in at least one of the owner service's S-CGs. The whole collection of CUPs of a service collectively represents how this service can interact/be composed with other services at the instance level. A CUP is formally defined as below:

Definition 4.5. Given a service s_k , a CUP of s_k (the owner service of the CUP) is defined as a binary $p=(S, \mathcal{L})$, where:

– $S=\{s_1, s_2, \dots, s_n\}$: a set of services that directly interact with s_k . Let x be the number of inputs of s_k and y be the number of outputs of s_k , we have $n \leq (x + y)$. If $s_k \in S$, it means that s_k can be composed with its duplicated copy or it is in a loop control structure.

- \mathcal{L} : a set of service links that link the services in S with s_k .
- Let G_{s_k} be the set of S-CGs of s_k , then $\exists g_{s_k} \in G_{s_k} \mid p \mapsto g_{s_k}$.

The expression “ $p \mapsto g_{s_k}$ ” is read as “ p complies with g_{s_k} ”, which means that for all the instance services in p , their service concepts or service concepts’ super concepts must appear in g_{s_k} .

According to the definition, a CUP describes only the relationships between the owner service and services that directly interact with the owner service. The indirect relationships are not described in a CUP because they can be inferred from the other service’s CUPs.

A CUP can be considered as a segment of a workflow. Service composition is about constructing suitable workflows and therefore we can say that CUPs can make the service composition process more efficient. The reason is that assembling segments of workflows in the service composition process is quicker than assembling individual services. In the next chapter, we will discuss this in detail.

SSDM is proposed based on definitions of atomic service and composite service and the definition of CUP. In SSDM, we assume that all the services are composite services and an atomic service is a special case of the composite service. Formally, the SSDM is defined as follows:

Definition 4.6. Given a service s_k , its SSDM is a 7-tuple $(IO, PE, M, \mathcal{O}, Str, C, B)$, where,

- IO : Inputs and outputs of s_k , including their data types and semantics.
- PE : A set of rules that describe pre-conditions and effects of s_k .
- M : A set of metadata that describe non-functional attributes of s_k , such as the quality of service (QoS), the service provider information, and the natural language based service description, etc.
- \mathcal{O} : A service ontology that defines the service concept of s_k .
- Str : The internal structure of s_k , which contains a set of services, control structures, and the data flow. An empty internal structure means that s_k is an atomic service.
- C : A set of CUPs associated with s_k .
- B : The service grounding information of s_k .

A graphical illustration of the SSDM is shown in Figure 4.2. The notations used in Figure 4.2 are listed below:

1. $Service_k$ is the described service.
2. $Concept_s$ can be either the direct or ancestor service concept of $Service_k$ in a service ontology.
3. S_1, S_2, \dots, S_n are the sub-services of $Service_k$.
4. $Service_{k-1}$ and $Service_{k+1}$ represent the services that are linked to $Service_k$ through service links in a CUP.
5. I and P are the inputs and pre-conditions, and O and E are the outputs and effects.
6. Services within the dashed border rectangle represent a CUP of $Service_k$.

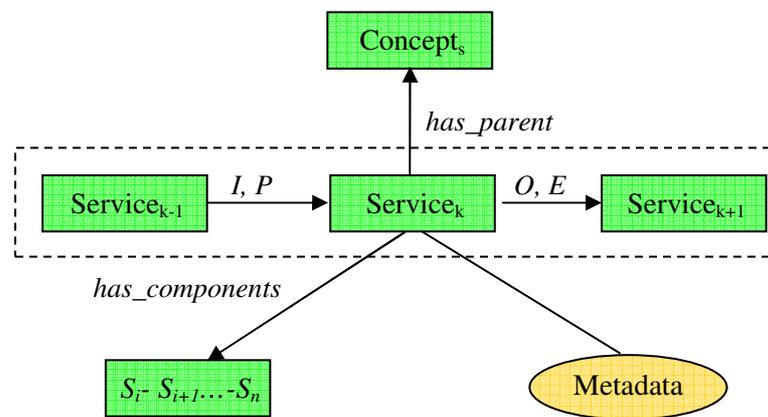


Figure. 4.2. A graphical illustration of SSDM

The SSDM has provided comprehensive information about a service so that the service users can have increased flexibility when searching for their required services with richer information to assist their search.

4.3.3 Non-Monotonic Rules

In order to correctly construct composite services and successfully execute them at runtime, the service pre-conditions and effects and the service composition rules must be represented by a suitable formalism. The web is a highly dynamic environment. In one situation, two services may be composable, whereas, in another situation the same two services may not be composable. Most of the information about service execution conditions on the web is incomplete information, i.e. when more information becomes available, the conditions may change. To deal with the dynamic nature of the web,

classical logic, i.e. monotonic logic, is insufficient because it cannot deal with conflicts and incomplete information of service execution conditions. The conflicts and incompleteness originate from the following aspects:

- **Business rules:** In real business scenarios, there are always cases where exceptions must be considered. These exceptions may be caused by applying rules or policies to different types of customer, different security constraints, different business partners, and so on. In those cases, a rule system with a priority mechanism is needed. Otherwise, conflicting conclusions may be drawn.
- **Incomplete information:** When designing real business applications or services, it is almost impossible to get complete information from customers or business partners. Therefore, some assumptions have to be made. These assumptions may lead to conclusions that are not supported by classical logic, for example drawing conclusions based on the priority order of conditions.
- **Runtime exceptions:** Due to the dynamic nature of the web, some information that is unavailable at design time may become available at runtime and vice versa. This can cause system failure if the system does not have a flexible way to cope at runtime.

In order to deal with conflicts and incompleteness, and still be able to correctly construct composite services, and enable service users to successfully execute their requirements, we adopt the non-monotonic reasoning approach and use Defeasible Logic [Nute, 1994], a non-monotonic formalism, to describe service pre-conditions and effects and the service composition rules. The basic detail of Defeasible Logic is discussed in Chapter 2, referring to [Nute, 1987] for further detail.

The advantages of Defeasible Logic among other non-monotonic approaches are its computational efficiency and its built in superiority handling mechanism [Brewka, 2001], these are crucial features needed to handle a situations where a large number of services are involved or processed. The recent implementation of Defeasible Logic – Deimos [Rock, 2000], a query answering system, demonstrates that the Defeasible Logic reasoning system is capable of efficiently dealing with 100,000s of defeasible rules.

The rules in the CbSSDF are divided into two categories: general rules and domain specific rules.

- **General rules**

General rules are used to govern and validate the service composition process. To achieve the desired level of flexibility and adaptability in the process of service composition, this type of rule is separated from the service description and stored in a rule repository as a part of the service composition engine. These rules govern the life cycle of the service composition process, from composite service planning, execution, to alternative service scheduling. They are applicable to all services. Some general rule examples are listed below:

r_1 : if a service's pre-condition is satisfied, then typically it can be executed.

$$satisfy(s.preCon) \Rightarrow executable(s)$$

r_2 : if a service is not available, then definitely it cannot be executed.

$$\neg available(s) \rightarrow \neg executable(s)$$

r_3 : if two services' input and output data types are compatible, i.e. the output data type of one service is the same type or a sub-type of the other service's input data type, typically these two services are composable.

$$type(s_1.Opt) \leq type(s_2.Ipt) \Rightarrow composable(s_1, s_2)$$

The rule r_2 has higher priority than the rule r_1 .

$$r_2 > r_1$$

- **Domain specific rules**

Domain specific rules are used to describe pre-conditions and effects of services and business rules and policies in a specific business domain. They are integrated into the service descriptions. Here we use the money transfer service discussed in Section 4.3.1 to illustrate some domain specific rules.

r_4 : if the service is supplied with valid account details, then typically the money will be transferred correctly.

$$valid(account\ details) \Rightarrow result(s)$$

r_5 : if accounts are in UK, then this service is definitely applicable.

$$location(UK) \rightarrow applicable(s)$$

r_6 : if a user chooses the money transfer service and the currency conversion service from the same bank, then typically a 20% discount on the total commission fees is applicable.

$$s_1.provider=s_2.provider \Rightarrow totalFees=(s_1.fees+s_2.fees) \cdot (1-20\%)$$

By performing reasoning and deduction on these rules, the system can derive conclusions that help to correctly construct composite services and validate the generated composite services. Defeasible Logic's built in priority handling mechanism enables the system to automatically arrange alternative services in case an error occurs during the service composition process or service execution. When the required services are located or generated, they will be validated by the appropriate rules. Two types of validation are performed on services:

- **Composable validation:** Two or more services may be syntactically composable according to their input and output data types and semantics. However, the generated composite services may be logically incorrect. In this type of validation, business rules and policies are used to ensure that the generated composite services satisfy the business logic and constraints in its domain.
- **Triggerable validation:** A service that satisfies a service requirement can still be an invalid service if the service's pre-conditions cannot be satisfied. Business rules and policies can affect the service execution and must be considered during the triggerable validation process. Furthermore, some services require authentication and authorisation and some services come with different security policies. If the security requirements cannot be satisfied, then the services cannot be executed.

4.4 Transformation Method

Web services as a distributed computing technology have been around for a long time. Many enterprises and organisations have adopted Web services as a crucial technology in areas such as enterprise application integration, business intelligence, and data integration [Linthicum, 2003]. Existing services have been described using a variety of service description frameworks. The most popular ones are WSDL and OWL-S. To enable an enterprise or an organisation to use the newly proposed service description framework without affecting their existing service operations, a sound transformation method is needed. In this section, we propose an agile method that

transforms the existing Web service descriptions into CbSSDF based service descriptions. This method uses a bottom up approach that will gradually collect the information required by CbSSDF from a service's existing description. The method has three steps: 1) the ontology based service classification step 2) the CUPS generation step, and 3) the S-CGs generation step.

4.4.1 Step One: Ontology based service classification

Suppose we have a service ontology \mathcal{O} that contains all the service concepts that are relevant to the services in a specific business domain, the classification process can be performed based on \mathcal{O} . Each service concept in \mathcal{O} has a set of data properties corresponding to the information described in SSDM, such as inputs, outputs, and metadata. Through semantic and keyword matching the information contained in the data properties and in the existing service descriptions, services can be linked to service concepts in \mathcal{O} . In order to obtain the best match, each data property has a set of semantically identical literals as its value rather than a single literal value. For example, a weather forecast service concept's name property may contain a set of literals, such as "weather", "weather forecast", and "weather report". If one or many of these words appear in a service's existing description, then this service may be relevant to the weather forecast service concept. The more data property values of a service concept that are matched in a service's description, the more relevant the service is to the service concept.

As the link between a service concept and a service is determined by a series of factors, (i.e. the data properties of the service concept), vector based similarity measurement methods [Berry et al. 1999; Joachims 1998; Xu et al. 2005] can be applied here to measure the relevance between a service and a service concept. Once a service is associated with a service concept, the best matched literal value in each data property of the service concept will remain as a part of the SSDM description of the service. The service classification process obeys the constraint that if a service is associated with a service concept and the service concept's super-concept, then the super-concept association will be ignored, i.e. the service will be the super-concept's indirect instance service.

How much information can be acquired from an existing service description depends on the type of the service description. A WSDL based service description can provide some basic information, such as service name, input and output data types, and some data semantics only when meaningful tags are used, for example, “<wSDL:operation name="GetLatestWeather">”. If a service’s WSDL document is accompanied with a natural language description, more information can be gained. Nowadays, as Semantic Web Services technologies are becoming mature, some organisations provide semantically annotated Web service descriptions, such as MINDSWAP⁴, which provides a list of Web services described using OWL-S. The semantically annotated service descriptions can make the classification process more accurate.

4.4.2 Step Two: CUPs generation

Let $S = \{s_1, s_2, \dots, s_n\}$ be a set of services that have been classified using the ontology \mathcal{O} . By applying the service link definition, we determine service links between services by evaluating their input and output compatibility and pre- and post conditions satisfaction. Using service links the services in S are linked into a set of graphs. These graphs represent the composability between services in S . We call them instance services graphs, which are defined as follows:

Definition 4.7. An instance services graph is a directed graph, denoted as $G = \langle V, E \rangle$, where:

– $V = \{s_1, s_2, \dots, s_m\}$: a set of instance services, $V \subseteq S$.

– $E = \{\ell(s_i, s_j) \mid s_i.out_p \gg s_j.in_q\}$, $s_i, s_j \in V$: a set of service links.

– Loops in E are allowed, e.g. $\ell(s_i, s_i) \mid s_i.out_p \gg s_i.in_q$, which means that s_i is repeatedly invoked until the desired condition is met.

The smallest instance services graph can be just one vertex, i.e. a single service. It means that the service is not able to be composed with other services in S and only works individually.

After services in S are connected into a group of instance services graphs, the CUPs of each service in these graphs can be generated based on the service links in the graphs.

⁴ <http://www.mindswap.org/2004/owl-s/services.shtml>

Let s_i be a service in an instance services graph, its CUP candidate services are the services that have service links with s_i . How many CUPs a service can have is dependent on how many inputs and outputs the service has and how many services are linked to its inputs and outputs through service links. Let N_{in} be the number of the inputs of s_i , N_{out} the number of the outputs of s_i , the number of CUPs of s_i at this stage is:

$$N_{CUP} = \prod_{k=1}^{N_{in}} I_k \cdot \prod_{h=1}^{N_{out}} O_h$$

where, I_k is the number of services connected to the k -th input of s_i and O_h is the number of services connected to the h -th output of s_i . However, not all CUPs generated at this stage are meaningful in a given business domain. They have to be re-evaluated once the appropriate S-CGs are generated.

4.4.3 Step Three: S-CGs generation

To generate S-CGs, we first replace instance services in the instance services graphs generated from the previous step with their associated service concepts allocated in step one. If two services s_i and s_j in an instance services graph have been replaced with the same service concept, these two services are considered as semantically equivalent services, written as $s_i \doteq s_j$. Let s_k be another service in the same instance services graph, if $s_i \doteq s_j$, then $l(s_i, s_k)$ and $l(s_j, s_k)$ are two semantically equivalent service links. Here we ignore how the inputs and outputs of s_k , s_i , and s_j are connected as it does not influence the conceptual relationship between services, only the order of services in service links matters. Now, if we merge the same service concepts and the semantically equivalent service links and replace service links with conceptual relations, the instance services graphs can be converted into service concepts graphs⁵. The conceptual relationships between services are business domain specific. Suppose we have a set of predefined service conceptual relations \mathcal{R} for a business domain and a labelling function $\mathcal{R}()$, then we can label the edges in each service concepts graph with conceptual relations by giving a pair of service concepts, i.e. $\mathcal{R}(c_{s_i}, c_{s_j}) \in \mathcal{R}$. During the

⁵ The service concepts graph is an intermediate stage towards S-CG. The difference between a service concepts graph and an S-CG is that a service concepts graph has conceptual relations labelled edges instead of relation nodes.

labelling process, if two service concepts cannot be labelled by a concept relation in \mathcal{R} , it means that their corresponding instance services' service links are irrelevant to the business domain and therefore need to be removed. CUPs generated from the previous step that contain irrelevant service links also need to be removed. Finally, to obtain S-CGs, we need to replace the conceptual relations, such as $\mathcal{R}(c_{si}, c_{sj})$, with relation nodes that can be generated based on $\mathcal{R}(sc_i, sc_j)$. According to the corresponding instance services' input and output data types and semantics, extra concept nodes may be introduced into the graphs.

Through the process of converting instance services graphs into S-CGs we can see that relations between instance services, i.e. the composability of those services, have significant impact on conceptual relations between service concepts. In other words, low level instance services determine what kind of high level business services that an enterprise can provide. If there are mismatches between instance service links and their corresponding service conceptual relations, it means that either the enterprise has proposed business services that cannot be achieved by their instance services, or the enterprise has the potential ability to provide more business services.

4.4.4 An Example

Suppose we have two "Money_Transfer" services and two "Currency_Conversion" services that are described using WSDL plus text description. The WSDL plus text description is a common approach to describe Web services [Hull et al., 2003].

To convert the example services' descriptions into the CbSSDF based description, the first step is the ontology based classification. By analysing the tags such as "<wsdl: types>", "<wsdl: message>", "<wsdl: operation>", and "<wsdl: service>" in the example services' WSDL documents and applying the vector based text similarity matching methods [Berry et al. 1999; Joachims 1998; Xu et al. 2005], we can get the results shown in Table 4.1.

As the WSDL based service description does not provide pre- and post-condition information, we need to analyse the text description of each service to find relevant information. For example, a part of the "Money_Transfer" service's description is "only operate on current accounts in Pound". Based on this information, one of the

pre-conditions for the “Money_Transfer” service is: *Currency.type = Pound*. All of these four services have no internal structure, i.e. they are atomic services.

Table 4.1. Acquired information from the WSDL and ontology based classification.

Web services	Service Concept	Inputs	Outputs
Money_Transfer (s_1)	Banking_Service	<i>Input₁</i> : Data Type: <i>Double</i> Concept: <i>Currency: Pound</i> <i>Input₂</i> : Data Type: <i>Account</i> Concept: <i>Bank_Account</i> <i>Input₃</i> : Data Type: <i>Account</i> Concept: <i>Bank_Account</i>	<i>Output₁</i> : Data Type: <i>Boolean</i> Concept: <i>Transfer_Status</i>
Money_Transfer (s_2)	Banking_Service	<i>Input₁</i> : Data Type: <i>Double</i> Concept: <i>Currency: Dollar</i> <i>Input₂</i> : Data Type: <i>Account</i> Concept: <i>Bank_Account</i> <i>Input₃</i> : Data Type: <i>Account</i> Concept: <i>Bank_Account</i>	<i>Output₁</i> : Data Type: <i>Boolean</i> Concept: <i>Transfer_Status</i>
Currency_Conversion (s_3)	Financial_Tool	<i>Input₁</i> : Data Type: <i>Double</i> Concept: <i>Currency: Dollar</i>	<i>Output₁</i> : Data Type: <i>Double</i> Concept: <i>Currency: Pound</i>
Currency_Conversion (s_4)	Financial_Tool	<i>Input₁</i> : Data Type: <i>Double</i> Concept: <i>Currency: Euro</i>	<i>Output₁</i> : Data Type: <i>Double</i> Concept: <i>Currency: Dollar</i>

The second step is to generate CUPs. First, we need to determine service links between services based on the services’ inputs, outputs, pre-conditions, and post-conditions. In this example, three service links are established: $\ell(s_3, s_1) \mid s_3.out_1 \gg s_1.in_1$, $\ell(s_4, s_2) \mid s_4.out_1 \gg s_2.in_1$, and $\ell(s_4, s_3) \mid s_4.out_1 \gg s_3.in_1$. The services are then connected and form an instance services graph, which is shown in Figure 4.3:

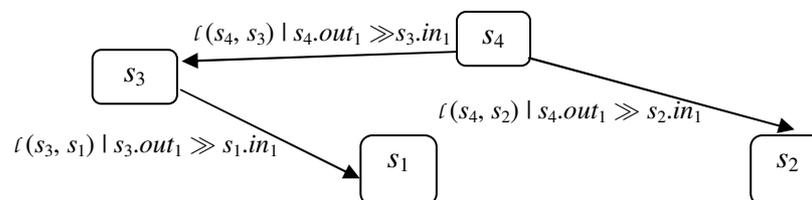


Figure 4.3. The generated instance services graph.

