

SEMANTICS-AWARE PLANNING METHODOLOGY FOR AUTOMATIC WEB SERVICE COMPOSITION

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Yan Leng

aus

Nanjing, China

Bonn, 2012

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelm-Universität Bonn

1. Referent: Prof. Dr. Armin B. Cremers, Universität Bonn
2. Referent: Prof. Dr. Thomas Rose, RWTH Aachen

Tag der Promotion: 29.06.2012

Erscheinungsjahr: 2012

Abstract

Service-Oriented Computing (SOC) has been a major research topic in the past years. It is based on the idea of composing distributed applications even in heterogeneous environments by discovering and invoking network-available Web Services to accomplish some complex tasks when no existing service can satisfy the user request. Service-Oriented Architecture (SOA) is a key design principle to facilitate building of these autonomous, platform-independent Web Services. However, in distributed environments, the use of services without considering their underlying semantics, either functional semantics or quality guarantees can negatively affect a composition process by raising intermittent failures or leading to slow performance.

More recently, Artificial Intelligence (AI) Planning technologies have been exploited to facilitate the automated composition. But most of the AI planning based algorithms do not scale well when the number of Web Services increases, and there is no guarantee that a solution for a composition problem will be found even if it exists. AI Planning Graph tries to address various limitations in traditional AI planning by providing a unique search space in a directed layered graph. However, the existing AI Planning Graph algorithm only focuses on finding complete solutions without taking account of other services which are not achieving the goals. It will result in the failure of creating such a graph in the case that many services are available, despite most of them being irrelevant to the goals.

This dissertation puts forward a concept of building a more intelligent planning mechanism which should be a combination of semantics-aware service selection and a goal-directed planning algorithm. Based on this concept, a new planning system so-called ***Semantics Enhanced web service Mining*** (*SEwsMining*) has been developed. Semantic-aware service selection is achieved by calculating *on-demand multi-attributes semantics similarity* based on *semantic annotations* (*QWSMO-Lite*). The planning algorithm is a substantial revision of the AI GraphPlan algorithm. To reduce the size of planning graph, a *bi-directional planning* strategy has been developed.

Acknowledgments

A PhD thesis is never the result of a single person's effort and it would have been impossible without the support and encouragement of many people. I would like to take this opportunity to express my deepest gratefulness to all of them.

First and foremost, I would like to acknowledge the contributions of my supervisor, Prof. Dr. Armin B. Cremers, in making this work possible. I would like to sincerely thank him for his visionary guidance, understanding and patience. I have benefited the most from his vision towards my research work. He has been putting my career in his mind and guiding me along the way. I have learned a lot working with him, from presentation skills, paper writing to approaches of conducting research. Additionally, I would also like to thank Prof. Dr. Thomas Rose from RWTH Aachen University for being my second supervisor and giving many insightful suggestions to this work.

I would also like to thank B-IT research school giving me the chance to do a PhD thesis and members of my doctoral committee, PD Dr. Volker Steinhage and Prof. Dr. Klaus Greve for the valuable discussions and accessibility.

My sincere thanks go to many friends and colleagues in the department of computer science for scientific discussion, advice and continuous support. In this regard, I am particularly grateful to Mahmoud El-Gayyar, Dr. Serge Shumilov, Dr. Alexandra Reitemann, Prof. Dr. Sascha Alda, Dr. Thomas Bode, Dr. Alexander Savinov and Ivan Denisovich for their collaboration, comments and critiques on my ideas and presentations. I also want to thank Dr. Dinan Wang, Zhengzheng Ding, Dr. Lihua Li, Dr. Yuan Wang, Yupeng Cun for their companionship and helps when I need them most.

Last but by no means least, I am and will always be grateful to my parents and family, who have been believing in my potential and giving me never-ending encouragement and ongoing support. Very special thanks go to my husband Guang Yang. He has always been there understanding and supporting me with everything I have ever needed. My daughter, Sonja, even though you cannot say more than a few "uh ah", I know that you meant: "I can always give you a nice smile when you need one".

Table of Contents

CHAPTER 1	
INTRODUCTION.....	1
1.1 MOTIVATION.....	2
1.2 RESEARCH QUESTIONS	3
1.3 CONTRIBUTIONS.....	4
1.4 OUTLINE	5
CHAPTER 2	
THE SEWSMINING FRAMEWORK FOR DYNAMIC SERVICE-ORIENTED COMPUTING	7
2.1 USE CASE OF SEWSMINING	8
2.2 SEWSMINING FRAMEWORK	10
CHAPTER 3	
STATE-OF-THE-ART	12
3.1 WEB SERVICES SEMANTICS	13
3.2 WEB SERVICES ANNOTATION	14
3.2.1 OWL-S	15
3.2.2 WSMO	15
3.2.3 SAWSDL	16
3.2.4 WSMO-Lite	16
3.3 MATCHMAKING METHODS FOR SERVICE DISCOVERY.....	17
3.3.1 Logic based Matchmaking	17
3.3.2 Non-logic based Matchmaking.....	18
3.4 WEB SERVICE COMPOSITION APPROACHES	19
3.4.1 Workflow based Composition.....	19
3.4.2 AI Planning based Composition	21
CHAPTER 4	
QWSMO-LITE: A QOS-AWARE ONTOLOGY FOR WEB SERVICE ANNOTATION.....	25
4.1 REQUIREMENTS ANALYSIS FOR SERVICE DESCRIPTION.....	26
4.2 QWSMO-LITE FRAMEWORK.....	27
4.3 ONTOLOGIES IN QWSMO-LITE	28
4.3.1 Modeling QoS Semantics in QWSMO-Lite.....	29
4.3.2 Specification of Domain Semantics	30
4.4 FORMALIZATION OF QWSMO-LITE	31
4.5 SUMMARY	33
CHAPTER 5	
SEWSDM: A MULTI-ATTRIBUTE SEMANTIC MATCHMAKING ENGINE	34
5.1 SEWSDM FRAMEWORK.....	35
5.2 SINGLE-ATTRIBUTE SEMANTIC MATCHMAKING	36
5.2.1 Traditional Logical Similarity Measures	37
5.2.2 Ontology based Single Attribute Matchmaking Algorithm	39
5.3 MULTI-ATTRIBUTE DECISION MAKING.....	43
5.3.1 Scoring Methods for Multi-Attribute Decision Making.....	44
5.3.2 TOPSIS based Algorithm for Multi-Attribute Decision Making.....	47
5.4 IMPLEMENTATION OF SEWSDM.....	53
5.5 SUMMARY	56

CHAPTER 6

WEB SERVICE COMPOSITION AS PLANNING IN SEWSPL	58
6.1 REQUIREMENTS ANALYSIS FOR AUTOMATIC PLANNING	59
6.2 SEWSPL FRAMEWORK	60
6.3 WEB SERVICE COMPOSITION FORMALISM	62
6.3.1 <i>Traditional Modeling of A Web Service Composition Problem</i>	62
6.3.2 <i>PDDL 3 based Planning Model for Web Service Composition</i>	64
6.4 GRAPHPLAN ALGORITHM	67
6.4.1 <i>Planning Graph</i>	68
6.4.2 <i>Graph Expansion</i>	69
6.4.3 <i>Solution Extraction</i>	70
6.5 ENHANCEMENTS OF GRAPHPLAN IN SEWSPL	70
6.5.1 <i>Simplified Ordered Planning Graph</i>	71
6.5.2 <i>Goal-Oriented Bi-directional Graph Expansion Algorithm</i>	74
6.5.3 <i>Workflow based Planning Extraction</i>	78
6.6 SELF-ADAPTIVE COMPOSITION	82
6.7 SUMMARY	84

CHAPTER 7

EVALUATION OF SEWSMINING.....	85
7.1 EVALUATION OF QWSMO-LITE	86
7.2 EVALUATION OF SEwsDM	87
7.3 EVALUATION OF SEwsPL	91
7.3.1 <i>Scalability Analysis</i>	91
7.3.2 <i>QoP Analysis</i>	94
7.3.3 <i>Dynamicity Analysis</i>	96

CHAPTER 8

CONCLUSION AND FUTURE WORK	98
8.1 SUMMARY	99
8.2 OUTLOOK ON FUTURE WORK	102
8.2.1 <i>Improvement for the Semantic Enhancement Layer</i>	102
8.2.2 <i>Enhancement of the WSD/WSC Layer</i>	102
8.2.3 <i>Enrichment of the Adaptation Layer</i>	103
REFERENCES	105

List of Figures

Figure 1-1 SOC research road map [Papazoglou et al., 2007].....	3
Figure 2-1 Use case diagram	8
Figure 2-2 System framework of <i>SEwsMining</i>	11
Figure 3-1 Ontology of QoS model [Yang et al., 2006].....	14
Figure 3-2 Taxonomy of semantic annotation approaches inspired by [Vladislava, 2006].....	14
Figure 4-1 QWSMO-Lite framework	28
Figure 4-2 Metadata model of <i>QoSParameter</i>	29
Figure 4-3 An example of the QoS specification in QWSMO-Lite.....	30
Figure 4-4 Ontology of <i>DomainFunctionalClassificationRoot</i>	31
Figure 4-5 An example of data semantic annotation	31
Figure 4-6 Description of an annotated function	32
Figure 4-7 Description of an annotated Web Service.....	33
Figure 5-1 <i>SEwsDM</i> matchmaking framework	36
Figure 5-2 Relationship between annotated services and ontology matchmaking.....	37
Figure 5-3 Multi-attributes decision making in <i>SEwsDM</i>	44
Figure 5-4 <i>MAMatching</i> algorithm.....	47
Figure 5-5 The execution time of normalization methods.	50
Figure 5-6 Normalization of data within interval [0,1]	51
Figure 5-7 Normalization of data within interval [50, 500].....	51
Figure 5-8 Activity diagram of <i>SEwsDM</i>	55
Figure 5-9 Matchmaking algorithms in <i>SEwsDM</i>	56
Figure 6-1 The <i>SEwsPL</i> framework	60
Figure 6-2 PDDL description for a meteorology service.....	63
Figure 6-3 QoS description using PDDL 2.1/PDDL 2.2.....	64
Figure 6-4 Simplified class diagram for the PDDL2Model	65
Figure 6-5 Mapping from QWSMO-Lite to PDDL2Model	66
Figure 6-6 An example of mapping to PDDL2Model	67
Figure 6-7 An example of a Planning Graph.....	69
Figure 6-8 Data structure of SOPG	72
Figure 6-9 Examples of SOPG: (a) F-SOPG (b) B-SOPG	74
Figure 6-10 A pseudo-code description of the bidirectional expansion.....	75
Figure 6-11 The comparison part of the expansion algorithm	76
Figure 6-12 Bidirectional expansion part of the expansion algorithm: (a) forward expansion (b) backward expansion	77
Figure 6-13 An example of bi-directional expansion algorithm	77
Figure 6-14 Redundant actions in F-SOPG and B-SOPG	79
Figure 6-15 The result of the meteorological scenario: (a) the workflow in SCUFL (b) the KML file	82
Figure 6-16 The original F-SOPG and B-SOPG.....	84
Figure 6-17 Updated SOPG Planning Graphs.....	84
Figure 7-1 Logical DoM using different systems.....	90
Figure 7-2 Graph size comparison among <i>SEwsPL</i> , BPGP and SPG/WPG	92
Figure 7-3 Execution time comparison among <i>SEwsPL</i> , BPGP and SPG/WPG	93
Figure 7-4 QoP comparison	94
Figure 7-5 QoS comparison between <i>SEwsPL</i> and MOP	95
Figure 7-6 Execution comparison between <i>SEwsPL</i> and MOP	95
Figure 7-7 The original Planning Graph.....	96

Figure 7-8 The Planning Graph grown by [Yan et al., 2010 a] 96

Figure 7-9 The Planning Graph generated by [Yan et al., 2010 b] 97

List of Tables

Table 4-1 A global weather service.....	28
Table 5-1 Comparison of the various measures.....	42
Table 5-2 Normalization methods.....	48
Table 5-3 Operation candidates	50
Table 5-4 Weight values of three decision makers	52
Table 5-5 Ranking results by different DMs.....	53
Table 6-1 A Web Service composition example	61
Table 6-2 Mapping between SOPG and SCUFL	80
Table 7-1 A comparison of existing approaches to QWSMO-Lite	87
Table 7-2 A comparison of <i>SEmsDM</i> with existing approaches.....	88
Table 7-3 Non-Logic DoM of two criteria	90
Table 7-4 Test set from WSC 2010	91

Chapter 1

Introduction

“The important thing in life is to have a great aim, and the determination to attain it.”

- Johann Wolfgang von Goethe (1749-1832)

Service-Oriented Computing (SOC) has been a major research topic in the past years. One of the core principles of SOC is the idea of assembling services to form a chain by discovering and dynamically invoking those multiple existing services which are platform-independent and self-describing applications to satisfy a single task, rather than building new applications from scratch. The problem of building such composition chains is known as a Web Service Composition (WSC) problem. Automation of this process is emerging as one of the most interesting challenges facing SOC today. On the other hand, a key component of effective Web Service composition, which has largely been ignored, is the consideration of quality of planning involving Quality of Service (QoS) of each component service and user preference. In this dissertation, we propose a means of performing automated Web Service composition within the so-called *SEmsMining* framework by specifying and integrating QoS and user preference. The key idea is that the composition is performed with an extended AI Planning Graph algorithm where the underlying semantics for the individual component services are exploited to enable discovering and selecting the most suitable services. A goal directed bi-directional planning algorithm is developed to chain the services in an automatic and more efficient way.

Contents

1	INTRODUCTION.....	1
1.1	MOTIVATION.....	2
1.2	RESEARCH QUESTIONS.....	3
1.3	CONTRIBUTIONS	4
1.4	OUTLINE	5

1.1 Motivation

Service-**O**riented **C**omputing (SOC) has been attracting tremendous attention from both industrial and academic communities. It is based on the idea of composing distributed applications even in heterogeneous environments by discovering and invoking network-available computational resources which are modeled as services to accomplish some complex tasks when no existing service can satisfy the user request. The visionary promise of SOC is that it will be possible to easily assemble application components into a loosely coupled network of services that can create dynamic business processes and agile applications that span organizations and computing platforms [Leymann, 2005]. The challenges of SOC can be illustrated using a research road map shown in Figure 1-1 which separates functionality into three layers:

- *Service Foundation Layer.* Applications are abstracted as Web Services to realize the run time **S**ervice-**O**riented **A**rchitecture (SOA) in this layer. Web Services are described using **W**eb **S**ervices **D**escription **L**anguage (WSDL) and published in **U**niversal **D**escription, **D**iscovery and **I**ntegration (UDDI) where the client is capable of querying and retrieving WSDL descriptions. These will be used to bind to the provider and invoke the service via **S**imple **O**bject **A**ccess **P**rotocol (SOAP) message. In many domains, multiple services are able to provide similar functional properties, although with different level of the quality. Accordingly, the selection of the appropriate service among these relevant services should consider not only the functional semantics, but also **Q**uality of **S**ervice (QoS), such as response time, throughput, price etc. However, previous traditional SOC approaches in this layer focused on the lower-level syntactic service discovery [Hang and Singh, 2010]. Unfortunately, plain syntactical information is not enough for quality based service selection. Therefore, an understanding of the underlying semantic concepts is required.
- *Service Composition Layer.* In this layer, service aggregators assemble multiple component Web Services which can be executed sequentially or concurrently to form a composite service to achieve complex behaviors. One major challenge is how to build such composite services. In SOC, composing services can be done either in a manual or an automatic way. In the manual approach, the composite service is built by domain experts. They are required to select Web Services and specify relationships among them. Such approach is error prone, time consuming and therefore not suitable for large-scale WSC problems. For an automatic approach, much effort is related to AI planning technologies. But most of previous AI planning based algorithms do not scale well when the number of Web Services increases. Furthermore, there is no guarantee that a solution for a composition problem will be found even if one exists [Oh and Kumara, 2006]. In addition, although a lot of previous research aim to find functionally satisfied composition chain, such functional composition is not sufficient in most cases, since many combinations of component services with different levels of quality might satisfy a given task. Users must be further assisted in selecting the best or near best composition chain satisfying their functional and non-functional requests. Adding QoS awareness to the service composition process will be the solution to this problem. However, this has been largely ignored by most of the previous work.

- *Service Management Layer*. Web environments are highly dynamic. Services can appear and disappear around the clock. Such dynamic changes can be monitored and recognized in the *Service Management Layer*. In SOC, most of WSC solutions endorse a static viewpoint of the composition, rather than capturing the dynamic nature, such as the status of services. Therefore self-adapting management is necessary for the composition. The system should adapt to changes in dynamic environments. For instance, at runtime, services with poor performance need to be detected and replaced with similar but more efficient services dynamically. On the other hand, suitable adaptations to meet changing end-user's requirements are also crucial.

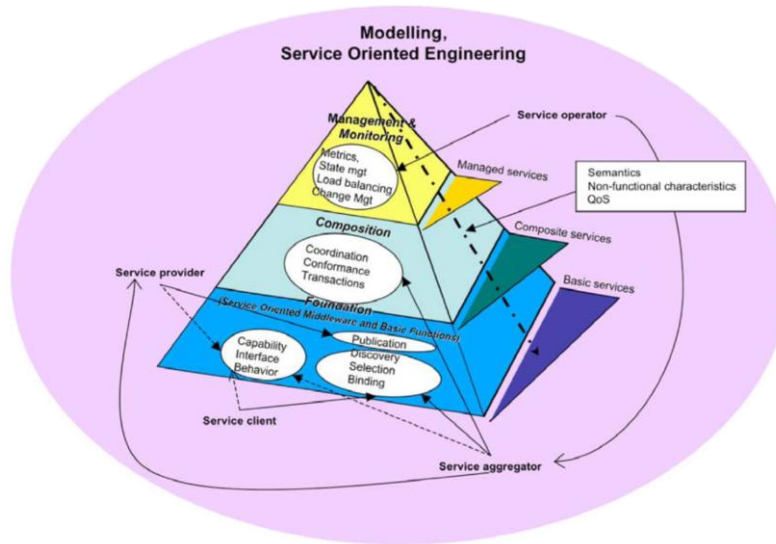


Figure 1-1 SOC research road map [Papazoglou et al., 2007]

1.2 Research Questions

Based on the research road map of SOC introduced above, [Papazoglou et al., 2007] classified and outlined the major research challenges of SOC. This dissertation is dedicated to addressing some of them to enrich the current SOC with semantic awareness and a more flexible composition strategy. In this part, those selected research questions of each layer that drive this dissertation's endeavors will be outlined.

Q1: *What kind of semantics have to be considered to enrich the description of Web Services and how to integrate them to the Web Service Description Language (WSDL)?*

Q2: *How could the annotated semantics help in discovering the appropriate services?*

The first two questions concerning service discovery are derived from the *Service Foundation Layer*. The main challenge of service discovery is how to accurately discover services with minimal user involvement. To this end, semantic annotation for the functional and QoS characteristics of services together with the user's preference needs to be annotated (as stated in

Q1). On the other hand, this also requires the enhancement of the understanding of and reasoning about these annotated semantics, which is addressed in **Q2**.

Q3: *How to facilitate the automation of service composition?*

Q4: *How can the annotated semantics, especially QoS aspects be considered to support the more flexible and accurate composition?*

Apparently, **Q3**, **Q4** address the research perspectives from the *Service Composition Layer*. One of the most notable research challenges for service composition is how to minimize user intervention during the service composition procedure and accelerate the process of creating a composition schema. A major drawback in most automated service composition techniques is the lack of scalability. This might be the main reason why in practice the manual composition techniques are still being widely adopted. **Q3** concerns the problem of composing service efficiently. Moreover, based on the semantics of the underlying services, especially the QoS aspects will help the system to determine the appropriate composition chains satisfying user's request. Unfortunately, this issue has been largely ignored in most of the current automated service composition techniques. Such research requirement is emphasized in **Q4**.

Q5: *How could the system support to adapt the composition chain to run-time dynamic changes?*

The research question addressed in **Q5** is related to the issuers in the *Service Management Layer*. Most of the current existing works produce only the static solution for the Web Service Composition problem without considering the dynamic changes of the distributed environments. The term “dynamic changes” means the availability and the QoS of the existing services will change from time to time. To get a more reasonable and flexible composition result, these changes should be detected and the composition chains should be capable of tuning themselves to meet the requirements of altered environments.

1.3 Contributions

With regard to the aforementioned research problems, a **Semantics Enhance Web Service Mining** (SEwsMining) system is desired to provide an intelligent planning strategy to enable an automated and flexible Web Service Composition. The contributions of this work can be summarized as follows:

- *Development of QWSMO-Lite ontology.* Addressing **Q1**, QWSMO-Lite is a service ontology which enhances WSMO-Lite with an ontology for modeling QoS properties. Both functional and non-functional semantics are allowed to be annotated to the services. Since QWSMO-Lite is based on WSMO-Lite [Kopecký and Vitvar, 2008], it simplifies the way to annotate Web Services with predefined annotation rules, which ensure that the annotations are complete and consistent. Moreover, in QWSMO-Lite, the QoS characteristics are modeled in a three layered modular framework. Such modular structure allows QWSMO-Lite to be extended and adapted easily with additional QoS properties. More details are presented in Chapter 4.
- *Realization of the multi-attribute semantic matchmaking engine for service discovery.* To address the research problem presented in **Q2**, we contribute a **Semantics Enhanced Web Service**

Decision **M**aking (SEwsDM) engine to facilitate the process of service discovery by calculating multi-attribute similarity. First of all, in *SEwsDM*, to increase the accuracy in assigning a matching degree, the semantic distance is calculated with an asymmetric matching formula, rather than assigning fixed numeric values to various logical filters, which is widely applied in most current work. Furthermore, other than recognizing the semantically related services, *SEwsDM* is also capable of ranking services. A **M**ulti-**A**tttribute **D**ecision **M**aking (MADM) based technique [Yoon and Hwang, 1995] where both functional and non-functional semantics of underlying services and the user's preference are taken into account to generate the similarity score which is integrated into *SEwsDM*. In addition, a flexible bidirectional search strategy is supported in *SEwsDM* by allowing both backward and forward service discovery. Technical details are shown in Chapter 5.

- *Enhancement of an AI GraphPlan based Planner for service composition.* We contribute a **S**emantics **E**nhanced **W**eb **S**ervice **P**Lanner (SEwsPL) which allows a goal-directed automated service composition [Leng et al., 2010]. This newly developed planner is a reconstruction of the AI GraphPlan algorithm [Blum and Furst, 1997]. Referring to **Q3**, in order to facilitate the automation of service composition, we define a two-step composition algorithm. In the first step, a Planning Graph storing all related services and their possible semantic links is created by a bidirectional expansion algorithm which alternately grows the graph from the initial states and from the goal states. With such strategy, the size of search space for the plans will be efficiently reduced. Afterwards, the procedure of extracting the plan is simplified by representing the generated graph as a workflow.
- *Specification of a new Planning Graph Model.* To a certain extent, **Q4** will be solved by means of a specific Planning Graph which is called **S**implified **O**rdered **P**lanning **G**raph (SOPG). Different from the Planning Graph in classic GraphPlan algorithm where services are randomly listed in the action layer, in the proposed SOPG all selected services are ordered in terms of their semantic distances which are calculated using *SEwsDM*. That means according to different user's QoS requirements various combination of services will be generated in the action layer of SOPG even if the user's functional requirements remain unchanged.
- *Providing a self-adaptive composition mechanism.* In addition, the requirement of adaptability mentioned in **Q5** can be met by a Plan Repair approach in *SEwsPL*. Though in theory modifying an existing plan is no more efficient than complete re-planning in the worst case [Nebel and Koehler, 1995], repairing the plan in practice is much better than planning from scratch, since most of the plan might still remain valid even if some situation changes as pointed in [Krogt and Weerdt, 2005]. Also, from the end-users' point of view, the modified plan might more easily be accepted than a complete new one.

1.4 Outline

The remainder of this dissertation is organized in the following way:

Chapter 2 is concerned with the conceptual approach of *SEwsMining*. The goal of this chapter is to give an overview of the system. It begins with the description of the use case

diagram where the supported functionalities are outlined with the scenario from the meteorology domain. Afterwards, the basic framework of *SEnsMining* along with the brief description of the main components in each layer is illustrated.

Chapter 3 introduces the related work classified into the core areas described earlier as part of the contributions.

Chapter 4 deals with the QWSMO-Lite ontology. First, the specification of QWSMO-Lite is presented in detail. It analyses the requirements of Web Service description for service discovery and composition. The QWSMO-Lite framework and its underlying ontologies are then discussed in detail. The remainder of this chapter summarizes the characteristics of QWSMO-Lite.

Chapter 5 comprises the contribution related to multi-attribute semantic matchmaking engine *SEnsDM*. This section illustrates the fundamental concept of the *SEnsDM* which is later used and integrated into the Web Service composition procedure. Again, after presenting the details of the matchmaking mechanism, we end this section with the summary of the features of *SEnsDM*.

Chapter 6 presents the considerations related to the AI planning based composition algorithm called *SEnsPL*. The details of the underlying techniques are introduced with a running example. Besides, a self-adaptive strategy is also presented. In the last part of this section, related work concerning the service composition and adaption is compared and evaluated.

Chapter 7 evaluates and compares the above three main components of the *SEnsMining* system with related work.

Chapter 8 summarizes the results of this dissertation and presents perspectives for future work.

Chapter 2

The *SEwsMining* Framework for Dynamic Service-Oriented Computing

“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.”

- Prof. C.A.R. Hoare

According to the current challenges of SOC discussed in Chapter 1, this dissertation puts forward a concept of building a more intelligent Web Service Composition strategy which should enhance semantic understanding of functional and non-functional aspects of Web Services and provide semantic enabled planning algorithm to facilitate the automation of the service discovery and service composition. For this purpose, a novel **S**emantics **E**nhanced **W**eb **S**ervice **M**ining (*SEwsMining*) is developed in this dissertation. The goal of this chapter is to give an overview of the system. It begins with the description of the use case diagram where the supported functionalities are outlined with the scenarios from a meteorology domain. Afterwards, the basic framework of the *SEwsMining* together with the main components in each layer is illustrated. The related work regarding the system is discussed in Chapter 3. More specific implementation details of each layer are depicted in Chapter 4, 5, 6 respectively.

Contents

2	THE SEwsMINING FRAMEWORK FOR DYNAMIC SERVICE-ORIENTED COMPUTING	7
2.1	USE CASE OF SEwsMINING	8
2.2	SEwsMINING FRAMEWORK	10

2.1 Use Case of *SEwsMining*

In this section, we present a motivating scenario in the meteorology domain. Let us assume that a renewable energy scientist attempts to do an experiment about the wind speed forecast for selected cities across the United States. Now he only has some local information about the selected cities, such as city name, IP address. And he expects to find an existing service containing the wind speed forecast data with high throughput. In order to achieve this goal, he searches for a Web Service which takes the available local information as inputs and produces the corresponding wind speed data in the *SEwsMining*. Finally a composition chain consisting of several existing services with the high-throughput is generated. And such chain can be later reused, visualized and executed in a workflow to get the final result. Moreover, in the case where the system detects that some of services are no longer available, the composition results will be updated. (See Chapter 6 for more details).

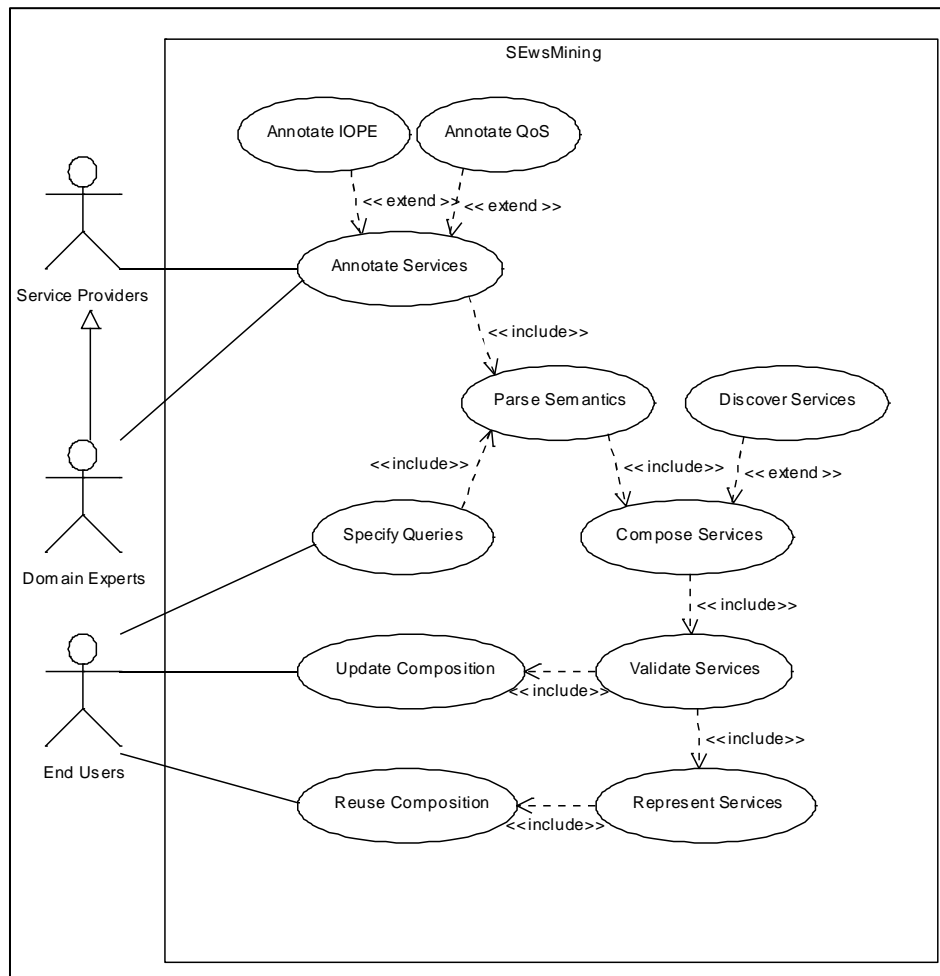


Figure 2-1 Use case diagram

Figure 2-1 depicts a use case diagram for *SEwsMining*. In general, there are two kinds of actors participating in this use case, namely service providers and end-users. Service providers

together with domain experts who are also treated as a kind of service providers are responsible for providing semantically annotated services for the whole system. In our above scenario, to use the composition system, available Web Services should be firstly annotated with semantics for *Inputs*, *Outputs*, *Preconditions* and *Effects* (IOPE). For instance, a function called *GmlLatLonList* returns digital weather *Geography Markup Language* (GML)¹ or *Keyhole Markup Language* (KML)² encoded *National Digital Forecast Database* (NDFD) data for a list of points at a single valid time in an NDFD XML Web Service. It can be annotated by meteorologists with the help of computer experts. Specifically, its inputs can be described by a Geo positioning ontology³ which represents a sequence of latitude and longitude pairs for given points, a time ontology⁴ pointed to the given valid time and a NDFD ontology linked to a set of feature parameters. Similarly, we annotate its outputs as GML and KML. Besides, QoS requirements such as high throughput in this scenario and QoS properties of each service such as cost, availability, and response time can also be added to the semantic annotations.