From the instance services graph, we obtain the CUPs of each service through their service links. For example, s_4 's CUPs are $\langle \{s_2\}, \{ \ell(s_4, s_2) \mid s_4.out_1 \gg s_2.in_1 \} \rangle$ and $\langle \{s_3\}, \{ \ell(s_4, s_3) \mid s_4.out_1 \gg s_3.in_1 \} \rangle$.

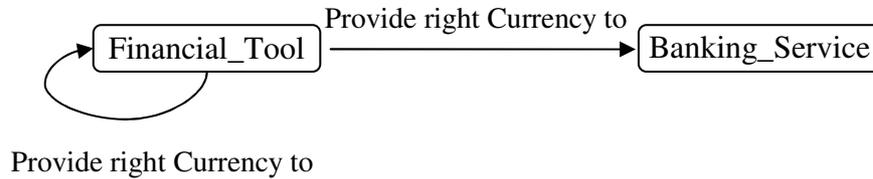


Figure 4.4. The generated service concepts graph.

The last step is to convert the instance services graph into an S-CG. We first replace services with their service concepts in the graph and merge the same service concepts and semantically equivalent service links, then we label the edges with the domain specific conceptual relations to form a service concepts graph, see Figure 4.4. In this example, s_1 and s_2 , s_3 and s_4 are semantically equivalent services and $\ell(s_3, s_1) \mid s_3.out_1 \gg s_1.in_1$ and $\ell(s_4, s_2) \mid s_4.out_1 \gg s_2.in_1$ are semantically equivalent service links. There is no service link that cannot be labelled by the conceptual relation and therefore no irrelevant service links and CUPs need to be removed.

Finally, we can get an S-CG by replacing the conceptual relation labels in the service concepts graph with the relation nodes, see Figure 4.5. The “Currency” concept is added according to the semantics of the instance services’ inputs and outputs.

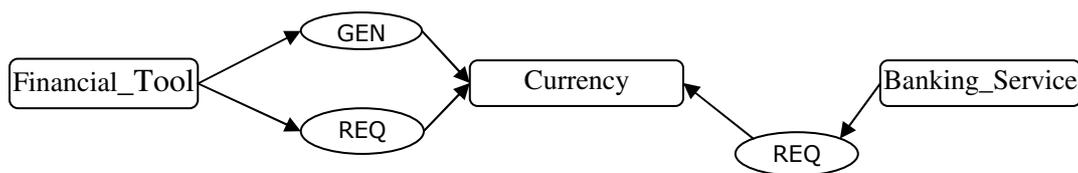


Figure 4.5. The generated S-CG.

4.4.5 A Note on Information Loss During Transformation

In this section, we discuss the information loss which may happen during the transformation process. After the transformation process, if all the information about service attributes, semantics, and service capabilities can be kept in the generated CbSSDF description, then there is no information loss, otherwise, information loss occurs. To discuss whether information loss happens when transforming from the original service description to the CbSSDF description, we need to categorise service

descriptions into two categories, i.e. the WSDL based service descriptions and the semantic service descriptions.

For a WSDL based service description, our transformation method can ensure that no information is lost. The reason is that the CbSSDF is a richer service description framework and all aspects of information described in a WSDL description will be fully described in the CbSSDF based description. The transformation methods will relocate the information in WSDL to relevant components in CbSSDF, such as inputs, outputs, and service banding. The transformation method will also extract semantic information from the content of a WSDL description. However, this step does not involve information loss since service semantics are not formally described (and hence not used by a WSDL parser) as a part of a WSDL description.

For a semantic service description, whether the information loss occurs or not depends on the type of semantic service description framework being used. As CbSSDF is generally a richer semantic service description framework, it subsumes all the machine-processable constructs of the other semantic service description frameworks, such as OWL-S and WSDL-S. Similar to the WSDL case, the transformation on these frameworks causes no information loss. However, some of the semantic service description frameworks cannot be transformed into CbSSDF without information loss. This is due to mismatched aspects. For example, the concept of a mediator [Fensel & Bussler, 2002] in WSMF is not addressed in CbSSDF. In this case, the mediator information will be lost during the transformation process.

4.5 Summary

In this chapter, we have proposed a comprehensive Web service description framework – CbSSDF. This framework takes into consideration not only the capability semantics (technical details) of Web services, but also the Service Usage Context (SUC). As discussed in Chapter 3, at the conceptual level, the SUC of a service provides the information that helps to identify whether a service is suitable for a service user's scenario. At the instance level, the SUC of a service can help to improve the efficiency and effectiveness of service discovery and composition. A transformation method is provided so that the existing service descriptions, such as in WSDL and OWL-S, can be transformed into the CbSSDF based descriptions.

The features of CbSSDF that make it distinct from the other service description frameworks are listed as follows:

- **Integrating SUC into the service description.** The conceptual level SUC, i.e. the \mathcal{T} -Context, is represented by a set of S-CGs; the instance level SUC, i.e. the \mathcal{A} -Context is represented by a set of CUPs in SSDM.
- **Richer semantics for service description.** SSDM gives a comprehensive description of a service's capabilities through four types of service semantics, i.e. data semantics, functional semantics, non-functional semantics, and execution semantics. It also addresses the inter-service relationships, which enhance the semantic description of the service's capabilities and improves the efficiency of service discovery and composition process. In existing service description frameworks, the inter-service relationships between services are not addressed.
- **Adopting the non-monotonic rules in describing service conditions.** Non-monotonic rules are used to represent the pre- and post-conditions of services as well as the general service composition rules. They are used to validate the correctness of the generated composite services and to ensure that the located or generated services can be executed in the given situation. In existing service description frameworks, monotonic rules are used to represent service conditions. However, due to the complexity of service conditions, monotonic rules are not sufficient to handle conflicts and incomplete information.

Although this framework is proposed for Web services, it can be applied to any component or object that has characteristics similar to a Web service, such as well defined interfaces, platform independence, programming language independence, composability, and self-containment. So far, we have applied the framework to Web services, software services in SaaS (Software as a Service [Choudhary, 2007]) [Du et al., 2008a], and learning objects in web-based learning systems [Du et al., 2007b].

The new features provided by CbSSDF enable us to develop a new and effective way to search services. In the next chapter, we will discuss a two-step service discovery mechanism that provides service users with a more flexible way to search services.

Chapter 5: Two-Step Service Discovery Mechanism

In this chapter, we will discuss a two-step service discovery mechanism, which is based on CbSSDF. It provides service users with a more flexible way to search services.

5.1 Overview

In this chapter, we introduce a two-step service discovery mechanism. The aim of the service discovery mechanism is to help service users to locate required services easily with more flexibility. By using this mechanism, service users are not required to provide detailed technical specifications about the required services at the beginning of the service discovery stage when they have little or no knowledge about the services. The two-step process will guide service users to gradually refine their requirements to determine the required services. The initial service query for the can be a natural language description or a list of keywords. This mechanism also improves the efficiency of the service composition process by using the features provided by the CbSSDF based service description. In the following sections, we discuss each step of the service discovery mechanism in detail.

5.2 Two-Step Service Discovery Mechanism

When service users search for services, usually they already have a picture in their mind about the goal they want to achieve. They also know what they already have that will aid them in achieving the goal and which additional functionalities are needed from the required services in order to achieve the goal. In other words, a usage scenario is already formed in a service user's mind regarding how the required services are going to be used. If the service discovery can be performed not only on the technical details of required services, but also on the service user's usage scenarios, i.e. how the services are going to be used, then it can be easier and more flexible. Currently, semantic service search engines and the ordinary UDDI service search engine all require technical details of the required services. To use these service search engines, service users need sufficient domain knowledge about the request services in order to provide technical specifications. For example, the OWL-S based semantic service discovery solution proposed by Paolucci et al. [Paolucci et al, 2002] requires service users to clearly state the semantics of a required service's inputs and outputs, data types of the service's inputs and outputs, and the service functionality etc. Further more, this information is usually required at the very beginning of each service discovery process. However, more often than not, service users are not domain experts in the required service's area and therefore, it is very hard for them to provide

the technical specifications at the very beginning of the service discovery process. Consequently, these service search engines have to carry out searching with inadequate requirement information and therefore return inaccurate results.

To solve this problem, we propose an enhanced (two-step) service discovery mechanism based on the CbSSDF based service description. S-CGs in the CbSSDF, i.e. the τ -Context, help the service search engine to locate services by using their concepts and conceptual relations, so in the first step a service user only needs to describe their requirements or scenarios in natural language without worrying about any technical detail. The search engine will then convert the natural language query into a CG and match with the S-CGs in the service repository to locate the relevant services. The located services may or may not be an exact match for the required services. However, the match suggests that these services are relevant to the user's query and therefore may provide a solution to the query. After the first step, the service user gets a list of relevant services with their technical descriptions attached. These service descriptions can act as hints to assist the user in providing detailed technical specifications according to their own situation. The second step is to refine the result from the first step using the provided technical specification, generate composite services, and rank the results according to their degree of similarity to the specification. The SSDM and the non-monotonic rules are indispensable in this step. The significant difference between the two-step service discovery mechanism and the traditional service discovery methods are that after the first step, the users can more easily propose detailed technical service specifications based on the initial results. Using this additional information the service composition process can be much more efficient due to the implementation of \mathcal{A} -Context in the SSDM. The two-step service discovery mechanism demonstrates how the features provided by CbSSDF can facilitate the service discovery and composition.

5.2.1 Step One: S-CG based Service Retrieval

In this step, the aim of the service search engine is to find the relevant services from the service repository. This can be achieved by using the CG matching mechanism [Montes-y-Gómez et al., 2001]. A service query is first converted into a CG and then matched with the S-CGs in the service repository. As discussed previously in Section

4.3.1, if a service query is conceptually satisfiable, its CG must have a projection⁶ on at least one S-CG. Therefore, in this step, we aim to finding an S-CG that contains the projection of the query CG. This S-CG can be either an existing S-CG in the service repository or a join of existing S-CGs. There are many well developed algorithms [Croitoru & Compatangelo, 2006] [Mugnier & Chein, 1992] that can be used to check for CG projections. In the first step we look for relevance rather than exactitude, this means that our CG matching method can be much more flexible than the formal CG projection checking methods proposed in the literature. There are six situations in which an S-CG's corresponding services can be considered as relevant to a service query:

- **Exact match:** A query CG is exactly matched with one or more S-CGs.
- **Projection:** A query CG has a projection in one or many S-CGs.
- **Composite projection:** A query CG has a projection in an S-CG that is generated by joining existing S-CGs.
- **Overlap:** A query CG has overlap concepts or relations with one or many S-CGs.
- **Concept match:** A query CG has only concept nodes matched with one or more S-CGs' concept nodes.
- **Relation match:** A query CG has only relation nodes matched with one or more S-CGs' relation nodes.

In the six situations, the relevance level of services is gradually decreases from “Exact match” to “Relation match”. We use an algorithm to categorise services in the repository into the different relevance situations and then we perform the CG similarity calculation to calculate the actual relevance degree of the services in each situation. The algorithm is shown in listing 5.1. The relevance levels are from 1 to 6, 1 represents “Exact match” and 6 represents “Relation match”. In order to improve performance, “Composite projection” is checked last and only on the services that are confirmed as relevant this is because if a set of S-CGs have no common concepts with a query CG, the derived graphs from them will not contain a projection of the query CG.

⁶ For the definition of projection, see Section 2.3.3.

```

List relevantServices = null;
CG q = query.CG;
S = {s1, s2, ..., sn}; //the service repository
for each s ∈ S do
{
    if projectionCheck(q, s.S-CG) == true then
    {
        if exactMatch(q, s.S-CG) == true then
        {
            s.setRelevanceLevel(1);
            relevantService.add(s);
            continue;
        }
        else
        {
            s.setRelevanceLevel(2);
            relevantService.add(s);
            continue;
        }
    }
    if overlapCheck(q, s.S-CG) == true then
    {
        s.setRelevanceLevel(4);
        relevantService.add(s);
        continue;
    }
    if commonConceptCheck(q, s.S-CG) == true then
    {
        s.setRelevanceLevel(5);
        relevantService.add(s);
        continue;
    }
    if commonConceptCheck(q, s.S-CG) == true then
    {
        s.setRelevanceLevel(6);
        relevantService.add(s);
        continue;
    }
}
if compositeProjectionCheck(q, relevantServices.S-CGList) == true then
{
    List participatedServices = compositeProject.getServices()
    for each s ∈ participatedServices do
    {
        if s.getRelevanceLevel() != 1 or s.getRelevanceLevel() != 2 then
        s.updateRelevanceLevel(3);
    }
}

```

Listing 5.1. CG similarity based service relevance classification algorithm.

After the relevance level categorisation process is completed, the actual relevance/similarity degree to the service query of each relevant service under each relevance level needs to be computed, except services with relevance level “1”. The relevance/similarity degree is calculated using the CG similarity measurement method.

The method we use to compute the CG similarity is proposed by Montes et al. [Montes-y-Gómez et al., 2001]. According to Montes et al., the similarity between two CGs, u and v , consists of a concept similarity S_c and a relation similarity S_r . The

concept similarity S_c is calculated using the Dice coefficient [Frakes & Baeza-Yates, 1992] expression:

$$S_c = 2 \left(\sum_{c \in \bigcup O} (weight(c) \times \beta(\pi_u c, \pi_v c)) \right) / \left(\sum_{c \in u} weight(c) + \sum_{c \in v} weight(c) \right)$$

where, O is a set of common overlap graphs of u and v ; $\bigcup O$ is the union of all of the common overlap graphs of u and v ; $\pi_u c$ and $\pi_v c$ represent the concepts coming from graphs u and v ; $weight(c)$ is the importance factor of the concept type c . Its value can be various in different applications. Currently, we distinguish two types of concepts:

$$weight(c) = \begin{cases} w_D & \text{If } c \text{ is a domain concept} \\ w_S & \text{If } c \text{ is a service concept} \end{cases}$$

The set of common overlaps O represents all common elements between u and v . The overlaps include not only the direct overlaps of the two graphs, but also the common generalisation of the two graphs.

The $\beta(\pi_u c, \pi_v c)$ function in the above expression calculates the semantic similarity between the two concepts $\pi_u c$ and $\pi_v c$, defined as follows:

$$\beta(\pi_u c, \pi_v c) = \begin{cases} 1 & \text{if } type(\pi_u c) = type(\pi_v c) \text{ and } referent(\pi_u c) = referent(\pi_v c) \\ depth / (depth + 1) & \text{if } type(\pi_u c) = type(\pi_v c) \text{ and } referent(\pi_u c) \neq referent(\pi_v c) \\ 2d_c / (d_{\pi_u c} + d_{\pi_v c}) & \text{if } type(\pi_u c) \neq type(\pi_v c) \end{cases}$$

The first condition indicates that the two concepts are exactly the same. The second condition indicates that the two concepts have the same type but refer to different instances, where $depth$ represents the number of levels in the ontology that contain both concepts. The third condition indicates that the two concepts have different types, where, d_c represents the distance from the least common super-type of $\pi_u c$ and $\pi_v c$ to the root of the ontology; $d_{\pi_u c}$ and $d_{\pi_v c}$ represent the distances from $\pi_u c$ and $\pi_v c$ to the root of the ontology.

The relation similarity S_r is calculated using the following expression:

$$S_r = \frac{2m(o)}{m_c(u) + m_c(v)}$$

where, $m(o)$ is the number of the relation nodes in the common overlap graphs of u and v ; $m_c(u)$ and $m_c(v)$ are the numbers of the relation nodes of the common overlap graphs of u and v and the overlap graphs' adjacent relation nodes. To examine the similarity between two relations, we need to compare not only the two relations themselves, but also the neighbour relations that linked to these two relations.

The actual relevance/similarity degree expressions under different situation are shown below:

$$Sim = \begin{cases} S_c = 2 \left(\sum_{c \in \bigcup O} (weight(c) \times \beta(\pi_u, c, \pi_v, c)) \right) / \left(\sum_{c \in u} weight(c) + \sum_{c \in v} weight(c) \right) & \text{if } S_r = 0 \text{ and } S_c \neq 0 \\ S_r = \frac{2m(o)}{m_c(u) + m_c(v)} & \text{if } S_c = 0 \text{ and } S_r \neq 0 \\ S_c \cdot S_r & \text{if } S_c \neq 0 \text{ and } S_r \neq 0 \end{cases}$$

The first condition applies when two CGs only have common concept nodes, i.e. the ‘‘Concept match’’ situation. The second condition applies when two CGs only have common relation nodes, i.e. the ‘‘Relation match’’ situation. The last condition applies when two CGs have overlap, i.e. the ‘‘Overlap’’ situation. However, the ‘‘Projection’’, and the ‘‘Composite Projection’’ situations are special cases of overlap, thus the last condition is also applicable on these situations.

After this step, the services in the service repository have been categorised as either relevant services or irrelevant services. The relevant services will be passed to the second step of the two-step service discovery mechanism for further refinement. These services have the potential to satisfy the service requirements. However, the irrelevant services will also be considered during the service composition process when relevant services cannot fully fulfil service requirements.

5.2.2 Step Two: SSDM based Service Composition and Ranking

The second step aims to refine the result from the first step. After the first step, services have been coarsely ranked according to the conceptual distance between them and the service requirement. However, being conceptually close to a service requirement does not necessarily mean that the service actually satisfies the requirement because some technical details of the service may not be compatible with the requirement's technical specification. In the second step, based on the user's further detailed technical specification, the services from the first step will be checked and services that satisfy or partially satisfy the technical specification will be selected and ranked. If the requirement cannot be satisfied by a single service, composite services may be generated. Another advantage that the first step provides is that it reduces the number of candidate services for the second step to process because only the conceptually relevant services are passed to the second step.

The refinement is based on semantic distance and technical detail matching between a user's specification and service attributes in each service's SSDM description. The major difference between the two-step service discovery method and the traditional service discovery methods is that by having the result from the first step, service users are edified and therefore able to provide further technical detail to describe their particular needs. Even if the service users still cannot provide the full technical specification, partial detail is acceptable and the step two can be repeated until an appropriate result is found.

As discussed previously in Section 4.3.2, the attributes provided in SSDM for service description are inputs, outputs, pre-conditions, effects, the service internal structure, CUPs, and the service metadata. If we use a vector v to represent the attributes of a service, then we can build up a $t \times m$ vector space V , where t is the number of terms in v and m is the number of services in a service repository (or the candidate services from step one). The vector space V is represented as below:

$$V = \begin{pmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1m} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{t1} & a_{t2} & \cdot & \cdot & a_{tm} \end{pmatrix}$$

where, the columns of V are service attribute vectors, the rows of V are term vectors, a_{ij} is the i -th attribute term of service j , and t represents the number of service attributes addressed in SSDM. A user requirement can also be represented as a vector $r=(q_1, q_2, \dots, q_t)$. Then, we apply the cosine similarity method to measure the distance between the service user's specification and the service attributes in SSDM. The cosine similarity method [Berry et al., 1999] uses the cosine value of the angle between two vectors to measure the similarity between these two vectors. The cosine value of the angle is computed using the following formula:

$$\cos \theta_{vr} = \left(\frac{v_j \cdot r}{\|v_j\|_2 * \|r\|_2} \right)_{j=1}^m = \left(\frac{\sum_{i=1}^t a_{ij} q_i}{\sqrt{\sum_{i=1}^t a_{ij}^2} * \sqrt{\sum_{i=1}^t q_i^2}} \right)_{j=1}^m$$

where, m is the number of services in the service repository (or the candidate services from step one), t is the number of attributes addressed in SSDM, a_{ij} is the i -th attribute term of service j , and q_i is the i -th term addressed in a query. A smaller angle represents a higher similarity between two vectors.

However, many attributes addressed in SSDM contain sub-attributes, such as inputs, outputs, the service internal structure, CUPs, and the service metadata. For example the service metadata attribute has many sub-attributes, such as the service provider's information, the service subject area, the region of the service, and versioning information. These sub-attributes themselves can also be considered as vector spaces. Therefore, V is actually a vector space with sub-spaces. A vector v_j in V is graphically illustrated in Figure 5.1.

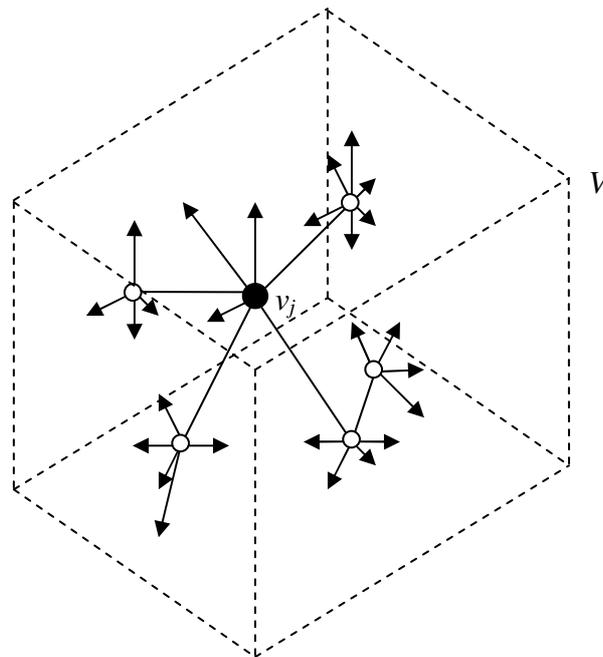


Figure 5.1. A graphical representation of part of the vector space V .

Each line or arrow in Figure 5.1 represents an attribute or a sub-attribute of a service. In reality, there should be many service vectors like v_j in the cube, i.e. V . We treat V and its sub-spaces as a tree structure and use a recursive algorithm to compute the cosine similarity between a service specification and each service attribute vector and its sub-vectors. Finally, an overall similarity degree for each service is calculated and the result services are ranked according to the similarity degree.

The cosine similarity calculation and the vector space model try to find the most suitable services for a service requirement. However, there are many cases where a service requirement cannot be satisfied by a single service, i.e. the required service does not exist in the service repository. In these cases, we need to employ the service composition approach [Dustdar & Schreiner, 2005] to dynamically construct composite services to fulfil the service requirement. In the following, we discuss how CbSSDF can facilitate the service composition process and improve its efficiency.

One of the most common approaches for service composition is to use AI planning techniques [Sirin et al., 2004] [Zhang et al., 2004]. Planning is about producing state changes through actions in order to achieve a desired goal [Yang, 1997]. A planning problem can be defined as below.

Definition 5.1. A planning problem is a 5-tuple (X, U, f, x_I, X_G) , where,

- X : is a set of states that represents all the distinct situations in a planning domain. X is finite or countably infinite and $X \neq \emptyset$
- U : is a set of actions. Each action $u_i \in U$ produces a new state x' after applied on the current state x , $x, x' \in X$.
- f : is a state transition function. $x' = f(x, u)$
- x_I : is an initial state, $x_I \in X$.
- X_G : is a set of goal states, $X_G \subset X$.

A planning algorithm's task is to find a sequence of actions in U that can transform an initial state x_I to a desired goal state x_g , $x_g \in X_G$, formally represented as below:

$$f(f(f(\dots f(x_I, u_1)\dots), u_{n-1}), u_n) \in X_G, u_i \in U, 1 \leq i \leq n$$

From the expression we can see that the time complexity of a planning algorithm is dependent on the number of steps required to achieve the goal and the number of candidate actions for each step. Using the Forward-Chaining Total-Order (FCTO) planning algorithm as an example, the worst case time complexity of FCTO [Yang, 1997] is:

$$T(FCTO) = O(B^N * t)$$

where, B is the number of candidate actions for each step, N is the number of steps in a plan for achieving the goal, and t is the average time spent by a planner on each step. However, irrespective of which planning algorithm is used, the time complexity is always related to N and B . If either of these two numbers can be decreased, especially N , the time complexity of the planning algorithm would be decreased.

In the Web service composition situation, a candidate service can be considered as an action, its pre-conditions are the states before its execution and effects are the states after its execution. A composite service can be considered as a plan to achieve a goal. The complexity of the service composition process can be higher than that of a normal planning task because in the existing service description frameworks, a composite service planner cannot determine which services are the potential candidates for the next step, therefore it has to check the whole service repository for each step it makes towards the composite service. This situation is improved using CbSSDF. The service composition process can become much more efficient by using the information provided in each service's SSDM. The key component for facilitating service

composition is the CUP. Each service has a set of CUPs associated with it. These CUPs improve service composition efficiency by reducing the number of potential candidate services to be considered for each step in a composite service and most importantly reducing the number of steps that the composite service planner takes to achieve the goal, see the analysis below:

- **Reducing the number of potential candidates:** In the set of CUPs for a service, all services that can potentially interact with the service based on the input, output data type compatibility and the pre-condition and effect constraint, have been listed. Let N be the number of services in a service repository S , S_c a set of services in the set of CUPs of a service s , and $N_C = |S_c|$, we always have $N_C \leq N$, if $s \in S$ and $S_c \subseteq S$. It is very unlikely that $N_C = N$, unless a service can interact with all other services in a service repository. Therefore, in most cases, $N_C < N$.
- **Reducing the number of steps:** Under existing service description frameworks, a sub-optimal will typically be produced because the planner cannot consider the inter-relationships between services. However, under CbSSDF, a composite service planner can construct a plan much faster than before because of the additional information provided in the CUPs. Each CUP can be considered as a fragment of a plan (or a workflow). When a service is located, its CUPs can tell the service planner what the possible services are for the next step. Therefore, the number of steps to reach a goal can be cut in half compared with the planning process under the existing service descriptions. Let B_C be the number of step that a planner need to go through to construct a composite service under CbSSDF, and B the number of steps under the existing service descriptions, we have $B_C = B/2$, if $B \geq 2$.

According to the above analysis, the time complexity expression for the FCTO algorithm under CbSSDF, can be rewritten as below:

$$T(FCTO) = O((B/2)^{N_c} * t)$$

However, the FCTO algorithm is only an example to illustrate the advantages of the CbSSDF in service composition. In fact, for any planning algorithm, if the time complexity is based on N and B , they can be more efficient when creating composite services using CbSSDF based service description.

5.3 Summary

In this chapter, we have introduced a two-step service discovery mechanism. Its aim is to demonstrate how CbSSDF can facilitate service discovery and composition. One issue of the current service discovery methods based on the existing service description frameworks is that they all require a large amount of technical detail in order to search for required services. However, sometimes it is very difficult to provide such domain specific technical information for a service user who is not a domain expert in the required area. Consequently, the service search engine has to carry out service discovery with insufficient information. By using the features provided by CbSSDF, the proposed two-step service discovery mechanism can assist service users in locating their required services in a more flexible and natural way.

The major differences between the two-step service discovery mechanism and current service discovery methods are 1) the two-step service discovery provides a more flexible way to search services. It allows service users to provide general and natural information instead of technical specifications for the required services at the beginning of the search; 2) the CbSSDF based service description can support service composition during the service discovery phase. The reason for this is that most of the solutions for service composition are based on planning algorithms which can be greatly improved using the CbSSDF based service description. The CUPs embedded in CbSSDF can reduce not only the number of candidate services in each step of a plan, but more importantly the number of steps for planner to reach a the desired goal.

We have implemented a prototype of the two-step discovery mechanism – ServiceComp, to demonstrate the advantages of CbSSDF, which will be discussed in Chapter 6. The performance evaluation result shown in Chapter 7 is also based on the implementation of the two-step service discovery mechanism.

Chapter 6: Implementation

In this chapter, we discuss the implementation details of the prototype that implements the major features of CbSSDF to facilitate service discovery and composition.

6.1 Overview

A prototype, called ServiceComp, is implemented as a part of our work to verify the applicability and performance of the proposed service description framework – CbSSDF. The main purpose of this prototype is to demonstrate that the CbSSDF based approach is feasible and applicable. The prototype is also used as a test-bed in the evaluation process to evaluate the performance and scalability of the proposed solution.

The prototype is a web based application built with Java and Java Applet technologies. It takes the advantages of the features provided by CbSSDF to facilitate service discovery and composition. One of its key features is that it implements the two-step service discovery mechanism for easy service discovery and efficient service composition. Another key feature is that it provides a graphical user interface for creating composite services manually without knowing the technical detail of service composition. Users can create composite services using drag and drop. All the services provided in ServiceComp including the composite services created by users can be directly executed. The type compatibility and SOAP message generation issues are automatically managed by ServiceComp.

The content of this chapter is organised as follow, as ServiceComp is a research prototype not a full implementation of the CbSSDF, we first discuss which features of the CbSSDF have been implemented in ServiceComp, we then introduce the interface and the system design of ServiceComp including the system architecture, the database design and the Java class diagram, finally we discuss the technologies that are used in ServiceComp implementation.

6.2 Implemented Features of CbSSDF

As ServiceComp is a research prototype for demonstration purposes, the features of CbSSDF have not been fully implemented. In this section, we summarise what features are implemented in the prototype, limitations of the implemented features, and what features have not been implemented. However, the implemented features are sufficient to examine and demonstrate the feasibility of the CbSSDF based solution.

The implemented features of the CbSSDF are as follows:

- **S-CGs:** S-CGs for each service are created and stored in the service repository. However, the example concepts in S-CGs contain only mathematical concepts for testing purposes.
- **SSDM described instance services:** All services in the service repository are described using the SSDM. The information in each service's description includes the input and output description, the service metadata, the internal structure, and CUPs. However, all services are mathematical services and the data types of each service's inputs and outputs are limited to numerical values, i.e. mathematical operands and results. There are no objects passing through services and therefore, the semantics of services and their inputs and outputs are simple.
- **Non-monotonic rules:** the non-monotonic rules are implemented in ServiceComp. However, they are only used to express the priority of mathematical operators.
- **Two-step service discovery mechanism:** The natural language query in the first step is simulated using mathematical expressions, which means that users can only use mathematical expressions to search services. However, the process used behind the scenes is the same as that discussed previously in Chapter 5, such as query interpretation, CG conversion, CG matching, and identifying related services. The second step provides an interface that collects information for matching the instance services described using SSDM and performs the vector based (cosine) similarity measurement.

The features that are not implemented are as follows:

- **S-CG join:** In the current implementation, S-CG join is not supported, which is a useful feature for dealing with complex business scenarios.
- **Alternative service allocation using non-monotonic rules:** If the non-monotonic rules feature was fully implemented, alternative services could be allocated when a service fails to deliver the expected result at runtime.

6.3 System Design and Architecture

In this section, we will introduce the interfaces of ServiceComp and its system design and architecture. The interfaces of ServiceComp include a service composition interface and a two-step service discovery interface. We will discuss their features with a series of screenshots. In the system design and architecture section, we will discuss the ServiceComp's system architecture, the implementation design, and the database (service repository) design in detail.

6.3.1 User Interface

ServiceComp provides a web based graphical user interface (GUI) to help users to search services, create composite services, and execute services. The service composition interface is shown in Figure 6.1. The main components of this interface are explained as followings:

- The panel to the left is the service repository panel. It contains a service ontology tab and an instance service tab. The service ontology gives an overall view of the type hierarchy of the services in the repository. When a class node is double clicked, its relevant instance services will be displayed in the instance service tab.
- The panel to the right is the service composition panel. Users can create composite services here by drag and drop.
- There are nine tool bar buttons on the top panel, see Table 6.1 for their functions.

Table 6.1. The description of tool bar buttons.

Tool bar Buttons	Description
	Creates new composite services.
	Saves newly created composite services into the service repository or saves the modifications of existing composition services back to the service repository.
	Switches between browsing mode and editing mode.
	Deletes the whole composite service. (only enabled in editing mode)
	Deletes a link between sub-services in a composite service. (only enabled in editing mode)

	Deletes a sub-service from a composite service. (only enabled in editing mode)
	Opens the natural language query dialog. (step-one)
	Opens the service specification based service discovery dialog. (step-two)
	Executes a service.

The GUI hides the technical details from users so that they can create composite services and then execute those services just by clicking the mouse. Those operations that require expert knowledge of Web services, such as type compatibility checking and SOAP message generation, are managed by the application automatically. However, if users do want to know the technical details, the interface also provides the option to see them.

Figures 6.2 and 6.3 show the interfaces for the two-step service discovery mechanism. In the first step, users can use a query analogous to natural language to directly query services by pressing the  button. As the current version of the prototype only supports mathematical Web services, natural language queries are simulated using mathematical expressions. After a query is proposed, ServiceComp will automatically locate relevant services, add them to the service composition pane, and provide the result if the required service had been successfully executed.

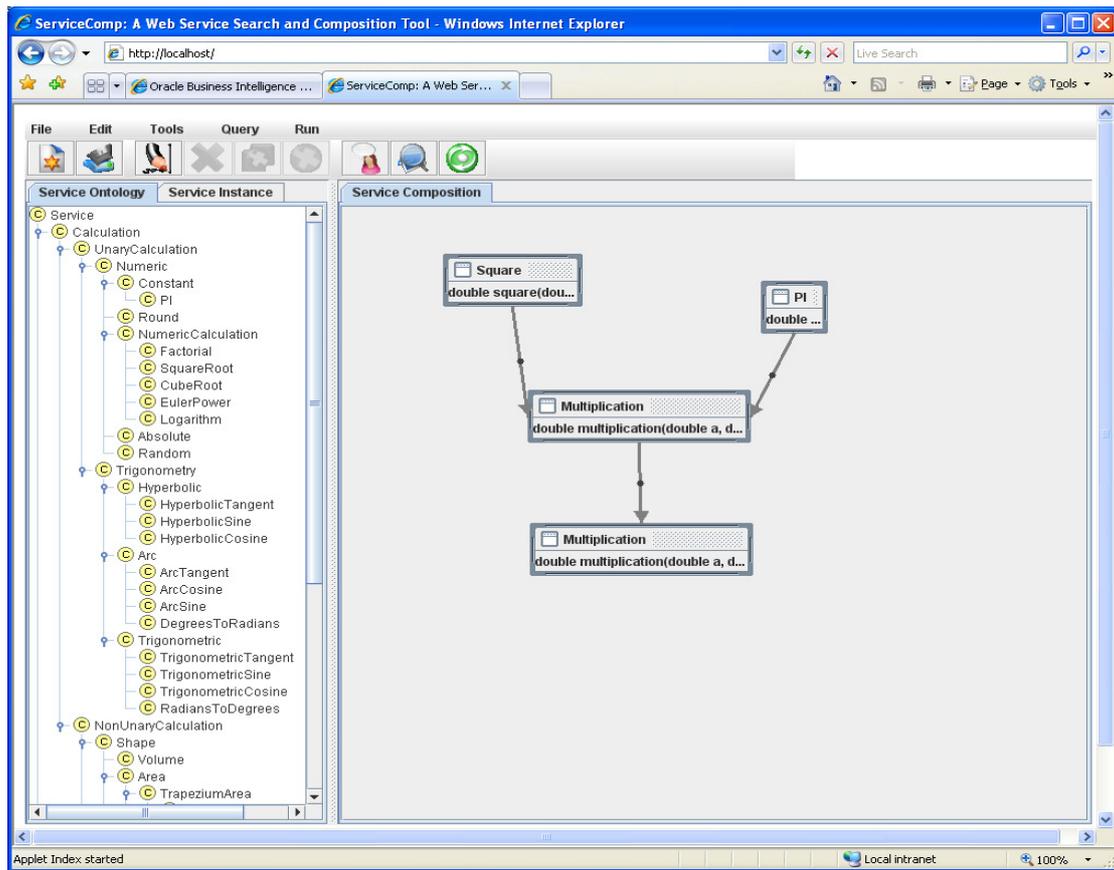


Figure 6.1. The user interface of ServiceComp.

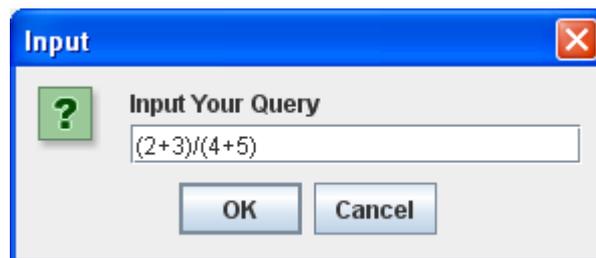


Figure 6.2. The first step query interface of ServiceComp

The screenshot shows a window titled "Search Service" with a blue border. It is divided into several sections for refining search criteria:

- Metadata Information:** Includes fields for "Service Name" (containing "addition"), "URL", and "Description Keywords". A button labeled "Individual Search" is to the right.
- Ontology Information:** Includes fields for "Direct Superclass" and "Ontology Class". A button labeled "Individual Search" is to the right.
- Data Type Information:** Includes a "Number of input" spinner (set to -1), "Inputs" and "Output Data Type" dropdown menus, and an "Input Data Type" dropdown menu (set to "double"). A button labeled "Individual Search" is to the right.
- Composite Information:** Includes fields for "Upper Pattern" and "Lower Pattern". A button labeled "Individual Search" is to the right.

At the bottom of the form is a large "Overall Search" button. Below the form, the results section is titled "Found 1 results" and displays the following information:

```

Satisfaction Rate: 1
(Name: 1, URL: 0, Keywords: 0)
Service Name: Addition
Inputs Data Type: (1) double, (2) double,
Output Data Type: double
Service Type: atomic
Keywords: +, add, plus
Service URL: http://localhost:9999/axis/services/Addition
Ontology: Service --> Calculation --> NonUnaryCalculation --> Arithmetic --> Addition

```

Figure 6.3. The second step query interface of ServiceComp

If the search results from the first step do not satisfy a requirement, the user can use the  button to start the second step interface to refine the search result with detailed technical specification.

All the services managed by ServiceComp can be directly executed. ServiceComp can dynamically generate and send the SOAP request messages at runtime based on the WSDL document of each service, process the SOAP response message contents, and return the result back to the user. To achieve this, dynamic Web service invocation technologies are used, which will be discussed in Section 6.3.