End-users in this scenario might be renewable energy scientists who need to observe the wind power in a certain place by recording wind speeds over different time periods. The main functionalities provided by the *SEwsMining* can be listed as follows:

- *Web Services Discovery*. The renewable energy scientist wants to find the zip code for a given city. He specifies the city name as the input and the zip code as the output to the system, and then a Web Service called *USZip* which is exactly satisfied with the requests will be retrieved.
- *Web Service Composition*. If there is no single service that fulfills user's requests, a composition chain containing a sequence of related services will be generated. For instance, the renewable energy scientist wants to get a sequence of wind speed values for a list of given cities. However, there is no exactly matched service. Instead, a composition chain consisting of a service called *ZipcodeLookupService* which is able to get location information according to a given city and a service called *GmlLatLonList* which uses the location information as inputs and returns wind speed values encoded in GML/KML, will be generated.
- *Adaptation*. The returned results, for either service discovery or service composition will be automatically updated when the service environments changes. For example, in this mentioned scenario, a used Web Service, *ZipcodeLookupService* becomes unreachable. *SEwsMining* will modify the generated chain above by only replacing this unavailable service, *ZipcodeLookupService*, with *USZip* which returns the zip code in terms of the city name and *LocationByZipService* which returns the location information according to the Zip code.
- *Reuse of Composition*. The generated results are allowed to be reused, visualized by some external workflow management systems. In our scenario, the generated composition schema based on *Simple Conceptual Unified Flow Language* (SCUFL)⁵ can be shared with other scientists by visualizing it in Taverna which is a domain independent tool for designing and executing workflow.

¹ GML is an Open Geospatial Consortium standard for encoding geospatial data.

² KML is a file format used to display geographic data in an Earth browser, such as Google Map, Google Earth.

³ Geo positioning ontology is available in http://www.w3.org/2003/01/geo/wgs84_pos#

⁴ Time ontology is available in <http://www.w3.org/TR/owl-time/>

⁵ The SCUFL is a dataflow-centric language, defining a graph of data interactions between different services.

2.2 SEwsMining Framework

According to use case described below, a **Semantics Enhanced Web Service Mining** (SEwsMining) system is developed focusing on the semantic-aware automated service discovery and composition. The overall architecture of the *SEwsMining* is presented in Figure 2-2. It consists of the following three layers:

- *The Semantics Enhancement Layer.* It aims to annotate Web Services with semantics involving both functional and non-functional characteristics. In this dissertation, a new specification so-called **QoS based WSMO-Lite** (QWSMO-Lite) is introduced to describe the annotated Web Services. Generally speaking, our newly developed QWSMO-Lite is an extension of WSMO-Lite with QoS awareness. More details about QWSMO are presented in Chapter 4.
- *Web Service Discovery and Composition Layer.* Based on the semantics obtained from the first layer, this layer provides the realization of the service discovery and composition. To this end, two main components, namely **Semantics Enhanced Web Service Decision Making** (SEwsDM) and **Semantics Enhanced Web Service PLanner** (SEwsPL) are developed for the service discovery and composition respectively. Specifically, the *SEwsDM* is a multi-attribute semantic matchmaking engine which is capable of discovering the related services against users' functional requirements and QoS Requests with bidirectional matching strategy. Moreover those recognized services are ranked according to their semantic closeness with TOPSIS based ranking algorithm. If the forward and backward ranked services are equal, it means there are existing services satisfying user's requests directly. Otherwise, the service composition component will be invoked to combine all related service to fulfill the use's goal. More details about the *SEwsDM* are described in Chapter 5. The *SEwsPL* is a semantic-aware goal-directed planner for service composition, which is a reconstruction of GraphPlan algorithm. First, a directed layered graph is created by expanding the graph forwards and backwards with the help of the ranking algorithm in the *SEwsDM*. Afterwards, the plan is extracted from the graph by representing the graph as a workflow. Such workflow can be reused, visualized and executed in the workflow management systems. The details of the planning algorithm are illustrated in Chapter 6.
- *Plan Validation Layer.* In *SEwsPL*, it provides the support for the plan adaptation as well as for the dynamic environments. It is achieved by implementing a Plan-Repair based self-adaptation algorithm which attempts to reuse the most part of the original composition results. The algorithm details are available in Section 6.6.

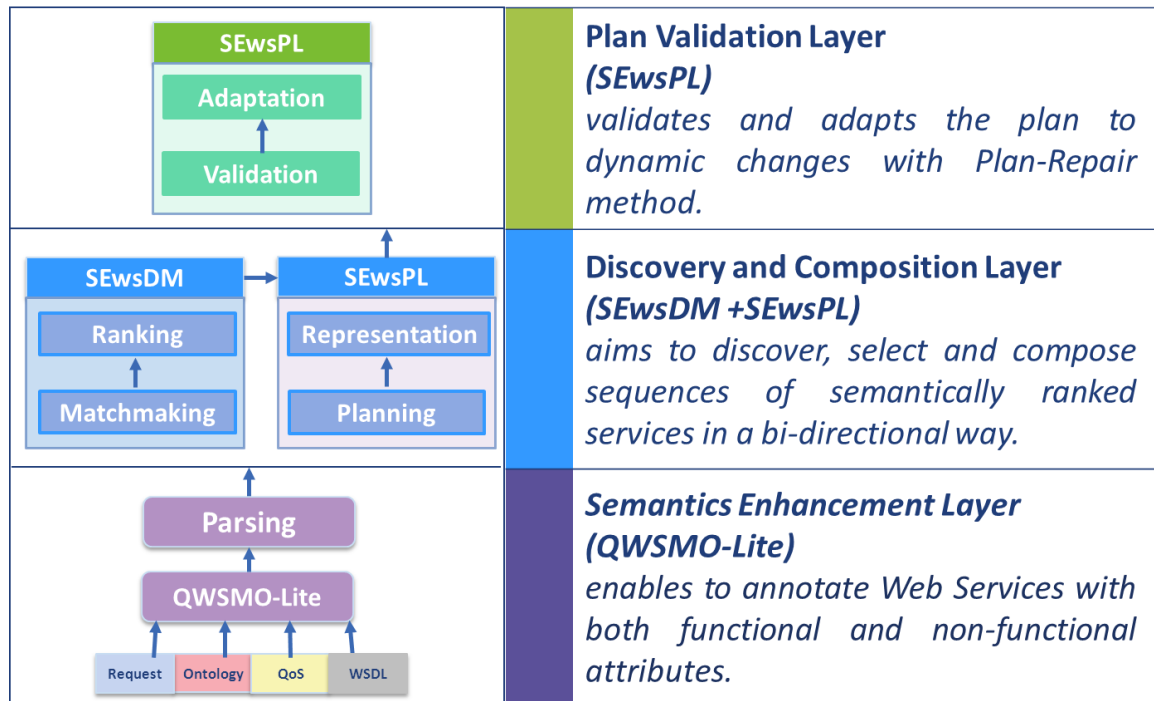


Figure 2-2 System framework of *SEwsMining*

Chapter 3

State-of-the-Art

“Those who cannot remember the past are condemned to repeat it.”

- George Santayana (1863-1952)

This chapter presents a review of the state-of-the-art technologies according to the contributions of this dissertation. In a nutshell, Web Services composition is a three-stepped process. Since Web Services standards lacking semantics operate only in the syntactical level, adding semantics to WSDL to enrich the machine readability is the most fundamental step. Given those service semantics, how to recognize the semantically related pairs between a service offer and a request, so called service matching is the second step. Based on the knowledge from the first two steps, the service composition can be invoked which concerns about how to make the composition process automatically and efficiently. The organization of this chapter depends on such three-stepped composition process. It begins with the introduction and classification of required semantics for automatic composition. And then the survey of most recent technologies according to each step will be presented.

CONTENTS

3	STATE-OF-THE-ART	12
3.1	WEB SERVICES SEMANTICS	13
3.2	WEB SERVICES ANNOTATION	14
3.2.1	OWL-S.....	15
3.2.2	WSMO	15
3.2.3	SAWSDL	16
3.2.4	WSMO-Lite	16
3.3	MATCHMAKING METHODS FOR SERVICE DISCOVERY.....	17
3.3.1	Logic based Matchmaking.....	17
3.3.2	Non-logic based Matchmaking.....	18
3.4	WEB SERVICE COMPOSITION APPROACHES	19
3.4.1	Workflow based Composition.....	19
3.4.2	AI Planning based Composition	21

3.1 Web Services Semantics

To enhance the machine readability of traditional Web Services, a variety of approaches have been developed to support semantic annotation of Web Services.

First of all, we should know which kinds of semantics are required to be annotated. The semantics can be divided into different types on the base of different indications [Vladislava, 2006].

- *Data semantics*, formally defining data in inputs and outputs messages of Web Services;
- *Operational semantics*, expressing business logics and corresponding to Web Services capabilities;
- *Execution semantics*, relating to execution and dynamic service invocation;
- *QoS semantics*, describing the quality aspect of a Web Service.

All these kinds of semantics play important roles in web process lifecycle. The first three kinds of semantics are functional semantic which specify the detail semantic information about the underlying functions supported by a service. To automatic Web Services discovery, *data semantics* and *operational semantics* with explicit meaning of input/output parameters and operations of Web Services will be useful to find a list of related services. When composing Web Services, rather than *data semantics*, *operational semantics* which are useful to build a composition chain, *execution semantics* can be employed to validate the generated chains.

The QoS semantic are non-functional semantics with multiple attributes of quality. It aims to evaluate and select a Web Service that is the most appropriate satisfied with users' requirements among Web Services candidates which are all functional similar to the requirements. Multiple attributes of QoS information, such as performance, accessibility, security, etc. [Wang et al., 2006][Menasce, 2002], are used to rank candidates according to the quality degree. However, since the definitions of these attributes are informal, it leads to the ambiguous interpretation of QoS attributes between services providers and consumers. To bring such gap, in paper [Yang et al., 2006], a general ontology of QoS attributes is presented (see Figure 3-1). All these attributes fall into four catalogs, namely performance, dependability, cost and security. *Performance* provides semantic about how the service is executed over the network. From a service provider's point of view, maximum throughput is mainly concerned to be able to serve the largest number of customs. On the other side, a service consumer wants to minimize the observed response time. *Dependability* integrating several properties deals with how good the service is. The expanse for a service execution is another important element associated with the *Cost*. Service consumers expect to obtain services with required functionalities and minimum cost, service provides intends to maximize the services' prices. *Security* considers how secure the service is.

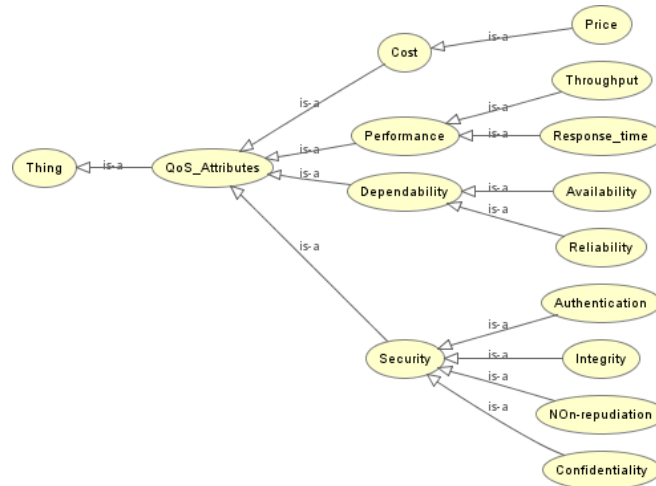


Figure 3-1 Ontology of QoS model [Yang et al., 2006]

3.2 Web Services Annotation

How these mentioned semantics are annotated to Web Services is concerned here. Recently, three main approaches have been developed to bring semantics to Web Service. They are OWL-S⁶, WSMO⁷ and SAWSDL⁸, which have reached the status of proposed recommendations within W3C. In this section, these three standards with their variant will be introduced and compared in terms of the taxonomy shown in Figure 3-2. In addition to those four kinds of semantics, *formalism* reflects the methodology how the semantics is annotated to Web Services. It can be annotated either in a top-level ontology which is a domain-independent and structured formal semantics or with a bottom-level extension.

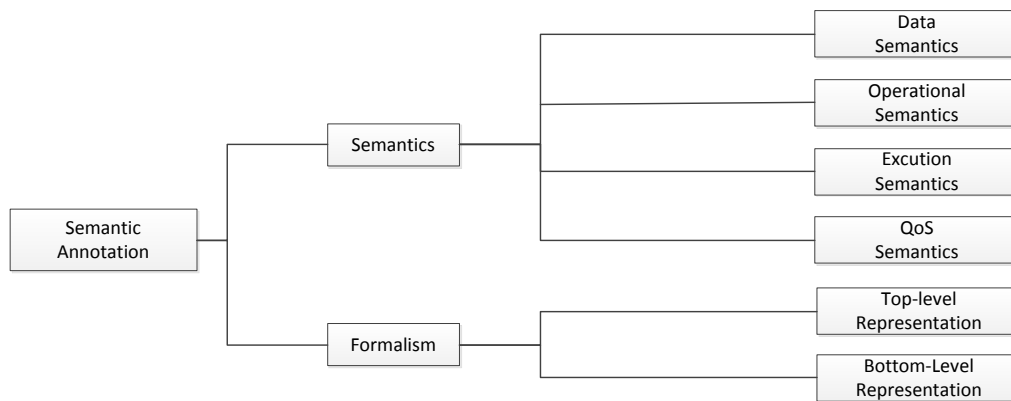


Figure 3-2 Taxonomy of semantic annotation approaches inspired by [Vladislava, 2006]

⁶ See <http://www.w3.org/Submission/OWL-S/> for details.

⁷ See <http://www.w3.org/Submission/WSMO/> for details.

⁸ See <http://www.w3.org/2002/ws/sawSDL/> for details.

3.2.1 OWL-S

OWL-S coalition defines a computer interpretable semantic markup language for describing Web Services, known as OWL-based Web Service Ontology (OWL-S) [Martin et al., 2007]. A top-level ontology with three essential elements is defined for semantic specification of Web Services: *ServiceProfile* presenting what the service does, *ServiceModel* describing how the service works and *ServiceGrounding* supporting how to access the service.

Rather than using a direct annotation of Web Service description itself, OWL-S depends on its upper ontology to semantically enrich Web Services. Data semantics is expressed in the *ServiceModel*. The *ServiceGrounding* maps the semantic representation to the underlying inputs and outputs parameters of Web Services. The operational semantics are represented in the *ServiceProfile* by giving three basic types of information: what organization provides the service, which function the service computes, and a host of features that specify characteristics of the service. The services are modeled as processes in the *Process Model* of the *Service Model* where declares the execution semantics by specifying the interaction of atomic functions in a service. Unfortunately, QoS semantic provided by OWL-S is without considering about QoS specification.

OWL-Q is a rich, extensible and modular ontology language that complements the OWL-S with QoS description [Kritikos and Plexousakis, 2009]. It is a top-level ontology comprising of several independent facets which are associated with a particular part of QoS specification. In addition, such modular-based approach makes this ontology more flexible to add new ontologies from application domain, such as time unit and currency ontologies.

Quality of Service Modeling (QoS-MO) [Tondello and Siqueira, 2008] allowing the extension of OWL-S description with well-defined QoS constraints is also a top-level ontology. Compared to OWL-Q, this ontology defines the concept of *QoSDimensionMapping* to specify the interdependent requirements of QoS between providers and consumers.

onQoS [Giallonardo and Zimeo, 2007] enhances OWL-S by providing QoS specification acting as its *ServiceParameters* in *ServiceProfile*. A powerful data type system to define quality properties values are strongly supported in this ontology. However, the main drawback of it is that the support for QoS constraints is limited.

3.2.2 WSMO

The Web Service Modeling Ontology (WSMO) [Roman et al., 2005] aims to describe various aspects of Semantic Web Services. Taking the Web Service Modeling Framework (WSMF) as a starting point, WSMO extends this framework and addresses to solve the Web service integration problem. To this end, four top level elements are defined to describe a semantic web service: *Ontologies* provide a formal domain specific machine-readable semantics of a shared conceptualization for both service providers and requesters. *Goals* specify intensions that need to be fulfilled using web services. Goals in WSMO are formulated by requesters without realizing the details of the underlying web services. *Web services* describe various aspects of a service, including the functionality and data interaction. *Mediators* resolve interoperability problems among different elements under data processing and protocol level.

Similar to OWL-S, WSMO defines a top-level ontology mentioned above to semantically enhance the WSDL. Data semantics and operational semantics are specified as *capability* in *Web Services* using formally defined terminologies in *ontology*. Execution semantics is annotated in

interface of *Web Services* in terms of a service's *choreography* indicating how to access the service from the user's perspective and *orchestration* focusing on how to integrate other Web Services to achieve its capability from the provider's point of view. QoS semantics is described as a part of non-functional properties using Dublin core⁹. However, such definition is not expressive and flexible enough for QoS characteristics. Therefore, the current support in WSMO to deal with such QoS description is rather limited.

WSMO-QoS [Wang et al., 2006][Li and Zhou, 2009] is a QoS upper ontology which specifies details of quality aspects about services in WSMO framework. A new class called QoS is created as a subclass of *nonfunctionalProperties* defined in WSMO. Moreover, this QoS Class can be attached to *Web Services* and *Goal*.

3.2.3 SAWSDL

Semantic Annotations for WSDL and XML Schema (SAWSDL) [Kopecký et al., 2007] intends to close the gap between Web Services and semantic web by specifying semantics for WSDL components using two annotation mechanisms: *modelReference* and *schema Mapping*. A *modelReference*, independent of any semantic technologies, specifies the association between WSDL components and concepts in semantic models which can be identified via URIs. The transformation from XML schema to an element of a concept and its reversion transformation are defined in *schemaMapping* using *liftingSchemaMapping* and *loweringSchemaMapping* respectively.

Different to OWL-S and WSMO which are complex top-down conceptual models for semantic annotation, SAWSDL defines a simple bottom-up schema extension of WSDL. SAWSDL itself does not provide specific service ontologies for annotation. It is independent of any concrete types of semantic models. Data semantics can be attached by WSDL message along with lifting and lowering schema mapping to enable data exchange between XML schema and semantic concepts. Operational semantics in SAWSDL has two levels of granularity, namely *categorization* and *capability*. *Categorization* aims to show which category the service belongs to, while *capability* specify preconditions and effects of a service. *ModelReference* are used to annotate these two kinds of semantics by referring to a concept in a semantic model. Execution semantics can be annotated either from WSDL service by explicitly describing the service's behaviour or from WSDL operation's *capability*. The QoS semantics can be annotated in WSDL service by pointing to an external QoS model.

3.2.4 WSMO-Lite

The Lightweight Web Service Modeling Ontology (WSMO-Lite) [Kopecký and Vitvar, 2008] is the next evolutionary step after SAWSDL. In SAWSDL there is no explicit mention of precondition and effects that are strongly supported in OWL-S and WSMO [Martin and Domingue, 2007] [Chabeb and Tata, 2008]. The reason behind this problem is because SAWSDL only provides the groundings for a bottom-up approach allowing adopting various solutions to semantic annotation. WSMO-Lite intends to fill the SAWSDL annotations with a subset of WSMO ontology.

Similar to SAWSDL, data semantics is attached to the message's part inside the types section with two types of annotations, namely *reference annotations* and *transformation annotations* corresponded to *modelReference* and *schemaMapping* in SAWSDL respectively. Although

⁹ See <http://www.dublincore.org/> for details.

operational semantics can be attached also with annotations of *categorization* and *capability* which is similar to SAWSDL, WSMO-Lite further precisely defines *capability* by adapting a subset of WSMO model to specify *conditions* and *effects* using WSML. In addition, WSMO-Lite extends WSMO with a new class, *FunctionalClassificationRoot* which is functionality taxonomies to describe *categorization* semantics. Instead of explicit representation of execution semantic, WSMO-Lite defines execution semantics through operational semantics, especially the descriptions of *capability*. Such descriptions are able to be transformed into a WSMO *chorography*. QoS semantics can be annotated to WSDL's service part with a reference pointed from a service component to a concrete QoS semantic model predefined using WSMO model. Due to such adaptation, the limitations of QoS annotations in WSMO are inherited to WSMO-Lite.

3.3 Matchmaking Methods for Service Discovery

Given the service semantics, how to recognize the semantically related services is one of the main issues of Web Service Composition (WSC). This issue can be seen as a variant of Web Service Discovery Problem (WSD). Supposed that we have a set of Web Services S and a service query $q(A, R)$ where available inputs A and requested outputs R are defined. WSD is to find a set of matched services S' , where $S' \subseteq S$. In the case that if there is no service to be found, this problem transforms to a WSC problem. The query q is split into two parts: $q(A, R) = q_1(A, X) + q_2(X, R)$. WSC intends to firstly find a set of services $S' \subseteq S$ satisfying with q_1 . And then q_2 is updated with effects of S' , $q_2(\infty, R) \Rightarrow q_2(S'_{out}, R)$. The same procedure will be repeated until the goals are reached. That means a WSC problem can be transformed to a series of WSD problems with continually changing queries. Therefore the semantic matchmaking in WSD can be adopted to the matchmaking in WSC. In this chapter, a series of approaches available for WSD will be illustrated.

3.3.1 Logic based Matchmaking

Logic based matchmaking is intended to automatically understand the semantics between a service s and a given query q by means of deductive reasoning on annotated ontological concepts. To determine the degree of matching, computing similarity is used in majority of current semantic service matchmakers. Similarity can be calculated either in a concept level or in a service level. In the first case, the similarity is computed for each I/O concepts of a service individually. Alternatively, as to the similarity in a service level, it considers a service as a whole.

OWLSM [Jaeger et al. 2005] is an example of computing concept-oriented service similarity. It supports the logic matching of service inputs, service outputs, services category and user-defined service matching criteria based on OWL-S services. The discovered services are ranked by aggregating these four matching results with user-defined weights. To classify semantic relations for the logic matching, it defines four relationships between two concepts: *fail* ($c_1 \not\equiv c_2$), *unknown* ($c_1.category \not\equiv c_2.category$), *subsumes* ($c_1 \supseteq c_2$) and *equivalent* ($c_1 \equiv c_2$). To distinguish the degree of matching, four numerical values are assigned to each of them. It is defined as:

$$Sim(c_1, c_2) = \begin{cases} 0, & \text{if } c_1 \not\equiv c_2 \\ 1, & \text{if } c_1.category \not\equiv c_2.category \\ 2, & \text{if } c_1 \supseteq c_2 \\ 3, & \text{if } c_1 \equiv c_2 \end{cases} \quad (2-1)$$

Here, *fail* means these two concepts have no relation with each other. If the used categorization is not supported by the matching algorithm, the relationship is *unknown*. The relationship of *subsumes* indicates concept c_1 is more general concept than c_2 . The relationship of *equivalent* shows these two concepts are identical.

OWLS-MX [Klusck et.al, 2009a], WSMO-MX [Klusck and Kaufer, 2009] and SAWSDL-MX [Klusck et al., 2009b] adopt the service-level similarity concepts. Specifically, they define five different matching filters to determine the degree of matching. They are exact match, plug-in match, subsumes match, subsumed-by match, logic-based fail, nearest-neighbor match and fail. Supposed a service S and a request R , S exact matches with R ($Exact(S, R)$), if the service I/O signatures fully match with each other. S plug-in matches with R ($Plugin(S, R)$), if preconditions of S are more general than that of R , and effects of S are semantically the same to or belong to the set of least specific concepts (LSC) of the required effects of R . Subsumes match ($Subsumes(S, R)$) relaxes the matching by allowing more specific effects of S than required in R . As to the subsumed-by match ($Sub_by(S, R)$), it tries to select services whose effects belong to the set of least generic concepts (LGC) of R which are slightly more general than requested. If there is no matching filter above to be found, it returns logic-based fail ($Logic_fail(S, R)$)

$$Exact(S, R) = \{(S, R) | \forall c \in S_{in}, \forall c' \in R_{in}: c \equiv c' \wedge \forall d \in S_{out}, \forall d' \in R_{out}: d \equiv d'\}$$

$$Plugin(S, R) = \{(S, R) | \forall c \in S_{in}, \forall c' \in R_{in}: c \supseteq c' \wedge \forall d \in S_{out}, \forall d' \in R_{out}: d \in LSC(d')\}$$

$$Subsumes(S, R) = \{(S, R) | \forall c \in S_{in}, \forall c' \in R_{in}: c \supseteq c' \wedge \forall d \in S_{out}, \forall d' \in R_{out}: d \sqsupseteq d'\}$$

$$Sub_by(S, R) = \{(S, R) | \forall c \in S_{in}, \forall c' \in R_{in}: c \supseteq c' \wedge \forall d \in S_{out}, \forall d' \in R_{out}: d \in LGC(d')\}$$

$$Logic_fail(S, R) = \{(S, R) | \forall c \in S_{in}, \forall c' \in R_{in}: c \not\equiv c' \wedge \forall d \in S_{out}, \forall d' \in R_{out}: d \not\equiv d'\}$$

These service matching degrees are sorted in terms of the semantic relevance as follows:

$$Exact > Plugin > Subsumes > Sub_{by} > Logical\ Fail$$

3.3.2 Non-logic based Matchmaking

Logic based matchmaking is not sufficient for matching, in the case of some services are not logically similar but syntactically related to a service query. They are based on the technologies from information retrieval (IR) domain.

OWLS-iMatcher [Kiefer and Bernstein, 2008] performs non-logic matchmaking of service inputs and outputs. Rather than defining fixed numerical values to various degree of matching in OWLSM, OWLS-iMatcher computes similarity scores to determine the degree of matching. It logically unfolds I/O concepts of two services, transforms them into two vectors. Then IR based algorithm, such as vector similarity measurements provided by SimPack¹⁰ will be used to calculate similarity score of these two services.

¹⁰ SimPack implements a set of similarity between entities (concepts in ontologies, classes in source code, etc.) SimPack is available at: <http://www.ifi.uzh.ch/ddis/simpack.html>

In OWLS-MX, Nearest-neighbor match ($NNeighbor(S, R)$) is defined for non-logic matching as below. It checks the degree of syntactic similarity ($SynSim$) between S and R .

$$NNeighbor(S, R) = \{(S, R) | SynSim_{IR}(S, R) \geq \alpha\}$$

Here, syntactic similarity is computed mainly with four text similarity measurement: cosine similarity, extended jaccard similarity, the intentional loss of information based similarity and Jensen-Shannon information divergence based similarity [Klusch et.al, 2009a].

Besides those text similarity based methods, in SAWSDL-MX structured graph matching can also been applied for non-logic based matching. More concrete, structured graph algorithm calculates the similarity between two labeled trees which represent WSDL descriptions of service. The comparison starts with the roots which are operational sets of Web Services and moves to nodes which refer to the inputs and outputs data of an operation. Then it traverses to the leaves which are data types of the objects communicated by messages [Klusch et al., 2009b].

3.4 Web Service Composition Approaches

In most of the real applications, it seems impossible to find a single Web Service to satisfy the users' requirement. To this end, combining and coordinating a set of services to provide novel functionalities that was not directly available from the existing services becomes even more indispensable. Due to the multitude and diversity of existing research approaches, two groups of approaches will be examined in this part respectively:

- *Web Service composition using Workflow technique*: approaches that handle the service composition with the help of existing workflow knowledge.
- *AI Planning based approaches*: approaches that model the service composition as an AI Planning problem.

3.4.1 Workflow based Composition

Workflow organization and management have drawn an enormous amount of attention for more than twenty years. The definition of a Workflow given by the **Workflow Management Coalition** (WfMC)¹¹ is as follows:

"The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"

That means a Workflow describes the order of a sequence of tasks performed by different organizations to complete the given requirements. In this context, the process of arranging tasks to form a Workflow is conceptually similar to the composition of Web Services. Therefore, it is possible to share knowledge from Workflow research domain with the

¹¹ WfMC: is a global organization of adopters, developers, consultants, analysts, as well as university and research groups engaged in workflow and BPM. See <http://www.wfmc.org/>

composition of Web Services. The recent work concerning workflow-based composition can be categorized into two classes:

- *Static Workflow Generation*: approaches focusing on static and manual composition.
- *Dynamic Workflow Generation*: approaches aiming to realize automated Web Service composition.

3.4.1.1 Static Workflow Generation

Most research work focusing on the static Workflow generation. Static composition means that the abstract process model including a set of abstract tasks, the underlying data dependency and the control flow is specified manually in graph by domain experts. Then such abstract process is converted to concrete process in the run-time by selecting and binding the appropriate real service to each abstract task. Many Workflow management system and tools are available for this purpose, such as Taverna¹², Kepler¹³, Triana¹⁴, Unicore¹⁵. A detailed survey of these existing systems is presented in [Shumilov et al., 2008]. However, those systems provide the general Workflow management functionalities without taking into account the real-time QoS constraints. In this part, works enabling QoS awareness will be introduced.

In [Ardagna et al., 2007], a service composition framework called **Process with Adaptive Web Services (PAWS)** is presented. It consists of two main components, namely design-time module and run-time module. In the design-time module, domain experts are allowed to create the abstract workflow with BPEL specification which is annotated with global and local constraints involving not only functional requirements but also QoS requirements. The service candidates are retrieved for each abstract task by verifying both functional and QoS constraints. In the run-time phase, the concrete workflow which is built by invoking one candidate service for each task is then executed by the BPEL engine. A framework of a QoS-aware WfMS from GridCC Project presented in [Guo et al., 2007] shares the similar idea. The system is allowed by users to produce a BPEL4WS document for the abstract workflow along with a QoS document containing a set of QoS constraints linked to the corresponded BPEL activities. Although the ideas are useful, no implementation is introduced in both works.

More recent works in this domain mainly focus on the selection of the most appropriate service with acceptable quality. Given an abstract workflow, the problem of QoS-Aware selection of services for each sub-task is modeled as an optimization problem. In [Jafarpour and Khayyambashi, 2010], the harmony search algorithm which is meta-heuristic evolutionary optimization algorithm is applied for this purpose. In [Zhang et al., 2010], authors illustrate a new version of **Ant Colony Optimization (ACO)** algorithm, so-called **MO_ACO** to optimize the service composition. The optimization is realized in two steps. First, an ant system is generated by decomposing the abstract Workflow into a set of parallel execution paths. Then **MO_ACO** is invoked in such ant system where each execution path is modeled as a multiple-objective vector. In [Syu et al., 2011], the selection of the best service in the design time is with

¹²Taverna is an open source and domain-independent Workflow Management System created by the my Grid team. See <http://www.taverna.org.uk/>

¹³Kepler is a java-based open source scientific workflow system. See <https://kepler-project.org/>

¹⁴Triana is an open source problem solving environment developed at Cardiff University. See <http://www.trianacode.org/>

¹⁵Unicore is an OGSA based Grid middleware system supporting Workflow Management. See <http://www.unicore.eu/>

the help of genetic algorithm. Authors point out that the approach guarantees that each selected service corresponding to user's requirements is globally fulfilling transactional and QoS request without asking for pre-defined workflow and possible termination states.