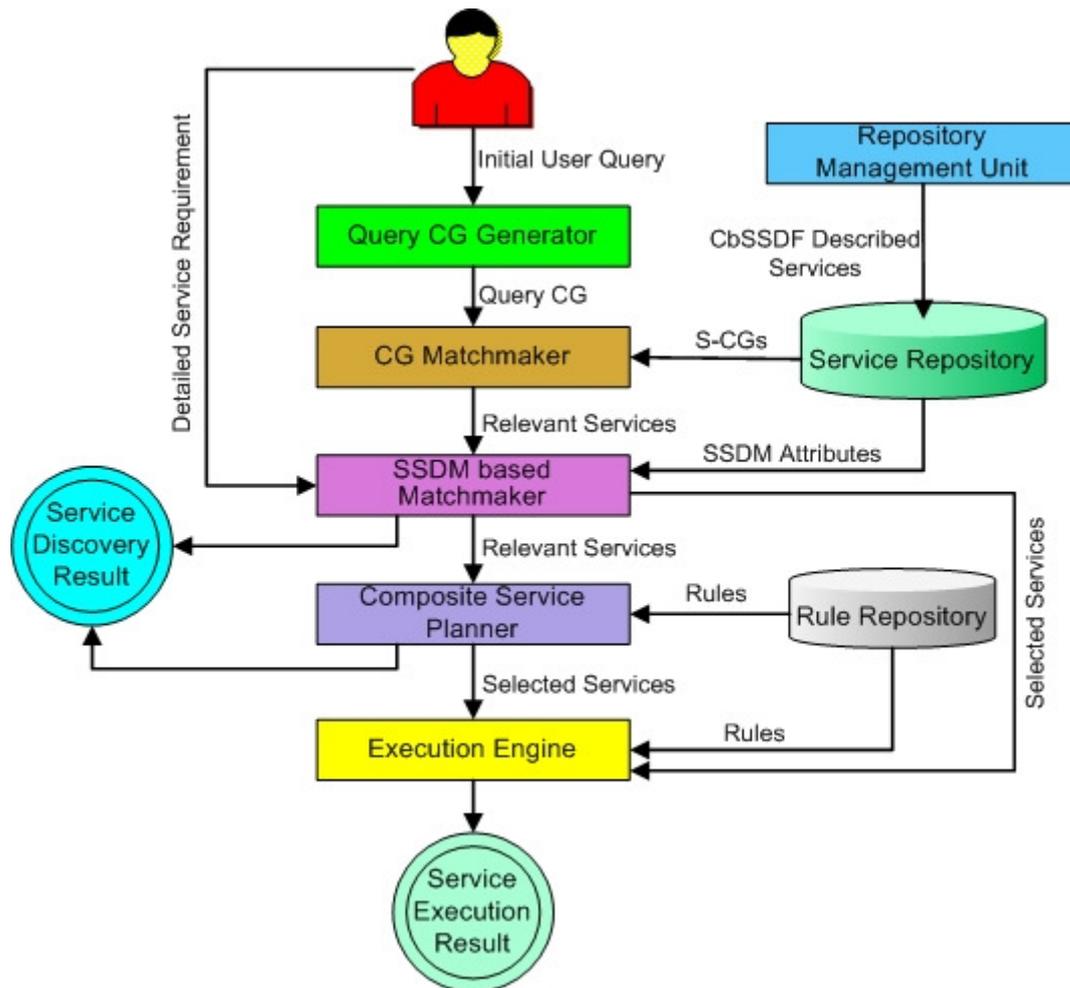


Figure 6.4. The system architecture of *ServiceComp*.

6.3.2 System Architecture

The system architecture of ServiceComp is shown in Figure 6.4. It illustrates the main components of the prototype and how they are related.

- **Query CG Generator** generates CGs from a natural language query.
- **CG Matchmaker** matches the generated CGs with the S-CGs in the service repository to locate relevant services.
- **SSDM based Matchmaker** matches the technical detail provided by the user in the second step with the SSDM in the service repository to refine the result from the first step and help the user to locate services precisely.
- **Composite Service Planner** generates composite services based on the information provided by SSDM.

- **Execution Engine** executes atomic or composite services and returns the result back to the user.
- **Repository Management Unit** provides the functionalities needed to manage the service repository, such as storing and retrieving CbSSDF based service descriptions, updating service descriptions, and creating or removing services. It also provides functions to convert other formats of service description into a CbSSDF based service description.
- **Rule Repository** stores the rules that are used for service composition and execution.

6.3.3 Implementation Design

The service repository database consists of seven tables: **Service**, **Service_Input**, **Service_Output**, **Composite_Service**, **Composite_Service_Input**, **Composite_Service_Output**, and **Graph**. The relationships among them are shown in Figure 6.5. These tables are used to store the service description information appropriate for CbSSDF.

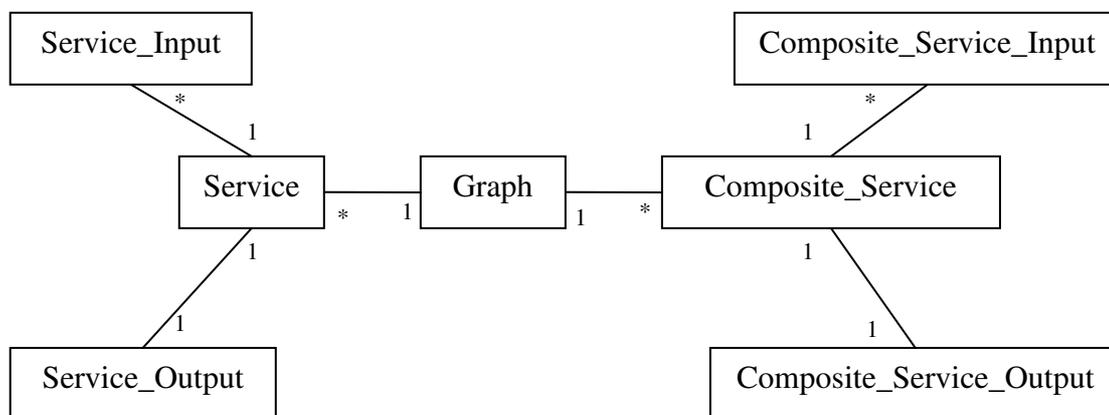


Figure 6.5. Service repository ER diagram.

The rule repository database consists of two tables: **General_Rules** and **Domain_Rules**. They are used to store the general and domain specific rules that govern the service composition and execution processes. For example, some of the general rules in ServiceComp are “ $r_1: priority ('*') = priority ('/')$; $r_2: priority ('-') = priority ('+')$; $r_3: r_1 > r_2$ ”. These tell the service composition engine and the service execution engine the priority of the arithmetic operators. The rules are described using

Defeasible Logic. However, only very basic features of Defeasible Logic are used here as there is no incomplete information in the arithmetic calculation supported by the prototype.

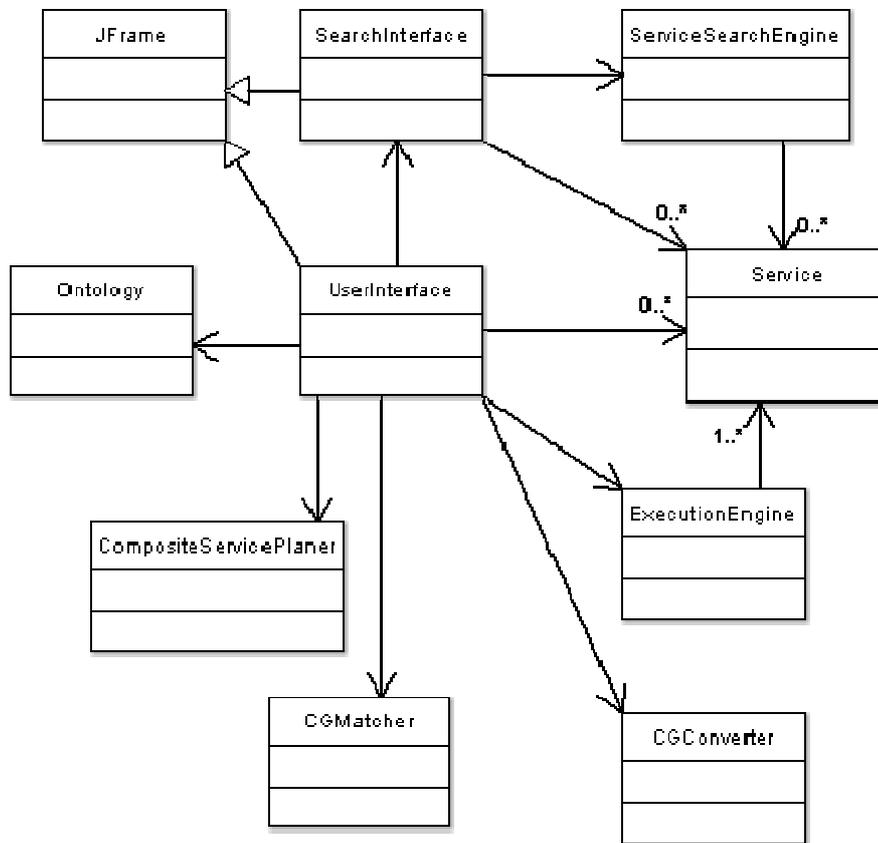


Figure 6.6. ServiceComp class diagram.

Nine Java classes have been created to implement ServiceComp. An abstract class diagram in Figure 6.6 shows the relationships among these classes. The use of each class is listed below:

- **UserInterface:** is the main class of ServiceComp that provides the user interfaces for service display and composition.
- **SearchInterface:** is the interface for the second step of the service discovery process. The interface for the first step, i.e. the natural language query dialog, is provided in the **UserInterface** class. Both the UserInterface class and the SearchInterface class are subclasses of JFrame.
- **Ontology:** is the class that provides service ontology for categorising services. It also helps the user interface to display services in a hierarchical way.

- **ServiceSearchEngine**: the class that performs the second step of the service discovery process.
- **Service**: is the class that keeps the attribute values of a service.
- **CGConverter**: is the class for converting a natural language query into CGs.
- **CGMatcher**: is the class that matches query CGs with the S-CGs in the service repository to locate relevant services.
- **ServiceComposer**: is the class that generates composite services based on the user's requirements and the general and domain specific rules.
- **ExecutionEngine**: the class that executes services, either atomic or composite services, and returns the result back to the user.

6.4 Implementation Technologies

The technologies used for implementing the prototype are chosen for their ability to create a friendly user interface, provide good accessibility and availability, carry out CG matching, and perform dynamic Web service invocation at runtime. The key technologies are listed as follows and their use is explained.

- **Java Swing provides the drag and drop and the flow chart style user interface for service composition**: all user interfaces in the ServiceComp are created using the Java Swing library. The version distributed with JDK 1.6 provides more reliable and better looking user interfaces. In ServiceComp, one of the key issues is how to help a user to perform service discovery and composition without professional knowledge of Web services. We use drag and drop and a flow chart style to help users to create composite services just by clicking the mouse without knowing any technical detail. The technical details are only provided on demand.
- **Java Applet provides the prototype with comprehensive functionality, high accessibility, and high availability**: in order to create a prototype with comprehensive functionalities with both high accessibility and availability, but lower implementation complexity, a Java Applet application is a good solution. It has most of the features that a desktop application has, but additionally is

accessible from anywhere with a Java enabled web browser. However, some features required by ServiceComp are limited by the Java security model. To overcome these constraints, we use a signed Java Applet [Sun, 2009] to make a Java Applet almost identical to a desktop application. To create a signed Java Applet, the normal applet application needs to be bound with a security certificate that is created by a trusted authority.

- **Notio provides the simple CG matching function:** Notio [Southey and Linders, 1999] is a Java API for creating and manipulating CGs. It addresses the widely varying needs of the CG community. In the prototype, the simple CG matching function is used.
- **WSDL4J and SAAJ provide the dynamic Web service invocation ability:** to statically invoke a Web service, usually client side programming language specific tools are used, such as the WSDL2JAVA [Axis, 2005] tool, which generates local proxy classes so that the client application can invoke Web services as if it is invoking methods on a local class. However, in ServiceComp users can select any services they want to execute at runtime or dynamically create composite services. It is impossible to generate proxy classes at runtime in this case. Therefore, dynamic Web service invocation techniques, such as WSDL4J and the SAAJ APIs, are used to solve this issue. WSDL4J [JWSDL, 2006] is an API that parses a WSDL document extracts information out of it. SAAJ [SAAJ, 2008] is an API that generates SOAP messages at runtime based on the information provided by a WSDL document to invoke target services. By combining these two technologies, Web services can be invoked dynamically at runtime.

6.5 Summary

In this chapter, we have presented a prototype – ServiceComp, which implements some of the features of CbSSDF and provides facilities for service discovery and composition. The purpose of ServiceComp is to firstly demonstrate that the solution proposed by CbSSDF is feasible and also show how CbSSDF can facilitate service discovery and composition, and secondly to, use it as a test bed to evaluate the CbSSDF approach.

ServiceComp implements the two-step service discovery mechanism with which service users can search services using either natural language queries (note that in ServiceComp the form of the natural language query is limited to arithmetic expressions) and/or technical service specifications. By using ServiceComp, service users can search services, create composite services, and execute services without requiring the domain specific knowledge of services.

In the next chapter, we will use the prototype to evaluate the performance of the CbSSDF based solution.

Chapter 7: Evaluation

In this chapter, we evaluate our solution by comparing it with an existing service description framework – OWL-S. We also evaluate our solution in terms of performance, scalability, and applicability based on the prototype ServiceComp.

7.1 Overview

In this chapter, we evaluate the proposed methods and solution. The evaluation mainly focuses on the criteria for success that are proposed in Section 1.6. For ease of reading, we list them here again and explain how we are going to evaluate the research work in this thesis against each of them.

- ***Technological novelty:*** *Our work must be novel in comparison with existing work.*

The novelty of our work has been discussed in many chapters, such as chapters 3, 4, and 5. Therefore it will not be addressed again here however, we will summarise our contributions in Chapter 8.

- ***A Context-based Semantic Service Description Framework:*** *one of the outcomes of our work is a framework that can better describe Web services in order to improve the efficiency of service discovery and composition.*

CbSSDF is proposed in Chapter 4. In this chapter, we will compare it with OWL-S to examine what the advantages and disadvantages of CbSSDF are.

- ***A suitable prototype:*** *A proof-of-concept prototype needs to be implemented in order to show that our service description framework can be actually realised and the result from the prototype should show that it is beneficial.*

The prototype ServiceComp proposed in Chapter 6 shows that the CbSSDF based solution is feasible and practicable. In this chapter, we use it as a test-bed to evaluate the proposed solution.

- ***Acceptable system performance:*** *The performance of the system has to be at an acceptable level.*

The performance of the CbSSDF based solution is evaluated in this chapter against three aspects: the accuracy of the search results, the response speed, and scalability both in terms of the number of services and the distribution of services.

The aim of the evaluation is to examine whether or not the CbSSDF based solution can improve the efficiency and effectiveness of service discovery and composition and investigate whether this solution is realistic and practicable.

The content of this chapter is organised as follows: we first discuss how our work will be evaluated through a rational process; then we compare our solution with OWL-S using a scenario with three tasks. After the comparison, we evaluate the performance of our solution and the applicability of the transformation method by carrying out a series of experiments on ServiceComp. Finally, we summarise the evaluation results by reflecting on the criteria for success.

7.2 Evaluation Strategy

Semantic Web Services description, discovery, and composition are brand new research topics. To our knowledge, they are still at a premature stage and there is still a long way to go before a full-fledged methodology and/or suitable applications come into use. There is no commercially released software or tools that comprehensively tackle these areas. There are only a few research prototypes and APIs, such as [OWL-S API, 2008] and [OWL-S Editor, 2008]. In other words, there is no existing system that our solution can be compared with. Hence, we need to design a rational evaluation process to evaluate our work.

The evaluation process is carried out in three stages. First, we make a scenario based comparison and analysis on how the CbSSDF based solution tackles problems, such as query interpretation, service discovery, and service planning and composition. We also consider whether the situation has been improved and by how much in comparison with existing service description frameworks. We set up the scenario with three tasks for the comparison and analysis to be based on. The three tasks include atomic service discovery, composite service discovery, and dynamic composite service generation. We choose OWL-S as representative of the existing semantic Web service description frameworks for comparison purposes. In the comparison and analysis, we assume that both frameworks' features are fully implemented. Secondly, by carrying out a series of experiments on the proposed prototype ServiceComp, we analyse the performance of the CbSSDF based solution in against the OWL-S based solution, for metrics including response time for queries and system scalability. Finally, we evaluate the applicability of the transformation method proposed in Chapter 4, which transforms the existing Web service descriptions into CbSSDF based service descriptions.

The reasons for choosing OWL-S as the comparison framework are discussed as follows:

1. The first and the most important reason is that OWL-S is a semantic Web service description framework and is comparable to CbSSDF. OWL-S aims to enable automatic service discovery, invocation, and composition through the integration of service semantics [Martin et al., 2004a], as does CbSSDF. Except for SUC, which is one component of CbSSDF, all of the aspects described in CbSSDF are also part of OWL-S, however, the specific details may be different. As the SUC can be considered as an extension of the service capability semantics, CbSSDF and OWL-S are two different solutions but tackle the same problem area and therefore, we can consider these two solutions as comparable;
2. The second reason is that OWL-S is a relatively mature research proposal in the Semantic Web Services subject area and is a more widely adopted semantic service description framework in comparison with other existing frameworks. It has been submitted to the World Wide Web Consortium (W3C) for consideration as a web standard;
3. The third reason is that OWL-S is an extension to OWL, which is already a standard for the semantic web. Therefore, the features provided by OWL are directly inherited by OWL-S for describing the semantics of services;
4. The final reason is that OWL-S provides rich service capability semantics, such as the service ontology, the service IOPE, the service profile, atomic and composite processes, and service grounding. It is the first framework that considers IOPE as key factors in addressing the semantics of a service's capabilities.

7.3 Scenario Based Comparison with OWL-S

The scenario for comparing the CbSSDF based solution and the OWL-S based solution is to perform a compound arithmetic calculation using mathematical Web services. Normally, an arithmetic calculation is described by a combination of symbols, mathematical operators, and rules. We use one Web service to represent each mathematical operator, where the service's inputs are the operator's operands and the output of the service is the result of the calculation. A compound arithmetic

expression can be represented as a composite service, where each operator in the expression is considered as a participant Web service in the composite service. The calculation result is produced by executing the participant services in the composite service in a certain order according to standard mathematical rules. Although this is not a complex scenario, it tackles all aspects of service discovery, composition, and invocation. It involves service query (either in natural language or formal mathematical expression) processing, service discovery based on semantic and technical information, data type compatibility checking during service composition, service planning, and rule regulated service invocation. Therefore, it requires a service description framework which provides sufficient information in order to achieve the above tasks with the minimum of human intervention. By going through the tasks from the scenario, we will assess which solution makes the tasks easier to achieve autonomously.