3.4.1.2 Dynamic Workflow Generation

Dynamic workflow generation attempts to create the abstract workflow and select the appropriate services automatically.

A two layered semantics-based dynamic service composition framework is described in [Fujii and Suda, 2009]. In the first layer, an approach called **Component Service Model with Semantics** (CoSMoS) is developed to model both function and semantics of components and the user's requirements. Then the modeled requests are sent to **Semantic Graph based Service Composition** (SeGSeC) where an execution path of the requested service is generated by discovering and interconnecting components based on the request and the components' semantics. In addition, to ensure that the generated execution path matches the user's request, a semantic matching procedure is executed.

In [Chiu et al., 2008], a framework for incorporating QoS in a dynamic workflow system is presented. A sequence of workflow candidates are created in workflow enumeration procedure which is essentially a depth-first traversal of all services with intermediate pruning. The selection of the best workflow is done by evaluating the QoS constraints of each candidate.

Eduardo et al. define in [Eduardo et al., 2009] [Eduardo et al., 2011] a framework called **Dynamic Composition of Service** (DyamiCoS) that aims at supporting service composition on demand and at runtime. Different to the time-consuming search algorithm applied in the approach presented above, to perform service composition, DynamicCoS provide a backward search strategy in a **Causal Like Matrix** (CLM) where all possible semantic connections of services with the corresponding similarities are stored. Though the use of the CLM helps to reduce the number of interactions with the service discovery procedure, it might be not efficient when the number of discovered service increases, since the execution time required to generate the CLM will be exponentially increases.

3.4.2 AI Planning based Composition

Web Service composition can be built either manually or automatically. In the manual approaches, each service is chosen and combined by domain experts relying on some GUI-based software to facilitate the composition process using Workflow techniques introduced above. However, it is error-prone and time-consuming task. It is not appropriate for large-scale problems. On the contrary, in automated Web Service composition, intelligent agents are enabled to select and build the multiple services chain. Referred to the approaches belonging to dynamic workflow generation, we noted that one of the main deficiencies of the current solutions for automation of the composition is the way the workflow is composed. Recently, the **Artificial Intelligence** (AI) planning techniques have been proposed by the most of efforts as a good way for the automated composition. In the rest part of this section, the updated automated composition approaches will be presented.

Inspired by the work in [Ghallab et al., 2004] and [Chan et al., 2006], recent AI techniques can be classified into the following four categories:

- *Classical planning* involves searching a state-space or a plan-space in order to find a series of actions transferred from an initial state to a goal state.
- *Non-classical planning* including *neoclassical planning* extends the classical notion of planning with some techniques, such as graph-based planning and constraint satisfaction, etc. *Heuristics and control strategies* utilize heuristic to manage the searching process and control rules, etc.

In the following sub-chapters, the related techniques in each category will be introduced in details.

3.4.2.1 Classical Planning

Classical planning approaches aim to search useful operations to achieve the desired goal through:

- *State-space planning*
- *Plan-space planning*

More formally, a state-space consists of a finite set of states $S = \{s_0, s_1, \dots, s_n\}$, a finite set of actions $A = \{a_0, a_1, \dots, a_n\}$ and a transition function $s_{n+1} = f(a_n, s_n)$ which defines how actions are transformed from one state to another. The state-space planning algorithms aim to search through the space of those possible states S for the path that can fulfill the goals. The solution of a state model π is a sequence of actions.

In [Sheshagiri et al., 2003], the authors attempt to present a logic based planner for DAML-S services which is the predecessor of OWL-S. The ServiceModel description of each DAML-S service is converted into STRIPS-style Verb-Subject-Object triples with **Java Expert System (JESS)**¹⁶. These notations corresponding to the services act as actions in the state-space. Given the goal and those available actions, their composition engine aims to solve a planning problem by adding the useful services to the plan which satisfy the existing goal. Goals will be then updated with preconditions and inputs of newly added services. This process will continue until there is no matched service for goals.

As for a Plan-space, it is composed of a set of actions $A = \{a_0, a_1, \dots, a_n\}$, a set of ordering constraints $O = \{o_0, o_1, \dots, o_k\}$ which define the order of the execution of those available actions, a set of variable binding constraints on the parameters of actions denoted as $B = \{b_0, b_1, \dots, b_k\}$ and a set of causal link L keeping track the connection information available in current plan. Different from the state-space planning algorithms, the plan-space planning search the space of partial plans which are set of unordered actions.

Peer [Peer, 2005] illustrates a plan-space based algorithm which improves the plan search with feedback gained from plan execution for the automatic Web Service Composition. To that end, the semantic annotation of service is translated into the **Planning Domain Definition Language (PDDL)**, an effort to standardize planning domain and problem description language. The algorithm begins to build the plan through those available partial plans in terms of initial states. To reduce the searching time, a runtime execution monitoring engine is embedded during the execution. Once the engine detects that there is no matched causal link to the goal, the planner will be terminated and move to the other related partial plans.

¹⁶ JESS is a rule engine for the Java platform. It is available from at <http://www.jessrules.com/jess/>.

3.4.2.2 Non-classical Planning

Techniques using in neoclassical planning extend the classical notion of planning. The most common techniques are Planning-Graph where graph structures has been utilized for the construction of all possible states and transitions and Constraint Satisfaction where the planning problem is expressed as a reasoning problem to find a suitable model satisfying all constraints.

Mathematically, a Planning-Graph is a directed layered graph consisting two types of nodes, namely action nodes A which are available actions and proposition nodes P which refer to existing states. The Planning-Graph is built by arranging these two types of nodes in alternating layers. It starts with proposition nodes in the initial level followed by layers of action nodes, and so forth. The solution of this planner is a sequence of sets of actions denoted as $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$. Here, each π_i is a subset of the actions nodes corresponding to each layer in the graph.

In [Wu et al., 2007], the authors present a Planning-Graph based on SAWSDL services. Unlike the classic Planning-Graph algorithm where only preconditions and effects of an action are taken into account, in their work data types of input/output message of actions are considered as well. That means the action can be added into the graph if and only if both its preconditions and data type of input message satisfying with current state. A recent planner WSC-WPG proposed in [Li et al., 2010] adds a weight which is the match degree of two concepts into the traditional Planning-Graph. The graph expansion can be improved by considering weight information. The action with smaller value of weight will be replaced by the action in the same level with the higher weight value.

The idea behind the Constraint Satisfaction is to find a solution satisfying all stated constraints. More formally, a **C**onstraint **S**atisfaction **P**roblem (CSP) is defined as a tuple (V, D, C) , where $V = \{v_1, v_2, \dots, v_n\}$ is a set of variables; D is a function that assigns possible values to each variable according to a specific domain $D(v_i)$; $C = \{c_1, c_2, \dots, c_m\}$ is a set of constraints which is a relation over variables specifying the valid combinations of value assignments to these variables.

Recently, a CSP based planer called GCSP is presented in [Mayer et al., 2009]. It extends the traditional CSP model by lifting constraints and variables to generic constraints and thereby overcoming the disadvantage of the traditional CSP where the number of services which is normally infeasible in real applications should be pre-specified. The composition process in GCSP is a two-phase algorithm. The first phase is called specification where service profiles, process constraints from associated domain knowledge and the composition goals as well must be translated into generic constraints. In the following composition phases, the initial constraint problem is extended to incorporate additional services until a consistent solution has been found that satisfied all generic constraints. The solution is expressed as a workflow in BPEL.

The planning algorithms belonging to Heuristics and control strategies employ heuristics to determine where to search next by estimating the usefulness of the alternative actions a planner can chose from. These algorithms include **H**ierarchical **T**ask **N**etwork (HTN) planning, where the desired task is decomposed into subtasks and such decomposition will be recursively applied until all sub-tasks can be directly performed using the planning operators.

HTN planning uses action and state description like the concepts in State-space planning. However the different definition is defined. In HTN planning, a planning domain is specified by a tuple (d, I, Op, Me) . Here d is the task network involving primitive and compound tasks. Primitive tasks are tasks that can be accomplished by a single operator while compound tasks need to be decomposed into smaller task using methods. I is initial states listing all propositions that are true. Op is a set of operations similar to PDDL operations indicating how to execute a primitive task. Me is a set of methods specifying how to decompose a compound task into a task network.

OWLS-Xplan introduced in [Klusch et al., 2005] allows for fast and flexible composition of OWL-S services with HTN planning. It takes three inputs: a set of OWL-S services, domain ontologies and a description of query and returns a sequence of services that fulfill the goal. To that end, a two-layered architecture is designed. In the first layer, those inputs mentioned above need to be converted to PDDL problem and domain descriptions. These PDDL descriptions are then used in the second layer to create plan that satisfies the goal. In addition, consider that in some case there is no decomposition method available, OWLS-Xplan integrates the also the Planning-graph algorithm which guarantees to find a solution if it exists in the state-space. A more recent work in [Xiao et al., 2010] presents a modified HTN-based planner for the composition of OWL-S Services, which enhances the traditional HTN planner with domain knowledge.

Chapter 4

QWSMO-Lite: A QoS-Aware Ontology for Web Service Annotation

“Rather than looking for a clear winner among various SWS approaches, I believe that in the post-SAWSDL context, significant contributions by each of the major approaches will likely influence how we incrementally enhance SAWSDL. Incrementally adding features (and hence complexity) when it makes sense, by borrowing from approaches offered by various researchers, will raise the chance that SAWSDL can present itself as the primary option for using semantics for real-world and industry-strength challenges involving Web services.”

- Prof.Dr.Amit.P.Sheth

The use of services without considering their underlying semantics can negatively affect composition processes by raising intermittent failures or leading to a slow performance. Recently, a new framework called WSMO-Lite has been defined for semantic Web Service annotation. WSMO-Lite, in turn is an integration with SAWSDL and WSMO which are two latest the World Wide Web Consortium (W3C) standards in this paradigm. However, due to the lack of quality of services descriptions, it is hard to determine the appropriate service among those functionally similar services. To this end, in this chapter we present QWSMO-Lite specification which extends WSMO-Lite with ontology for modeling QoS properties. Such specification is a foundation for the service discovery and ranking presented in Chapter 5 as well as the service composition approach described in Chapter 6.

CONTENTS

4	QWSMO-LITE: A QOS-AWARE ONTOLOGY FOR WEB SERVICE ANNOTATION.....	25
4.1	REQUIREMENTS ANALYSIS FOR SERVICE DESCRIPTION	26
4.2	QWSMO-LITE FRAMEWORK	27
4.3	ONTOLOGIES IN QWSMO-LITE	28
4.3.1	Modeling QoS Semantics in QWSMO-Lite.....	29
4.3.2	Specification of Domain Semantics	30
4.4	FORMALIZATION OF QWSMO-LITE.....	31
4.5	SUMMARY	33

4.1 Requirements Analysis for Service Description

In this section, the most important requirements of adding semantics to Web Services compared to the existing solutions introduced in Section 3.2 are stated.

As to the semantic model which can be used for annotations, the requirements are analyzed from four perspectives according to the classifications used to evaluate existing solutions. They are *data semantic*, *operational semantics*, *execution semantics* and *QoS semantics*.

From the *data semantic perspective* it is clear that many challenges for Web Service composition arise from their lack of understanding of the underlying data that the service exchanges. Therefore to better understand and reuse a service, the interface of functionalities should be annotated with the terms from domain ontologies.

Taking into account the *operational semantics perspective*, the service description should be able to describe the categorization information of the functionalities. Since for the service composition, this kind of semantics enables to reduce the searching space by only considering the services falling into the required category. Moreover, to successfully invoke a service, the conditions of inputs should also be taken into account. In addition, the effects which define the state after the service invocation are useful to find succeeding services to build a composition chain. Consequently, these three kinds of semantics should also be annotated in service's operations.

The requirements of *execution semantics* are concerned about how to externally and internally consume the Web Service. The semantics of external invocation can be inferred from operational semantics. The semantics of internal invocation is described as a workflow indicating how the functionalities from other services are composed. Such static semantics has some limitations. For instance, it only can be provided by the users who have pre-knowledge over all underlying services. And such static semantic doesn't adjust to the dynamic change of the environments. Therefore, this kind of knowledge is not of interest to the automatic service composition.

Another important issue that we should consider is to annotate services with QoS semantics. First of all due to the fact that the requirements of QoS vary with individual domains, the annotation of QoS model should be able to adapt for different domains. Secondly, QoS semantics is a multi-dimensional property as depicted in Figure 3-1. The same quality dimensions are represented using different metric measurements in most cases. To evaluate and measure all provided qualities, metric should be specified in QoS model. Moreover, to accurately interpret such metric measurement, additional information about various types of unit and data type should be described. In additions, the system should also support the annotation of the QoS priority for each quality detention, since QoS properties often have different important levels in terms of different service users.

After the comparison and evaluation of a set of existing solutions, we can figure out that WSMO-Lite is a good candidate for semantic annotation of Web Services. It adopts the idea from SAWSDL which defines the simple extensions for WSDL and XML Schema by referring the WSDL components to arbitrary semantic descriptions. Moreover, it tries to enhance SAWSDL with concrete lightweight service ontology from WSMO. However, due to the lack of quality of services descriptions, it is hard to determine the most appropriate service among those functionally similar service candidates. Therefore, we define QWSMO-Lite specification

which enhances WSMO-Lite with ontology for modeling QoS properties. These annotated semantics will be exploited for enhancing the automatic Web Service composition.

4.2 QWSMO-Lite Framework

In this chapter, QWSMO-Lite service ontology is presented in details. We adopt the base model of WSMO-Lite which fills SAWSDL annotations with concrete lightweight WSMO ontologies. The usage of WSMO-Lite based description makes users easier for users to annotate WSDL in SAWSDL framework. The mechanism only requires that the concepts in the semantic models can be identified by URIs. Semantic models using in WSMO-Lite are defined using a subset of WSMO which complements SAWSDL with explicit meaning of the particular service annotations. In WSMO, the quality aspects are part of the non-functional information of a Web Service description and are simply defined as: *Accuracy, Availability, Financial, Network-related QoS, Performance, Reliability, Robustness, Scalability, Transactional and Trust*. However, such QoS definition is neither expressive nor flexible enough for QoS properties to distinguish functionally similar services or operations for service discovery and composition [Wang et al. 2006]. Moreover, annotation methodology provided in WSMO-Lite also needs to be evolved and refined. For instance, in WSC the annotation of operations plays a more crucial role than others. Nevertheless, there is no available non-functional description attached to the operation elements in WSDL. In addition, WSMO-Lite allows services elements to be annotated with functional capabilities, which makes no sense for those services consisting of more than one operation with various functionalities. To solve these problems, we develop a QWSMO-Lite, an extension of WSMO-Lite. The key characteristics of QWSMO-Lite are summarized in Figure 4-1:

- **Extensible QoS ontology.** Inspired by the work presented in [Li and Zhou, 2009], a new ontology depicted in the figure with *QoSParameters* are extended to the WSMO-Lite vocabulary consisting of a set of generic and domain specific QoS dimensions for both service requesters and service offers.
- **Domain specific annotations in the operational level.** Noted that both functional semantics and QoS semantics differ from one domain to another, we should distinguish such domain specific knowledge from general service descriptions. Considering the nature of a WSC problem which is to find linked operations among services, the annotations of those domain specific operation elements are viewed as more important than the others. Therefore we attach operation elements in WSDL with domain knowledge consisting of *DomainFunctionalClassificationRoot* which refers to the category of the underlying operations and *QoSParameters* which specifies domain specific QoS criteria.
- **Generic annotations at service level.** The annotation at service level should provide general descriptions of functional and non-functional attributes of the underlying operations. *FunctionalClassificationRoot* is a function annotation representing an abstract type of the service. Here, we distinguish the single domain services consisted of operations belonging to one single domain from the multi-domain services whose operations belong to multi-domains. Such information will be beneficial to reduce the search space in the service discovery procedure. The details will be discussed in the

next chapter. In addition, *NonfunctionalParameter* with nonfunctional semantic, such as the service provider's information can also be linked to the service elements.

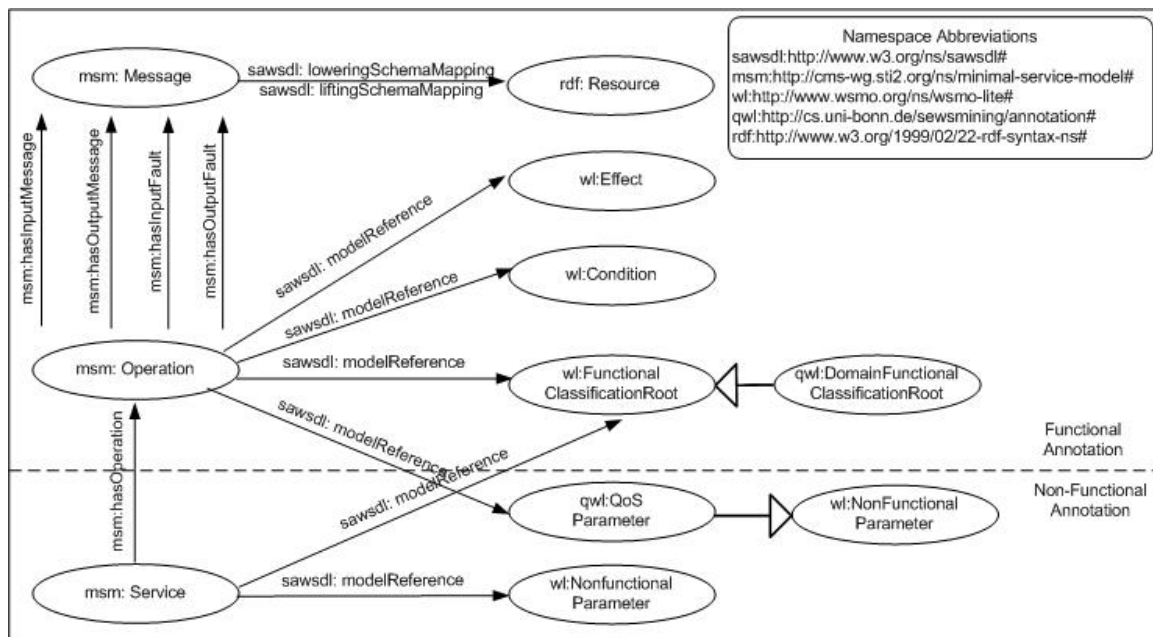


Figure 4-1 QWSMO-Lite framework

4.3 Ontologies in QWSMO-Lite

Briefly speaking, QWSMO-Lite extends the WSMO-Lite specification with two ontologies:

- *QoSParameter* representing QoS semantics
- *DomainFunctionalClassificationRoot* specifying the underlying domain information.

In this section, these two extensions are presented in details. In order to better illustrate how to annotate WSDL with QWSMO-Lite we take a real Web Service for global weather¹⁷ as a running example. Table 4-1 shows the general description of the service.

Table 4-1 A global weather service

Function	Inputs	Outputs
GetWeather	CountryName(String)	GetWeatherResult(String)
	CityName(String)	
GetCitiesByCountry	CountryName(String)	GetCitiesByCountryResult(Sting)

¹⁷ This Web Service is based on WSDL1.1.

It is available from <http://www.webservices.com/globalweather.aspx?WSDL>

4.3.1 Modeling QoS Semantics in QWSMO-Lite

Inspired by the work shown in [Li and Zhou, 2009], QoS ontology named *QoSParameter* was developed as an extension of the WSMO-Lite framework. Unlike Li's approach which only deals with some generic qualities of the services, *QoSParameter* aims to address domain specific qualities as well. Moreover, in order to facilitate extensibility and reusability, *QoSParameters* has been designed to be a modular in nature. Each modular can be easily extended as a normal ontology. Such ontology falls into three layers shown in Figure 4-2.

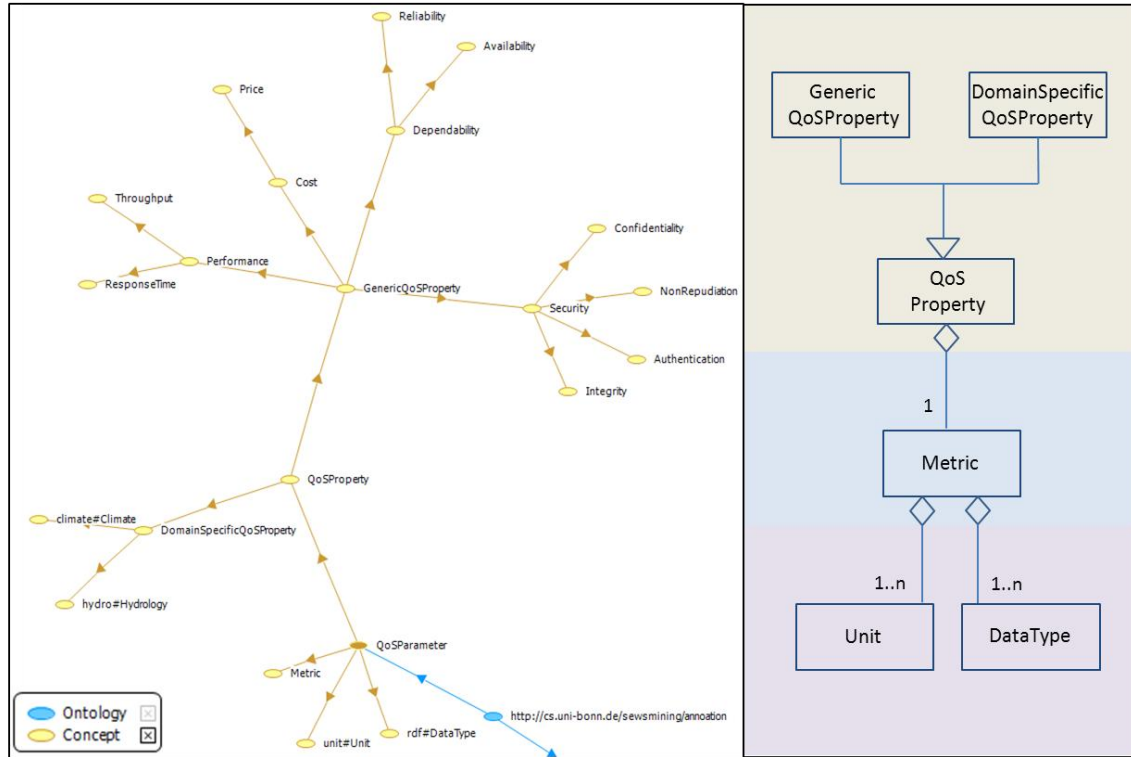


Figure 4-2 Metadata model of *QoSParameter*

QoSProperty is an ontology for modeling service quality dimension in *QoSParameter*. To make a clear distinction between the generic QoS semantics and domain specific quality information *QoSProperty* is split into two parts: *GenericQoSProperty* containing generic quality criteria, such as the elements shown in Figure 3-1, which is applicable to all services and *DomainSpecificQoSProperty* relating to domain specific quality attributes. In some domains, such generic attributes are not sufficient. For instance, to get weather information, users are interested in services with high accessibility and accuracy. And such qualities are domain specific requirements which can be obtained from *DomainSpecificQoSProperty*. Adding a new domain-specific ontology can be easily realized by importing ontologies to the framework.

Considering that there is no standard means to measure quality dimensions, even the same dimensions are probably evaluated in different ways by different service providers. For example in global weather service, the response time is recorded using millisecond, while in the requirements document the response time might be measured by second. Therefore, in QWSMO-Lite, each *QoSProperty* has an associated *Metric* ontology. Such *Metric* ontology is

defined to recognize these mismatches by specifying the *Unit* and *DataType* the providers used to measure their qualities. These two ontologies can be easily extended by importing new ontologies to a framework without changing any existing values. In addition, each QoS property may have different important level in various use cases. Accordingly, in *Metric*, it enables to assign the case-specific weight values to the corresponding QoS qualities. Now four levels of important are supported, namely:

$$weight \in \{\text{"strong"}, \text{"less strong"}, \text{"weak"}, \text{"very weak"}\}$$

Ontologies defined in QWSMO-Lite will be helpful to formulate concrete service descriptions tailored to annotate Web Services. Taken the global weather service as an example, Figure 4.3 shows the snippet from its service description ontology which contains QoS specification with a flavor of WSM. QoS descriptions for the individual domain named *DomainQoS* and the generic characteristics denoted as *GenericQoS* are specified separately. Such structure facilitates the annotation process on WSDL level. As depicted in Figure 4-1, the *DomainQoS* needs to be referred to the operational elements of WSDL while *GenericQoS* is used for service level annotation. Both *GenericQoS* and *DomainQoS* are instantiated as instances filling the required attributes with a set of case-specific values in the service descriptions. Note that such QoS description should be available in both service providers and service consumers. To advertise a service, the QoS values are collected mainly through active monitoring by service providers. On the other hand, it also enables the service consumers to customize their specific QoS values.

```
concept DomainQoS subConceptOf QoS
concept GenericQoS subConceptOf QoS

instance ResponseTimeMetricInstance memberOf { GenericQoS, qwl#Metric }
{
  qwl#hasUnit hasValue "milliseconds"
  qwl#hasDataType hasValue "long"
  rdf#hasWeight hasValue "Less Strong"
  qwl#hasValue hasValue "2000"
}

instance ResponseTimeQoSInstance memberOf { GenericQoS, qwl#GenericQoSProperty }
hasMetric hasValue ResponseTimeMetricInstance
...

instance DomainMetricInstance memberOf { DomainQoS, qwl#Metric }
{
  qwl#hasUnit hasValue "percentage"
  qwl#hasDomainQoS hasValue "Accessibility"
  rdf#hasWeight hasValue "Strong"
  qwl#hasDataType hasValue "long"
  qwl#hasValue hasValue "89,65"
}

instance DomainQoSInstance memberOf { DomainQoS, qwl#DomainSpecificQoSProperty }
hasMetric hasValue DomainMetricInstance
...
```

Figure 4-3 An example of the QoS specification in QWSMO-Lite

4.3.2 Specification of Domain Semantics

In QWSMO-Lite, it defines a new class called *DomainFunctionalClassificationRoot*, stating the taxonomy of a specific domain. As shown in Figure 4-4, in climate domain, a concept of *Climate* stands for a class of all climate-related terminology. Concepts can be put in a hierarchy by means of the *subConceptOf* construct. For instance, *Weather* is a subclass of *Climate*, meaning that any weather information is part of climate knowledge and it is allowed to have its own

attributes. This ontology is used to point to operational elements in WSDL. A list of all referred domain ontologies can be applied to service components in WSDL.

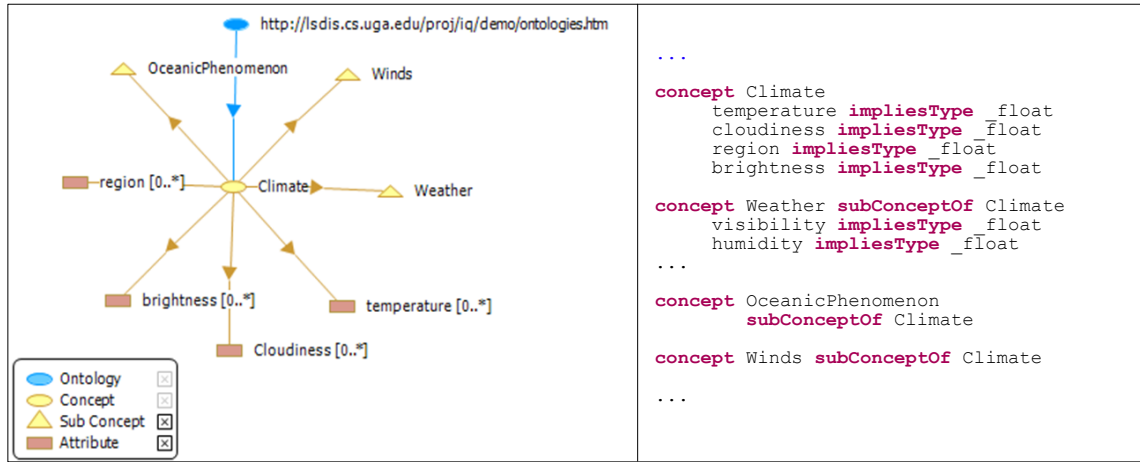


Figure 4-4 Ontology of *DomainFunctionalClassificationRoot*

4.4 Formalization of QWSMO-Lite

In this subsection, we discuss how we can annotate WSDL with those mentioned ontologies in QWSMO-Lite framework. A running example for the annotation is illustrated along with the definition of each element in an annotated service.

Definition 4-1 (Annotated Function). An annotated function of a Web Service is described as a tuple $f = (C_{in}, C_{out}, pre, eff, cat, q)$ with

- $C_{in} = \{c_1^{in}, \dots, c_n^{in}\}$: is a set of concepts referred to input parameters of WSDL,
- $C_{out} = \{c_1^{out}, \dots, c_m^{out}\}$: is a set of concepts referred to output parameters of WSDL,
- pre is the precondition of the function f ,
- cat is the category of the function f ,
- eff is the effect of the function f ,
- q is the sequence of QoS dimensions. ■

In QWSMO-Lite, data semantics represented as a set of inputs and outputs concepts can be used to describe XML messages in WSDL. If necessary, the transformation which enables the match between the schema and the underlying XML data can also be described. Figure 4.5 shows a data semantics annotation example for the global weather service.

```

<s:sequence>
  <s:element maxOccurs="1" minOccurs="0" name="CityName" sawsdl:modelReference="http://daml.umbc.edu/
  ontologies/ittalks/address#city" type="s:string"/>
  <s:element maxOccurs="1" minOccurs="0" name="CountryName" sawsdl:modelReference="http://daml.umbc.edu/
  ontologies/ittalks/address#country" type="s:string"/>
</s:sequence>

```

Figure 4-5 An example of data semantic annotation

Functional and Executional semantics are provided in preconditions, effects and category information of an operation. Unlike the annotation defined in WSMO-Lite, preconditions and effects in QWSMO-Lite contain not only the axioms that need to be satisfied to successfully invoke a Web Service, but also describe the required domain ontologies to invoke a service and available domain information after the invocation respectively. The reason behind this mechanism is that when composing services, such information facilitates the automation of finding preceding and succeeding operations by confining the search spaces to the domains described in the annotation file. Figure 4-6 illustrates an example for annotating *GetWeather* operation available in the global weather service.