Suppose we have a student who wants to find a Web service to calculate the volume of a cone. He knows how this can be done, but he wants a Web service do it for him. He proposes a query as follows:

“Cone volume calculation service: multiply a cone’s base circle area by its height and divide by 3”

This query states which kind of service he is looking for and how the service should work. Now let us analyse what the possible situations of the returned result are:

- One or many existing atomic services from the service repository are located for the student’s requirement.
- One or many existing composite services from the service repository are located for the student’s requirement.
- There are no existing services that can satisfy the requirement, but a composite service is constructed dynamically for the requirement.
- No (satisfiable) result returned, i.e. neither an existing service nor a dynamically constructed composite services can satisfy the requirement. (This outcome will not be considered in this section).

Suppose we have the query interfaces for both the CbSSDF based solution and the OWL-S based solution and a repository service containing the aforementioned mathematical Web services. Some of them are atomic services, such as the addition service, the multiplication service, and the square root service. Some of them are composite services, such as the circle area service and cylinder the volume service. Both the atomic services and the composite services can be composed to construct more complicated composite services. Two examples of CbSSDF based service descriptions are shown in Table 7.1.

Table 7.1. Two examples of the CbSSDF based service description.

Atomic Service		Composite Service	
Name:	Addition Service	Name:	Circle Area Service
Type:	Arithmetic	Type:	Area
Input Data Type:	In1: double In2: double	Input Data Type:	In1: double In2: double
Input Semantics:	In1: Addend In2: Summand	Input Semantics:	In1: Radius In2: Pi
Output Data Type:	Out1: double	Output Data Type:	Output1: double
Output Semantics:	Out1: Summation	Output Semantics:	Out1: Circle Area
Pre-condition:	$\text{isDouble}(\text{in1}) \wedge \text{isDouble}(\text{in2})$	Pre-condition:	$\text{isDouble}(\text{in1}) \wedge \text{isDouble}(\text{in2})$
Effect:	$\text{isDouble}(\text{out1})$	Effect:	$\text{isDouble}(\text{out1})$
CUP Input Services:	In1: Subtraction, Multiplication, ... In2: Subtraction, Multiplication, ...	CUP Input Services:	In1: Subtraction, Multiplication, ... In2: Subtraction, Multiplication, ...
CUP Output Services:	Subtraction Service, Multiplication Service, ...	CUP Output Services:	Subtraction Service, Multiplication Service,
Internal Structure:	null	Internal Structure:	Multiplication, Square, and PI
Metadata:	QoS, Natural language description, Service provider information, ...	Metadata:	QoS, Natural language description, Service provider information, ...
Resource:	Not provided	Resource:	Not provided
S-CGs:	[Arithmetic: Addition Service] ← (REQ) ← [Area: Trapezium Area Service] → (REQ) → [Arithmetic: Multiplication Service]	S-CGs:	[Area: Circle Area Service] ← (REQ) ← [Volume: Cylinder Volume Service]

Most of the information in Table 7.1 is obvious. “CUP Input Services” means that these services can provide input data for the listed services. “CUP Output Services” means that these services can consume the output data from the listed services. The

mathematical services do not have strong data semantic restrictions, thus in the above case any services in the service repository that can provide or consume the *double* data type are in the “CUP Input Services” list and/or the “CUP Output Services” list. An atomic service does not have an internal structure and therefore, the Addition Service’s “Internal Structure” information is *null*. OWL-S based service description examples can be found on the MindSwap website⁷.

In the following sections, each of the tasks will be discussed and the results of the two solutions are compared.

7.3.1 Task 1: Locating an Existing Atomic Service

In the first task, we assume that there is at least one atomic service in the service repository that can satisfy the requirement, i.e. performing the calculation of the volume of a cone. This task requires a service description framework that has the capability to support query interpretation and specification matchmaking.

7.3.1.1 Solution Comparison

The comparison result for these two different solutions for solving task 1 is shown in Table 7.2.

Table 7.2. The comparison of CbSSDF and OWL-S based solutions for task 1.

CbSSDF based Solution	OWL-S based Solution
<p>1. Query Interpretation:</p> <p>A given query Q is converted into a CG:</p> $Q \Rightarrow CG$	<p>1. Query Interpretation:</p> <p>A given query Q is converted into a set of concepts:</p> $Q \Rightarrow C = \{c_1, c_2, \dots, c_n\}$
<p>2. Match Making:</p> <p>Step one:</p> <p>By CG matching, a set of relevant services $S_r = \{s_1, s_2, \dots, s_n\}$ is obtained. Then the services in S_r are ranked according to their S-CGs' similarity to the query CG.</p>	<p>2. Match Making:</p> <p>Not Applicable.</p>

⁷MindSwap OWL-S example: <http://www.mindswap.org/2004/owl-s/services.shtml>

<p>Step two: Based on the further technical specification provided by the service user, S_r is refined, ranked according to similarity, and returned to the service user.</p> <p>The specification matching is performed based on the attributes addressed in the SSDM, such as the IOPE, the service concept, the service metadata, the service internal structure, and CUPs.</p>	<p>Matchmaking cannot be performed based on natural language query. Therefore, the technical specification is required at the same time when the query is proposed.</p> <p>The matchmaking is performed based on the IOPE and the service metadata. A set of result services is returned to the service user and they are ranked according to the similarity to the specification.</p>
--	--

7.3.1.2 Summary

From the result shown in table 7.2, we can see that for locating a single atomic service, there is no significant difference between these two solutions. They both require the service user to propose a query followed by a detailed technical specification of the required service. However, in the CbSSDF based solution, the matchmaking can be performed based on imprecise information, such as a natural language query. The service user can provide technical specification later based on the initial result. The OWL-S based solution requires the service user to give the technical specification of the required service at the very beginning of the search. It could be a difficult task for the service user to give the detailed technical specification at that time, especially when the service user is not a domain expert in the required service area.

7.3.2 Task 2: Locating an Existing Composite Service

In the second task, we assume that in the service repository there is at least one composite service that can perform the calculation of the volume of a cone. This task requires a service description framework that has the capabilities to support query interpretation, specification matchmaking, and internal structure and sub-services matching with composite services if applicable.

7.3.2.1 Solution Comparison

The comparison result for these two different solutions for solving task 2 is listed in Table 7.3.

Table 7.3. The comparison of CbSSDF and OWL-S based solutions for task 2.

CbSSDF based Solution	OWL-S based Solution
<p>1. Query Interpretation:</p> <p>A given query Q is converted into a CG:</p> $Q \Rightarrow CG$	<p>1. Query Interpretation:</p> <p>A given query Q is converted into a set of concepts:</p> $Q \Rightarrow C = \{c_1, c_2, \dots, c_n\}$
<p>2. Matchmaking:</p> <p>Step one – CG Matching: By CG matching, a set of relevant services $S_r = \{s_1, s_2, \dots, s_n\}$ is obtained. Then the services in S_r are ranked according to their S-CGs' similarity to the query CG.</p> <p>Step two – Specification Matching: Based on the further technical specification provided by the service user, S_r is refined, ranked, and returned to the service user.</p> <p>However, in this step, if the service user is familiar with the required service and able to provide detail about the internal sub-services' detail, the result can be more accurate.</p> <p>For example, if service s is a composite service and consists of s_i and s_j, then s_i and s_j's detail can also be used to locate s.</p>	<p>2. Matchmaking:</p> <p>Not Applicable.</p> <p>The internal detail of services are hidden from service users in the OWL-S based solution, thus for a service user, the atomic service and the composite service are not distinguished. For this reason, the matchmaking process in this task is exactly the same as the one described in task 1.</p>

7.3.2.2 Summary

The comparison result in Table 7.3 shows the difference between these two solutions when dealing with the composite service discovery. In the CbSSDF based solution, the rich service description enables the service user to use extra information, such as the internal structure of composite services, to more precisely locate required services. In reality, it is not necessary for a user to know whether the required service is an atomic service or a composite service. However, if the user does know the extra information, it can be used to obtain a better search result. In the OWL-S based solution, the composite service and the atomic service are not distinguishable from the service user's perspective. The advantage of this is that it can simplify service discovery and service description. CbSSDF tries to use all of the information available to assist the service discovery. The disadvantage of this is that it increases the

complexity of the service description and discovery processes. However, the complex service description is compensated for by the more accurate service discovery result.

7.3.3 Task 3: Dynamically Constructing Composite Service

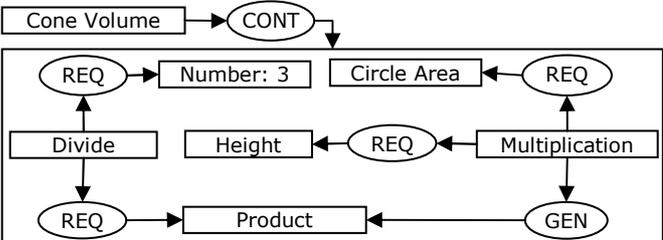
In the third task, we assume that there is no existing service that can perform the calculation of the volume of a cone. Therefore, a composite service needs to be dynamically constructed from the existing services in the service repository.

This task requires a service description framework that has the capabilities to support not only query interpretation, and specification matchmaking, but also service planning.

7.3.3.1 Solution Comparison

The comparative results for the two different solutions are listed in Table 7.4.

Table 7.4. The comparison of CbSSDF and OWL-S based solutions for task 3.

CbSSDF based Solution	OWL-S based Solution
<p>1. Query Interpretation:</p> <p>A given query Q is converted into a CG:</p> $Q \Rightarrow CG$ 	<p>1. Query Interpretation:</p> <p>A given query Q is converted into a set of concepts:</p> $Q \Rightarrow C = \{c_1, c_2, \dots, c_n\}$
<p>2. Matchmaking:</p> <p>Step one – CG Matching: By CG matching, a set of relevant services $S_r = \{s_1, s_2, \dots, s_n\}$ is obtained. The services are ranked according to their S-CGs' similarity to the query CG.</p> <p>The CG matcher will join single S-CGs together into larger S-CGs in order to achieve the best possible match.</p> <p>Step two – Specification Matching: Based on the further technical specification provided by the service user, S_r is refined and ranked.</p> <p>If there is no matched service or only matches with very low similarity rate, the system will start the service planning process to generate composite</p>	<p>2. Matchmaking:</p> <p>Not Applicable.</p> <p>Based on the technical specification provided by the service user, the OWL-S based solution will try to find a set of best matched services.</p> <p>If there is no matched service or matched with very low similarity</p>

services.	rate, service composition will be attempted.
3. Planning and composition:	3. Planning and composition:
<ul style="list-style-type: none"> – The planning is based on reduced service range S_r, i.e. the relevant services from the step one are considered first in the planning process. – When a service is located, its CUPs can tell the planner where to go next, only the services in its CUPs are compatible, so there is no need to go through all the services in the repository. 	<ul style="list-style-type: none"> – The planning is based on the whole service repository. – The planner in the OWL-S based solution needs to go through the whole service repository every time it tries to locate a service for a task.
4. Rule evaluation	4. Rule evaluation
<ul style="list-style-type: none"> –Pre-conditions and effects of each service are evaluated during the service planning process. – The general and domain specific rules are evaluated to filter out invalid composite services. 	<ul style="list-style-type: none"> –Pre-conditions and effects of each service are evaluated during the service planning process.

7.3.3.2 Summary

In dynamic composite service construction, the CbSSDF based solution clearly demonstrates its advantages. In comparison with the OWL-S based solution, the information provided by CbSSDF can greatly improve the performance of service composition and the accuracy of the result. The inter-relationships between services addressed by the CUPs can decrease the number of candidate services in each step of planning. As each CUP can be considered as a segment of a plan, the actual number of planning steps is reduced. The general and domain specific rules can be used to both describe the pre-conditions and effects of services, and also to verify the correctness of the generated composite service(s).

In the OWL-S based solution, each service is treated completely separately. The inter-relationships between services in the real world will not be considered. As there is no information to indicate the relationships between services, we have to search the whole set of services in the service repository for one candidate service that possibly matches each sub-task in a composite service. Obviously, this search and match process greatly increases the performance overhead of an OWL-S based system.

7.3.4 Discussion of the Scenario based Comparison

In the previous sections, we used an arithmetic calculation scenario to compare the CbSSDF based solution with an OWL-S based solution. Three tasks are proposed to examine how each solution deals with the following situations:

- **Locating an atomic service:** in this situation, the service description framework needs to assist the service search engine in locating existing atomic services that can fulfil the service user's requirement.
- **Locating a composite service:** in this situation, the service description framework needs to assist the service search engine in locating existing composite services that can fulfil the service user's requirement.
- **Dynamically constructing a composite service:** in this situation, there is no existing service that can fulfil the service user's requirement. Therefore, the service description framework needs to assist the service search engine in dynamically constructing one or more composite services which can fulfil the service user's requirement.

By analysing the two different approaches through the three tasks, we have several pros and cons of these two solutions, which are summarised as follows:

- 1) With regards to atomic service discovery, the two solutions have no discernible differences. However, in general the CbSSDF solution and its two-step service discovery mechanism gives service users more natural ways to search for services in more natural ways (using natural language) and only give precisely specified technical information later if they can. In the OWL-S solution, service users must provide precisely specified technical information at the very beginning of each search, which can be hard for users who are not familiar with the technical detail of their required services.
- 2) The composite service and the atomic service are not distinguished from the service user's perspective in the OWL-S solution. Therefore, for a service user, there is no difference between searching for an atomic service or a composite service. The advantage of this is that it makes the searching process simpler and the user does not need to be aware of the differences.

One of the principles of the CbSSDF is to use all possible information to assist in service discovery. Therefore, if service users know the internal detail of the services they are looking for, they can provide relevant information which may make the discovery result more accurate. However, the disadvantage of this is that it increases the complexity of the service description and discovery process.

- 3) When the situation is more complicated, i.e. when dynamic composite service construction is required, the advantages of the CbSSDF based solution become obvious. First of all, the two-step service discovery mechanism can filter out irrelevant services by CG matching so that the number of the candidate services for service composition is reduced. Secondly, CUPs can further reduce the number of candidate services in each step of service planning. They also reduce the number of steps that a planner needs to take to reach the goal. Thirdly, the non-monotonic rules in CbSSDF can help to identify invalid composite services, which make the resulting service more accurate and more reliable.

A significant defect of the OWL-S solution is that it does not consider the inter-relationships between services. A consequence of this is that for all stages of the planning process the planner has to search through the whole service repository for candidate services.

7.4 Prototype based Performance Study

In the previous sections, we have analysed the differences between the CbSSDF solution and the OWL-S solution in solving service discovery and composition problems. In this section, we evaluate the CbSSDF solution from a performance point of view in comparison with the OWL-S solution. We carry out a series of experiments on the ServiceComp prototype. The purpose of the experiments is to evaluate the CbSSDF solution from the following three aspects:

- The accuracy of the service discovery result.
- The performance efficiency.
- The system scalability.

7.4.1 Experiment Environment

We set up a lab based test-bed. The test bed consists of five server machines and a client machine. All the machines used in the test-bed have the same configuration: HP Compaq DC7600S, Intel Pentium4 HT 3.00GHz processor, 3G RAM. One server machine is the main server and the rest four servers are assistant servers that are used for simulating the distributed service repository environment. The service repository is deployed on the main server and duplicated on each assistant server. The ServiceComp prototype is deployed on the client machine. In order to compare with the OWL-S solution, we use the WSDL2OWL-S [Srinivasan et al., 2006] tool to generate OWL-S based service descriptions and implement a very simple service searching interface using the OWL-S/UDDI Matchmaker and Client API [Srinivasan et al., 2006].

7.4.2 Design of the Experiment

7.4.2.1 Analytical Model

To evaluate the accuracy of the service discovery result, the precision and recall model is used. Precision is used to measure how relevant a retrieved service is to a user's need, i.e. exactness or fidelity. Recall is used to measure how many services relevant to the query are successfully retrieved, i.e. completeness.

Let A be a set of relevant services to a user's need in the service repository, B a set of retrieved services from the service repository. According to [Van Rijsbergen, 1979] the precision and recall is calculated as below:

$$Precision = \frac{|A \cap B|}{|B|}, \quad Recall = \frac{|A \cap B|}{|A|}$$

The accuracy of a search result is measured by the combination of the values of precision and recall. Under the same recall, the higher precision value indicates a more accurate search result [Van Rijsbergen, 1979].

The precision and recall model is widely used as a measurement model for the accuracy of result in areas such as information retrieval and statistical classification. As Web services are considered a dynamic format of information, the model is

applicable to Web service discovery. The values of precision and recall are explained as follows: a maximum precision score of 1.0 means that all of the results retrieved by a search are relevant (but it does not mean that all the relevant documents have been retrieved), whereas a maximum recall score of 1.0 means that all of the relevant documents are retrieved by a search (but it does not mean that no irrelevant documents have been retrieved). A common use of precision and recall is to form a precision and recall space (PR Space), where we can draw PR curves to compare the accuracy of different searching methods.

The performance efficiency and system scalability are examined through average query response time. The average query response time for evaluating the performance efficiency is calculated based on a centralised service repository. The average query response time in the centralised service repository situation (suppose involving service composition) will be the sum of the average time for performing the CG matching (T_{CG}), the average time for performing the specification matchmaking and semantic similarity ranking based on SSDF (T_{m_rank}), the average time for constructing composite services ($T_{Composite}$), and the average time for the non-monotonic rule reasoning (T_{reason}). Let $T_{Central}$ be the average query response time in the centralised service repository situation, it can be expressed as following:

$$T_{Central} = \text{AVG}(T_{CG}) + \text{AVG}(T_{m_rank}) + \text{AVG}(T_{Composite}) + \text{AVG}(T_{reason})$$

where, $\text{AVG}()$ is the average function.

The evaluation for system scalability is twofold. First, we increase the number of services in the centralised service repository to see how fast the query response time is. The calculation expression is the same as $T_{Central}$. Second, we distribute the services to five servers to simulate a decentralised service repository environment, such as the Internet. Then, we examine how fast the query response time is. The average query response time in the decentralised service repository situation (suppose involving service composition) will be the sum of $T_{Central}$ on each server, the network latency for connections to each server ($T_{Latency}$), and the time for combining the result from different service repositories ($T_{Combine}$). Let $T_{Decentral}$ be the average query response time in the decentralised service repository situation, it can be expressed as follows:

$$T_{Decentral} = \sum_{i=1}^n T_{Central}^i + \sum_{i=1}^m T_{Latency}^i + T_{Combine}$$

where, n is the number of service repositories and m is the number of connections to the repositories.

We will discuss the experimental results in the next section using the analytical model.

7.4.2.2 The Independent and Dependent Variables and the Treatment

In evaluation of the search result accuracy:

- **Independent variables:** are the number of the services that have been retrieved and the number of the relevant services in the service repository. We change the values of these two variables to observe the different outcomes from the experiment.
- **Dependent variables:** are the recall and precision. These are the values that we are observing during the experiment in order to examine the accuracy of the search result.
- **Treatments:** are the CbSSDF based solution and the OWL-S based solution. These are the two solutions that we compare in the experiment.

In evaluation of the performance and scalability:

- **Independent variable:** is the number of the services in the service repository (100-1000 services for the performance evaluation and 1100-2000 services for the scalability evaluation). We change the number of services to observe the different outcomes from the performance and scalability experiments.
- **Dependent variable:** is the query response time. We observe the response time of each service query to examine the system performance and scalability.
- **Treatments:** are the CbSSDF based solution and the OWL-S based solution. These are the two solutions that we compare in the experiment.

7.4.3 Experiment Results

Experiments are performed in a controlled laboratory environment, so we can clearly know how many relevant services for each service query are in the service repository and how many of them have been retrieved. Therefore, the precision-recall curve

diagrams of the results based on different approaches can be precisely drawn, see Figure 7.1. The data shown on the diagram are also shown in Table 7.5.

Table 7.5. Precision-recall table for CbSSDF and OWL-S solutions.

		The values of the recall and precision								
CbSSDF	<i>Recall</i>	0.08	0.19	0.32	0.44	0.51	0.6	0.69	0.8	0.92
	<i>Precision</i>	0.9	0.7	0.55	0.48	0.38	0.35	0.32	0.2	0.16
OWL-S	<i>Recall</i>	0.08	0.15	0.24	0.42	0.49	0.59	0.71	0.81	0.91
	<i>Precision</i>	0.78	0.55	0.37	0.28	0.2	0.17	0.14	0.08	0.02

From the diagram shown in Figure 7.4, we can see that the CbSSDF based solution can significantly improve the accuracy of the service discovery result in comparison with the OWL-S based solution. During the experiment we observed that a large number of the attribute fields provided in the OWL-S based search interface for the technical specification of the required service are left empty by the users. As discussed previously, if service users are not familiar with the required services, it is difficult for them to provide technical information without any hints. The consequence of leaving some of the required search criteria empty is that the search engine lacks sufficient information to accurately locate services, which is one of the reasons for the lower accuracy of the OWL-S approach. By using the two-step service discovery mechanism, users can be guided step by step from the conceptual description of their needs to the technical specification and therefore more detailed information can ultimately be provided. The other reason for the better accuracy of the CbSSDF solution is that CbSSDF provides a richer service description, especially considering that the search engine will have information on inter-related services.

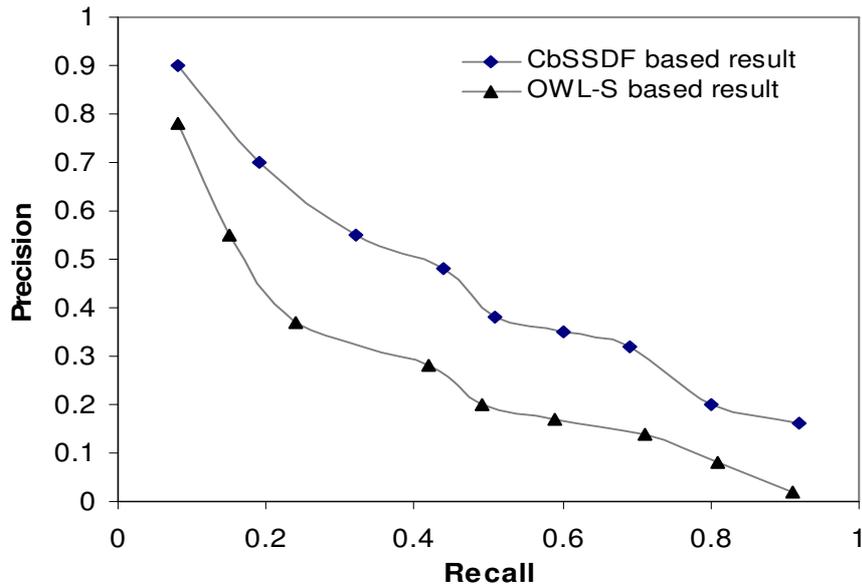


Figure 7.1. Precision-Recall curves.

The performance of the system is evaluated by comparing the query response time with the OWL-S solution using a centralised service repository. The CbSSDF based solution's query response time is examined in three stages: first, recording the response time for the first step search; then, recording the response time for the second step search; and finally, calculating the overall response time of the CbSSDF solution. The results from all the three stages are compared with the OWL-S solution. Figure 7.2 shows the comparative result and some of the sample data used to draw the diagram is shown in Table 7.6, where n is the number of services and t is the response time in milliseconds.

Table 7.6. System performance evaluation result – data samples.

	n	100	200	300	400	500	600	700	800	900	1000
Step one	t (ms)	180	196	202	220	217	234	240	246	258	264
Step two	t (ms)	135	141	177	144	145	166	171	139	163	166
Overall	t (ms)	315	337	379	364	362	400	411	385	421	430
OWL-S	t (ms)	181	193	199	215	220	228	231	235	244	256

From Figure 7.2, we can observe that the first step in particular and the overall search time in general of the CbSSDF solution takes more time than the OWL-S based approach. This is due to the extra complexity of the CG matching algorithm and the algorithm required for converting natural language to a CG. The second step, however, is much faster than the OWL-S based approach because after the first step, only relevant services are passed to the second step, thus the search space is relatively

small. However, sometimes composite services are generated in the second step and therefore the time curve for the second step is fluctuant.

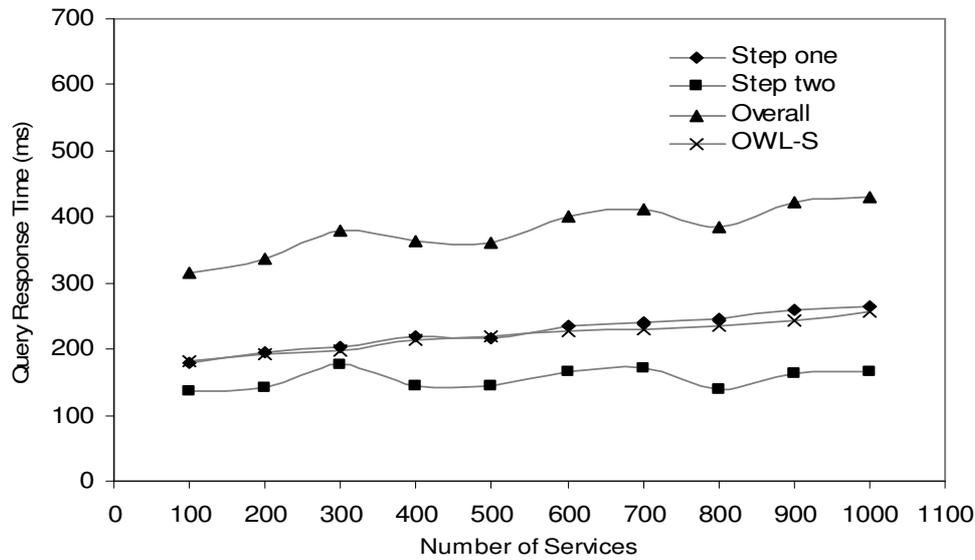


Figure 7.2. Query response time for performing on a centralised service repository.

The scalability of the system is evaluated in two ways: first, we increase the number of services in the service repository on the main server and get the overall system time curve for searching with both CbSSDF and OWL-S; then, we distribute the services to the assistant servers to simulate a decentralised service repository and get the alternative time curves for both CbSSDF and OWL-S (see Figure 7.3). Some of the data used to draw the diagrams is listed in Table 7.7.

Table 7.7. System scalability evaluation result – data samples.

		<i>n</i>	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
Centralised	CbSSDF	<i>t (ms)</i>	353	376	421	654	843	976	1123	1525	2343	3108
	OWL-S	<i>t (ms)</i>	154	200	380	610	890	1420	2160	4341	6540	8550
Distributed	CbSSDF	<i>t (ms)</i>	424	531	576	693	938	1023	1347	1696	2941	3801
	OWL-S	<i>t (ms)</i>	261	330	450	725	1032	1610	2930	4620	7250	9530

The diagram shows that the response time of the CbSSDF solution increases rapidly when the number of services is grows large (>1700). The reason is that when the service repository gets larger, the S-CGs in the repository are also get larger and more complex. Therefore the CG matching algorithm's complexity increases due to the larger CGs. The decentralised service repository also affects performance. The overhead comes from the network latency and time required to assemble results from each of the distributed servers. However, for the OWL-S based solution, the performance declination is due to the inefficient service discovery and composition

process. From the diagram we can observe that it brings more overhead onto to the overall query response time than the CG matching algorithm. When the number of services is larger, the response time increases almost exponentially.

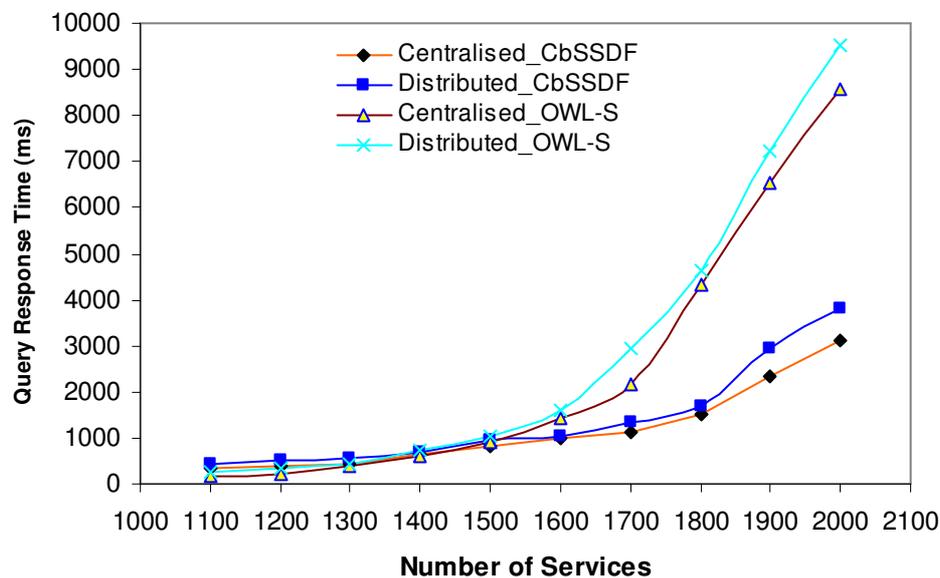


Figure 7.3. Query response time for performing on both the centralised and decentralised service repositories after increasing the number of services.

The results from the evaluation are very promising. However, the evaluation techniques and methods have some limitations, which may reduce the accuracy of the results. In the next section, we will highlight the limitations we have considered in our evaluation process.

7.4.4 Limitations of the Experiments and Threats to Validity

The limitations of the experiments mainly come from three aspects. These aspects are summarised as following:

- **Hardware limitation:** The computers used in the experiments are normal desktop computers. No specific high standard server machines have been used.

For a desktop computer hardware configuration, we need to choose algorithms that do not require such a high amount of computational power that it may exceed the capacity of the machines. We also need to carefully consider the number of services to be hosted on each server in the experiments because a large number of services, e.g. 100,000 services, may exceed the server's capability.

- **Software limitation:** ServiceComp is only a research prototype and has not implemented all the features that are supported by CbSSDF.

The simple interface and search engine for OWL-S are implemented using very basic libraries and these libraries may not be up to date. This could cause lower performance than a properly implemented OWL-S based search engine.

- **Evaluation method limitation:** There are about 200 Web services originally added to the service repository. However, in the system scalability experiment, there are more than 2000 services involved. As a large amount of development work would have been required, we did not actually create all these services, most of them are generated from the original 200 Web services through duplication and minor modification.

As the experiments have taken place in controlled conditions, there are threats to the validity of the evaluation results that we need to consider and control. The threats to the internal validity are summarised as follows:

- **Selection of scenario:** The simplified mathematical calculation scenario for the comparison and analysis between CbSSDF and OWL-S may hide issues that can only be explored in the real world complex scenarios.
- **Selection of services:** The services used in the evaluation are simple mathematical calculation services and hosted in the same environment without interference from other applications or systems, this could increase the performance of the system.
- **Instrumentation:** The evaluation result may contain deviations caused by the implementation of the prototype and the measurement methods.
- **Experimenter Bias:** The expectations of the outcomes may influence the experimenter to view result data in a subjective way.

The threats to the external validity are summarised as follows:

- **Generalisation:** the conclusion drawn from the comparison between CbSSDF and OWL-S may not be applicable for newly proposed research works on Semantic Web Services description.

7.5 Transformation Method Applicability Evaluation

In Chapter 4, we introduced a transformation method that can transform existing Web service descriptions into CbSSDF based descriptions. In this section, we evaluate the applicability of this transformation method on different types of service description. We divide the services that are used in the experiment into three groups. The services in the first group are described using WSDL, the second group using WSDL with additional natural language description, and the third group using OWL-S. Each group has 500 sample services. The metric for the experiment is the average percentage of the required information in CbSSDF that can be found by extracting information from the different types of service description using the transformation method. The required information in CbSSDF includes the information needed for generating S-CGs and the information needed for completing the SSDM, such as service semantics, service interface data types and semantics, the service metadata, and service relationships. The result is shown in Table 7.8. If the required information is fully obtained from the service description frameworks, a “Yes” will be given to that field; if the required information is partially obtained, a “Partial” will be given; if the required information is not available, then a “No” will be entered. At the end of the table, we calculate the percentage of the required information that can be obtained. In order to calculate the percentage, we assign ‘1’ to “Yes”, ‘0’ to “No”, and ‘0.5’ to “Partial”. The result is then graphically illustrated in Figure 7.4.

Table 7.8. Percentages of the required information in CbSSDF obtained from different service description frameworks.

	Input		Output		Service Metadata	Service Concepts	Service Relation	CUP	(%)
	Data Type	Semantics	Data Type	Semantics					
WSDL	Yes (1)	No (0)	Yes (1)	No (0)	No (0)	No (0)	No (0)	No (0)	25
WSDL +NL	Yes (1)	Partial (0.5)	Yes (1)	Partial (0.5)	Partial (0.5)	Yes (1)	No (0)	No (0)	56.3
OWL-S	Yes (1)	Yes (1)	Yes (1)	Yes (1)	Yes (1)	Yes (1)	Partial (0.5)	No (0)	81.3

The experimental results show that the WSDL based description only provides (on average) a quarter of the required information through the transformation method; the WSDL with additional natural language provides 56.3% of the required information, whereas the OWL-S based description doubles that of the WSDL based descriptions by providing 81.3% of the required information. However, just because a high

percentage of the required information can be obtained from a service description framework does not mean that the framework is identical to CbSSDF. Some of the obtained information in the original framework has been used in a different way. For example, some of the information in WSDL is only used for keyword matching, whereas in the CbSSDF, it may be used to build up the conceptual relationships between services.

The CbSSDF is a semantically rich service description framework and there is no other existing service description framework that provides 100% of the required information. The experimental result indicates that the transformation method is applicable to a majority of the existing service description models, although some of the service description models may need more manual manipulation after applying the transformation method than others.

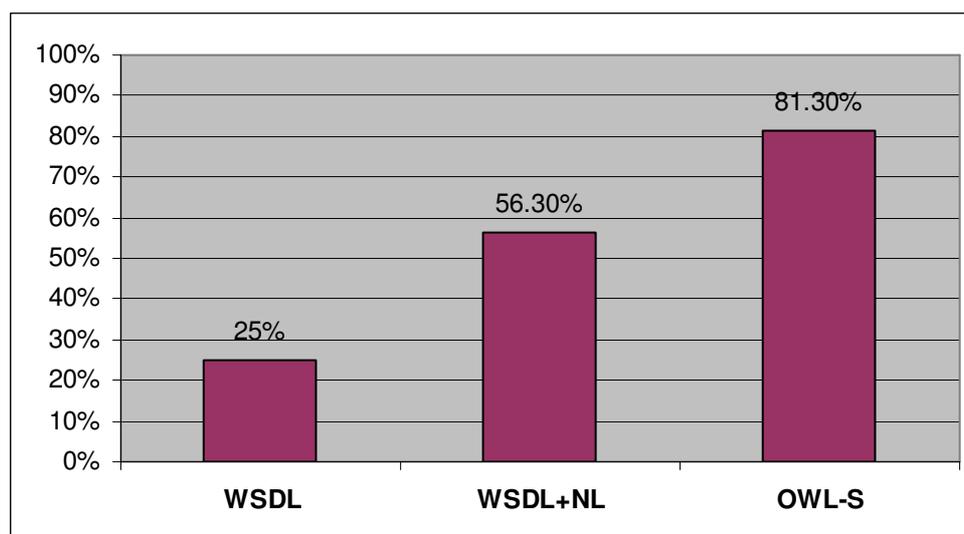


Figure 7.4. Percentage of acquired information from different service descriptions.

7.6 Summary

In this chapter, we have evaluated CbSSDF through a scenario with a set of tasks and a series of experiments. As previously discussed, the evaluation has been conducted using the following three stages.

First, we use an arithmetic calculation scenario and three tasks to analyse the differences between the CbSSDF solution and the OWL-S solution. The three tasks

cover atomic service discovery, composite service discovery, and dynamic composite service construction situations. The analysis result is summarised as follows:

- 1) In atomic service discovery, these two solutions have no major differences. However, the two-step service discovery mechanism provided by the CbSSDF solution gives service users a greater flexibility in service discovery in contrast to the OWL-S based solution.
- 2) Using the OWL-S based solution, composite services and atomic services are not distinguished from the service user's perspective, so there is no difference between atomic service discovery and composite service discovery in the OWL-S based solution. The advantage of this is that it makes the search simpler and users do not need to know the difference between a composite and an atomic service.

In the CbSSDF solution, users can use details of the internal structure to assist service discovery. One of the design principles of CbSSDF is to use all the available information to support service discovery, including the internal structure of the service. However, the disadvantage of this is that it increases the complexity of the service description.

- 3) The advantages of using CbSSDF to dynamically construct composite services include: i) a significant reduction of the size of search space due to the two-step service discovery mechanism; ii) an improvement in the efficiency of the service planning and composition processes; and iii) effective identification of invalid composite services with the help of the non-monotonic rules.

Second, the service search accuracy, the system performance, and the system scalability were studied through a series of experiments and the results compared with the OWL-S based solution. The findings from the experiments include 1) the CbSSDF solution can considerably improve the accuracy of the service discovery results; 2) due to the complexity of the CG matching algorithm and the NL-to-CG converting algorithm, the performance of the CbSSDF solution is lower than the OWL-S solution; 3) the CbSSDF solution can handle a relatively large amount of services (a service space with more than 1500 services) in a decentralised service repository. However, some extra performance overhead is observed due to network latency and the time spent on assembling results from the decentralised service repository.

Third, the applicability of the transformation method has been examined. The result shows that the transformation method works best on existing semantic service description frameworks, such as OWL-S. The efficiency of the transformation method is lower if the service description framework does not contain semantic information, (e.g. WSDL). One of the key concepts for CbSSDF is the use of inter-service relationships to provide the SUC information about the services. However, without the semantics from the existing service description, it is very difficult to capture meaningful inter-service relationships with an automatic procedure. Therefore, more human intervention is required to complete the description after the transformation method has completed.