Service Description Ontology	Web Service Description
<pre> concept Precondition subConceptOf wl#Condition hasDomain impliesType _"http://dam1.umbc.edu/ ontologies/ittalks/address#address" isAvailable impliesType _boolean nfp dc#relation hasValue {IsAvailable} endnfp </pre>	<pre> <wsdl:operation name="GetWeather"> <sawSDL:attrExtensions sawSDL:modelReference="template#Precondition"/> </pre>
<pre> concept Effect subConceptOf wl#Effect hasDomain impliesType _"http://lstdis.cs.uga.edu/proj/iq/demo/ ontologies.htm#weather" </pre>	<pre> <sawSDL:attrExtensions sawSDL:modelReference="template#Effect"/> </pre>
<pre> concept category subConceptOf qwl#DomainFunctionalClassificationRoot hasDomain impliesType _"http://lstdis.cs.uga.edu/proj/iq/demo/ ontologies.htm#weather" </pre>	<pre> <sawSDL:attrExtensions sawSDL:modelReference="template#Category"/> </pre>
<pre> concept DomainQoS subConceptOf QoS ... axiom isAvailable definedBy ?country memberOf address#Country and ?x[isAlive hasValue _boolean("true")] :- ?country[hasName hasValue "Germany"]. </pre>	<pre> <sawSDL:attrExtensions sawSDL:modelReference="template#QoS"/> ... <wsdl:input message="tns:GetWeatherHttpGetIn"/> <wsdl:output message="tns:GetWeatherHttpGetOut"/> </wsdl:operation> </pre>

Figure 4-6 Description of an annotated function

Definition 4-2 (Annotated Web Service). An annotated Web Service of a Web Service is described as a tuple $w = (F, NF, cat_w)$ with

- $F = \{f_1, \dots, f_n\}$ is set of annotated functions as defined above,
- $NF = \{nf_1, \dots, nf_n\}$ is a set of generic non-functional descriptions for w ,
- cat_w is a list of domains that a service contains. ■

The description of an annotated Web Service is shown in Figure 4-7. Similar to WSMO, the non-functional properties are represented using Dubin core. Unlike the category information referred to functional elements, here cat_w is a collection of ontologies used to annotate underlying operations.

Service Description Ontology	Web Service Description
<pre> ontology - "http://cs.uni-bonn.de/annotationTemplate" nonFunctionalProperties dc#title hasValue "QWSMO-Lite Template ontology" dc#date hasValue _date(2011,02,01) dc#format hasValue "text/html" ... endNonFunctionalProperties concept Categories subConceptOf qwl#DomainFunctionalClassificationRoot hasDomain impliesType { "http://lsdis.cs.uga.edu/proj/iq/demo / ontologies.htm#weather", "http://dam1.umbc.edu/ontologies/ itTalks/address#" } ... </pre>	<pre> <wsdl:service name="GlobalWeather"> <sawSDL:attrExtensions sawSDL:modelReference="template"/> <sawSDL:attrExtensions sawSDL:modelReference="template#Categories"/> <wsdl:port binding="tns:GlobalWeatherSoap" name="GlobalWeatherSoap"> <soap:address location= "http://www.webservicex.com/globalweather.asmx"/> </wsdl:port> </wsdl:service> </pre>

Figure 4-7 Description of an annotated Web Service

4.5 Summary

QWSMO-Lite is an extension of WSMO-Lite devoted to provide expressive representation of semantics on the one hand, and facilitate the annotation task and the automation of Web Service composition on the other hand. The technique comparison of related approaches is presented in Section 7.1. To sum up, the main contributions of QWSMO-Lite are listed as follows:

- **Supports a modular structure of QoS ontology.** Unlike the approaches introduced in Section 3.2, to facilitate reusability and extensibility the QoS ontology has been designed from the beginning to be modular in nature. QoS related semantics including domain specific knowledge, general QoS specifications and measurement semantics for different domains and applications can be easily described in the ontology. In addition, within this model, users are allowed to specify units for each QoS dimension. QoS priority can also be customized by assigning weights. (see Section 4.3.1)
- **Simplifies the representation of semantic information.** Inspired by the minimal Web Service Model in WSMO-Lite, a more simplified and specific version of the Web Service annotation model is defined in QWSMO-Lite. With such model, users become aware of the underlying referred semantics, rather than only list them using SWSDL specification. Moreover, it simplifies the annotation work by annotating services with minimal and most important semantics. (see Section 4.4)
- **Facilitates the automation of Web Service Composition.** A service is annotated with category information containing all involved domain specifications. Meanwhile, each operation is annotated by one specific domain information. The category semantics at service level helps facilitate the service discovery by narrowing the range of candidate services. In addition, unlike the specification of operations' preconditions and effects in WSMO-Lite, in QWSMO-Lite, a set of required and affected domain ontologies are listed in the operations' preconditions and effects parts respectively. This type of semantics indicates the searching ranges of finding the matched proceeding and succeeding services to build a composition chain. (see Section 4.3.2 and Section 4.4)

Chapter 5

SEwsDM: A Multi-Attribute Semantic Matchmaking Engine

“The world can be changed by man's endeavor, and that this endeavor can lead to something new and better. No man can sever the bonds that unite him to his society simply by averting his eyes. He must ever be receptive and sensitive to the new; and have sufficient courage and skill to novel facts and to deal with them.”

- Franklin D. Roosevelt (1882-1945)

An important challenge of Web Service composition is that the system needs to locate, find, select and invoke appropriate services to build a chain. This process is called semantic matchmaking. Two key issues are required to be addressed in this semantic knowledge lifecycle: a semantic matchmaking mechanism for computing matching degree between two single-attribute semantics and a semantic engine for processing multi-attribute semantics. As noted in Section 2.3, most of the current solutions dealing with semantic matchmaking focus on measuring single-attribute semantics. However, in real dynamic environments, especially for the Web Service Composition problem, those individual single-attribute semantics should be taken into account as a whole. Therefore in *SEwsDM* we developed a multi-attribute semantic matchmaking engine including these two engines. Briefly speaking, logic and non-logical reasoning algorithms are adopted to the single attribute semantic engine while the *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS) is applied to calculate semantic score for the multi-attribute semantic matching.

CONTENTS

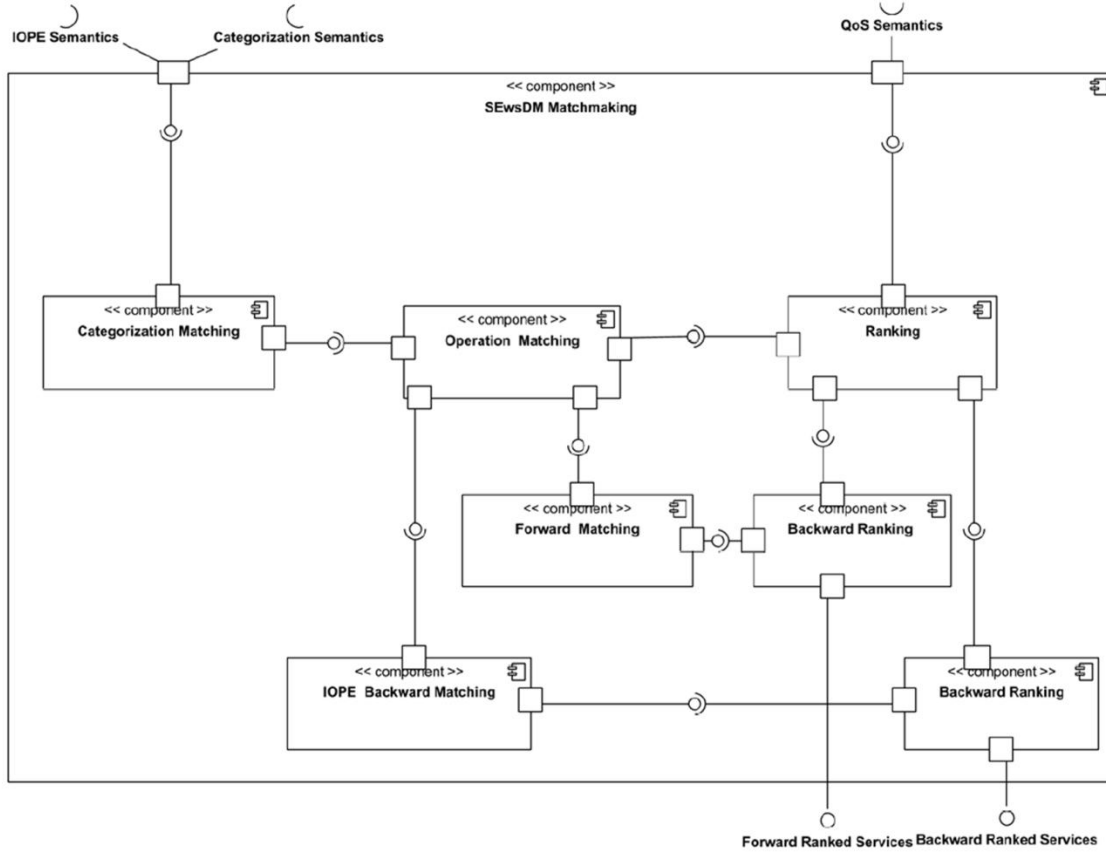
5	SEwsDM: A MULTI-ATTRIBUTE SEMANTIC MATCHMAKING ENGINE	34
5.1	SEwsDM FRAMEWORK.....	35
5.2	SINGLE-ATTRIBUTE SEMANTIC MATCHMAKING	36
5.2.1	Traditional Logical Similarity Measures	37
5.2.2	Ontology based Single Attribute Matchmaking Algorithm	39
5.3	MULTI-ATTRIBUTE DECISION MAKING.....	43
5.3.1	Scoring Methods for Multi-Attribute Decision Making.....	44
5.3.2	TOPSIS based Algorithm for Multi-Attribute Decision Making.....	47
5.4	IMPLEMENTATION OF SEwsDM.....	53
5.5	SUMMARY	56

5.1 *SEwsDM* Framework

Matchmaking is the process of comparing the service request against the available service advertisements by calculating the semantic similarity and finding the most appropriate one. Different to the Web Service Discovery problem which intends to find a service whose parameters satisfy with a service query, in Web Service composition problem, it is devoted to finding a preceding or a succeeding service of the current service using matching mechanism.

A Web Service can be added to the candidate list of succeeding service of the current service, if and only if all the parameters of its functions are satisfied by the current service. On the other hand, a service whose affected parameters are available from the current service can be put into the current service's preceding candidate list. Note that to find the semantically related services, calculating the similarity of the Web Service operations plays a crucial role. That is to say, the services similarity can be measured as an aggregation of the similarity of a series of operations supported in a service. Consequently, in *SEwsDM*, the operational level similarity algorithms rather than a traditional service's level matchmaking are concentrated on.

The semantics management architecture based on the Web Service operations is depicted in Figure 5-1. The matchmaking component comprises three phases. The first phase is to filter the services with *Categorization Matching*. If the annotated categorization information of a service semantically relates to the current service, we mark it as usable. Otherwise, that service remains unusable. All usable services will go to the second phase which is *Operational Matching*. Here, two matching mechanisms are developed to check correspondent operation's parameters for backward and forward matchmaking respectively. Afterwards, QoS specifications offered by this checked service together with the results from the first two phases are aggregated to measure the similarity between these two services in the last so-called *Ranking* process. Due to the various criteria for ranking backward and forward matching services, two ranking algorithms are implemented for these two matching purposes. Those ranked services will be used at a later stage to compose the composition chain. The underlying technical details of the matchmaking will be illustrated in the following subchapters.

Figure 5-1 *SEwsDM* matchmaking framework

5.2 Single-Attribute Semantic Matchmaking

As mentioned in the last chapter, *Categorization Matching* acts as a preprocessing procedure in *SEwsDM* framework. The annotated categorization semantic is the only attribute to be checked. With regard to the *Operation Matching*, to find matched succeeding services of current available service s_a , the annotated outputs of s_a and the inputs of the service candidates need to be concentrated on to calculate the similarity. Similarly, to determine the matched proceeding services of s_a , the similarity between the outputs of a service candidate to the inputs of s_a should be considered. All of these matching methods share one thing in common. Only a single dimensional attribute needs to be taken into account to measure the similarity. We call this kind of matchmaking as single attribute semantic matchmaking. Ontology-based approaches are often adopted for this paradise.

The definition of ontology as a technical term for computer science is provided by Tom Gruber in [Gruber, 2008].

"In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational

primitives include information about their meaning and constraints on their logically consistent application.”

In our system, the underlying semantics of Web Services are expressed as ontologies which are used to annotate services in a QWSMO-Lite flavor. As shown in Figure 5-2, after annotating services, the relevant ontologies are referred to the outputs of available services and the inputs of a service candidate. The degree of match between these two services is calculated with ontology matchmaking algorithms.

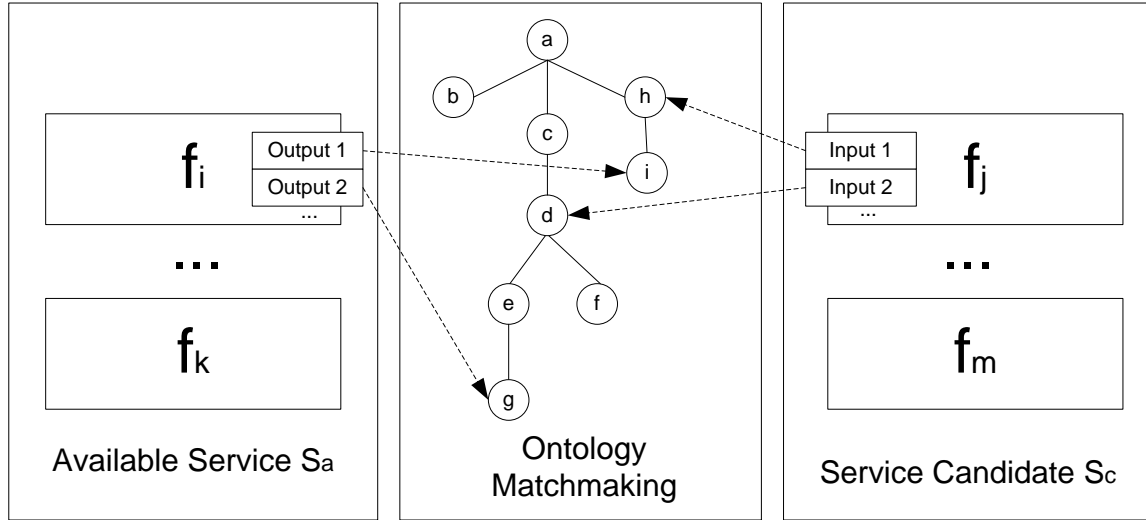


Figure 5-2 Relationship between annotated services and ontology matchmaking

5.2.1 Traditional Logical Similarity Measures

Similarity denotes how similar the two concepts are. Determining the similarity of the two concepts is a crucial issue in Web Service Composition where we need to select the most appropriate service to connect with the current available service. The greater the value of similarity is, the more similar two sets of concepts are. In this section, several traditional notion of similarity is introduced. We classified these notions into three categories:

- **Bag-of-words based similarity**
- **Vector-space based similarity**
- **Hierarchy based similarity**

Let us start with the similarity based on bag-of-words (BOW) which is a set of weighted terms that best describe the entity in Information Retrieval domain [Thiagarajan et al., 2008]. There are many different measures in use, which differ primarily in the way they normalize this intersection value [Rijsbergen, 1979]. Jaccard's coefficient [Doan et al., 2002] and Dice's coefficient belong to this category.

Given two BOW collections X and Y , Jaccard's coefficient is defined as the size of the shared information over the size of the union of these two collections as following:

$$Sim_{Jacc}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (5 - 1)$$

The Jaccard's coefficient is practically applied to define similarity measure in many systems such as the GLUE system which employs this formula for machine learning techniques to find mapping in the ontology model. A methodology of ontology mapping proposed in [Kong et al., 2004] is based on similarity measurement by Jaccard's coefficient as well.

Dice's coefficient is similar to Jaccard's coefficient which is defined as twice the intersection divided by total number of terms in both tested sets.

$$Sim_{Dice}(X, Y) = \frac{2 * |X \cap Y|}{|X| + |Y|} \quad (5 - 2)$$

Both of the above mentioned notions of similarity concentrate on the intersection part of two compared BOW sets. However, the similarity also depends on features that are unique to each set. To this end, Tversky's model [Tversky, 1977] is considered as the most suitable approach to match such semantics which is defined as:

$$Sim_{Tversky}(X, Y) = a * f(X \cap Y) + b * f(X - X \cap Y) + c * f(Y - X \cap Y) \quad (5 - 3)$$

where the model is specified by the features that is common in both sets, those in X but not in Y and those in Y but not in X . Here, a , b and c are parameters that provide for differences focusing on the different components. Such Tversky's model based notion also can be adapted to compute similarity between Web Services. [Cardoso, 2006] introduces the matching functions for Web Service Discovery based on this model, where not only the concept itself but also the unique properties associated with two compared concepts are taken into account to compute the similarity.

In the BOW based solutions, in the case that $X \cap Y = \emptyset$, the similarity might be zero according to the Tversky's model based notion. But some concepts which have no intersection with each other but are siblings should have nonzero similarity.

The other category for expressing similarity is based on vector-space model. Cosine similarity is often be utilized which defines the similarity between two vectors to be the cosine of the angle between them. Formally, this similarity is given by the formula:

$$Sim_{cos}(X, Y) = \frac{\vec{V}(X) \cdot \vec{V}(Y)}{|\vec{V}(X)| |\vec{V}(Y)|} \quad (5 - 4)$$

where $\vec{V}(X)$ and $\vec{V}(Y)$ is the vector representation of entity X and Y respectively. $|\vec{V}(X)|$ is the Euclidean length and can be computed by $|\vec{V}(X)| = \sqrt{\sum_{i=1}^n x_i^2}$. In recent work [Gulla et al., 2009] [Liu and Shao, 2010], the cosine similarity is adopted to compute similarity on mapped concepts. However, if the dimension of the vector increases, it will make the computing of such cosine similarity more problematic.

The approaches in the third category are to make use of a hierarchy to compute similarity. There have been several research efforts to develop matchmaking algorithms as introduced in Section 3.3. These approaches share one thing in common: based on the subsumption relation

in a taxonomy tree, four degrees of match are defined as below [Paolucci et al., 2002] [Şenvar and Bener, 2006]:

- **Exact Match:** the requests are equivalent to the service's advertisement.
- **Plug-in:** the service's advertisement is more general than the requested service.
- **Subsume:** the service's advertisement is more specific than the requested service.
- **Fail:** If no subsumption relation is found between service's advertisement and the requested service.

Degree of math are organized along a discrete scale where exact matches are of the best one among others; Plug-in matches are the next best level since the returned output can probably be used instead of what the requester expects. Subsumes is the third best level because the requirements of requester are only partially satisfied. Fail is the lowest level and it indicates an unacceptable result. In real applications, typically, to distinguish these four scales, four numerical values are assigned to each of them. One major concern with such approach is that it does not consider the semantic distances of the properties involved.

5.2.2 Ontology based Single Attribute Matchmaking Algorithm

Referring to Figure 5-1, in both *Categorization Matching* and *Operational Matching*, ontology based matchmaking play a critical role to determine matched services. Such matching engine can be established by calculating path distance between concepts on their shared hierarchical semantic structure. However, measuring similarity between Web Services differs slightly from calculating their underlying concept semantic similarity. The Web Service matching is characterized by the following features:

- **An asymmetric match.** Most of the current ontology matching algorithm are for symmetric matching [Algergawy et al., 2010] [He et al., 2008] which means that the similarity between a and b should be equal to the similarity between b and a denoted as $sim(a, b) = sim(b, a)$. However, the match between Web Services is asymmetric. Let us assume that we have two operations f_i and f_j as depicted in Figure 5-2. In this scenario, f_j matched to f_i , since all the inputs of f_j are subclasses of the outputs of f_i . That means f_j can be invoked after f_i . But, the converse is not necessarily true. There is no evidence indicating that f_i can be also invoked after f_j . Such asymmetrical property can be expressed as:

$$Operation_Match(f_i, f_j) \neq Operation_Match(f_j, f_i).$$

- **Comparing semantics with a common ontology commitment.** In SEwsDM, corresponding operations' parameters are related to one global ontology. For instance, in the example shown in Figure 5-2, f_i 's outputs and f_j 's inputs are referred to one ontology. The degree of match is translated into the problem of measuring the distance between two concepts in one ontology.

As introduced in the last section, all the discussed similarity notions can precisely determine the closeness between two compared concepts except those hierarchy based solutions which are the only asymmetric similarity notions. To increase the accuracy of the measurement of

matching degree, semantic distances should be taken into consideration. The works presented in [Ganesan et al., 2003], [Wen et al., 2006] and [Li et al., 2003] introduce different methods to compute a so-called matching score to precisely calculate similarity with semantic distances.

In this section we present an ontology based single attribute matchmaking algorithm noted as **SAMatching** for an asymmetric match which is inspired by the above three works. To begin with, instead of defining degree of match at the service level, we define degree of match for the underlying concepts. It is based on the fact that when building a chain for the composition, only one type of matching is required to take into account. For instance, to determine the succeeding services, only services with a *Plug-in match* against the last service of the chain can be selected, as more general inputs of a service might be able to be satisfied by the more detailed outputs of a service. In the same way, to find preceding services, only services which subsume the last service in the chain should be considered. Compared to the traditional filters for matchmaking, the matching filter defined in *SEwsDM* extends it with a sibling relationship where two concepts share at least one ancestor. Next, matching score is calculated by aggregating a set of concept similarities involved in a service.

Definition 5-1 (Degree of Match between Concepts). Given two concepts c_i, c_j based on a shared ontology O .

- If c_i and c_j are equivalent, then c_i **Exactly Matches** to c_j : ($c_i \equiv c_j$),
- If c_i is a subclass of c_j , then c_j could be **Plugged In** place of c_i : ($c_i \sqsubseteq c_j$),
- If c_i is a superclass of c_j , then c_i **Subsumes** c_j : ($c_i \sqsupseteq c_j$),
- If c_i and c_j have intersections and share ancestors, then c_i and c_j are **Siblings**: ($c_i \sqcap c_j$),
- If no subsumption relation exists between c_i and c_j , then **Failure** occurs: ($c_i \not\sqsubseteq c_j$). ■

Five degrees of match in the concept level is defined above. Considering that in some cases two concepts which are siblings can also indicate semantic relations between them, a relationship of sibling is added to our matching filters. These filters information will be used to find matched categories and operations. Taken the case shown in Figure 5-2 as an example, since concept i associated with *output1* of S_a is a subclass of h referred by *input1* of S_c , *output1* plugin matches with *input1*, *output1* \sqsubseteq *input1*. Conversely, it also indicates that *input1* subsumes *output1*, *input1* \sqsupseteq *output1*.

Definition 5-2 (Similarity between Concepts). Given two concepts c_i, c_j based on a shared ontology O .

$$Sim(c_i, c_j) = \begin{cases} 1, & \text{if } Exact(c_i, c_j) \\ \frac{1}{2} + \frac{1}{2} * e^{\alpha d} \cdot \frac{e^{\beta d} - e^{-\beta d}}{e^{\beta d} + e^{-\beta d}}, & \text{if } Plugin(c_i, c_j) \\ \frac{1}{2} * e^{\alpha d} \cdot \frac{e^{\beta d} - e^{-\beta d}}{e^{\beta d} + e^{-\beta d}}, & \text{if } Subsume(c_i, c_j) \\ \log \left(e^{\alpha d} \cdot \frac{e^{\beta d} - e^{-\beta d}}{e^{\beta d} + e^{-\beta d}} \right), & \text{if } Sibling(c_i, c_j) \\ 0, & \text{if } Fail(c_i, c_j) \end{cases}$$

where

$$d = \frac{2d_{c_i c_j}}{d_{c_i} + d_{c_j}},$$

$$d_{c_i c_j} = \begin{cases} \text{depth}(c_i), c_i \supseteq c_j \\ \text{depth}(c_j), c_i \sqsubseteq c_j \\ \max(\text{depth}(\text{anc}(c_i, c_j))), \text{others} \end{cases},$$

$d_{c_i} = O(c_i)$ is the height of the node c_i ,

$d_{c_j} = O(c_j)$ is the height of the node c_j ,

$l = \min(\text{dis}(c_i, c_j))$ is the shortest path between c_i and c_j ,

$\alpha \geq 0, \beta > 0$: scales the contribution of the shortest path length and height. ■

This definition is a combination of the similarity measures proposed by [Ganesan et al., 2003] and [Li et al., 2003]. In the first work the measure is based on the assumption that concepts at upper layers of the hierarchy have more general semantics and less similarity between them, while concepts at lower layers have more concrete semantics and stronger similarity. The similarity is computed as following:

$$d = \frac{2d_{c_i c_j}}{d_{c_i} + d_{c_j}} \quad (5-1)$$

However, this method doesn't take into account of the direct path length between two compared concepts. In Li's work, the shortest path length l as well as the depth of the lowest common ancestor are taken into consideration to compute the similarity:

$$\text{Sim}(c_i, c_j) = e^{\alpha l} \cdot \frac{e^{\beta d_{c_i c_j}} - e^{-\beta d_{c_i c_j}}}{e^{\beta d_{c_i c_j}} + e^{-\beta d_{c_i c_j}}} \quad (5-2)$$

It scales down similarity for subsuming concepts at upper layers and scales up the similarity at upper layers. However, this measure does not scale the similarity, the work presented in [Wen et al., 2006] intends to combine these above two measures together. The depth of the lowest common ancestor, $d_{c_i c_j}$, in (5-2) is placed with the relative depth, d , in (5-1) between these two computed concepts. Unfortunately, all above measures are more suitable for evaluating similarity at the leaf level where the depth of the lowest common ancestor in most cases is greater than 1. In **SAMatching**, we define a smooth function where the lowest common ancestor between c_i and c_j can be determined from one of three cases:

- If c_i is the super concept of c_j , we assign the depth of c_i to $d_{c_i c_j}$,
- If c_j is the super concept of c_i , we assign the depth of c_j to $d_{c_i c_j}$,
- If there is no relationship between c_i and c_j , we assign the depth of the lowest common ancestor of c_i and c_j to $d_{c_i c_j}$.

Unlike those existing solutions which are symmetric, considering the nature of matching among Web Services, our solution supports asymmetric matchmaking. The information of degree of match is integrated into the concept-level similarity. It is worth noting that the value of semantic similarity will be equal to 1 when it belongs to "Exact" match and in range of (0.5, 1.0) if it falls into "Plugin" match and in range of (0, 0.5) if it is classified as "Subsume" match. If the similarity value is smaller than 0, it means maybe these two concepts are siblings.

Table 5-1 Comparison of the various measures

Similarity	Ganesan et al.	Li et al.	Wen et al.	<i>SAMatching</i>
$Sim(d, g)$	0,33	0,79	0,29	0,29
$Sim(g, d)$	0,33	0,79	0,29	0,79
$Sim(d, h)$	0	0	0	0
$Sim(e, g)$	0,57	0,15	0,40	0,30
$Sim(h, i)$	0	0	0	0,23
$Sim(i, h)$	0	0	0	0,73

Table 5-1 shows the comparison of the similarity using various measures. Here, we compute similarity between four pairs of concepts in terms of the ontology available in Figure 5-2. From the ontology structure, we would expect to find that concepts d and e and g are more similar to each other than concept d and g . From Table 5-1, we see that except for the measures proposed in [Wen et al., 2006], all the other measures produce this result. As we mentioned before, since the lowest common ancestor of h and i is the root concept a with depth value equal to zero, the similarity between concepts h and i becomes to zero in the first three measures. Our measure can successfully compute the similarity with the smooth function of $d_{c_i c_j}$. Moreover, our solution also can be used as an asymmetric measure as mentioned above. It conveys more information to distinguish different logic relations hiding behind concepts.

Definition 5-3 (Operational Similarity). Consider that we have two annotated Web Services operations f_1, f_2 , the similarity between these two functions is defined as

$$OpSim(f_1, f_2) = \min_{c_i \in C_{out}^1 \cup C_{in}^1, c_j \in C_{in}^2 \cup C_{out}^2} Sim(c_i, c_j),$$

where

$$f_1 = (C_{in}^1, C_{out}^1, pre^1, eff^1, cat_1, q_1),$$

$$f_2 = (C_{in}^2, C_{out}^2, pre^2, eff^2, cat_2, q_2). \quad \blacksquare$$

As defined in Definition 4-1, an operation of a Web Service consists of a set of functional and non-functional elements. To determine a matched operation to the current available function, firstly, we need to evaluate the matching between two categories using the degree of matches defined in Definition 5-1. Next, the matching algorithm will go into the concept level to find the detailed matching information by calculating concept similarity. At this level, the referenced concepts for inputs and outputs of the service candidate and the available service will be compared to each other. The matching degree of each pair of concepts is determined by the generated similarity value. In the last step, as shown in Definition 5-3 the similarity of these two functions is assigned by the minimum value of similarity among all compared pairs over corresponding inputs and outputs.

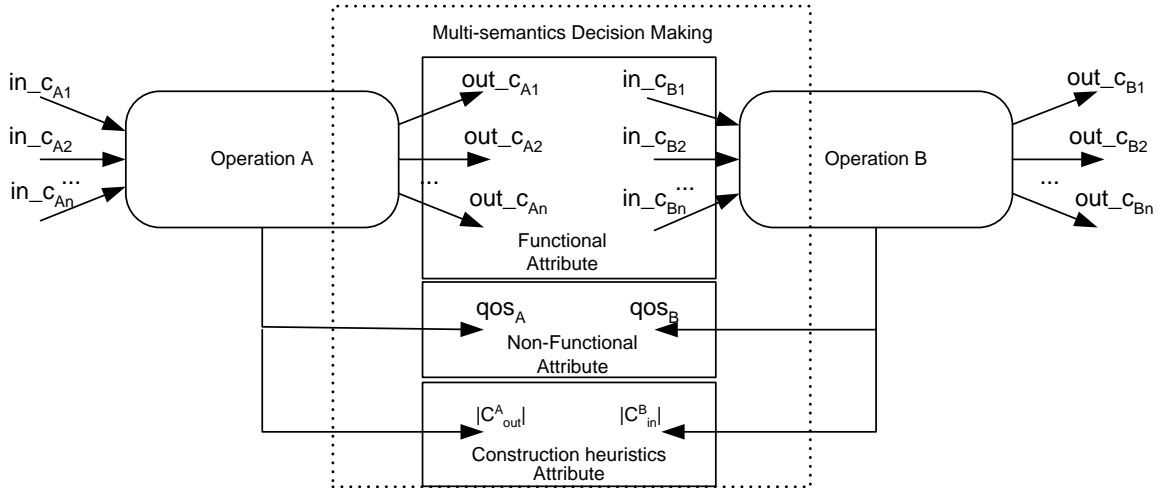
For instance, again referring to Figure 5-2, to compare the functions f_i and f_j , we skip step 1, since we assume that the categories of these two functions are matched. As calculated in Table 5-1, the similarity of corresponding concepts are $Sim(g, d) = 0.79$ and $Sim(i, h) = 0.73$. As declared in Definition 5-3, $OpSim(f_i, f_j) = \min(0.79, 0.73) = 0.73$. In addition, this similarity measure is also asymmetric, since the similarity between f_g and f_i is $OpSim(f_j, f_i) = \min(Sim(d, g), Sim(h, i)) = 0.23 \neq OpSim(f_i, f_j) = 0.73$.

5.3 Multi-Attribute Decision Making

The two main stages of composing Web Services are service discovery and service selection. In the service discovery process, a set of available Web Services are proposed as potential candidates mainly by the functional matchmaking algorithm explained in the previous section. In a succeeding selection, for each task, the most appropriate candidate is chosen to form the optimal composition due to the selection criteria. A selection may not be properly made if only considering one single attribute. For example, in the case that we aim to compose a chain with the maximum execution time, each operation we choose should functionally match to the available operation. Meanwhile, this chosen operation should also contribute to obtain the maximum execution time of the whole chain. Generally speaking, to efficiently compose Web Services all related attributes are classified into three groups depicted in Figure 5-3:

- **Functional attribute** indicates functional closeness of two operations. The similarity can be obtained by the formula defined in Definition 5-3.
- **Non-Functional attribute**, on the one hand, provides non-functional constraints to select the best service. The chosen service should satisfy with the QoS requirements of the available service. On the other hand, the aggregation of QoS elements of all chosen service to build the chain should satisfy the user's non-functional requirements of the whole composition task.
- **Construction heuristics attribute** is the knowledge of efficiently building a composition chain. When composing operations with backward chaining, if a service requires many inputs, it seems to be harder to satisfy. In contrast, if a service can provide lots of data, it will be easier to find successors [Yan et al., 2009] [Bless et al., 2008].

This section discusses how the selection of these multiple attributes can determine the best candidates for the composition.

Figure 5-3 Multi-attributes decision making in *SEmsDM*

5.3.1 Scoring Methods for Multi-Attribute Decision Making

Zeleny [Zeleny, 1982] opens his book “Multiple Criteria Decision Making” with a statement:

“It has become more and more difficult to see the world around us in a unidimensional way to use only a single criterion when judging what we see.”

It was also suitable for the service selection problem. To determine the most appropriate service, the ranking problem can be mapped to a **Multi-Criteria Decision Making (MCDM)** problem. The definition of MCDM is presented [Evangelos, 2002]:

“Multi-Criteria Decision Making (MCDM) has been one of the fastest growing problem areas in many disciplines. It is assumed that the decision maker is capable of expressing his/her opinion of the performance of each individual alternative in terms of each one of the decision criteria. The problem then is how to rank the alternatives when all the decision criteria are considered simultaneously.”

According to the different aspect of multiple and conflicting criteria, MCDM can be classified into two categories:

- **Multi-Objective Decision Making (MODM)** addresses the continuous decision spaces, primarily on mathematical programming with several objective functions.
- **Multi-Attribute Decision Making (MADM)** concentrates on problems with discrete decision spaces. It is applied to preferable decisions among available classified alternative by multiple attributes [Yoon and Hwang, 1995].

In the Web Service Composition paradigm, MADM has been adopted to optimize decisions under complex environment. They are able to evaluate a series of discrete candidates with predefined attributes. In a nutshell, there are three steps in applying MADM to evaluate the operation candidates:

- Step 1: Determine the relevant attributes and operation candidates.

- Step 2: Assign user-defined importance weights to the attributes.
- Step 3: Attach the numerical values to determine a ranking of each candidate.

Given a set of m candidates denotes as O_1, O_2, \dots, O_m and their individual attributes denoted as A_j . Assumed that decision maker has determined the relatedness of attributes denotes as w_1, w_2, \dots, w_n . In the end of step 2, the decision matrix has been generated as follows:

$$M = \begin{bmatrix} Norm(O_1, A_{11}) * w_1 & Norm(O_1, A_{12}) * w_2 & \dots & Norm(O_1, A_{1n}) * w_n \\ Norm(O_2, A_{21}) * w_1 & Norm(O_2, A_{22}) * w_2 & \dots & Norm(O_2, A_{2n}) * w_n \\ \dots & \dots & \dots & \dots \\ Norm(O_m, A_{m1}) * w_1 & Norm(O_m, A_{m2}) * w_2 & \dots & Norm(O_m, A_{mn}) * w_n \end{bmatrix}$$

where, $Norm(O_i, A_{ij})$ denotes the normalization procedure to transfer the distinct scales of attributes to a numerically comparable scale. Moreover, Score techniques are utilized in the step 3 to determine the affinity of each candidate. Many research efforts have been made for MADM. Different methods vary in their normalization schemas and in the manner of scoring candidates. In the rest of this section, some well-known MADM methods will be introduced.

Simple Additive Weighting (SAW) is the best known and most widely used method for MADM [Jaeger and Rojec-Goldmann, 2005]. In the normalization procedure, it considers both negative and positive attributes denoted as A_{pos} and A_{neg} respectively. Negative attributes mean that the higher the value, the worse off they are. The normalization function for negative and positive attributes is given by:

$$Norm(O_i, A_{ij}) = \begin{cases} \frac{A_{ij} - A_j^{min}}{A_j^{max} - A_j^{min}}, & A_{ij} \in A_{pos} \\ \frac{A_j^{max} - A_{ij}}{A_j^{max} - A_j^{min}}, & A_{ij} \in A_{neg} \\ 1, & A_j^{max} = A_j^{min}, A_{ij} \in A_{pos} \text{ or } A_{ij} \in A_{neg} \end{cases} \quad (5-3)$$

According to the generated decision matrix, the candidates are ranked based on the score method $V(O_i)$ defined as:

$$V(O_i) = \frac{1}{n} * \sum_{j=1}^n Norm(O_i, A_{ij}) * w_j \quad (5-4)$$

Weighted Product Model (WPM) processes sound logic and is computationally simple but has not been widely utilized [Yoon and Hwang, 1995]. Contrary to the SAW method, the transformation to a numerically comparable scale is not necessary when we use WPM where attributes are connected by multiplication. The scoring method is defined as:

$$V(O_i) = \begin{cases} \frac{\prod_{j=1}^n A_{ij}^{w_j}}{\prod_{j=1}^n (A_j^+)^{w_j}}, & A_{ij} \in A_{pos} \\ \frac{\prod_{j=1}^n A_{ij}^{(-1)*w_j}}{\prod_{j=1}^n (A_j^-)^{(-1)*w_j}}, & A_{ij} \in A_{neg} \end{cases} \quad (5-5)$$

where A_j^+ and A_j^- is the best value among all candidates of the j^{th} attribute for positive and negative attribute respectively. The weights become exponents associated with each attribute value. Positive power is for positive attributes, while negative power is for negative attributes.

Technique for **Order Preference by Similarity to Ideal Solution** (TOPSIS) initially proposed in [Hwang and Yoon, 1981] is also the practical and useful MADM method. It is based on the idea that the most preferred candidate should be close to its positive ideal solution (PIS) and far away from the negative ideal solution (NIS). TOPSIS method has been successfully used in dealing with multiple attributes issues in many domains due to:

- its theoretical rigorousness [Olson, 2004],
- a sound logic that represents the human rationale in selection [Jahanshahloo et al., 2009]
- the fact that it has been proved in [Zanakis et al., 1998] as one of the most appropriate methods in solving traversal rank.

Similar to SAW, TOPSIS needs to construct the normalized and weighted decision matrix in the first two steps. The normalization function can be defined as follows:

$$Norm(O_i, A_{ij}) = \frac{A_{ij}}{\sqrt{\sum_{i=0}^m (A_{ij})^2}} \quad (5-6)$$

Considering that the TOPSIS score of each candidate depends on the distance to PIS and NIS, two more steps are required before calculating the final similarity. The determination of PIS denoted as O^+ and NIS denoted as O^- should be the third step, which can be obtained from the following formula:

$$\begin{aligned} O^+ &= \left\{ \left(\max_{i=1, \dots, m} Norm(O_i, A_{ij}) * w_j \mid A_{.j} \in A_{pos} \right), \left(\min_{i=1, \dots, m} Norm(O_i, A_{ij}) * w_j \mid A_{.j} \in A_{neg} \right) \right\} \\ O^- &= \left\{ \left(\min_{i=1, \dots, m} Norm(O_i, A_{ij}) * w_j \mid A_{.j} \in A_{pos} \right), \left(\max_{i=1, \dots, m} Norm(O_i, A_{ij}) * w_j \mid A_{.j} \in A_{neg} \right) \right\} \end{aligned} \quad (5-7)$$

The fourth step is to measure the distance of each candidate to PIS and NIS respectively. The distance of each candidate from both ideal solutions are given by the n-dimensional Euclidean distance:

$$\begin{aligned} d_i^+ &= \sqrt{\sum_{j=1}^n (Norm(O_i, A_{ij}) * w_j - O_j^+)^2}, \\ d_i^- &= \sqrt{\sum_{j=1}^n (Norm(O_i, A_{ij}) * w_j - O_j^-)^2} \end{aligned} \quad (5-8)$$

The last step is to calculate the similarity to PIS which is measured by the relative distance of each candidate to PIS and NIS. The most appropriate candidate is the one that has the highest value. The score method $V(O_i)$ utilized to compute closeness is finally defined as below:

$$V(O_i) = \frac{d_i^-}{d_i^+ + d_i^-} \text{ with } 0 \leq V(O_i) \leq 1 \quad (5-9)$$

Among these three methods, the WMP method yielded results that were too extreme to be taken into account. The reason behind that is the use of weights as exponents in the mathematical formulation [Azar, 2000]. In [Simanaviciene and Ustinovichius, 2010], the

performed sensitive analysis in terms of initial data enables the authors to figure out that TOPSIS method is more sensitive than SAW method, especially when the initial data differ by 10% from the average criterion values. However, one of the drawbacks of classic TOPSIS is that the normalization formula is complicated [Saghafian and Hejazi, 2006]. Moreover, the use of Euclidean distance may have the problem associated with weights calculated twice [Lo et al., 2010]. Accordingly, the modifications and extensions of TOPSIS are highly required for Web Service Composition. In the following section, our scoring method implemented in *SEwsDM* which is inspired by TOPSIS will be illustrated.

5.3.2 TOPSIS based Algorithm for Multi-Attribute Decision Making

Generally speaking, the proposed matchmaking algorithm called *MAMatching* is based on classic TOPSIS including three major steps similar to other MADM algorithms shown in Figure 5-4. In the first step the related attributes of a compared operation are gathered and refined to conform to a standard. Then a series of user-predefined importance weights are assigned to these modified attribute values. A transformation procedure is invoked to transform the attribute values to a linear scale. After the first two steps, the decision matrix will be constructed. The scoring method implemented in the third step is then employed in this normalized matrix to compute similarity of each candidate. As a result, a set of ranked operations will be obtained. The operation listed in the first place which is the closest to 1 among other candidates will be selected. In the rest of this section, these three steps will be illustrated in details.

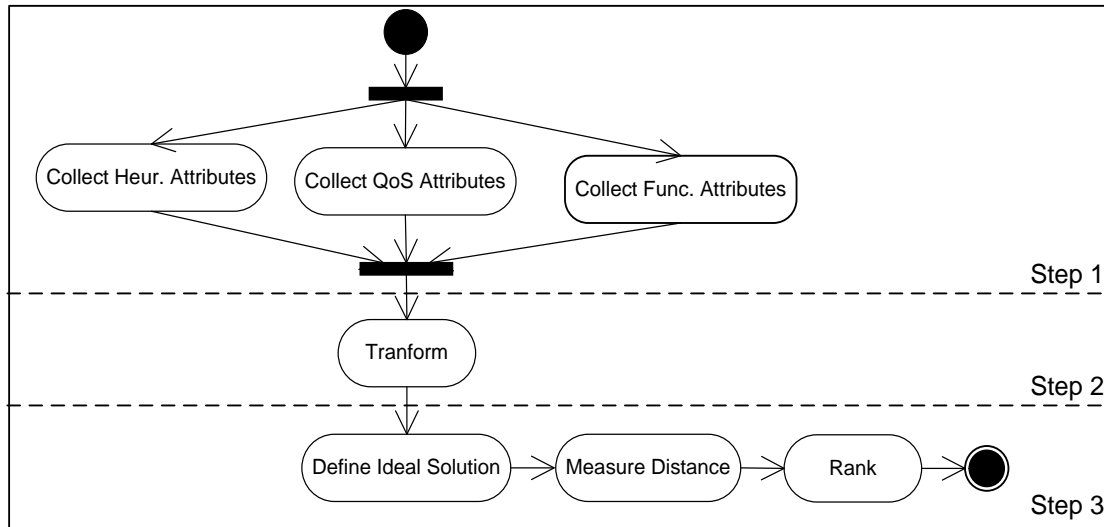


Figure 5-4 *MAMatching* algorithm

Assumed that we have a set of annotated operations as formulated in Definition 4-1, the functional attributes concerned the functional semantics, the QoS attributes containing the QoS specifications and heuristic attribute with semantics to efficiently build a service chain are extracted from the annotation file of each operation to construct a $m \times n$ decision matrix M , where m and n are the cardinality of a candidate-set and an attribute-set respectively. Each row of the matrix $(A_{i1}, A_{i2}, \dots, A_{in})$ indicates an operation candidate represented as an attribute vector. In the original TOPSIS method, the raw data in M are directly normalized using the formula 5-6. However, it should be noted that there is heterogeneity existing in those attribute

data of an operation, especially for the QoS attributes. For instance, the units such as second or millisecond to measure response time of a function maybe selected by different service providers. To ensure that each column vector uses the same unit, the first step of our multi-attribute semantic matchmaking is devoted to unit conversion among those operation candidates. On the other hand, the intervals of the underlying attributes are quite different as well. In this step, additional transformation will be conducted over the underlying original attribute values to keep the values in the interval of $[0, 10]$.

The next step is to normalize the generated decision matrix with normalization functions. The normalization is the process which transforms the attribute values to a linear scale. In the traditional TOPSIS, the vector normalization method as shown in formula 5-6 has been utilized. Unfortunately, it makes no difference when normalizing between positive attributes and negative attributes. Various normalization functions are designed to distinguish the negative attributes from positive attributes shown in Table 5-2.

Table 5-2 Normalization methods

Norm. Method	Positive Attributes	Negative Attributes	Proposed by
Vector based	$\frac{A_{ij}}{\sqrt{\sum_{i=1}^m (A_{ij})^2}}$	$1 - \frac{A_{ij}}{\sqrt{\sum_{i=1}^m (A_{ij})^2}}$	[Brauers et al., 2008]
		$\frac{(A_{ij})^{-1}}{\sqrt{\sum_{i=1}^m (A_{ij})^{-2}}}$	[Li et al., 2009]
Sum-based	$\frac{A_{ij}}{\sum_{i=1}^m (A_{ij})}$	$\frac{(A_{ij})^{-1}}{\sum_{i=1}^m (A_{ij})^{-1}}$	[Jiang and Li., 2006]
Minimum-Maximum based	$\frac{A_{ij} - A_j^{min}}{A_j^{max} - A_j^{min}}$	$\frac{A_j^{max} - A_{ij}}{A_j^{max} - A_j^{min}}$	[Zeng et al., 2004]
	$\frac{A_{ij}}{A_j^{max} + A_j^{min}}$	$1 - \frac{A_{ij}}{A_j^{max} + A_j^{min}}$	[Tong and Zhang, 2006]
Logarithmic based	$\frac{\ln(A_{ij})}{\ln(\prod_{i=1}^n A_{ij})}$	$1 - \frac{\ln(A_{ij})}{\ln(\prod_{i=1}^n A_{ij})}$	[Zavadskas and Turskis, 2008]
		$n - 1$	

After evaluating these existing methods, we recognize some limitations listed as below:

- **Out-of-range problem.** These methods share one thing in common that the normalized values are dependent on some global values, such as the sum or the maximal and minimal value over a certain dimension of attribute. It would easily result in so-called out of range value when adding candidates with their corresponding attribute data. Out of the range values are those newly added values which beyond out of the limits defined for the original values. The normalized valued for these out of the range values will probably be outside of the range of $[0, 1]$.

1]. In this context, a re-normalization process needs to be performed to get rid of these out-of-range values by recalculating those global values.

- **Lack of accuracy.** Some methods, like logarithmic based normalization function, might not precisely normalize the values when they are smaller than 1. The main reason behind is the use of the nature logarithm in the mathematic formulation. If the attribute value is smaller than 1, its nature logarithm is negative. Therefore, when applying the logarithmic based function for positive attribute, the larger the attribute value is, the smaller its normalized value is. Accordingly, when normalizing the attributes with value smaller than 1, we should use the function implemented for the negative attributes and vice versa.

In *SEwsDM*, different to other methods, to avoid such out-of-range in the beginning, softmax scaling is adopted to build our normalization function. The definition of “softmax scaling” is introduced in [Pyle, 1999]:

“Softmax scaling is so called because, among other things, it reaches “softly” toward its maximum value, never quite getting there. It also has a linear transformation part of the range. The extent of the linear part of the range is variable by setting one parameters. It also reaches “softly” towards its minimum value. The whole output range covered is 0-1.”

In Pyle’s book, it also indicates that a logistic function can be modified to perform the softmax scaling as described above. That is to say, those variable’s instance value can be transformed into the required value though logistic function. In our system the logistic function can be defined for both positive and negative attributes as below.

$$Norm(O_i, A_{ij}) = \begin{cases} \frac{1}{1 + e^{-A_{ij} * \alpha}}, & \text{if } A_{ij} \in A_{pos} \\ 1 - \frac{1}{1 + e^{-A_{ij} * \alpha}}, & \text{if } A_{ij} \in A_{neg} \end{cases} \quad (5 - 10)$$

To evaluate these existing methods, we did a small experiment. Assume that we have 6 operational candidates with those annotated attribute values as shown in Table 5-3. To simplify, we assume that all the corresponded attributes have already been transformed into the same scale.

Table 5-3 Operation candidates

Operation	Functional Attribute	QoS Attribute					Heuristic Attribute	
		Resp. Time	Avail.	Cost	Through-put	Reputation	In.	Out.
O ₁	0,179	350	0,98	90	8	0,91	3	3
O ₂	0,467	260	0,97	110	10	0,98	10	4
O ₃	0,462	490	0,99	110	11	0,93	10	4
O ₄	0,493	300	0,95	130	12	0,86	9	4
O ₅	0,325	170	0,96	95	4	0,62	10	5
O ₆	0,472	360	0,97	120	17	0,47	10	4

The bar chart shown in Figure 5-5 illustrates that in terms of the execution time of different normalization methods, our proposed normalization method and logarithmic based method require much less execution time than vector based, sum based and max-min based methods. For instance, in vector based method, ratio obtained by the square root method is more complicated. Sum-based method aims to reduce the execution time by computing ratio with the sum of data. However, it is still a time-consuming approach.

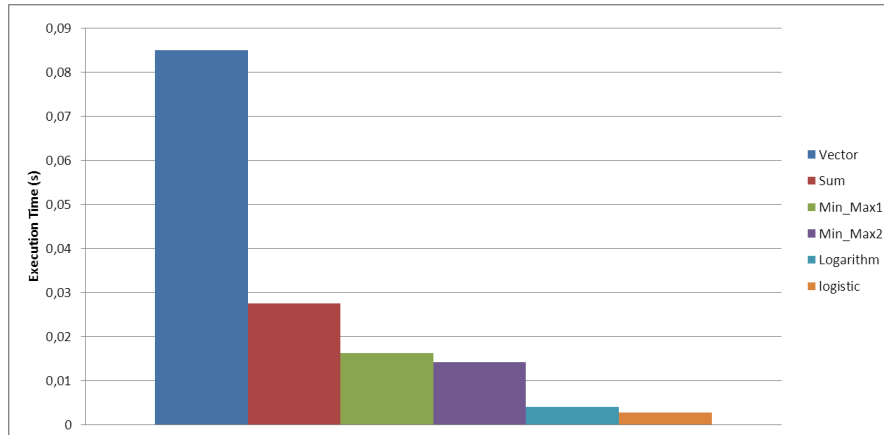


Figure 5-5 The execution time of normalization methods.

Figure 5-6 indicates the normalization results using all mentioned methods. When dealing with data within the interval $[0, 1]$, it is obvious that all mentioned methods, except logarithmic based method can be used to normalize data. Because of the nature logarithm used in logarithmic based method, it shows complete opposite results. Though it is proved that logarithmic based method yields more stable results in resolving multi criteria decision making problem in the paper [Zavadskas and Turskis, 2008], we found that when handling attribute data which are larger than 10, the normalized results will be approximately equal to each other as shown in Figure 5-7. In this context, because of the preprocessing in the first step, the normalized values generated by our proposed logistic based method are more segregated.

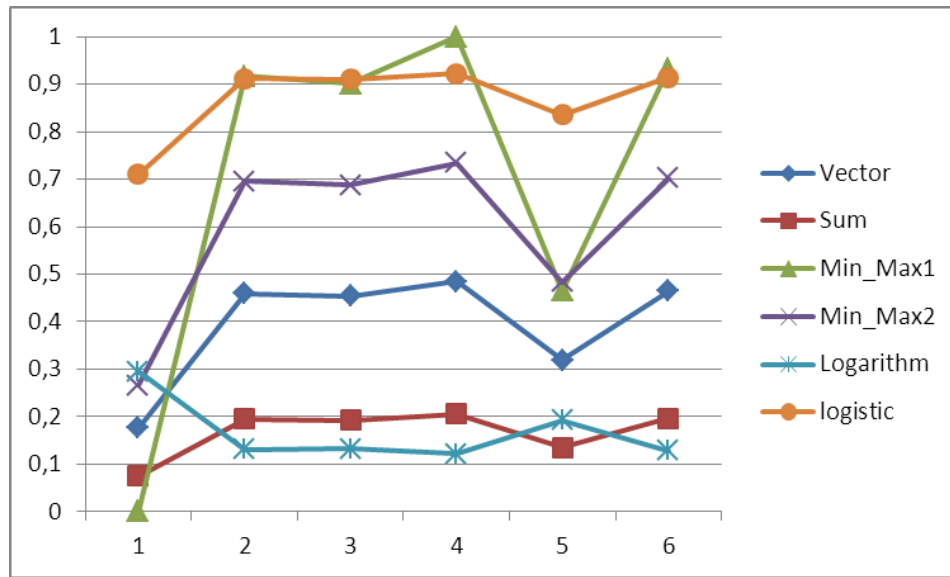


Figure 5-6 Normalization of data within interval [0,1]

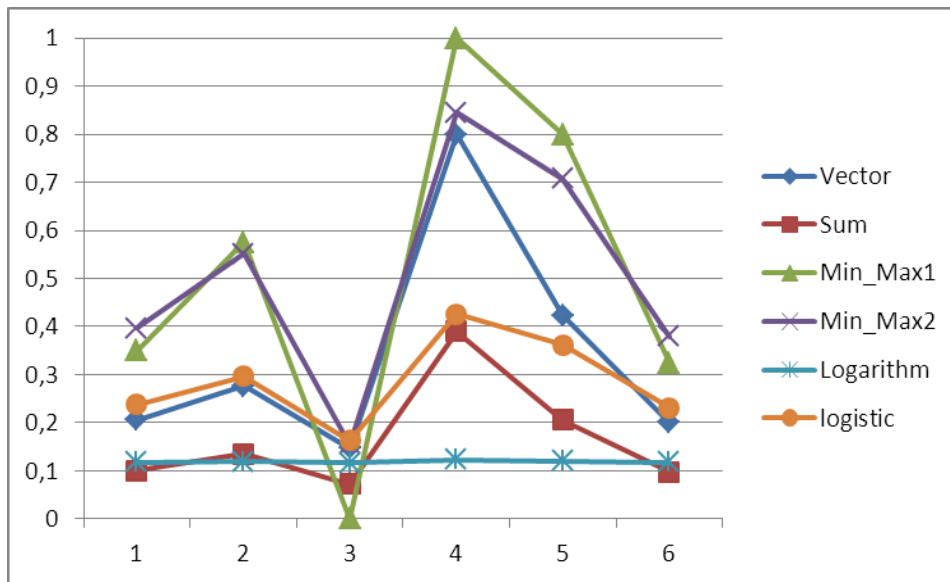


Figure 5-7 Normalization of data within interval [50, 500]

To sum up the above arguments, our proposed logistic appears to be more competitive than others. To avoid the out-of-range problem, the normalized values are generated depending only on the current original values. From the performance perspective, the newly method requires less execution time than other methods. Moreover, the normalized values generated by logistic base method can still remain segregated as in the other methods no matter if the original attribute values are smaller than 1 or larger than 100.

From now on, the decision matrix can be constructed by assigning the weight value of each attribute dimension to the corresponding normalized values of each candidate. Take the operation candidates listed in Table 5-3 as an example, the weights of attributes, elicited by three decision makers are shown in Table 5-4.

Table 5-4 Weight values of three decision makers

DMs	Functional Attribute	QoS Attribute					Heuristic Attribute	
		Resp. Time	Avail.	Cost	Through-put	Reputation	In.	Out.
DM ₁	0,2	0,4	0,05	0,15	0,05	0,05	0	0,1
DM ₂	0,2	0,05	0,05	0,15	0,4	0,05	0	0,1
DM ₃	0,4	0,15	0,05	0,2	0,05	0,05	0,05	0,05

In the end of step 2, the decision matrix M which will be used in the following steps is obtained as below:

$$M = \begin{bmatrix} 0.128979 & 0.094983 & 0.048163 & 0.063834 & 0.028314 & 0.047703 & 0 & 0.015887 \\ 0.165175 & 0.118379 & 0.048103 & 0.061402 & 0.029129 & 0.048163 & 0 & 0.020861 \\ 0.164693 & 0.06535 & 0.048221 & 0.061402 & 0.029533 & 0.047845 & 0 & 0.020861 \\ 0.167598 & 0.170223 & 0.047978 & 0.059 & 0.029934 & 0.047309 & 0 & 0.020861 \\ 0.149425 & 0.144803 & 0.048042 & 0.063223 & 0.026664 & 0.044381 & 0 & 0.015887 \\ 0.165652 & 0.09259 & 0.048103 & 0.060197 & 0.0319 & 0.041365 & 0 & 0.020861 \end{bmatrix}$$

The last step is to rank the candidates by measuring their distances to PIS and NIS respectively. Since in the last steps, both positive and negative attributes are normalized in the same scale, the definition of two solutions in Eq. 5-7 is modified by:

$$\begin{aligned} O^+ &= \{m_1^+, m_2^+, \dots, m_n^+\} = \{\max_{i=1, \dots, m} m_{ij}\} \\ O^- &= \{m_1^-, m_2^-, \dots, m_n^-\} = \{\min_{i=1, \dots, m} m_{ij}\} \end{aligned} \quad (5-11)$$

In traditional TOPSIS algorithm, the Euclidean distance is used to measure the distance shown in Eq. 5-8. However, the problem may occur when using such distance, as the weight values have been calculated twice [Lo et.al 2101]. To overcome this problem, in our system the weighted Minkowski distance [Steuer, 1986] shown in below has been utilized.

$$\begin{aligned} d_i^+ &= \left[\sum_{j=1}^n w_j * \left(\frac{m_{ij}}{w_j} - \frac{o_j^+}{w_j} \right)^p \right]^{\frac{1}{p}} \\ d_i^- &= \left[\sum_{j=1}^n w_j * \left(\frac{m_{ij}}{w_j} - \frac{o_j^-}{w_j} \right)^p \right]^{\frac{1}{p}} \end{aligned} \quad (5-12)$$

where $p=2$ which is known as the weighted Euclidean distance. Again referring to the example, by applying Eq. 5-12, the distance matrix can be obtained as follows:

$$D = \begin{bmatrix} 0.119042 & 0.430556 \\ 0.044363 & 0.490157 \\ 0.126402 & 0.471422 \\ 0.060937 & 0.524887 \\ 0.052308 & 0.472951 \\ 0.088583 & 0.470614 \end{bmatrix}$$

Here, the first column represents the distance from O_i to PIS while the second column indicates the distance to NIS. According to such distance matrix, by applying the Eq.5-9 the relative closeness coefficient for each candidate can be obtained as follows:

$$\begin{aligned} V(O_1) &= 0.129936, & V(O_2) &= 0.147923, & V(O_3) &= 0.142269 \\ V(O_4) &= 0.146745, & V(O_5) &= 0.142731, & V(O_6) &= 0.142026 \end{aligned}$$

Therefore, based on the coefficients above, these six candidates are arranged in the following order: $O_2, O_4, O_5, O_3, O_6, O_1$.

For each set of weight values, the TOPSIS based ranking algorithms are applied. Table 5-5 shows the ranking result using different criteria.

Table 5-5 Ranking results by different DMs

DMs	No.1	No.2	No.3	No.4	No.5	No.6
DM ₁	O_2	O_4	O_5	O_3	O_6	O_1
DM ₂	O_6	O_4	O_2	O_3	O_5	O_1
DM ₃	O_4	O_2	O_3	O_6	O_5	O_1

5.4 Implementation of *SEwsDM*

In this section, the implementation of semantics enhanced decision making algorithm will be introduced in details. To be brief, the introduced single attribute matchmaking algorithm aims to find a set of semantically related functions, and the ranking of those correlated services are executed by multiple attributes matchmaking algorithm.

Note that in most cases more than one suitable function will be invoked in terms of a set of available concepts denoted as $F_a = \{f_a^1, f_a^2, \dots, f_a^n\}$. To find the related functions using operational similarity defined in Definition 5-3 will lead to redundant comparison work between available functions and those compared functions. Therefore, to simplify the matching process, we choose a set of current available concepts denoted as $C_a = (Cat, C)$ as the start point of the matchmaking algorithm. Here, Cat is a set of required category information, and C is a set of expected concepts. How to define such available concepts will be discussed in the next section. The reason behind this strategy is to consider all the available functions as a whole. In addition, to speed up the composition in the next phase, in *SEwsDM* we distinguish the backward matching from the forward matching. Before going into details, let us define some kinds of matchmaking algorithms which are utilized in *SEwsDM*.

Definition 5-5 (Categorization Match). Assumed that $C_a = (Cat, C)$ is a set of current available concepts and the compared function $f = (C_{in}, C_{out}, pre, eff, cat, q)$,

$Cat_Match(C_a, f)$ is a boolean function which compares the categorization semantic between the function and the current available information using similarity function defined in Definition 5-2. For forward matching, $Cat_Match(C_a, f)$ is true if the following conditions hold:

$$\exists c_i \in Cat \wedge \exists c_j \in f.pre.cat : Sim(c_i, c_j) \neq 0$$

For backward matching, $Cat_Match(C_a, f)$ is true if the following condition is fulfilled:

$$\exists c_i \in Cat \wedge \exists c_j \in f.eff.cat : Sim(c_i, c_j) \neq 0 \quad \blacksquare$$

Definition 5-6 (Forward Matching). Given an annotated current available set of concepts $C_a = (Cat, C)$. Let $f_s = (C_{in}^s, C_{out}^s, pre^s, eff^s, cat_s, q_s)$ be an annotated function. Then f_s is a succeeding function of C_a , if the following conditions hold:

- i) $Cat_Match(C_a, f_s) = true$,
- ii) $SSim(C_a, f_s) = OpSim(C_a, f_s) = \min_{c_i \in C, c_j \in C_{in}^s} Sim(c_i, c_j) \geq 0.5 \quad \blacksquare$

Forward matching aims at finding succeeding services whose input concepts *plugin* with the current available concepts. Therefore according to the Definition 4-2, the similarity between these two operations should be greater than 0.5.