In conclusion, it has been demonstrated through evaluation that CbSSDF can significantly improve the efficiency and effectiveness of service discovery and composition. The two-step service discovery mechanism gives service users increased flexibility to search services. The comprehensive service description emphasises not only the semantics of services, but also the SUC of services, i.e. the inter-service relationships including conceptual relationships and technical relationships. However, we have identified some problems with CbSSDF that need further investigation. First, the service description in CbSSDF is complex, which results in a larger storage space and increased processing time. Second, highly complex algorithms, such as the CG matching algorithm and the NL-to-CG conversion algorithm, are required to process the service description. This leads to a performance overhead. However, the performance is compensated for by the more accurate search results.

Now let us look back to examine whether we have satisfied the criteria for success proposed at the beginning of the thesis.

- ***Technological novelty***: many features addressed in CbSSDF are unique and innovative, such as the SUC, the S-CG and the CUP.
- ***A Context-based Semantic Service Description Framework***: CbSSDF is proposed to improve the efficiency and effectiveness of service discovery and composition. The evaluation results demonstrate that CbSSDF does improve the situation.

- ***A suitable prototype:*** the prototype ServiceComp is proposed to demonstrate that the CbSSDF based solution is feasible and the performance is acceptable.
- ***Acceptable system performance:*** The performance of the CbSSDF based solution is examined in the evaluation (and contrasted to other solutions such as OWL-S). It is observed through comparison with the existing work that a more accurate search result can compensate for the increased performance.

As per the above, we believe that all of the criteria for success have been satisfied.

In the next chapter, we will summarise our work and indicate possible future research directions for Semantic Web Services.

Chapter 8: Conclusion and Future Work

In this chapter, we summarise our proposals and discuss the remaining problems, and point out some future research directions.

8.1 Overview

In this chapter, we first summarise the ideas, methods and solutions proposed in this thesis, stressing our main contributions. Then, we briefly analyse the remaining problems that need to be solved in future work and also look at what needs to be further developed in terms of research and development. Finally, we will discuss future research directions for Web services and Semantic Web Services description, discovery, and composition.

8.2 Summary and Contributions

One goal of this thesis is to present a comprehensive semantic Web service description framework in order to increase the efficiency and effectiveness of the Web service discovery and composition.

Web services are simple, self-contained applications that perform functions, from simple service requests to complicated business processes. They can be considered as the dynamic side of the web, this is in comparison with the static side of the web in forms such as information on web pages. XML encoded communication, i.e. SOAP messages enables Web services to be programming language, operating system, and hardware independent. As Web services enable computer-to-computer communication in a heterogeneous environment, it is ideally suited to provide dynamic information and functionalities over the web. The main advantages of Web services mean this technology is the natural choice for the implementation of a new application design and development paradigm – Service Oriented Architecture (SOA). With SOA, an enterprise can modularise their core business into services and reuse them in different business processes and applications. However, to fully achieve the features of SOA, the modules in the SOA, i.e. the services, must be well described to support service discovery and composition. Currently, the most promising research efforts in the area of service description are semantic service description frameworks, such as OWL-S, WSMF, and WSDL-S. These integrate machine understandable semantics into service description to enhance the expressiveness of WSDL, which makes automatic service discovery and composition become possible.

We carried out a comprehensive literature review on the existing Semantic Web Services frameworks and found a list of problems that they are not adequately addressed.

- **Insufficient context information:** The current semantic Web service descriptions focus on ontology based data and capability semantics. They do not sufficiently address the service context information, such as how a service should be used, where the service should be used, and what are the common usage scenarios that the service participates in, i.e. the service applicability/usability/composability.
- **Precise requirements required to locate services:** In order to locate a required service, the current service discovery methods require precisely defined technical information, such as service input and output data types and service capabilities. This kind of information may be difficult for a service user to provide, especially when the service user is not a domain expert in the required service's area.
- **Insufficient information about inter-relationships among services:** The current work has inadequately addressed inter-relationships among services. As Web services are functional units, they must interact with the external environment, such as other Web services. For a given service, there is always a certain group of services that the service can interact with in order to achieve certain tasks. If each Web service is considered as an isolated individual and the potential relationships with other services ignored, the efficiency of the service discovery and composition process will be reduced.
- **Insufficient incomplete information handling:** The rules provided in the current service description frameworks are based on monotonic logic. They are not suitable for handling incomplete information and conflict conditions.

To address the above issues, we proposed a Context based Semantic Service Description Framework (CbSSDF) that provides service usage context (SUC) information about services, adequate semantics of services, and a non-monotonic rule system for handling incomplete information. We also proposed a two-step service discovery mechanism based on CbSSDF to demonstrate how the proposed framework can improve service discovery and composition. A prototype – ServiceComp, is

implemented to demonstrate the practicability of CbSSDF and evaluate its performance. In the following, we describe the main contributions of the work proposed in this thesis.

As discussed previously, Web services, as functional units, must interact with the external environment. Each service has its live context, i.e. which kind of problems this service intends to solve and what relationships with other services are involved in solving the problems. It is very rare that a service can function in all scenarios and interact with any possible service. Therefore, a service's context information, especially how the service can be used, is very important for identifying an appropriate service. The problem of representing the service context leads to our first contribution [Song et al., 2009] [Du et al., 2006c].

Contribution 1 – A new concept of service context, i.e. the Service Usage Context (SUC), is proposed. It addresses two levels of service context from the usage perspective. The conceptual level service usage context, i.e. \mathcal{T} -Context, defines the conceptual relationships between a service concept and other service concepts and entities, such as users, service providers, and other business application related concepts. The instance level service usage context, i.e. \mathcal{A} -Context, defines the interactions between an instance service and other instance services at runtime, i.e. the composability between instance services.

By identifying the shortcomings of the existing service description frameworks, we propose a comprehensive service description framework, which leads to our second and most important contribution [Du et al., 2006b] [Du et al., 2007a] [Du et al., 2007b] [Du et al., 2008a].

Contribution 2 – A Context based Semantic Service Description Framework (CbSSDF) is proposed to address not only the semantics of Web services, but also the SUC. The key features that distinguish CbSSDF from other semantic service description frameworks are: 1) it addresses the inter-relationships between services at both the conceptual level and the instance level through the SUC; 2) it uses non-monotonic rules to describe service pre- and post-conditions and service composition conditions, which makes the handling of conflict conditions caused by incomplete information possible. We have also

applied CbSSDF to describe other software components which have the same basic characteristics as services, such as SaaS and learning objects.

The primary goal of CbSSDF is to integrate the SUC into service descriptions so that services and their usage can be easily mapped into business scenarios. The first benefit of integrating service usage context is that at the conceptual level, it brings the service description closer to real business scenarios. In other words, it brings the service description closer to what a service user wants. This leads to our third contribution [Song et al., 2009] [Du et al., 2007a] [Du et al., 2008a].

Contribution 3 – The conceptual graph (CG) formalism has been applied to represent the \mathcal{T} -Context of services, i.e. the Service Conceptual Graphs (S-CGs). S-CGs create a conceptual layer on the top of the technical service description to bridge the gap between the technical service description and the high level service requirements and business scenarios. S-CGs also represent the conceptual relationships between services.

The second benefit of integrating the SUC is to improve the efficiency of the service discovery and composition. This leads to our fourth contribution [Du et al., 2006a] [Du et al., 2008b].

Contribution 4 – Common Usage Patterns (CUPs) are proposed as a part of CbSSDF to represent the \mathcal{A} -Context of services. A CUP defines an instance service's \mathcal{A} -Context by describing the way it directly interacts with a given set of other instance services. The information provided in the CUPs can significantly improve the efficiency of the service discovery and composition process.

Another goal of CbSSDF is to handle incomplete information. This leads to our fifth contribution [Du et al., 2007a].

Contribution 5 – Non-monotonic rules are used to represent the pre-conditions and effects of services and the service composition conditions. By using non-monotonic rules, the conclusions drawn from the rules can be automatically adjusted when new information becomes available or the environment changes.

To take the advantages of CbSSDF and give the service users more flexibility in service discovery, an enhanced service discovery mechanism is proposed. This leads to our sixth contribution [Du et al., 2007a].

Contribution 6 – A two-step service discovery mechanism is proposed to give service users much more flexibility to search for their required services. Under the two-step service discovery mechanism, if a service user is a domain expert in the required services area, he can directly provide a detailed technical specification with which to query services, otherwise, natural language based queries can be used to locate services at the preliminarily stage of the service discovery process.

On top of our research, development work has also been done to demonstrate the practicability of CbSSDF. First, we proposed a transformation method to convert existing service descriptions into CbSSDF based service descriptions [Du et al., 2008b]; then a prototype – ServiceComp was implemented to demonstrate the features of CbSSDF and the two-step service discovery mechanism.

8.3 Remaining Problems

Although the project proposed in this thesis has been researched and developed for three years, some issues still remain that could be solved if more time and development work were allocated. We summarise the main remaining issues as follows:

- **A descriptive language needs to be created to represent the CbSSDF.** At the moment, the CbSSDF descriptions in the prototype are stored in a database for demonstration purposes only. However, if CbSSDF is applied in real SOA applications, a standard descriptive language needs to be created to represent it. The language must have the ability to represent CGs, ontologies, and non-monotonic rules. It should also be able to link to WSDL in order to make the described services invoke-able. An ideal language could be based on XML. According to current research work, there are XML based languages for representing CGs, such as CGXML [CGXML, 2008]; there are XML based languages for representing rules, such as SRML [SRML, 2001] and RuleML

[RuleML, 2008]; and there are XML based languages for representing ontologies, such RDFS and OWL. Therefore, we cannot see any difficulties in creating a compound language to represent CbSSDF.

- **Performance needs to be improved by applying faster algorithms.** In this research project all CG matching and query processing algorithms are from existing resources. The performance of CbSSDF depends heavily on how well these algorithms are designed and implemented. To improve the performance, we need to develop new algorithms that best suit CbSSDF based service discovery and composition. The complexity of CbSSDF also brings an overhead in terms of performance. In the future, the CbSSDF needs to be made more concise in order to be more efficient.
- **Quality of Service (QoS) needs to be considered in the service description.** In our current work, QoS has not been considered. However, in reality, QoS is a very important criterion in the service selection process. A service matched with a user's requirement does not necessarily mean that the service is the right service for the user. It is a right service only if it provides the user with the desired QoS in terms such as the cost, performance, and stability.

8.4 Future Research and Development Directions

Since SOA is described as the new enterprise application development paradigm, Web services technology is gaining more and more attention by both industry and the academia. Following our research, we have identified a number of future research directions in the Semantic Web Services, and service discovery and composition areas.

8.4.1 Service Level Agreement Enhanced Service Registry

A proper service registry is crucial for effective and efficient service discovery and composition. Currently, service registries are implemented using UDDI, which provides information such as the service provider's information, a brief description of services, and the URLs of each registered service's WSDL. The idea behind UDDI is to provide a place that service providers can register their services and service users can search for their required services. However, UDDI has several problems meeting its pre-set goals. In addition to its lack of semantic description support for services as

discussed previously, it has another two major problems. First, it does not have proper management facilities. Anyone can register their services with UDDI. One can even register their services with “localhost” as WSDL’s URL. Second, the services registered in UDDI have no guarantees as to their quality. One of the reasons that users do not use UDDI to search for services is that they do not trust the service providers on UDDI. Most of the Web services applications in real life are based on Service Level Agreement (SLA), which acts as a contract between a service provider and a service user to guarantee the quality of the provided services.

Ideally, a service registry should be open and supported with a sufficient management mechanism so that there are facilities for the service providers and the service users to create their SLA and other relevant agreements.

8.4.2 Business Patterns in SOA

When we talk about the business in an enterprise, we are talking about routines and processes. Although business is about innovation, the outcomes of the innovation are generally newer, or re-designed, more efficient routines and processes. Therefore, we could say that the business is a set of routine based operations and the innovation is a jump from one set of routines to a better set of routines. Business patterns encapsulate the best practice solutions for certain business tasks. They are the best practices within an enterprise or across enterprises. Business patterns help an enterprise to run their business smoothly and quicken the restructuring process to meet new demands.

Business patterns in SOA are enterprise-focused best practice SOA based solutions. These patterns include the solutions for business under the SOA infrastructure and the technologies for these solutions that have accumulated over the years. These patterns help an enterprise to understand and analyse the complex business problems and break them down into smaller functions and then modularise them into services for future reuse. SOA is about agility, i.e. how an enterprise can quickly reconstruct their business to meet the new demands. If an enterprise can encapsulate their SOA solutions into patterns, it can make the business even more agile.

Different levels of SOA business patterns should be constructed. At a high level, the patterns should describe what are the best practices are under the SOA approach. At a low level, the patterns should give a guideline for how to solve particular technical

problems, such as application integration, Enterprise Service Bus setup, and resource sharing. Ideally, the business patterns in SOA should be described in a suitable, machine understandable language so that they can be processed by computer programmes if required.

8.4.3 Web service Monitoring

Under the SOA paradigm, most applications are distributed applications. The services of each business process are most likely remote services and not under the control of the business process's owner. If any service within the process fails or performs badly, the holistic business process will be affected. Therefore, a business process should have the ability to identify failures and give an appropriate response as quickly as possible, or even predict failures before they happen. To achieve this, monitoring each service in the process is necessary.

In an ideal SOA application, a service monitoring system should be able to collect real-time information about each service and store it in a database. The collected information would include average failure rate, average response time, throughput, and cost etc. When required, such information can then be retrieved to provide suggestions about the current status of the monitored services.

The benefits of the monitoring system are: 1) to identify faulty services quickly - using the monitoring information, a faulty service in a business process can be quickly identified and the process owner can modify the process before it delivers the wrong result to their customers; 2) to prevent failures - because the monitoring system provides the information for all services in a process, a faulty service can be identified even before it is executed; 3) to prevent the design of low performance and/or unreliable business processes - the monitoring system has information about each service's status, this means it can produce warning messages for potential failures during the design phase of a business process. For example, if a service has been offline or not working properly for a week, and a new business process uses it as a component, a warning message should be issued regarding potential unreliability of that service in the future. This allows business process designers to arrange alternative solutions early in the design phase. This could significantly reduce the failure recovery cost.

References

- Abbate J. (1999) *Inventing the Internet*, MIT Press, Cambridge, 1999.
- Agarwal, S., Handschuh, S., and Staab, S. (2004) Annotation, composition and invocation of Semantic Web Services, *Journal of Web Semantics* 2004 2(1): pp. 31-48
- Agosti, M. and Smeaton A. F. (1996) (Edt.) *Information Retrieval and Hypertext*, Kluwer Academic Publishers, 1996.
- Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., & Verma, K. (2005) *Web service Semantics – WSDL-S*, A joint UGA-IBM Technical Note, version 1.2, April 18, 2005.
- Antoniou, G., Billington, D., Governatori, G., and Maher, M. J. (2001) Representation Results for Defeasible Logic, *ACM Transactions on Computational Logic*, Vol. 2, No. 2, April 2001, Pages 255–287
- Antoniou, G. and Harmelen, F. (2004) *A Semantic Web Primer*, the MIT Press, 2004.
- Axis (2005), Apache, <http://ws.apache.org/axis/java/reference.html>
- Baader, F., Horrocks, I., and Sattler, U. (2002) Description logics for the semantic web, *KI - Künstliche Intelligenz*, 16(4):57-59, 2002.
- Baader, F. and Nutt W. (2003) Basic Description Logics, In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors 2003, *The Description Logic Handbook*, Cambridge University Press
- Berners-Lee, T. (1991) *World Wide Web Seminar*, <http://www.w3.org/Talks/General.html>
- Berners-Lee, T., Hendler, J., and Lassila, O. (2001) The Semantic Web, *Scientific American*, May 2001, pp. 34–43.
- Berry, M. W., Drmac, Z., and Jessup, E. R. (1999) Matrices, Vector Spaces, and Information Retrieval, *SIAM Review*, Vol. 41, No. 2 (Jun., 1999), pp. 335-362.
- Bieberstein, N., Bose, S., Fiammante, M., Jones, K., and Shah, R. (2005) *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*, IBM Press, October 19, 2005.
- Binildas C. A. (2008) *Service Oriented Java Business Integration: Enterprise Service Bus integration solutions for Java developers*, Packt Publishing, March 2008.
- Brachman, R. J. (1979) Structured inheritance network, In Woods, W. A. and Brachman, R. J. editors, *Research in Natural Language Understanding*, Annual Report. Technical Report 4274, Bolt Beranek and Newman, Cambridge, 1979

- Brachman, R. J. and Levesque, H. J. (2004) Knowledge Representation and Reasoning, Morgan Kaufmann; 1 edition (May 19, 2004).
- Brewka, G. (1991). *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, 1991.
- Brewka, G. (2001) On the Relationship between Defeasible Logic and Well-Founded Semantics, Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, Vienna Austria, Sep 2001, LNCS 2173, pp. 121-132.
- Brickley, D. and Guha, R.V. (2004) RDF Vocabulary Description Language 1.0: RDF Schema, W3C 2004, <http://www.w3.org/TR/rdf-schema/>
- Brown, P.J., Bovey, J.D., and Chen, X. (1997) Context-Aware Applications: From the Laboratory to the Marketplace. *IEEE Personal Communications*, 4(5) (1997) 58-64
- Bruijn, J., Lara, R., Arroyo, S., Gomez, J. M., Han, S. K. and Fensel D. (2005) A Unified Semantic Web Services Architecture based on WSMF and UPML, *International Journal of Web Engineering and Technology* 2005 - Vol. 2, No.2/3 pp. 148 – 180.
- Calvanese, D., De Giacomo, G., Lenzerini, M., and Nardi, D. (2001) Reasoning in expressive description logics, In Robinson, Alan and Voronkov, Andrei, editors 2001, *Handbook of Automated Reasoning*, The MIT Press.
- Calvanese, D. and Giacomo, G. (2003) Expressive Description Logics, In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors 2003, *The Description Logic Handbook*, Cambridge University Press.
- Cardoso, J. and Sheth, A. (2003) Semantic e-Workflow Composition. *J. Intelligent Information Systems* 21, pp.191-225, 2003
- Cardoso, J. and Sheth, A. (Eds.) (2006) *Semantic Web Services, Processes and Applications, Semantic Web and Beyond Computing for Human Experience Vol. 3* Springer 2006.
- CGXML (2008),<http://tockit.sourceforge.net/cgxml/index.html>
- Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. (1999) What are Ontologies, and Why do We Need Them? *IEEE Intelligent System*, 1999, pp.20-26.
- Charfi, A. and Mezini, M., Mezini, (2004) Hybrid Web service Composition Business Process Meet Business Rules, Proceedings of the 2nd International conference on Service oriented computing, 2004.
- Chein, M. and Mugnier, M. L. (1992) Conceptual Graphs: Fundamental Notions, *Revue d'Intelligence Artificielle*, Vol. 6, Issue 4, 1992, pp 365--406.