Definition 5-7 (Backward Matching). Let $C_a = (Cat, C)$ be a set of available concepts. Let $s_p = (C_{in}^p, C_{out}^p, pre^p, eff^p, cat_p, q_p)$ be an annotated service. Then s_p is a preceding service of C_a , if the following conditions are satisfied:

- i) $Cat_Match(C_a, f_s) = true$,
- ii) $PSim(C_a, f_s) = OpSim(C_a, f_s) = \min_{c_i \in C_{in}^p, c_j \in C} Sim(c_i, c_j) \in (0, 0.5] \quad \blacksquare$

Backward matching targets to find the preceding services whose output concepts *subsume* the input concepts of current operations. Based on the Definition 4-2, the similarity between these two operations should be distributed between 0 and 0.5.

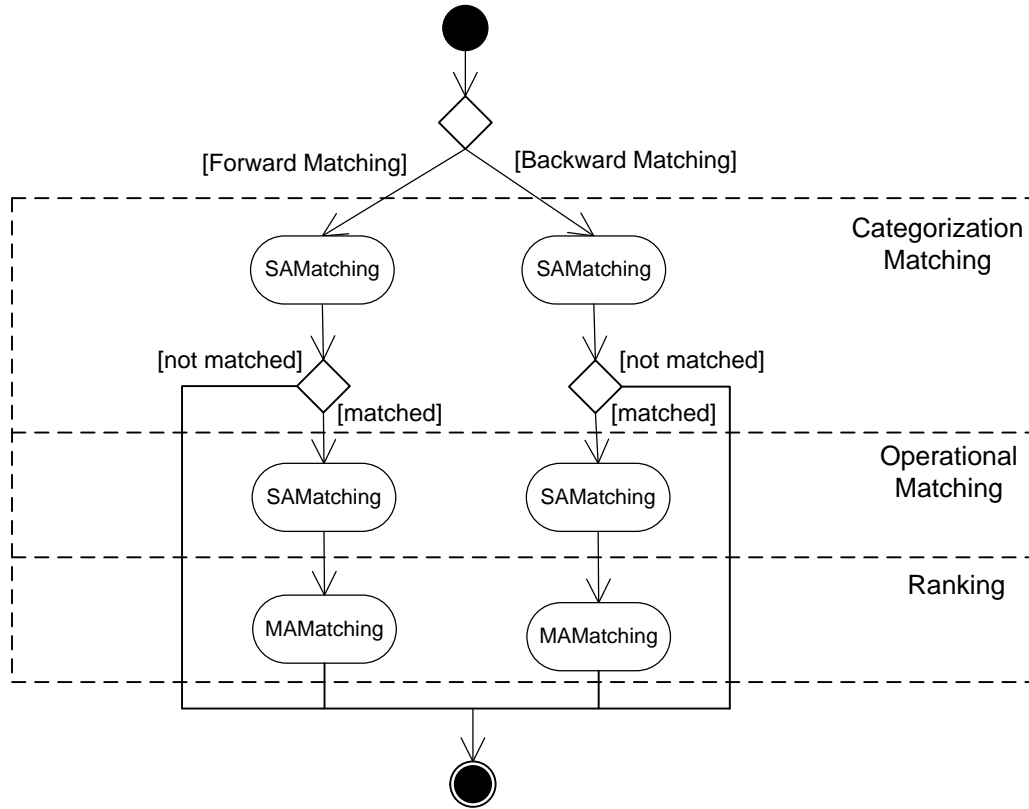
Figure 5-8 Activity diagram of *SEwsDM*

Figure 5-8 depicts the activity diagram of *SEwsDM*. The matchmaking algorithm is split into two parts: **Forward Matching** (*FMatch*) and **Backward Matching** (*BMatch*). Figure 5-9 shows the pseudo-code for these two matchmaking algorithms. Each matchmaking algorithm consists of three matching processes, namely categorization matching, operational matching and ranking.

More specifically, it starts with the recognition of those category-matched operations. Those found operations will be sent to the *Operational Matching* component where based on the ontology based single attribute matching algorithm (*SAMatching*) introduced in Section 5.2.2, the operational level similarity for forward and backward matchmaking is calculated by applying Definition 5-5 and Definition 5-6 respectively. Note that, in *FMatch*, a successor is added to the result set if and only if all its input concepts are satisfied by the existing concepts while in *BMatch*, an operation is added to proceeding operation set if at least one of its outputs concepts plugin the input concepts of current operations.

Because of this slight difference, in the *Ranking* process, various construction heuristics attributes will be selected together with functional similarity generated from *Operational Matching* and QoS information from the annotation of the operation itself to perform the multi-attributes semantic matchmaking (*MAMatching*). Regarding the *FMatch*, the number of output concepts can be chosen as the heuristics attribute. It is based on the conjecture that if an operation provides more output concepts, the probability that the final goal is achieved will increase. Concerning the *BMatch*, the effected output concepts and the number of input concepts might be selected for the heuristic attributes. Note that the *BMatch* differs from *FMatch*. There is no need that all the outputs of the preceding candidates are matched to all

inputs of current available operations. That is to say, the preceding candidates must be combined together to jointly satisfy the inputs concepts of existing operations. In this context, the more the effected outputs the operation has, the better the operation is for the future process. On the other hand, if a service requires many inputs, it seems to be harder to satisfy. Therefore the number of inputs also needs to be taken into account.

Algorithm 1 Forward Matchmaking: FMatch(C,F,W)

Inputs: a set of existing concepts, C
a set of available functions, F
a set of weight vectors, W

Data: degree of functional matching, ssim
degree of final matching degree, d

Output: a set of successor operations, S

Begin

```

foreach function f do
  if (Cat_Match (C, f) == true) then
    ssim = SSim(C, f)
    d = MAMatching(ssim,f.qos, f.outputSize(),W)
    add f to S
  end if
end for
  rank f in S according to d in descending order

```

return S

End

Algorithm 2 Backward Matchmaking: BMatch(C,F,W)

Inputs: a set of concepts, C
a set of available functions, F
a set of weight vectors, W

Data: degree of functional matching, psim
degree of final matching degree, d

Output: a set of successor operations, S

Begin

```

foreach function f do
  if (Cat_Match (C,f) == true) then
    psim = PSim(C,f)
    d = MAMatching(psim,f.qos,f.OutputSize(),f.inputSize(),W)
    add f to S
  end if
end for
  rank f in S according to d in descending order

```

return S

End

Figure 5-9 Matchmaking algorithms in *SEnsDM*

5.5 Summary

In this chapter, a multi-attribute semantic matchmaking engine called *SEnsDM* is presented. The matchmaking engine is a key step for the further Web Service composition process. To facilitate the automated composition process, the matchmaking algorithm should be able to retrieve relative services efficiently and to select the most suitable service for each single step as well. Therefore, our newly developed engine, *SEnsDM*, consists of two sub-engines called *SAMatching* which aims to find matched services including category and operation level matching. The other engine called *MAMatching* is focusing on ranking results among the discovered operations based on multi-attribute semantics of services, such as functional closeness, QoS semantics and construction heuristics.

Generally speaking, our newly developed semantic matchmaking engine *SEnsDM* attempts to improve the traditional matching algorithms in the following aspects:

- **Makes the logic-based DoM more precisely.** According to the technical survey in Section 3.3, existing matching efforts determine the logical DoM share one thing in common that several logical filters in terms of the concepts referred to the ontology are defined. Typically, to measure the DoM, various numerical values are assigned to each of those pre-defined logical filters. One major concern of such approach is that it does not consider the semantic distances of the properties involved. To increase the accuracy in assigning matching degree, semantic distances should be taken into consideration. To solve this problem, in *SEnsDM*, a similarity model which is capable of providing more precisely DoM by integrating non-logical based similarity into the logical filters. By doing so, on the one hand, the accuracy of similarity is dramatically

improved as shown in Table 7-3. On the other hand, the similarity between services or operation will still remain asymmetric in *SEwsDM*, which is not supported by pure non-logical methods (see Section 5.2).

- **Enhances the ranking mechanism with multi-attribute decision making algorithm.** The ranking algorithm in existing system mainly depends only on the non-logical DoM which examines only the functional aspects such as IOPE information between service providers and service requirements. However, other semantics for instance QoS attributes and construction heuristics are also play a crucial role to select the appropriate services. In *SEwsDM*, we treat the service selection problem as a multi-attribute decision making problem where all these mentioned semantics are taken into account in a TOPSIS based ranking algorithm (see Section 5.3). The comparison with other existing solutions is shown and discussed in Section 7.2.

Chapter 6

Web Service Composition as Planning in *SEwsPL*

“By describing a service as a process in terms of inputs, outputs, preconditions and effects, using the metaphor of an action, composition can be viewed as a planning problem.”

-Mark Carman

Most research efforts for managing complex distributed resources have been focusing on Web Services technology. As such, the number of available Web Services increased dramatically during the recent years. However, in almost all modern applications, especially in those complicated scientific applications, it is often impossible to find fully satisfied Web Services. Therefore, composing the most appropriate Web Services from existing services to fulfill the complex requirements of a scientific task becomes even more indispensable. The major challenge when composing web services is scalability, as the search space will exponentially increase if the number of available Web Services increases. Toward this problem, efficient composition algorithms are highly preferred.

CONTENTS

6	WEB SERVICE COMPOSITION AS PLANNING IN SEwsPL.....	58
6.1	REQUIREMENTS ANALYSIS FOR AUTOMATIC PLANNING	59
6.2	SEwsPL FRAMEWORK.....	60
6.3	WEB SERVICE COMPOSITION FORMALISM.....	62
6.3.1	<i>Traditional Modeling of A Web Service Composition Problem.....</i>	<i>62</i>
6.3.2	<i>PDDL 3 based Planning Model for Web Service Composition</i>	<i>64</i>
6.4	GRAPHPLAN ALGORITHM	67
6.4.1	<i>Planning Graph</i>	<i>68</i>
6.4.2	<i>Graph Expansion</i>	<i>69</i>
6.4.3	<i>Solution Extraction.....</i>	<i>70</i>
6.5	ENHANCEMENTS OF GRAPHPLAN IN SEwsPL	70
6.5.1	<i>Simplified Ordered Planning Graph.....</i>	<i>71</i>
6.5.2	<i>Goal-Oriented Bi-directional Graph Expansion Algorithm.....</i>	<i>74</i>
6.5.3	<i>Workflow based Planning Extraction</i>	<i>78</i>
6.6	SELF-ADAPTIVE COMPOSITION.....	82
6.7	SUMMARY	84

6.1 Requirements Analysis for Automatic Planning

This chapter begins with the requirements analysis for the automated Web Service composition derived from the literature review presented in Section 3.4. To automate the composition of services, AI approaches were often been exploited. Considering the characteristics of Web Services, additional requirements for the AI Planning techniques are identified as follows:

- ***Enables Quality of Plan (QoP) awareness in the composition (R1).*** In some previous work, QoS has been considered to find the most suitable service for Web Service discovery. However, QoP, which is a logical combination of QoS has been largely ignored in the most cases of AI planning [Baryannis and Plexousakis 2010]. Notice that in practice, a number of combinations of component services might satisfy a given task with different levels of quality. Some of them probably result in higher cost but quicker response time. Some may lead to better accuracy with a long response time. Meanwhile, different users can customize the quality requirements in different ways. Therefore, to build a chain fulfilling the user's requirements, a good automated service composition system should take care of and understand the quality of services.
- ***Allows the dynamic composition of Web Service (R2).*** The composition results generated by most of the research efforts that use AI planning are static without taking account of the dynamic changes of services. However, in the real applications, due to the dynamic environment of Web service, such as the shutdown of the server, service upgrade, etc., the availability of service will change frequently. Therefore, there is a need to enable dynamic generation of the composition chains adapting to such dynamic changes, which have not been adequately explored in traditional AI planning approaches.
- ***Achieves scalability during the composition process (R3).*** Due to the enormous number of existing services and their applicability in many different processes, scalability problem is raised to new heights for real world problems [Hennig and Balke, 2010]. Owing to the lack of scalability in most of the AI planning techniques, this has been one of the reasons behind the fact that automated composition techniques have not been widely adopted in practice. Instead, the manual service composition tools have been often applied in this area. A new mechanism to provide the ability to solve large real world problem is highly required in an automated composition system.

In the following sections, we present the design and implementation of AI planning based service composition framework called **S**emantics **E**nhanced **W**eb **S**ervice **P**lanner (*SEwsPL*), which tries to achieve the mentioned requirements.

6.2 SEwsPL Framework

Our research work addressed the problems mentioned in the previous section by introducing a novel AI Planning Graph based planner called *SEwsPL*. Generally speaking, it enables the automated composition by extending the classic AI GraphPlan algorithm with a goal-directed bidirectional expansion strategy and representing the final plan as a scientific workflow for the future reuse.

An abstract framework is presented in Figure 6-1. The *SEwsPL* system consists of four functional modules: *Problem Modeling*, *Graph Expansion*, *Plan Extraction* and *Plan Repair*.

First, required data structures for planning are created and filled in the *Problem Modeling* modular. *Graph Expansion* and *Plan Extraction* are core modules concerning the creation of a Planning Graph and the search for the final plan. Different to traditional GraphPlan algorithm where expansion is performed with a forward-chaining strategy while the extraction of the final plan is achieved by a backward-chaining algorithm, in *SEwsPL*, a bidirectional expansion algorithm is developed for the expansion procedure, which aims to reduce the size of whole Planning Graph. Rather than invoking the time-consuming backward search, we intend to represent the final plan directly from the generated Planning Graph to a scientific workflow which can be easily visualized in the workflow management system. Details will be illustrated in Section 6.4.

In addition, considering that the vast majority of AI Planning techniques produce only a static plan without taking into account of the dynamic changes, in *SEwsPL*, an additional modular called *Plan Repair* is developed to support self-adaptation. This modular aims to find an alternative solution when inconsistencies occur. A plan repair algorithm is proposed by reusing most of the original plan as possible. All technical details can be found in Section 6.5.

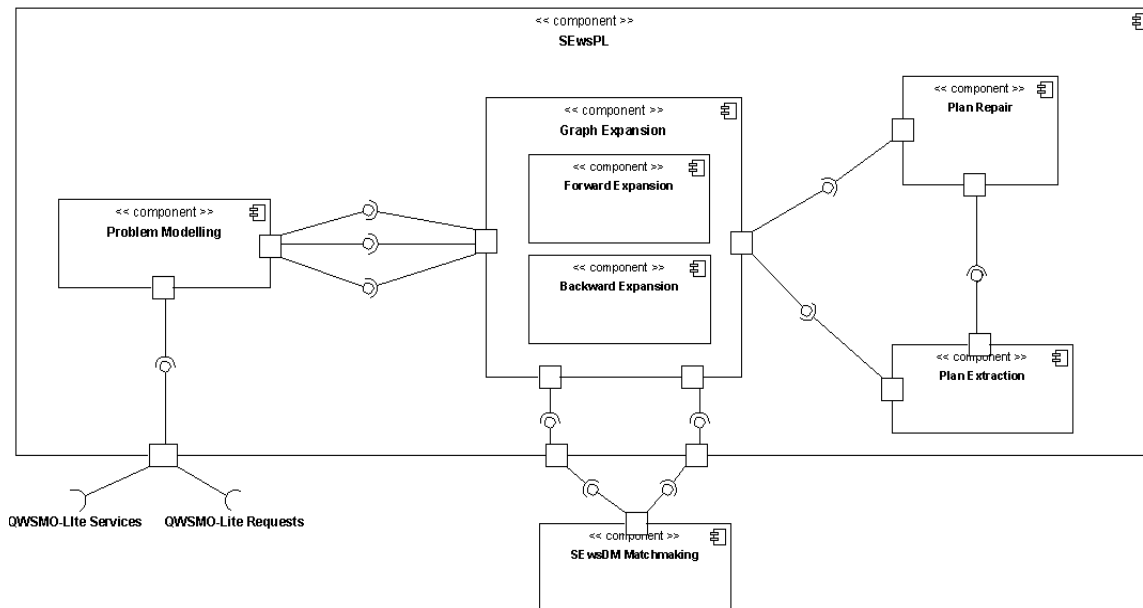


Figure 6-1 The *SEwsPL* framework

We use a service composition scenario mentioned in Section 2.1 as a running example to illustrate our approach.

Let us assume that a renewable energy scientist attempts to do an experiment about the wind speed forecast for selected cities across the United States. Now he only has some local information about the selected cities, such as city name, IP address. And he expects to find an existing service containing the wind speed forecast data with high throughput. To satisfy the requirements denoted as W_r in the following Table 6-1, five Web Services with six operations will be involved. Here, we assume that all the Web Services are already annotated with semantics using QWSMO-Lite (see Chapter 4). The details of semantics including both the functional and non-functional descriptions are presented in the Table. To simplify the use case, we consider only two important generic QoS attributes, namely throughput and response time.

Table 6-1 A Web Service composition example

Web Service	Operation	Input	Output	QoS	
				Thr. Put	Resp. Time
W_r : Request	-	“San Jaun” “130.14.160.0” 2011-09-22T23:59:59, “Wind Speed”	KML-NDFD	-	minimal
W_1 : USZip	GetInfoByCity	City name	ZipCode	17.5k	0.87s
W_2 : ZipcodeLookupService	CityToLatLong	City name	Latitude, Longitude	20k	1.5s
W_3 : ip2loc	GetIPCountry	IP-Address	Latitude, Longitude, Country, State, City	18k	0.35s
W_4 : LocationByZipService	getLocationByZip	ZipCode	State, Latitude, Longitude	15k	0.25s
W_5 : ndfdXML	LatLonListZipCode	ZipCode	Latitude, Longitude	20k	0.24s
	GMLLatLonList	Latitude, Longitude, RequiredTime, PropertyName	KML-NDFD	20k	0.24s

6.3 Web Service Composition Formalism

Before discussing the details of our framework, let us model a **Web Service Composition (WSC)** problem to an AI Planning problem. A planning problem has been defined by [Weld, 1994] as requiring the following inputs:

- A description of the world start state, S_0
- A goal state, S_g
- A set of actions, A

6.3.1 Traditional Modeling of A Web Service Composition Problem

Stanford Research Institute Problem Solver (STRIPS) notation which was used to describe the planning domain for a robot system called “Shakey” in the 1970ies [Fikes and Nilsson, 1971] are now widely applied to formalize a composition problem. A Web Service composition problem in STRIPS model [Russell and Norvig, 2002] is represented by $\Pi = \langle S, A, S_0, S_g \rangle$ where S is a set of states, A is a set of available actions which are Web Services in WSC domain, $S_0 \subset S$ is a set of initial states and $S_g \subset S$ is a set of goal states [Oh and Kumara, 2006] [Hatzil et al., 2009]. To represent the example in Table 6-1 as a composition problem:

$$S = \{City, IP - Address, ZipCode, Latitude, Longitude, \dots\};$$

$$A = \{W_1, W_2, W_3, W_4, W_5\};$$

$$S_0 = \{City, IP - Address, StartTime, EndTime, PropertyName\};$$

$$S_g = \{KML - NDFD\}.$$

It is noticed that in AI planning domain, STRIPS allows to define the actions by specifying preconditions, an ADD-list and a DELETE-list, such a schema will be also used to formalize the behavior of Web Services. The semantics behind it is that the Web Service W_i can only be applicable if its preconditions which are the service’s inputs concepts are satisfied by the current states S_n . After the execution of the action, the service’ output concepts will be added to ADD-list. Considering that services do not generate any negative effects, the DELETE-list will remain empty. In addition, the successor state S_{n+1} will be updated with newly generated ADD-list: $S_{n+1} = S_n \cup Add(W_i)$. For instance the atomic service $ip2loc:W_3 = (Pre(W_3), Add(W_3), Del(W_3))$ may be defined as follows:

$$Preconditions: Pre(W_3) = \{IP - Address\}$$

$$ADD - list: Add(W_3) = \{Latitude, Longitude, Country, State, City\}$$

$$DELETE - list: Del(W_3) = \emptyset$$

Planning Domain Definition Language (PDDL) [Ghallab et al., 1998] is a de-facto standard planning domain and problem description language. Compared with STRIPS, it provides support for more expressive action descriptions including the definition of universally quantified and conditional expression effects. This language is now widely accepted and used for the representation of planning problems.

The description of PDDL is separated into two files:

- **A domain file:** consists of the definition of all language constructs referenced in the action including types, predicates, function, etc. and the definition of the action itself.
- **A problem file:** contains mainly the objects of the problem, the initial states and the goal.

The PDDL description for the above meteorology service is shown as follows:

<pre>(define (domain meteorologyService) (:requirements [:strips]) (:action ip2loc :parameters (?p - IP_Address) :precondition (IP_Address ?p) :effect (and (Latitude ?lat) (Longitude ?lon) (Contury ?c) (State ?s) (City ?c))) ...)</pre>	<pre>(define(problem getKMLfile-Service) (:domain meteorologyService) (:init (and (Cityname ?a) (IP_Address ?p) (StartTime ?t_start) (EndTime ?t_end) (PropertyName ?prop))) (:goal(KML_NDFD ?k)))</pre>
--	--

Figure 6-2 PDDL description for a meteorology service

PDDL 2.1 [Fox and Long, 2003] and PDDL 2.2 [Edelkamp and Hoffmann, 2004] have been designed to be backward compatible with the original PDDL. It extends PDDL with time and resources. It allows for the numeric expressions for the additional properties of an action. The effects of an action can make use of a selection of assignment operations in order to update the values of primitive numeric expressions. Moreover, derived predicates are also supported which enables the handling of domain axioms. In the new version, deterministic unconditional exogenous events are expressed using timed initial literals. In addition, considering that the same initial and goal states might yield different plans, another extension so-called plan metric is specified a particular problem to evaluate the generated plans. These extensions allow us to express QoS semantics of Web Services and the plan. In [Naseri and Towhidi, 2007], the additional quality of service processor updates the domain and problem files with the initial and goal QoS ontology in PDDL 2.1 style. Let's refer to the example again. Assume that users attempt to get KML files with the maximum throughput, Figure 6-3 shows the descriptions in PDDL 2.1/2.2 version. Different from the original PDDL, in PDDL 2.1/PDDL 2.2, the domain file is extended by defining a function for each of the QoS dimension and the initial and goal descriptions in problem file are modified with additional requirements of users' preferred QoS semantics as well.

<pre> (define (domain meteorologyService) ... (:functions (Response-Time) (Total-Throughput)) (:action ip2loc ... (assign (Response-Time) 0.35) (increase (Total-Throughput) 18000)) ...)</pre>	<pre> (define(problem getKMLfile-Service) (:domain meteorologyService) (:init (and ... (= (Total-Throughput) 0))) (:goal(KML_NDFD ?k)) (:metric maximize (Total-Throughput))))</pre>
--	---

Figure 6-3 QoS description using PDDL 2.1/PDDL 2.2

PDDL 3.0 [Gerevini and Long, 2006] and PDDL 3.1[Helmert et al., 2008] extend the previous versions of the PDDL language by increasing its expressive power about the plan quality specification. It allows us to distinguish those constraints and goals which must be achieved from those so-called preferences which may not be satisfied, but are desired. The quality of a plan is improved by identifying the best plan which satisfies with all the strong constraints and has the best subset of the preference. In other words, the more preferences a plan satisfies, the better quality it processes. To this end the optimization of the plan quality is performed by defining plan metrics with preference marks. As for the WSC domain, unfortunately, these extended descriptions which should also be possible to apply to augment the quality of composition chains are ignored by the most of the recent efforts in this field. In [Lin et al., 2008] and [Sohrabi and Mcilraith, 2009], PDDL 3.0 is only be used to specify the user preferences rather than encoding the problem domain as a whole. In their systems, to generate a plan OWL-S based service descriptions and PDDL 3.0-style users' preferences should be translated to HTN-based constraints respectively.

6.3.2 PDDL 3 based Planning Model for Web Service Composition

As mentioned in the previous section, PDDL language now becomes the de-facto standard supported by a range of AI planners. However, the enhanced version PDDL 3.1 with more expressive power to describe QoP is ignored by most of the recent research work. On the other side, since pure WSDL which lacks semantics annotations fails in the task of efficiently and automatically Web Services composition, in Chapter 4, an annotation language called QWSMO-Lite has been introduced to semantically describe the given services. Inspired by the works done in [Hatzi et al., 2009] and [Klusch et al., 2005] where OWL-S atomic processes correspondent to WSDL operations are translated into original PDDL and PDDL 2.1 respectively, in *SEwsPL* system, the planning data are extracted and collected from the operational level of the service descriptions along with the users' requests in QWSMO-Lite to a PDDL 3 based format.

To simplify the parsing and communication with PDDL descriptions, we define a data model called PDDL2Model to store the planning data for the service composition in PDDL style as shown in Figure 6-4. Concretely, a planning class consists of a *Domain* class and *Problem* class. Every *Domain* class contains many *Actions* specified by their *Parameters*, *Preconditions*, *Effects* and *QoS Metric*. It is worth noting that to enhance the expression of QoS criteria, the original PDDL 3 model is extended with additional class *QoS Metric* where the detailed

information about QoS semantics, such as the unit, the data type, the value, etc. are stored. Moreover, due to the reason that the involved category data might be used to reduce the searching space for the service discovery as we mentioned in Section 5.4, in our model, we present awareness of the current category information of a concept with a special *categoryIsAvailable()* predicate. For instance, if a service takes concept A as an input and generates concept B as an output, *categoryIsAvailable(A)* will be added to *Precondition* and *categoryIsAvailable(B)* will be added to *Effect*. Every *Problem* class contains a collection of initial states and goal states. We distinguish the soft goals from strong goals using the newly introduced concepts named preference and constraint in PDDL 3. For example, “we would like to get a plan with maximum throughput and availability”. In this case, throughput and availability with a given weight values will be added to *Preference* class. The final goal should be the combination of both preferences and constraints.

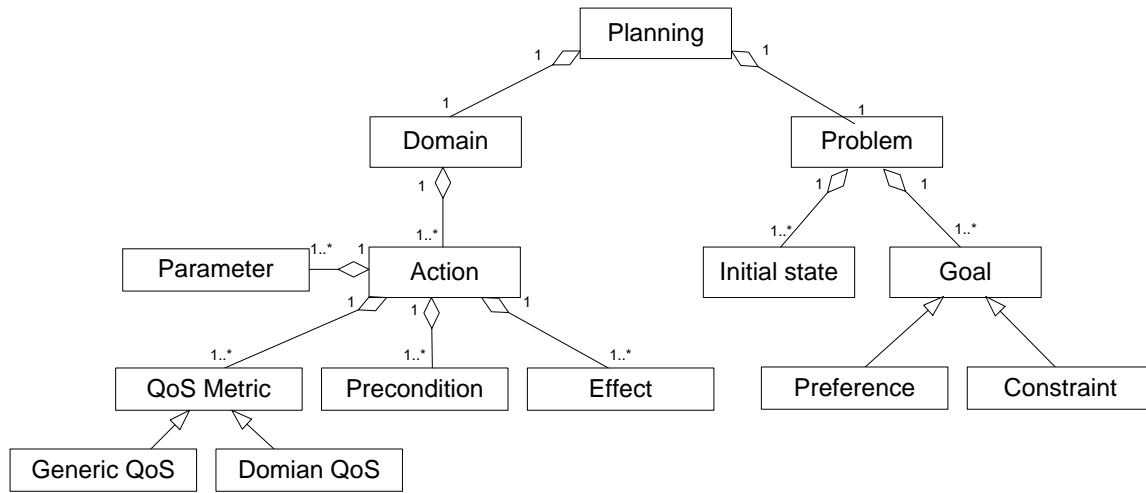


Figure 6-4 Simplified class diagram for the PDDL2Model

Figure 6-5 presents correlating notions of the conceptual model of QWSMO-Lite and PDDL2Model. A set of annotated services $W = \{w | w = (F, NF, cat_w)\}$ defined in Definition 4-2, where an annotated function set $F = \{f | f = (C_{in}, C_{out}, pre, eff, cat_f, q_f)\}$ is defined in Definition 4-1 as shown on the upper left side. The users' composition requests are specified as a simple service also in the flavor of QWSMO-Lite shown on the lower left side. On the left side is the target PDDL2Model. The dash lines in between represents how to map between each other. Specifically, each *Operation* of the QWSMO-Lite corresponds to an *Action* in the domain file. The concepts captured from inputs/outputs are converted to PDDL types. Moreover, QoS semantics either generic specified at the service level or domain specific available in each operation are transformed as functions in the domain file and are assigned values in the *Action* part. Similarly, input concepts along with their categories and output concepts can be directly transformed to the *Initial* part and *Goal* part of the problem file respectively. Furthermore the users' QoS requests can be converted to *Goal* part as well with *Preference* referring to soft requirements and *Constraints* for the strong requirements.

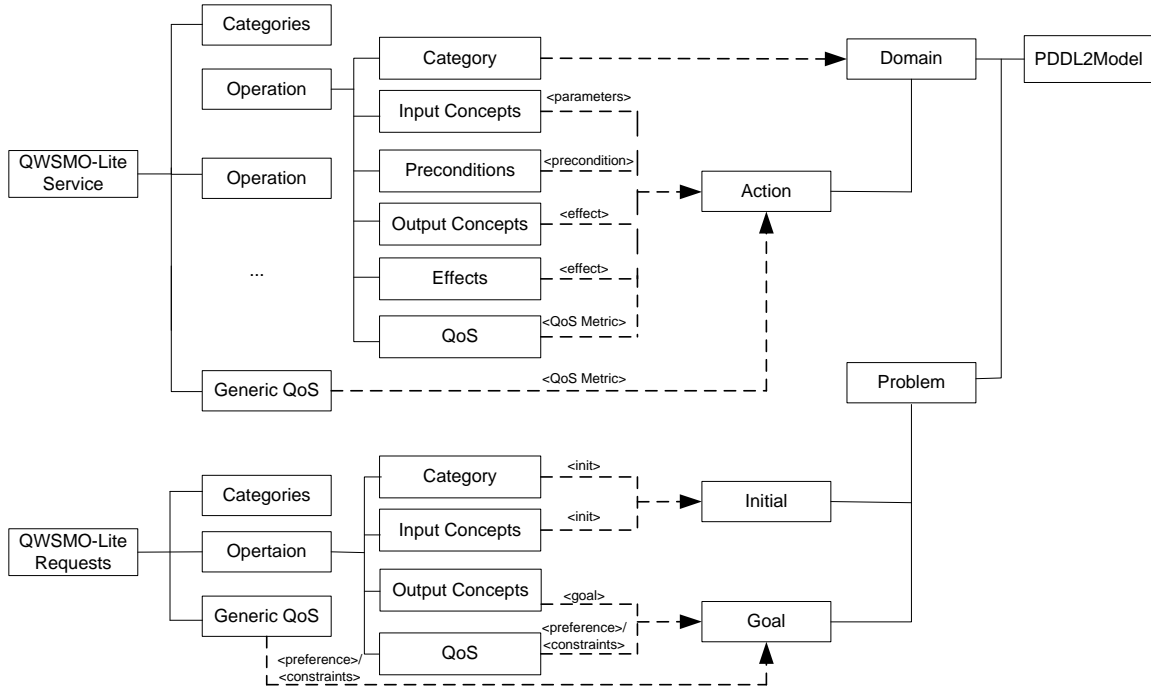


Figure 6-5 Mapping from QWSMO-Lite to PDDL2Model

Based on the model above, the planning problem $\Pi = \langle A, S_0, G \rangle$ in our system can be formally defined as follows:

Definition 6-1 (Planning Action A). Given an annotated service $W = \{w | w = (F, NF, cat_w)\}$, where $F = \{f | f = (C_{in}, C_{out}, pre, eff, cat_f, q_f)\}$, an action for the Web Service composition is described as a tuple $a = (O_{in}, pre_a, O_{out}, eff_a, q)$ with

$O_{in} = \langle Cat, C \rangle$: $O_{in}.Cat \leftarrow f.pre, O_{in}.C \leftarrow f.C_{in}$, : is a set of concepts referred to input parameters of a function f ,

$pre_a \leftarrow categoryIsAvailable(O_{in})$: is the precondition to invoke a function f ,

$O_{out} = \langle Cat, C \rangle$: $O_{out}.Cat \leftarrow eff, O_{out}.C \leftarrow f.C_{out}$, : is a set of concepts referred to output parameters of a function f ,

$eff_a \leftarrow categoryIsAvailable(O_{out})$: is the effect after executing f ,

$q \leftarrow f.q_f$: specify QoS metric. ■

Definition 6-2 (Planning Initial States S_0). Given an annotated service $W = \{w | w = (F, NF, cat_w)\}$, where $F = \{f | f = (C_{in}, C_{out}, cat_f, q_f)\}$, the initial states of the planning requests are described as a tuple $S_0 = (Cat, C)$ with

$Cat \leftarrow Null$: category information is not available from the requirements,

$C \leftarrow f.C_{in}$: is the set of input concepts referred to the input parameters of f . ■

Definition 6-3 (Planning Goal States G). Given an annotated service $W = \{w | w = (F, NF, cat_w)\}$, where $F = \{f | f = (C_{in}, C_{out}, cat_f, q_f)\}$, the goals for Web Service composition is described as a tuple $G = (S_g, G_{pref})$ with

$S_g = \langle Null, f.C_{out} \rangle$: is a set of concepts referred to the output parameters of an unknown function f ,

$G_{pref} \rightarrow f.q_f$: is the QoS requirement of the goal. ■

Taking the scenarios in Section 6.2 as an example again, Figure 6-6 illustrates how to map the operations and requests to PDDL2Model. To simplify the problem, only one operation named “GetIPCountry” of the service “ip2loc” was mapped to PDDL2Model. Here, each action is identified by a name which is a combination of the service name and its operation’s name. A list of “set” functions implemented in the *Action* class, *InitialStates* class and *GoalStates* class is used to instantiate the object.

<pre> <wsdl:operation name="GetIPCountry"> <sawSDL:attrExtensions sawSDL:modelReference="template#Precondition"/> <sawSDL:attrExtensions sawSDL:modelReference="template#Effect"/> <sawSDL:attrExtensions sawSDL:modelReference="template#Category"/> <sawSDL:attrExtensions sawSDL:modelReference="template#QoS"/> <wsdl:input message="tns:GetIPCountrySoapIn"/> <wsdl:output message="tns:GetIPCountrySoapOut" /> </wsdl:operation> ... <wsdl:operation name="GetKML"> <sawSDL:attrExtensions sawSDL:modelReference="template#Category"/> <sawSDL:attrExtensions sawSDL:modelReference="template#QoS"/> <wsdl:input message="tns:GetKMLSoapIn"/> <wsdl:output message="tns:GetKMLSoapOut" /> </wsdl:operation> </pre>	<pre> public Action MappingAction() { ReadWebService read = new ReadWebService(ip2locwsdl); Action action = new Action(); action.setName(read.getInputs().getName()); action.setParamter(read.getInputs()); action.setPrecondition(categoryIsAvailable(read.getInputs())); action.setEffect(read.getOutputs()); action.setEffect(categoryIsAvailable(read.getOutputs())); action.setQoSMetric(read.getQoS()); return action; } public InitialStates MappingInit() { ReadWebService read = new ReadWebService(Requestwsdl); InitialStates init = new InitialStates(); init.setInputs(read.getInputs()); init.setCates(read.getCates()); return init; } public GoalStates MappingGoal() { ReadWebService read = new ReadWebService(Requestwsdl); GoalStates goal = new GoalStates(); goal.setCons(read.getOutputs()); goal.setPref(read.getQoS()); return goal; } </pre>
--	--

Figure 6-6 An example of mapping to PDDL2Model

6.4 GraphPlan Algorithm

The GraphPlan algorithm is based on constructing and analyzing a compact structure so-called Planning Graph. Unlike the state-space graph, where a plan is a path through the graph, in Planning Graph, a plan is a kind of flow in the network flow sense [Blum and Furst, 1997]. GraphPlan algorithm exploits the information captured in such a Planning Graph by alternating between two main steps which are graph expansion and solution extraction. Briefly speaking, in the phase of graph expansion, the Planning Graph is extended until a necessary condition of plan existence is achieved, where all goals are stratified by the current available states. Then solution extraction is started to search for a valid plan that solves the problem

with a backward-chaining strategy. If no solution is found, the whole process will be repeated by further expanding the Planning Graph. It is also worth pointing out that the GraphPlan is characterized by the features of soundness, completeness, generation of shortest plans and termination on unsolvable problems. The details of these two steps will be introduced in the rest of this section.

6.4.1 Planning Graph

Mathematically, a Planning Graph is a directed layered graph containing two types of nodes, namely proposition nodes and action nodes. The layers alternate between these two kinds of nodes. Figure 6-7 shows an example of Planning Graph for the scenario above. It starts with a proposition layer denoted as P_0 consisting of one node for each proposition in the initial states. The second layer is an action layer, A_0 , containing possible action nodes whose preconditions are satisfied by the proposition nodes in P_0 . In our example, W_1, W_2, W_3, W_4, W_5 are added into A_0 , since their preconditions are present in P_0 . Similarly, a proposition layer P_1 will be the third layer which comprises the propositions nodes in P_0 and the proposition nodes representing the effects of action nodes in A_0 . In our use case, P_1 is updated with $\{Country, Latitude, Longitude, State\}$ which are effects of those newly added actions. According to the proposition nodes in P_1 , the operation “GMLLatLonList” of W_5 is available in the new action layer A_1 . In the followed proposition layer P_2 , the proposition nodes are updated with “KML”. Since all goal states are now reachable, the construction of the Planning Graph stops in the current layer P_2 .

Moreover, the relations between action nodes and proposition nodes are defined by three kinds of edges. An action node in A_i is connected by incoming precondition edges to its preconditions in P_i , is connected by add-edges to its add-effects in P_{i+1} and by delete-edges to its delete-effects in P_{i+1} . Take W_2 in A_0 as an example, the precondition edge shown as a blue line in Figure 6-7 connects to its precondition “City” in the current available proposition layer P_0 , while the add-edges show also in a blue line connect to its add-effects, $\{Latitude, Longitude\}$, in the next proposition layer P_1 .

Furthermore, mutual exclusion relations are defined in a Planning Graph for action nodes and proposition nodes in the same layer. That means if two actions or two propositions in a given action layer and proposition layer respectively cannot be contained in a valid plan, these two actions and propositions are mutually exclusive. The possible mutual relations between two action nodes a_i^k, a_j^k in the k^{th} action layer are interference, where a_i^k (or a_j^k) deletes a precondition or an add-effect of a_j^k (or a_i^k) and competing needs, where p , a preconditions of a_i^k is mutually exclusive with q , a precondition of a_j^k in P_k . A mutex relation holds between two propositions p_i^k, p_j^k , if p_i^k (or p_j^k) is the negation of p_j^k (or p_i^k) or if all ways of achieving these two propositions are pairwise exclusive.

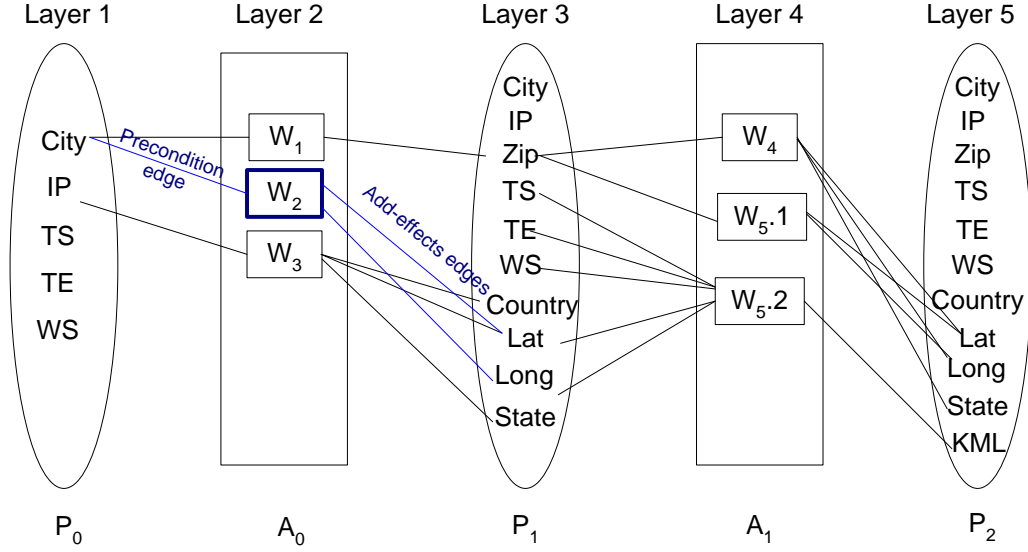


Figure 6-7 An example of a Planning Graph

6.4.2 Graph Expansion

Graph Expansion aims to generate a Planning Graph which guides its search for a plan in the next phase.

Let $\Pi = \langle A, S_0, G \rangle$ be a planning problem as presented in Section 6.3, the Graph expansion algorithm aims to grow the Planning Graph, until all goal propositions are present. In the end of this stage, a Planning Graph which is a sequence of layers of nodes and mutex pairs are obtained: $PG = \{P_0, A_0, \mu A_0, P_1, \mu P_1, \dots, P_i, \mu P_i\}$. It is worth noting that such Planning Graph does not depend on the goal states, G . The same Planning Graph will be used for different planning problems that have the same set of actions A and initial state S_0 .

The expansion starts from the initial state S_0 . The procedure includes the generation of a set of actions A_i , mutex actions μA_i , proposition nodes P_{i+1} and mutex propositions μP_{i+1} . The action layer A_i is generated by adding action nodes a , whose preconditions are present at current proposition layer and no two of them are exclusive: $A_i \leftarrow \{a \in A \mid \text{precond}(a) \subseteq P_i \text{ and } \text{precond}(a) \cap \mu P_i = \emptyset\}$. The detected exclusive actions are added to μA_i . Creating a proposition layer P_{i+1} is performed by adding both add-effects and delete-effects of the inserted actions in A_i : $P_{i+1} \leftarrow \{p \mid \forall a \in A_i: p \in \text{effect}^+(a) \text{ or } p \in \text{effect}^-(a)\}$. Similarly, the exclusive propositions are added to μP_{i+1} . The expansion terminates if all goal states appear in the proposition layer or the graph reaches a fixed-point level where two consecutive layers are identical.

To keep the matters simple, Figure 6-7 illustrates an example expansion procedure without taking account of the mutex relations between actions and propositions. It starts with a set of known concepts listed in P_0 , the Web Services whose preconditions are available in P_0 are added to create the action layer A_0 , the expansion continues, until P_2 is generated where all goals are present.

6.4.3 Solution Extraction

When the Planning Graph reaches a proposition level where all goal propositions are present and none are pairwise mutex, the search for a solution plan proceeds with layer-by-layer backward-chaining approach.

In particular, given a Planning Graph, $G = \langle P_0, A_0, \mu A_0, P_1, \mu P_1, \dots, P_i, \mu P_i \rangle$, where the goals for time i , $g_i = S_g$, $S_g \in P_i$ and $S_g \cap \mu P_i = \emptyset$, the solution extraction procedure starts from the proposition layer at time i and attempts to look for a set of actions at time $i - 1$, $\pi_{i-1} \in A_{i-1}$ that achieve all these goals, S_g as add-effects and are not exclusive with any other actions already selected. Afterwards, the preconditions of these actions in π_{i-1} make up new goals at the time $i - 1$: $g_{i-1} = \{p | \forall a \in \pi_{i-1}: p \in \text{precond}(a)\}$. This procedure continues, until layer 1 is successfully reached. Then a solution plan is obtained $\{\pi_0, \pi_1, \dots, \pi_{i-1}\}$ which is the corresponding sequence of actions. This strategy is based on the idea that if those newly formed goals can be achieved at time $j - 1$, then the original goals can be solved at the time j . In the case that the goals in $j - 1$ cannot be satisfied, then the algorithm will backtrack over other combinations of actions of A_{j-1} which can achieve the goals at time j . If there is no solution found by time i and the PlanningGraph is not leveled off, then the Graphplan's algorithm will extend the PlanningGraph with additional actions and a proposition layer, and then tries to extract the solution again.

To make this more clearly, let us consider the meteorological problem above again. According to the Planning Graph generated in the first phase shown in Figure 6-6, we notice that $S_g = \{KML\}$ is in P_2 without mutex. The only action in A_1 that achieves S_g is the operation "GMLLatLonList" denoted as $W_5.2$, hence $\pi_1 = \{W_5.2\}$. At time 1, the preconditions of the action in π_1 form the new goal: $g_1 = \{Lat, Long, TS, TE, WS\}$. Since $\{TS, TE, WS\}$ are available in S_0 , the possible action set for achieving this goal is $\pi_0 = \{W_2\}$. The new goals become: $g_0 = \{City\}$ which are present in P_0 already. Now the layer at time 0 is successfully reached. On the other words, the solution plan for solving this problem is: $\Pi = \{\{W_5.2\}, \{W_2\}\}$.

6.5 Enhancements of GraphPlan in SEwsPL

As introduced in the Section 6.4, the GraphPlan's algorithm is characterized by the features of soundness, completeness, generation of shortest plans and termination on unsolvable problems. Furthermore, those Planning Graphs which reveal constraints with polynomial size can be built in polynomial time. Accordingly, in our SEwsPL planner, GraphPlan's algorithm is adopted to find the solution plan efficiently. However, besides the lack of quality which has been addressed by the semantic annotations mentioned in Chapter 4 and semantics matchmaking introduced in Chapter 5, the original GraphPlan still has some limitations:

- **Irrelevant information in the Planning Graph.** Due to the forward-chaining used in GraphPlan for the expansion of the Planning Graphs, the size of the graph only depends on the number of initial states and the available actions. Too much irrelevant information will be contained in the Planning Graphs in the case that the many actions and their propositions are available, but only a few subsets of those actually correspond to the goals, which will probably lead to a state-space explosion during the

graph expansion phase of the planning. [Kambhampati et al., 1997][Parker, 1999][Gupta et al., 2007][Feng and Sun, 2010].

- ***Redundant search in the solution extraction phase.*** The original GraphPlan's algorithm for the extraction of solution plans is quite time-consuming. The big source of inefficiency in the search algorithms is the branching factors of the graph [Kambhampati, 1999] [Ghallab et al., 2004] [Yan and Zheng, 2008], since the backward search may try a redundant search among actions or propositions that cannot be reached from the initial states.

This section addresses these problems by providing a so-called *SEwsPL* planner which is an enhancement of the original GraphPlan in both graph expansion phase and solution extraction phase. In the rest of this section, these enhancements will be introduced in details.

6.5.1 Simplified Ordered Planning Graph

Before discussing the details of the algorithm, let us consider the specific features of Planning Graphs used in Web Service Composition domain first.

As we introduced before, in a classical Planning Graph, each action may have both add and delete effects. For instance, the add-effect of the action “unstack block A from block B” in the block world will be “block A is on the table”, while its delete-effect might be “there is no block on block B”. However, in the Web Service Composition domain, each Web Service only has a set of add-effects.

Moreover, since there is no action or proposition which is the negation of the other, it is not necessary to consider the mutual relation between either action nodes or proposition nodes in the Web Service paradise.

Furthermore, in the original Planning Graphs, since all selected actions are exactly matched with the given proposition nodes, there is no need to express additional information about the degree of matching in the graph. However, for automated Web Service Composition such matching information plays a significant role in recognizing the most reasonable solution plans.

Various approaches have been proposed to define the Planning Graph in the Web Service Composition domain. In [Yan and Zheng, 2008], such a Planning Graph is defined as a Simplified Planning Graph with the definition shown below:

“A simplified planning graph is a planning graph where each pair of actions is independent and no mutex relations exist between actions or propositions.”

Unfortunately, the Simplified Planning Graph still lacks the matching semantics. In [Li et al., 2010], a **W**eighted **P**lanning **G**raph (WPG) is proposed. The WPG is constructed by adding the degree of service matching as the weight to the add-effect edges of the original Planning Graph. As a result, it facilitates the simplification of the graph by avoiding adding the actions whose effects are already present in the proposition layer, and their weights are less than the previous added actions. However, as pointed out by the authors, the matching weight is calculated without considering the non-functional natures of the services. On the other hand, how to reduce the time to build such a WPG is another issue needed to be taken into account

in their future work, since when adding a new action much more time will be spent to compare the weights of different effects.

In *SEwsPL*, we define the Planning Graphs as a **S**implified **O**rdered **P**lanning **G**raph (SOPG) shown in Figure 6-8. A SOPG grows with a sequence of ordered *Action nodes* and *Proposition nodes*. Unlike the classic Planning Graph, only two kinds of edges are available in SOPG, namely *Precondition edge* and *Add edge*, since there are no delete-effects in Web Services. Each action node is connected to the current and the next proposition layer with a set of associated *Precondition edges* and *Add edges*. It also should be noted that to facilitate the discovery of actions, Category information is also available in SOPG.

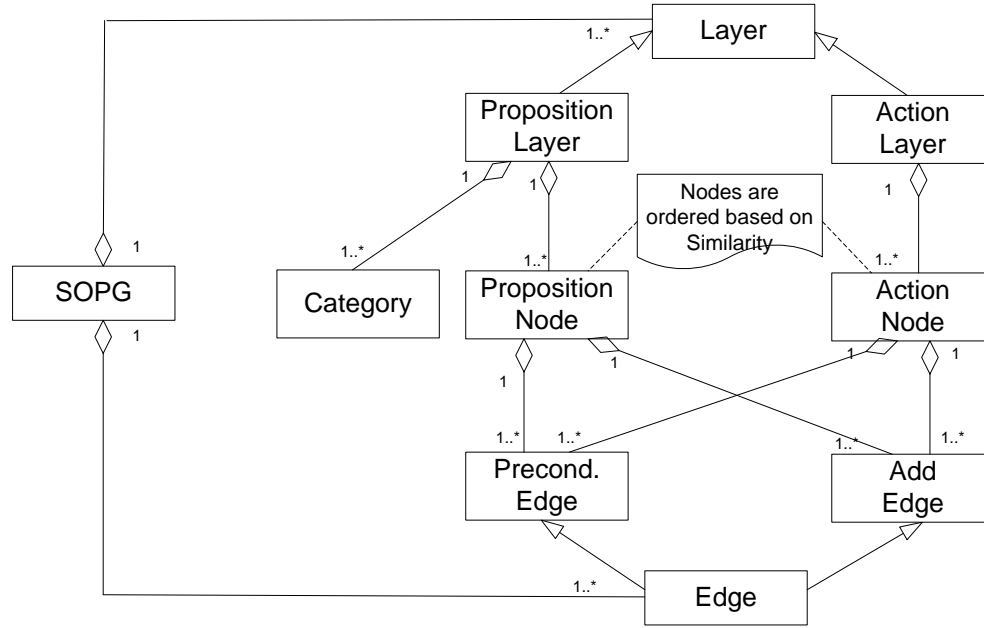


Figure 6-8 Data structure of SOPG

To facilitate the extension of the planning graph, we distinguish the forward graph from the backward graph. Their definitions are presented as below:

Definition 6-4 (A Forward SOPG: F-SOPG). Given a planning problem $\Pi = \langle A, S_0, G \rangle$ defined by Definition 6-1, 6-2 and 6-3. *F-SOPG* is a layered graph where the layers of nodes form an alternating sequence of propositions and actions: *F-SOPG* = $\langle FP_0, FA_0, FP_1, FA_1, \dots, FA_{k-1}, FP_k \rangle$ where $FP_i = \langle Cat, N_{fp} \rangle, FA_i = \langle N_{fa} \rangle$

$Cat \leftarrow \{cat | \forall a \in FA_{i-1}: cat \in a.O_{out}.Cat\},$

$N_{fp} = \langle n_{fp} = \langle concept, Pre, Eff \rangle \rangle$ is a set of Forward Proposition Node:

$$\begin{cases} concept \leftarrow c, \exists a \in FA_{i-1}, c \in a.O_{out}.C \\ Pre \leftarrow \{a | \exists a \in FA_{i-1} \text{ and } concept \in a.O_{out}.C\} \\ Eff \leftarrow \{a | \exists a \in FA_i \text{ and } concept \in a.O_{in}.C\} \end{cases}$$

$N_{fa} = \langle n_{fa} = \langle action, Pre, Eff \rangle \rangle$ is a set of ordered Forward Action Node:

$$\begin{cases} \text{action} \leftarrow a, a \in A, \exists n_{fp} \in FP_i. N_{fp}, \forall c \in \text{action}. O_{in}. C, c \supseteq n_{fp}. \text{concept} \\ \text{Pre} \leftarrow \{n_{fp} | \forall n_{fp} \in FP_i. N_{fp} \text{ and } n_{fa}. \text{concept} \subseteq \text{action}. O_{in}. C\} \\ \text{Eff} \leftarrow \{n_{fp} | \forall n_{fp} \in FP_{i+1}. N_{fp} \text{ and } n_{fa}. \text{concept} \subseteq \text{action}. O_{out}. C\} \end{cases} \quad \blacksquare$$

Definition 6-5 (A Backward SOPG: B-SOPG). Given a planning problem $\Pi = \langle A, S_0, G \rangle$ defined by Definition 6-1, 6-2 and 6-3. $B - SOPG$ is a layered graph where the layers of nodes form an alternating sequence of propositions and actions: $B - SOPG = \langle BP_0, BA_0, BP_1, BA_1, \dots, BA_{k-1}, BP_k \rangle$ where $BP_i = \langle N_{bp} \rangle$, $BA_i = \langle N_{ba} \rangle$:

$N_{bp} = \langle Cat, C \rangle$ is a set of Backward Proposition Node:

$$Cat \leftarrow a. O_{in}. Cat, \exists a \in BA_{i-1},$$

$C = \langle \text{concept}, Pre, Eff \rangle$ is a list of concepts of a :

$$\begin{cases} \text{concept} \leftarrow c, c \in a. O_{out}. C \\ Pre \leftarrow \{a | \exists a \in FA_{i-1} \text{ and } \text{concept} \in a. O_{in}. C\} \\ Eff \leftarrow \{a | \exists a \in FA_i \text{ and } \text{concept} \in a. O_{out}. C\} \end{cases}$$

$N_{ba} = \langle n_{ba} = \langle \text{action}, Pre, Eff \rangle \rangle$ is a set of ordered Backward Action Node:

$$\begin{cases} \text{action} \leftarrow a, a \in A, \exists n_{bp} \in BP_i. N_{bp}, \forall c \in a. O_{out}. C, c \supseteq n_{bp}. \text{concept} \\ Pre \leftarrow \{n_{bp} | \forall n_{bp} \in BP_i. N_{bp} \text{ and } n_{ba}. \text{concept} \subseteq \text{action}. O_{in}. C\} \\ Eff \leftarrow \{n_{bp} | \forall n_{bp} \in BP_{i+1}. N_{bp} \text{ and } n_{ba}. \text{concept} \subseteq \text{action}. O_{out}. C\} \end{cases} \quad \blacksquare$$

These two kinds of the Planning Graphs, illustrated in Figure 6-9, aim to capture the information from the initial states and goal states respectively. Unlike an action layer and a proposition layer in a classic Planning Graph, in a SOPG, the action nodes of an action layer are arranged according to their degrees of matching with the current proposition nodes. Meanwhile, a proposition layer consists of not only the proposition nodes obtained from the previous action layer, but also the information of the associated categories. The advantage of creating such additional knowledge in the graph is to recognize the related actions among a large amount of available actions more efficiently. Details of the discovery procedure will be discussed in the next section.

Owing to the various ways to build the Planning Graph, discrimination between F-SOPG and B-SOPG is listed below:

- Starting and end points: Building a F-SOPG starts from the initial situation and stops when all its goal states are reachable, while creating a B-SOPG begins with the goal states and terminates when it reaches the initial states.
- The semantics of an operation: Intuitively, the semantics of an operation in F-SOPG is that an operation is only applicable if its preconditions are satisfied by the current proposition nodes. However, in B-SOPG, an available action means there is at least one of its effects presenting in the current proposition layer.
- The construction of a proposition layer: In F-SOPG, the proposition layer is updated by adding the effects of action nodes in the previous action layer to the previous proposition layer. Nonetheless, in B-SOPG, the new proposition layer is created by only the associated preconditions and categories of the actions in the previous action

layer. Moreover, a proposition layer in F-SOPG contains a set of categories and concepts, while in SOPG, a proposition layer consists of a sequence of sets of categories and concepts. With such structure, the size of the graph can be easily reduced when applying the expansion algorithm. Details will be given in the next section.

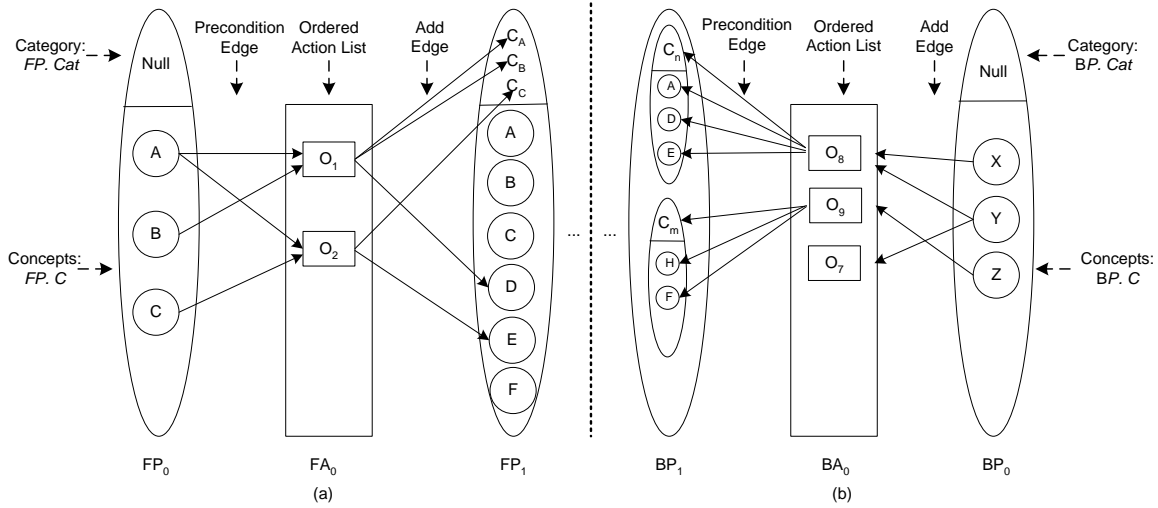


Figure 6-9 Examples of SOPG: (a) F-SOPG (b) B-SOPG

6.5.2 Goal-Oriented Bi-directional Graph Expansion Algorithm

As we mentioned above, one of the limitations of the traditional GraphPlan is the possible irrelevant action and proposition nodes generated in the Planning Graph. The main reason behind it is because of the forward chaining strategy applied in the expansion phase. The goal-directed planner called Bsr-graphplan [Parker, 1999] expands its graph from the target sets with backward chaining. Although it is capable of reducing the size of the Planning Graph, it makes the solution incomplete. Accordingly, to build an efficient Planning Graph, the trade-off between the forward and the backward chaining needs to be addressed.

A novel bidirectional expansion algorithm based on the Simplified Ordered Planning Graph is applied in *SEwsPL*. Let $\Pi = \langle A, S_0, G \rangle$ be a planning problem as presented in Section 6.3 such that S_0, G are the initial states and the final states respectively and A is a set of available operations. A forward expansion of F-SOPG starting initially from S_0 and a backward expansion of B-SOPG from the goal states are executed alternatively. Such repeatable process will continue until there is no active proposition node existing in the B-SOPG or these two graphs are leveled off.

Algorithm 1 shown in Figure 6-10 is the main algorithm for the expansion procedure. First, FP_0, BP_0 , the proposition layers of an F-SOPG and a B-SOPG are initialized with the initial conditions and goals respectively. A comparison function between these two proposition layers presented in Figure 6-11, is then proceeded to reduce the number of involved active proposition nodes in the B-SOPG by removing those proposition nodes already achieved in the F-SOPG.

Second, a forward expansion algorithm, FExpand, and a backward expansion algorithm, BExpand, are used to expand the F-SOPG and B-SOPG iteratively. It is worth noting that these two algorithms expand the graph with different paces. Specifically, the backward expansion is much faster than the forward expansion. An F-SOPG grows dramatically by adding the effects of the newly added actions to the proposition layer, while the size of a B-SOPG increases relative slowly by creating a new proposition layer. Hereby, slowing down the forward expansion and speeding up the backward expansion will help to reduce the size of the whole search space. In addition, within each loop, to control the size of the B-SOPG, the current proposition layer of B-SOPG is checked with the comparison function by removing those redundant propositions.

The expansions of the F-SOPG and the B-SOPG continue until they are subject to the termination conditions. In our algorithm, three kinds of loop termination conditions are defined as below:

- Both directions of the expansion terminate, if current proposition layer of the B-SOPG is empty: $BP_j = \emptyset$. It means that all proposition nodes in the B-SOPG are available in the F-SOPG.
- The forward expansion, FExpand, terminates, if the current F-SOPG reaches a fixed point where two consecutive layers are identical: $FP_j = FP_{j+1}$.
- The backward expansion, BExpand, terminates, if the current B-SOPG is leveled off: $BP_j = BP_{j+1}$.