- Choudhary, V. (2007) Software as a Service: Implications for Investment in Software Development, in Proc. of the 40th Hawaii International Conference on System Sciences, 2007.
- Collins COBUILD English Dictionary (1995), HarperCollins Publishers Ltd 1995.
- Coulouris, G., Dollimore, and Kindberg, T. (2001) Distributed Systems Concepts and Design, 3rd edition, Addison-Wesley published, 2001.
- Croitoru, M. and Compatangelo, E. (2006) A tree-decomposition algorithm for Conceptual Graph projection, In Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'2006), Lake District of the United Kingdom, June 2-5, 2006.
- DCMI Metadata Terms (2005), Dublin Core Metadata Initiative 2005, <http://dublincore.org/documents/dcmi-terms/>
- Dey, A. and Abowd, G. (2000) Towards a better understanding of context and context-awareness, Proceedings of Workshop on the What, Who, Where, When and How of Context-Awareness, The Hague, Netherlands. April, 2000.
- Du, X., Song, W., and Munro, M. (2006a) Using Common Process Patterns for Semantic Web Services Composition, in Proc. of 15th International Conference on Information System Development (ISD2006), Budapest, Hungary, Aug. 31 - Sept. 2, 2006.
- Du, X., Song, W., and Munro, M. (2006b) Service Composition in the Context of Grid, in Proc. of UK e-Science Program All Hand Meeting (AHM2006), Nottingham, UK, Sept. 18-21, 2006.
- Du, X., Song, W., and Munro, M. (2006c) Semantics Recognition in Service Composition Using Conceptual Graph, in the Proc. of International workshop on Semantics in Virtual Organizations and Web services (SVO&WS), held in conjunction with the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI-06), Hong Kong, China, Dec. 18-22, 2006.
- Du, X., Song, W., and Munro, M. (2007a) Semantic Service Description Framework for Addressing Imprecise Service Requirements, in proceedings of the 16th International Conference on Information Systems Development, Galway, Ireland, Sept. 2007.
- Du, X., Song, W., and Zhang, M. (2007b) A Context-based Framework and Method for Learning Object Description and Search, in Proc. of 6th International Conference on Web-based Learning (ICWL 2007), LNCS Vol. 4823, Springer, Edinburgh, United Kingdom, Aug. 15-17, 2007.
- Du, X., Song, W., and Munro, M. (2008a) An Innovative Approach for Service Description and Discovery in Context of Software as a Service, accepted by the 1st IEEE International Workshop on Barriers towards Internet-Driven Information Services (BINDIS2008), held in conjunction with IEEE COMPSAC 2008, Turku, Finland.

- Du, X., Song, W., and Munro, M. (2008b) A Method for Transforming Existing Web service Descriptions into an Enhanced Semantic Web service Framework, in Proc. of 17th International Conference on Information System Development (ISD2008), Paphos, Cyprus August 25-27, 2008.
- Dustdar, S. and Schreiner, W. (2005) A Survey on Web services Composition, Int. J. Web and Grid Services, Inderscience , Vol. 1, No. 1, pp.1–30.
- Ehrig, M. and Sure, Y. (2004) Ontology mapping - an integrated approach, First European Semantic Web Symposium, pages 76–91, 2004
- Erl T. (2005) Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, August 04, 2005.
- Fensel, D. (2001) Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag Berlin Heidelberg 2001.
- Fensel, D., Lausen H., Polleres A., Buijn J., Stollberg M. Roman D., and Domingue J. (2007) Enabling Semantic Web Services: the Web service Modeling Ontology, Springer.
- Fensel, D. and Bussler, C. (2002) The Web service Modeling Framework WSMF, Electronic Commerce Research and Applications Volume 1, Issue 2, Summer 2002, Pages 113-137.
- Foster I, Kesselman C, and Tuecke S (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, 15(3), 2001.
- Frakes, W. B. & Baeza-Yates, R. (1992) Information Retrieval: Data Structures & Algorithms, Prentice Hall, 1992.
- Genesereth, M. R. and Nilsson N. J. (1987) Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publishers, Aug 1987.
- Gray, N.A.B. (2004) Comparison of Web services, Java-RMI, and CORBA service, Fifth Australasian Workshop on Software and System Architectures. In conjunction with ASWEC 2004, Melbourne, Australia, 2004.
- Gruber, T. R. (1995) Toward Principles for the Design of Ontologies Used for Knowledge Sharing, International Journal of Human and Computer Studies, 43(5/6): 907-928.
- Guarino, N. (1998) Formal Ontology and Information Systems, Proceedings of FOIS'98, Trento, Italy, 1998, Amsterdam, IOS Press, pp. 3-15.
- Guha, R., McCool, R., & Fikes, R. (2004) Contexts for the Semantic Web. In Proceedings of the ISWC'04, Hiroshima, Japan, Nov 2004, Lecture Notes in Computer Science Vol. 3298, pp. 32 – 46, Springer.
- Gunzer H. (2003) Introduction to Web services, Borland Web service White Paper, March 2003.

- Hage, W. R., Katrenko, S., and Schreiber, G. (2005) A Method to Combine Linguistic Ontology-Mapping Techniques, ISWC 2005, LNCS 3729, pp. 732–744
- Horrocks, I., Patel-Schneider, P. F., and Harmelen, F. (2003) From *SHIQ* and RDF to OWL: The making of a web ontology language, *J. of Web Semantics*, 1(1):7-26, 2003.
- Horrocks, I. (2005) Description Logics in Ontology Applications, Presentation at KI/Tableaux 2005, Koblenz, Germany, September 2005
- HTML 4.01 Specification (1999), W3C 1999, <http://www.w3.org/TR/html4/>
- Huhns, M. N. and Singh, M. P. (2005) Service Oriented Computing: key concepts and principles, IEEE Intelligent System, January-February 2005, pp. 75-81.
- Hull, R., Benedikt M., Christophides V., and Su, J. (2003) E-Services: A look behind the curtain, , in proceedings of the 22nd ACM SIGMOD International Conference on Management of Data / Principles of Database Systems, June 9-12, 2003, San Diego, CA.
- IBM (2006) IBM Web services Tutorial, IBM, <http://www-128.ibm.com/developerworks/>
- Joachims., T. (1998) Text categorization with support vector machines: learning with many relevant features. European Conf. Mach. Learning, ECML98, Apr. 1998.
- Josuttis N. M. (2007) SOA in Practice: the Art of Distributed System Design, O'Reilly, August 2007.
- JWSDL (2006), Java Community Process, <http://www.jcp.org/en/jsr/detail?id=110>
- Keidl, M. and Kemper, A. (2004) Towards Context-Aware Adaptable Web services, WWW2004, May 17–22, 2004, New York, New York, USA.
- Klyne, G. and Carroll, J. J. (2004) Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C 2004, <http://www.w3.org/TR/rdf-concepts/>
- Kreger H. (2001) Web services Conceptual Architecture, IBM software group.
- Lara, R., Lausen, H., and Arroy, S., Semantic Web Services: description requirements and current technologies, In International Workshop on Electronic Commerce, Agents, and Semantic Web Services, September 2003
- Leymann, F. (2001) Web services Flow Language (WSFL 1.0), IBM Software Group 2001.
- Linthicum, D. S. (2003) Next Generation Application Integration: From Simple Information to Web services, Addison Wesley, September 2003
- Lu, Z., Ghose, A., Hyland, P., and Guan, Y. (2006) Using Assumptions in Service Composition Context, in Proc. of the IEEE International Conference on Services Computing (SCC'06), Sep. 2006, Chicago, USA.

- Lusch, R. F. and Vargo, S. L. (2006) *The Service-dominant Logic of Marketing: Dialog, Debate, And Directions*, M.E. Sharpe published, February 28, 2006.
- Maamar, Z., AlKhatib, G., and Mostefaoui, S.K. (2005a) Context-based Personalization of Web services Composition and Provisioning, IEEE 30th EUROMICRO Conference, Rennes, France, August/September, 2004.
- Maamar, Z., Mostefaoui, S. K. and Yahyaoui, H. (2005) Toward an agent-based and context-oriented approach for Web services composition, IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 17, no. 5, pp. 686–697, 2005.
- Maamar, Z., Benslimane, D., Thiran, P., Ghedira, C., Dustdar, S., and Sattanathan S., (2007) Towards a Context-based Multi-type Policy Approach for Web services Composition, *Data & Knowledge Engineering*, 62(2): 327-351 (2007).
- MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., and Metz R. (2006) Reference Model for Service Oriented Architecture 1.0, OASIS Committee Specification 1, 2 August 2006
- Manola, F. and Miller, E. (2004) RDF Primer, W3C 2004, <http://www.w3.org/TR/rdf-primer/>
- Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., & Sycara, K. (2004a) OWL-S: Semantic Mark-up for Web services, <http://www.daml.org/services/owl-s/1.0/owl-s.html>
- Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K. (2004b) Bringing Semantics to Web services: The OWL-S Approach, Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 6-9, 2004, USA.
- McGuinness D. L. and van Harmelen, F. (2004) OWL Web Ontology Language Overview, W3C 2004, <http://www.w3.org/TR/owl-features/>
- McIlraith S, Son TC, and Zeng H (2001) Semantic Web Services, IEEE Intelligent System Special Issue Semantic Web 16(2):46–53.
- Medjahed, B. and Atif, Y. (2007) Context-based matching for Web service composition, *Distributed and Parallel Databases*, Volume 21, Number 1, February 2007, pp. 5-37(33).
- Mike Uschold, Michael Grüninger (1996) Ontologies: Principles, Methods, and Applications, *Knowledge Engineering Review*, Vol. 11, No. 2. (1996), pp. 93-155.
- Mineau, G. and Gerbe, O. (1997) Contexts: A formal definition of worlds of assertion. In D. Lukose et.al., editor, *Conceptual Structures: Fulfilling Peirce's Dream*, number 1257 in *Lecture Notes in Artificial Intelligence*, pages 80–94. Springer-Verlag, 1997.

- Montes-y-Gómez, M., Gelbukh, A., López-López, A., & Baeza-Yates, R. (2001) Flexible Comparison of Conceptual Graphs, Proceeding of 12th International Conference and Workshop on Database and Expert Systems Applications. LNCS 2113, Springer, 2001, pp. 102-111.
- Mugnier M. (2000) Knowledge Representation and Reasonings Based on Graph Homomorphisms, in Proc. 8th Int. Conf. on Conceptual Structures, ICCS'2000, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, 1867, 2000, pp. 172-192.
- Mugnier, M. and Chein, M. (1992) Polynomial Algorithms for Projection and Matching, Proceedings of the 7th Annual Workshop on Conceptual Graphs (AWCG'92), New Mexico State University, Las Cruces, New Mexico, USA, juillet 1992, pp. 49-58.
- Nagarajan, M. (2006) Semantic Annotations in Web services, Cardoso, J. and Sheth, A. (Eds.) Semantic Web Services, Processes and Applications, Semantic Web and Beyond Computing for Human Experience Vol. 3, Chapter 2, Springer 2006.
- Nardi, D. and Brachman, R. J. (2003) An Introduction to Description Logics, In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors 2003, The Description Logic Handbook, Cambridge University Press
- Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003) SHOP2: An HTN Planning System, Journal of Artificial Intelligence Research 20 (2003) pp. 379-404
- Newcomer E. and Lomow G. (2004) Understanding SOA with Web services, Addison Wesley Professional, December 14, 20.
- Nute, D. (1994) Defeasible Logic, in Handbook of logic in artificial intelligence and logic programming (vol. 3): Non-monotonic Reasoning and Uncertain Reasoning, Oxford University Press.
- Orriens, B., Yang, Orriens, J., Yang, and Papazoglou, M.P. Papazoglou. (2003) A Framework for Business Rule Driven Web service Composition, Proceedings of the 4th International Workshop on Conceptual Modeling Approaches for e-Business Dealing with Business Volatility, 2003.
- OWL-S API (2008) Mindswap Project, <http://www.mindswap.org/2004/owl-s/api/>
- OWL-S Editor (2008) <http://owlseditor.semwebcentral.org/related.shtml>
- OWL-S Upper-ontology (2006), <http://www.daml.org/services/owl-s/1.0/Service.owl>,
<http://www.daml.org/services/owl-s/1.0/Profile.owl>,
<http://www.daml.org/services/owl-s/1.0/Process.owl>,
<http://www.daml.org/services/owl-s/1.0/Grounding.owl>

- Paolucci, M., Kawamura, T., Payne, T., & Sycara, K. (2002) Semantic Matching of Web services Capabilities, in proceeding of 1st International Semantic Web Conference (ISWC2002), LNCS 2342, pp. 333-347, Springer, 2002.
- Paolucci, M., Srinivasan, N., and Sycara K. (2004) Expressing WSMO mediators in OWL-S, In Semantic Web Services Workshop at ISWC 2004, 2004
- Paolucci, M., Sycara, K., and Kawamura, T. (2003) Delivering Semantic Web Services, in Proc. of WWW 2003, May 20-24, 2003, Budapest, Hungary.
- Peer, J. (2005) Web service Composition as AI Planning - a Survey, University of St.Gallen, Switzerland, 2005.
- Peirce, C. S. (1936-58) 1936-58 Collected Papers of C. S. Peirce, v. 1-6 ed. Charles Hart-shorne and Paul Weiss, v. 7-8 ed. Arthur Burks, Cam-bridge: Harvard.
- Pinto, H. S., Gomez-Perez, A., and Martins, J. P. (1999) Some issues on ontology integration, Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI, 1999, Stockholm, Sweden.
- Pinto, H. S. and Martins, J. P (2001) A methodology for ontology integration, Proceedings of the international conference on Knowledge capture, K-CAP, ACM Press, 2001.
- Rao, J. and Su, X. (2004) A survey of automated Web service composition methods, In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004 Springer-Verlag, USA
- Rock, A. (2000) Deimos: Query answering defeasible logic system.
<http://www.cit.gu.edu.au/~arock/defeasible/Defeasible.cgi>
- Roman, D., Keller, U., Lausen, H., Bruijn, J., and Lara, R. (2005) Web service Modeling Ontology, J. Applied Ontology, vol. 1, no. 1, 2005, pp. 77–106.
- RuleML (2008), <http://www.ruleml.org/>
- Ryan, N., Pascoe, J., and Morse, D. (1997) Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney, V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology, 1997.
- SAAJ 1.3 (2008), <https://saaj.dev.java.net/>
- Sayah, J. Y. and Zhang, L. (2005) On-demand business collaboration enablement with Web services, Decision Support Systems 40 (2005) pp.107– 127, Elsevier B.V. 2005
- Sheth A. and Miller J. A. (2003) Web services: Technical Evolution yet Practical Revolution? IEEE Intelligent Systems, January/February 2003, p78-p79
- Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D. (2004) HTN Planning for Web service Composition Using SHOP2, Web Semantics Journal, Volume 1, Issue 4, pp. 377-396 October 2004.

- Smith, M. K., Welty, C., and McGuinness, D. L., (2004) OWL Web Ontology Language Guide, W3C 2004, <http://www.w3.org/TR/owl-guide/>
- SOAP (2007), SOAP 1.2 Specification, <http://www.w3.org/TR/soap12-part1/>
- Song, W. and Li, X. (2005) A Conceptual Model for Virtual Organizations in the Grid, in the proceedings of the fourth international conference on Grid and Cooperative Computing (GCC05), Beijing, China, Nov. 2005. Zhuge, H., Fox, G.C. (Eds.), Lecture Notes in Computer Science Vol. 3795, Springer-Verlag.
- Song, W., Du, X., and Munro, M. (2009) A Concept Graph Approach to Semantic Similarity Computation Method for e-Service Discovery, to appear in International Journal of Knowledge Engineering and Data Mining, Inderscience publishers.
- Southey F. and Linders J. G. (1999) NOTIO - a Java API for developing CG tools, Lecture Notes in Computer Science, Volume 1640/1999, pp. 262-271.
- Sowa, J. F. (1984) Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Canada.
- Sowa, J. F. (1987) Semantic Networks, Encyclopedia of Artificial Intelligence. Ed. Stuart C Shapiro.
- Srinivasan, N., Paolucci, M., and Sycara, K. (2006) Semantic Web service Discovery in the OWL-S IDE, Proceedings of the 39th Annual Hawaii International Conference on System Sciences, Hawaii, 2006.
- Srivastava, B., and Koe J. (2003) Web service Composition – Current Solutions and Open Problems, in proc. of 13th International Conference on Automated Planning & Scheduling, June 9-13, 2003, Trento, Italy.
- SRML (2001), May 17, 2001, <http://xml.coverpages.org/srml.html>
- Strang, C. J. (2005) Next generation systems architecture — the Matrix, BT Technology Journal, Vol 23 No 1, January 2005.
- Sun Microsystems Inc. (2009) Signed Java Applet, Sun Developer Network, <http://java.sun.com/developer/onlineTraining/Programming/JDCBook/signed.html>
- Thatte, S. (2001) Web services for Business Process Design (XLANG), Microsoft Corporation 2001
- Tous, R. and Delgado, J. (2006) A Vector Space Model for Semantic Similarity Calculation and OWL Ontology, DEXA 2006, LNCS 4080, pp. 307–316, 2006.
- Tidwell, D. (2006) IBM Web services Tutorial, IBM, <http://www-128.ibm.com/developerworks/>
- UDDI (2004), UDDI Version 3.0.2 Specification, http://www.uddi.org/pubs/uddi_v3.htm

- UDDI.org (2006) The Evolution of UDDI, The Stencil Group, Inc.
- Uschold, M., and Grüninger, M. (1996) Ontologies: principles, methods and applications, Knowledge Engineering Review, 1996, Volume 11, No. 2, pp. 93-155.
- Van Rijsbergen, C. J. (1979) Information Retrieval, 2nd ed. London: Butterworths, 1979.
- Wohed, P., Aalst, W.M.P. van der, Dumas, M., and Hofstede, A. H. M. (2003) Analysis of Web service Composition Languages: The Case of BPEL4WS, LNCS 2813, pp. 200-215, 2003
- WS-BPEL (2007) Web services Business Process Execution Language Version 2.0, <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- WSCI (2002) Web service Choreography Interface (WSCI) 1.0, <http://www.w3.org/TR/wsci/>
- WSCL (2002) Web services Conversation Language (WSCL) 1.0, <http://www.w3.org/TR/wscl10/>
- WSDL (2007), Web services Description Language Version 2.0 Specification, <http://www.w3.org/TR/wsdl20/>
- Webservice.org, (2006) Web service Definition, <http://www.webservices.org/>
- W3C, Web services Architecture, 2004, <http://www.w3.org/TR/ws-arch/>
- Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D. (2003) Automating daml-s Web services composition using shop2. In 2nd International Semantic Web Conference (ISWC2003), 2003
- Wu, Z. and Palmer, M. (1994) Verb Semantics and Lexical Selection, Proc. of the 32nd Annual Meeting of the Associations for Computational Linguistics, 1994.
- Xu, B., Wang, Y., Zhang, P., and Li, J. (2005) Web services Searching Based on Domain Ontology, in proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05).
- Yang, Q. (1997) Intelligent Planning: A Decomposition and Abstraction Based Approach, Springer Verlag, Berlin, Germany, 1997.
- Zhang, J., Zhang, S., Cao, J., and Mou Y. (2004) Improved HTN planning approach for service composition, in Proceeding of IEEE International Conference on Services Computing, 15-18 Sept, 2004
- Zhou, T., Zheng, X., Song, W., Du, X., and Chen, D. (2008) Policy-based Web service Selection in Context Sensitive Environment, accepted by IEEE International Conference on Services Computing (SCC 2008) July 8-11, 2008, Honolulu, Hawaii, USA.