Algorithm Expand(A, S_0, G)

Inputs: a set of available actions A ,
initial states, S_0
goal states, G

Data: pace of the backward expansion, β

Output: a F-SOPG, $FG = \langle FP_0, FA_0, \dots, FP_i \rangle$
a B-SOPG, $BG = \langle BP_0, BA_0, \dots, BP_j \rangle$

Begin:

```

 $i \leftarrow 0, j \leftarrow 0, FP_i \leftarrow S_0, BP_j \leftarrow G.S_g, BP_j \leftarrow \text{CompareProps}(FP_i, BP_j)$ 
 $F\text{-go} \leftarrow \text{true}, B\text{-go} \leftarrow \text{true},$ 
while ( $BP_j.size() > 0 \parallel (F\text{-go} \ \&\& \ !B\text{-go}) \parallel (!F\text{-go} \ \&\& \ B\text{-go})$ ) do
  if  $F\text{-go}$  then  $FG \leftarrow \text{FExpand}(FG)$ 
  if  $B\text{-go}$  then  $BG \leftarrow \text{BExpand}(FG, BG, \beta)$ 
  update  $i, j$ 
   $BP_j \leftarrow \text{CompareProps}(FP_i, BP_j)$ 
  if (Fixedpoint ( $FG$ )) then
     $F\text{-go} \leftarrow \text{false}$ 
    continue
  end if
  if Fixedpoint ( $BG$ ) then
     $B\text{-go} \leftarrow \text{false}$ 
    continue
  end if
end while
return  $FG, BG$ 

```

End

Figure 6-10 A pseudo-code description of the bidirectional expansion

Algorithm CompareProps(FP_i, BP_j)

Inputs: a proposition layer in F-SOPG, FP_i
a proposition layer in B-SOPG, BP_j
Data: a set of category, Cat
a set of concept, C
Output: an updated proposition layer of B-SOPG, BP_j
Begin:
 foreach (Cat, C) in BP_j **do**
 if there is no related category between cat and $FP_i.Cat$ **then**
 continue
 else foreach concept c in C **do**
 if c presents in $FP_i.C$ **then**
 remove c from BP_j
 end if
 end for
 end if
end for
return BP_j
End

Figure 6-11 The comparison part of the expansion algorithm

Figure 6-12 shows the forward expansion and backward expansion sections of the algorithm $\text{Expand}(A, S_0, G)$. These two expansion procedures correspond to generate $FA_i, FP_{i+1}, BA_i, BP_{i+1}$, from the current proposition layer FP_i and BP_i respectively.

Specifically speaking, FA_i and BA_i are generated according to the FMatch and BMatch algorithms which are TOPSIS based matchmaking algorithms illustrated in Chapter 5.4. These matching functions allow searching the related operations in a large operation repository and ordering found operations in terms of a multi-attributes similarity calculated by TOPSIS. The only difference between them is the rule by which they discover the succeeding actions. For the forward matchmaking, all the inputs of an action should be satisfied by the current propositions, while in the backward matchmaking, the valid succeeding actions are those actions whose effects are completely or partly available in the proposition layer.

Adding the new proposition layer FP_{i+1} involves two steps. In the first step, all the category information of the newly added actions are inserted into the category set of the previous proposition layer FP_i . Afterwards, the propositions representing the effects of those new actions are added to the concept set of FP_i . Compared to the strategy of generating a forward proposition layer, rather than updating the previous layers with new elements, the BP_{i+1} is a completely new layer containing only the related concepts and category information of the action in BA_i .

Furthermore, it is worth noting that as we discussed above, in order to reduce the size of the graph, these two graphs are allowed to be expended at different paces. Since the size of F-BOSG increases faster than the size of a B-SOPG, in our expansion algorithm, the forward expansion is slowed down, while the backward expansion is speeded up. To this end, we introduce an expansion pace, *beta* to the BExpand procedure. It is a user-defined parameter to control the pace of backward expansion.

Algorithm FExpand(FG)

Inputs: Current F-SOPG, $FG = \langle FP_0, FA_0, \dots, FP_i \rangle$
Data: a set of available operations, A
 Goal states, G
Output: a F-SOPG, $FG = \langle FP_0, FA_0, \dots, FP_i, FA_i, FP_{i+1} \rangle$
Begin:
 $FA_i \leftarrow \text{FMatch}(FP_i, A, G.\text{Gpref})$
for all a in FA_i **do**
 $FP_{i+1}.\text{Cat} \leftarrow FP_i.\text{Cat} \cup \{a.O_{out}.\text{Cat}\}$
 $FP_{i+1}.C \leftarrow FP_i.C \cup \{a.O_{out}.C\}$
end for
return FG
End

(a)

Algorithm BExpand(FG, BG, beta)

Inputs: Current F-SOPG, $BG = \langle BP_0, BA_0, \dots, BP_i \rangle$
 Expansion pace, beta
Data: a set of available operations, A
 Goal states, G
Output: a B-SOPG, $BG = \langle BP_0, BA_0, \dots, BP_i, BA_i, BP_{i+1} \rangle$
Begin:
 $BA_i \leftarrow \text{BMatch}(BP_i, A, G.\text{Gpref})$
while beta > 0 **do**
for all a in BA_i **do**
 $BP_{i+1} \leftarrow \{(Cat, C) \mid Cat = a.O_{out}.\text{Cat}, C = a.O_{out}.C\}$
 $BP_{i+1} \leftarrow \text{CompareProps}(FP_i, BP_{i+1})$
end for
 beta = beta - 1
end while
return BG
End

(b)

Figure 6-12 Bidirectional expansion part of the expansion algorithm:
 (a) forward expansion (b) backward expansion

Again referring to the scenario in Table 6-1, Figure 6-13 shows how an F-SOPG and a B-SOPG grows based on the bidirectional expansion. First of all, these two graphs initiate with the initiate states and the goal states respectively. Then the B-SOPG begins to grow at the pace of 2. That means every time, the B-SOPG expands two layers. Recall that during the expansion, the comparison function will be invoked to refine the proposition layer by removing those nodes which are already present in the F-SOPG. Therefore, in this case, we found in BP_i , only $\{Lat, Long\}$ are available. Afterwards, the F-SOPG expands forwards with an action layer, FA_0 and a proposition layer FP_i . Since until now, the entire proposition nodes in the B-SOPG are reached from the F-SOPG, and both expansion processes terminate. Furthermore, it is important to note that, in the action layer of both graphs, action nodes are arranged according to the closeness of the current proposition layer and the use's nonfunctional requirements. In addition, compared to the classic Planning Graph generated by the traditional expansion procedure shown in Figure 6-6 which contains totally 26 proposition nodes, 6 action nodes, the SOPG graphs generated by our bi-directional expansion consists of only 18 proposition nodes, and 8 action nodes. Thus, it can be proved that our solution can successfully reduce the size of the graph in a certain extent.

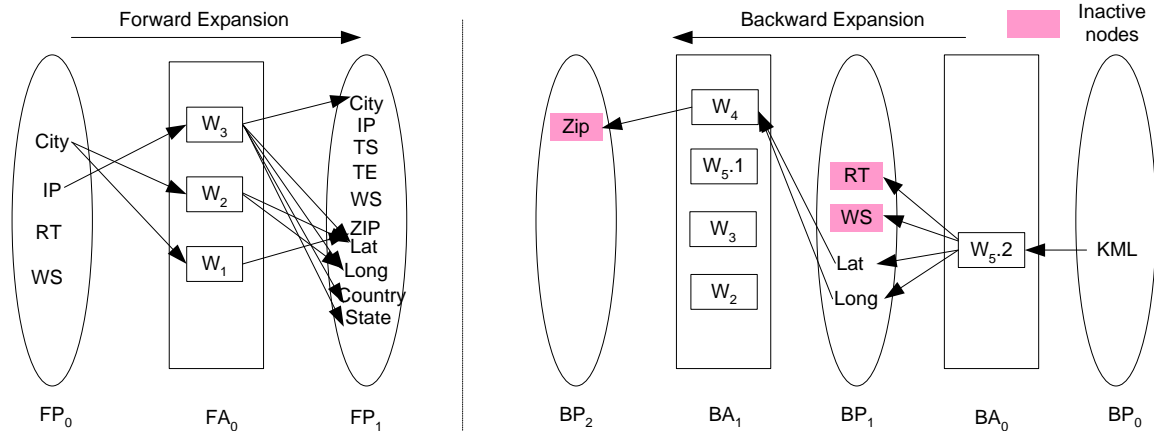


Figure 6-13 An example of bi-directional expansion algorithm

6.5.3 Workflow based Planning Extraction

Assuming that the expansions of F-SOPG and B-SOPG have accomplished, the next activity is to identify a near-optimal plan which involves the following two sub-steps:

- *Solutions Extraction*: helps find all the candidates that satisfy the requirements from the Planning Graphs.
- *The best solution selection*: tries to determine the most optimal solution from all the candidates. In many situations, multiple composition results can be used to achieve the same goal. Thus, an additional step to select the best one in terms of both the functional and QoS requirements is required, which is unfortunately usually ignored by the classic GraphPlan.

Currently, efforts on the solution extraction part are divided into two groups:

- *Improving the extraction strategy*: In the traditional GraphPlan algorithm, a backward-chaining is employed for the solution extraction procedure. However, this backward style of search tends to be quite time-consuming and inefficient, since due to the chronological backtracking, it will lead to a large amount of wasted effort on exploring the same nodes several times during the extraction procedure. Thus, to eliminate this problem, in [Kambhampati, 1999], authors present some augmentations which add explanation-based learning and dependency-directed backtracking capacities to GraphPlan. In Bsr-graphplan [Parker, 1999] and Gdi-graphplan [Feng and Sun, 2010], instead of a backward-chaining, a forward-chaining with a new data structure called constrained tree is applied to discover the solutions. Though it can guarantee the generated plan is goal-directed, it still results in low efficiency, because at each time step, only a single action is considered using such constrained tree structure.
- *Pruning redundant nodes of the Planning Graph*: In [Yan and Zheng, 2008] and [Li et al., 2010], rather than improving the extraction algorithm, they aim at removing redundant Web Services contained from the graph in terms of a series of pre-defined strategies.

To select a desired composition plan, several approaches are proposed. In [Pop et al., 2009], an immune-inspired algorithm is integrated into the AI Planning Graph to find the most appropriate solution satisfying both user's functional and QoS requirements. In [Zhang, et al. 2010], the selection problem is converted to a multi-objective optimization problem. The Ant Colony Optimization (ACO) algorithm is presented to solve this problem. If there is no optimal QoS can be found, then the near-optimal QoS solutions will be selected.

Unfortunately, there is no approach available for these two steps. To bridge this gap, our mechanism is presented aiming at putting these two steps together to identify the most appropriate solution. Briefly speaking, to make a good use of the SOPGs obtained from the previous steps, extracting plans is achieved by removing redundant nodes. Afterward, the workflow perspective will be applied to represent the collaboration among actions. In the rest of the section, our approach will be illustrated in more details.

Firstly, the redundant nodes in the SOPG graphs obtained from the previous step need to be identified. The definitions of redundant actions are given in below:

Definition 6-6 (A redundant action in F-SOPG) Given a Forward SOPG: $F-SOPG = \langle FP_0, FA_0, FP_1, FA_1, \dots, FA_{k-1}, FP_k \rangle$, an action $a_n \in FA_i$ is redundant in F-SOPG, iff $a_n.O_{out}.C = a_m.O_{out}.C$ and $m < n$. ■

Definition 6-7 (A redundant action in B-SOPG) Given a Backward SOPG: $B-SOPG = \langle BP_0, BA_0, BP_1, BA_1, \dots, BA_{k-1}, BP_k \rangle$, an action $a_n \in BA_i$ is redundant in B-SOPG, iff $a_n.O_{out}.C \cap a_m.O_{out}.C \neq \emptyset$ and $m < n$. ■

To better illustrate these two definitions, let's see the example in Figure 6-14. Here, after the expansion procedure, an F-SOPG and a B-SOPG graph are ready for the plan extraction. We figure out that the outputs of O_3 , marked in yellow, is exactly same as the output of O_2 and $Sim(FP_0, O_2) > Sim(FP_0, O_3)$, therefore, according to Definition 6-6, we set O_3 as a redundant action of O_2 . In the meanwhile, this action is stored into a redundant map which will be used for the self-adaptive composition of the plan in the future. The semantics behind it is that only the new propositions can make it possible to achieve the goal.

Regarding O_{11} , marked in yellow in the B-SOPG, since its outputs overlap with the outputs of O_9 and O_{10} which are more related to the goal and have already satisfied all goals, thus, based on definition 6-7, O_{11} can be regarded as a redundant action for both O_9 and O_{10} . Different to the definition of the redundant action defined in F-SOPG, in B-SOPG the redundant action is defined in terms of the associated propositions. Similarly, the redundant information will be stored in a redundant map for the future. The idea behind this is that in the B-SOPG where the graph grows backward from the goal state, at each step, actions can be inserted into the graph, as long as their effects contain at least one goal proposition. Only the preconditions of these newly added actions are helpful to reach the initial states.

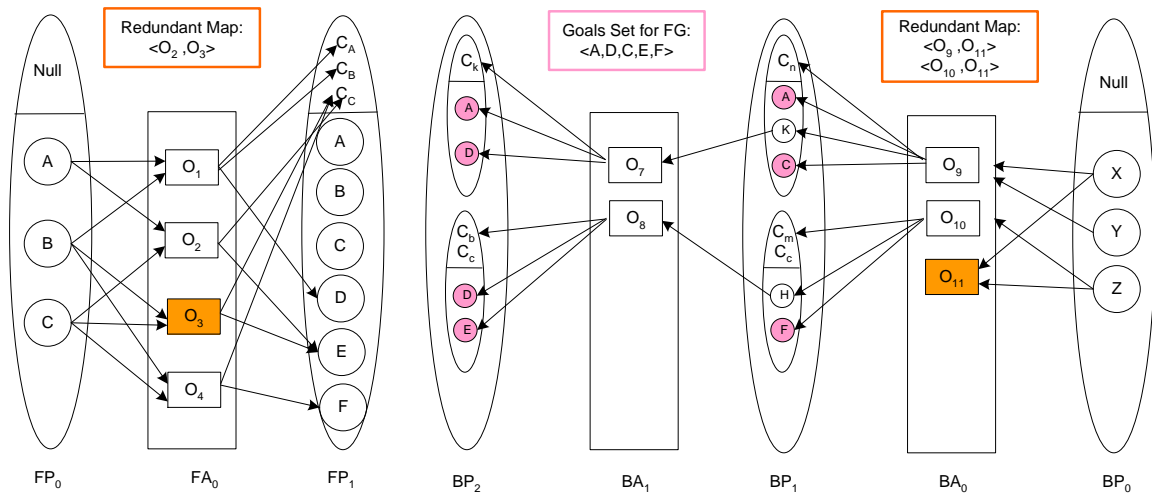


Figure 6-14 Redundant actions in F-SOPG and B-SOPG



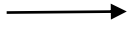


Assuming that the SOPG has been refined by marking redundant nodes, the next step is to extract the most optimal plan from the graph. The main idea is that we transform such extraction procedure to the process of building a workflow which is conceptually similar to a

composition chain. The advantages of using such a workflow based mechanism are listed as follows:

- *Reuse of the existing approaches.* Workflow organization and management have been a major research topic for more than twenty years. Accordingly, a lot of techniques are available to represent a sequence of actions, execute the plan and instead of creating the work from the scratch, some existing solutions can be easily reused.
- *Make the generated plan visible.* It allows visualizing the composition chain in some workflow management systems, such as Taverna¹⁸, Kepler¹⁹, Unicore²⁰, SWIMS [El-Gayyar et al., 2009] [El-Gayyar et al., 2010] which makes the plan more readable and understandable.

In *SEwsPL*, we transform the SOPG to SCUFL language which is now used in Taverna and SWIMS. The mapping between them is illustrated in Table 6-2.

Table 6-2 Mapping between SOPG and SCUFL

Element in SOPG	SCUFL		
	Element	Representation	Diagram
Action	processor	<s:processor name=" " />	
Active action node	arbitrarywsdl	<s:arbitrarywsdl> <s:wsdl> ... </s:wsdl> <s:operation> ... </s:operation> </s:arbitrarywsdl>	
Redundant/Inactive action node	alternate	<s:alternate> <s:arbitrarywsdl> <s:wsdl>... </s:wsdl> <s:operation>... </s:operation> </s:arbitrarywsdl> <s:outputmap key=" " value=" " /> <s:inputmap key=" " value=" " /> </s:alternate>	
Proposition node	source/sink	source=" " / sink=" "	
Edge	link	<s:link source=" " sink=" " />	
Initial state	source	<s:source name=" " />	
Goal state	sink	<s:sink name=" " />	

Let us recall the meteorological scenario in Table 6-1 again, after exploring the bidirectional expansion algorithm, we got the F-SOPG and B-SOPG graphs as shown in Figure 6-13. According to the Definition 6.6 and 6.7, we recognize that W_1 , W_2 are redundant actions W_3 , since all the outputs of W_1 and W_2 can be replaced by W_3 . Thus, $\langle W_3, W_1 \rangle, \langle$

¹⁸ Taverna: <http://www.taverna.org.uk>

¹⁹ Kepler: <https://kepler-project.org/>

²⁰ Unicore: <http://www.unicore.eu>

W_3, W_2 are stored into the redundant map. Now, the transformation procedure begins from the B-SOPG and moves backwards until reaching the FP_0 in the F-SOPG. Specifically, all the proposition nodes in BP_0 are converted to “sink” in the SCUFL. For instance, the proposition “KML” are transformed to `<s:sink name="KML" />`. Then, it moves to BA_0 where $W_{5,2}$ is present. A new process named “GmlLatLonList” is created as follows:

```
<s:processor name="GmlLatLonList">
  <s:description>Returns National Weather Service digital weather forecast data encoded in GML for a
  single time</s:description>
  <s:arbitrarywsdl>
    <s:wsdl>http://www.nws.noaa.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl</s:wsdl>
  </s:arbitrarywsdl>
</s:processor>
```

In the meanwhile, according to the precondition edges and effects edges defined in $W_{5,2}.Pre$ and $W_{5,2}.Eff$, the data flow are specified as a sequence of links as below:

```
<s:link source="RT" sink="GmlLatLonList: RequiredTime " />
<s:link source="WS" sink="GmlLatLonList: PropertyName " />
<s:link source="GmlLatLonList: KML-NDFD " sink="KML" />
```

This process continues until it reaches the last proposition layer BP_2 . The new goal states for the followed F-SOPG are created with all inactive nodes of the B-SOPG. In this case $\{ZIP, RT, WS\}$ are set as new goals. The transformation process moves to the F-SOPG with the new goals starting from the last proposition layer FP_1 . Here, in terms of the proposition nodes’ effect edge set, RT, WS can be regarded as initial states, since there is no effect edges associated with them. Therefore two new sources are generated: `<s:source name="RT" />`, `<s:source name="WS" />`. With regard to “ZIP”, we notice that both W_3 and W_7 have it as an effect. In this case, the algorithm needs to search for the redundant information from the redundant map. As a result, we recognize that W_3 is an active action, while W_7 is a redundant action of it. Such semantics can be represented as:

```
<s:processor name="IP2Location">
  <s:arbitrarywsdl>
    <s:wsdl>http://ws.fraudlabs.com/ip2locationwebservice.asmx?wsdl</s:wsdl>
    <s:operation>IP2Location</s:operation>
  </s:arbitrarywsdl>
  <s:alternate>
    <s:arbitrarywsdl>
      <s:wsdl>http://www.websvcex.com/uszip.asmx?WSDL</s:wsdl>
      <s:operation>GetInfoByCity</s:operation>
    </s:arbitrarywsdl>
    <s:outputmap key=" Location " value="attachmentList" />
    <s:inputmap key=" City name " value=" City name " />
  </s:alternate>
</s:processor>
```

Afterwards, the data flow of W_3 will be specified. The transformation procedure then terminates when it reaches FP_0 .

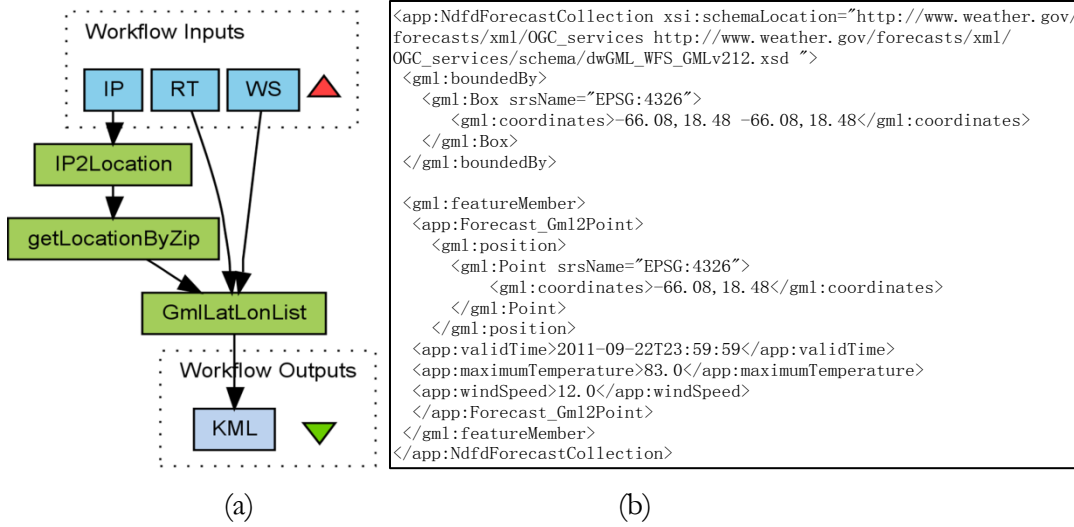


Figure 6-15 The result of the meteorological scenario: (a) the workflow in SCUFL (b) the KML file

6.6 Self-adaptive Composition

Supporting self-adaptation is becoming more and more important due to the dynamic nature of the Web Service environment. For instance, used services may become unreachable or their QoS specifications change frequently. Accordingly, there is a need to enable the Web Service Composition to deal with changing situations. Unfortunately, this kind of requirement hasn't been adequately explored in the most of AI planning techniques which provide only static one-time solution.

Basically, adaptation can be achieved in two ways:

- *Re-planning*: the plan is re-generated from the state where inconsistencies occur. Such as in OWLS-XPlan presented in [Klusch et al., 2005], during the plan execution, in the case when an agent detects that an action's preconditions have not satisfied, the re-planning component is informed of the position of the error and tries to fix the problem by searching for an alternative plan in the connectivity graph from the current position to the goal state.
- *Plan Repair*: rather than building the plan from the scratch, Plan Repair tries to reuse most of the original plan as possible.

Though in theory modifying an existing plan is not more efficient than complete re-planning in the worst case [Nebel and Koehler, 1995], repairing the plan in practice is much better than planning from scratch, since the most part of the plan might still remain valid even if some situation changes as pointed in [Krogt and Weerd, 2005]. On the other hand, from the end-users' point of view, the modified plan might more easily be accepted than a complete new one. Therefore, Plan Repair is adopted in our work to efficiently adapt the plan in a changing world.

In *SEmsPL*, the Plan Repair is based on the F-SOPG and B-SOPG graphs which are generated in the graph expansion phase as presented in Section 6-5. The process starts from

the lowest action layer BA_m in B-SOPG which contains disabled actions. The B-SOPG is then divided into two parts:

- *The head* of the plan consisting of actions that can be executed from the goal state:
 $BG_{head} = \{BP_0, BA_0, \dots, BP_m, BA_m\}$
- *The tail* that contains all inconsistent actions and propositions:
 $BG_{tail} = \{BP_{m+1}, BA_{m+1}, \dots, BP_j, BA_j\}$

During the adaptive algorithm, the tail part is repaired for the changing situations, while the head of the plan will remain unchanged. Specifically, the system tries to find alternative actions of those disabled ones from the redundant map which is generated from the plan extraction procedure. The following proposition layer BP_{m+1} is updated with the preconditions of those newly added actions. Similarly, the adaptive process begins from the lowest action layer PA_m where disabled actions exist. The F-SOPG is then also divided into two parts as follows:

- $FG_{head} = \{FP_0, FA_0, \dots, FP_m, FA_m\}$
- $FG_{tail} = \{FP_{m+1}, FA_{m+1}, \dots, FP_i\}$

The head part is kept in the new plan, and the tail need to be refined. The refinement is to remove all associated edges, action nodes and propositions in the next layers. Finally, we get an updated F-SOPG FG . The last step is to invoke the Expand function as described in the previous section, which takes the set of propositions in the last proposition layer FP_i of F-SOPG as the new initial states and the propositions in BP_{m+1} as the new goal states. The newly generated FG and BG will be the new planning graph adapting to the new situation.

To better illustration our algorithm, we reuse the example shown in [Yan et al., 2010 a] and [Yan et al., 2010 b]. To simplify, only inputs and outputs parameters are taken into account. Assume that we have nine available actions: $A2BC: a \rightarrow bc, A2D: a \rightarrow d, C2E: c \rightarrow e, D2E: d \rightarrow e, D2F: d \rightarrow f, F2G: f \rightarrow g, F2H: f \rightarrow h, G2E: g \rightarrow e, H2I: h \rightarrow i$. The initial state is a and the goal state is e . Our adaptive algorithm is based on the F-SOPG and B-SOPG graphs as we mentioned before. Figure 6-16 shows the generated graphs for this example. Assume that action $C2E$ is not available, then the next action $D2E$ will be invoked and BP_1 is updated with d which is the preconditions of $D2E$. In the case that both $C2E$ and $D2E$ are becoming unavailable, a straightforward method is proposed in our system which continues extending F-SOPG and B-SOPG until the new termination conditions are hold. The new graphs to the changes are shown in Figure 6-17. As we mentioned above, since the unavailable action exists in BA_0 , The B-SOPG is divided into two parts: $BG_{head} = \{BP_0, BA_0\}$ and $BG_{tail} = \{BP_1\}$, which means that the tail part needs to be repaired for the new change. The Plan Repair of B-SOPG is achieved by a normal backward extension algorithm (see Figure 6-11). As shown in here, the extension of the B-SOPG continues, until it reaches BP_3 where all the proposition nodes are present in F-SOPG.

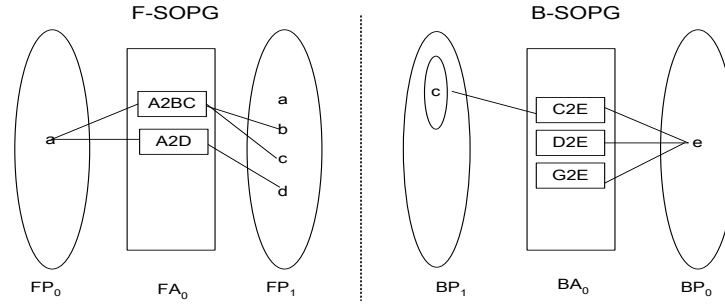


Figure 6-16 The original F-SOPG and B-SOPG

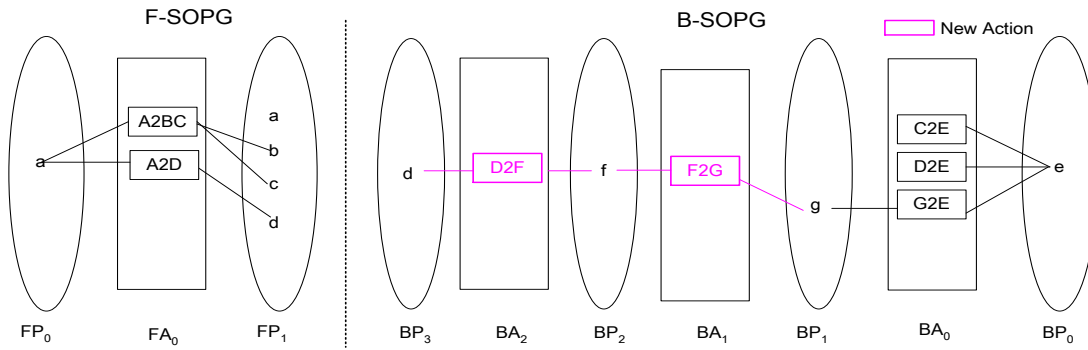


Figure 6-17 Updated SOPG Planning Graphs

6.7 Summary

To sum up, a goal-directed GraphPlanning algorithm called *SEwsPL* is presented in this chapter. It enhances the traditional GraphPlanning algorithm with a so-called Simplified Ordered Planning Graph (SOPG) which stores both the planning information and their multi-attribute similarity (see Section 6.5.1). Furthermore, in our new system, to reduce the search space a bidirectional expansion procedure is introduced in the expansion phase (see Section 6.5.2). Moreover, Rather than searching the plan in a separated step, we transform the generated Planning Graphs to a scientific workflow directly (see Section 6.5.3). In addition, to facilitate the adaptation of the dynamic changes, a Plan-repair based approach has been developed (see Section 6.6).

Chapter 7

Evaluation of *SEwsMining*

"Too much of the research in computing education ignores the hundreds of years of education, cognitive science, and learning sciences research that have gone before us. If we want our research to have any value to the researchers that come after us, if we want to grow a longstanding field that contributes to the improvement of computing education, then we have to 'stand on the shoulders of giants,' as Newton put it, and stop erecting ant hills that provide too little thought."

- Mark Guzdial

Research is too important to rely on subjective judgments. This chapter provides the evaluation of *SEwsMining* system. Section 7.1 presents the evaluation of the semantic annotation language, QWSMO-Lite. The evaluation of *SEwsDM* is illustrated in Section 7.2. Finally, Section 7.3 evaluates the performance of *SEwsPL*.

CONTENTS

7	EVALUATION OF SEWSMINING	85
7.1	EVALUATION OF QWSMO-LITE	86
7.2	EVALUATION OF <i>SEwsDM</i>	87
7.3	EVALUATION OF <i>SEwsPL</i>	91
7.3.1	Scalability Analysis	91
7.3.2	QoP Analysis	94
7.3.3	Dynamicity Analysis	96

7.1 Evaluation of QWSMO-Lite

QWSMO-Lite is a complementary ontology that provides detailed semantic specification of QoS constraints for WSMO-Lite. Based on the survey of the-state-of-the-art approaches discussing in Section 3.2, we compare these reviewed ontologies with our QWSMO-Lite. The comparison results are basically categorized into two fundamental questions: how to model semantics and how to annotate the service with modeled semantics.

Table 7-1 depicts the summary of this comparison. We figured out that WSMO and SAWSDL support more semantic annotations than original OWL-S which does not provide explicit semantics for execution and QoS aspects of Web Services. Some OWL-S variants are the complement of OWL-S with additional QoS models. Among them, OWL-Q is more flexible and extensible because of its modular based structure where new ontologies are easily adapted. It also allows various types of units. Concrete QoS can be specified in the flavor of QoS-MO and onQoS. Only onQoS enables to assign priorities over different QoS characteristics. Dublin core is used to specify non-functional semantics in WSMO, which is not expressive for the nature of QoS. WSMO-QoS has been developed to bridge this gap. However, it still lacks flexibility and priority information of QoS dimensions.

From the annotating point of view, since SAWSDL is a bottom-level extension of WSDL, its annotation can be directly used for invocation, and discovery of Web Services. Nevertheless, for those top-level approaches such as WSMO and OWL-S, the additional efforts for grounding which maps the semantic framework to WSDL are required. On the other hand, one of the disadvantages of exploiting bottom-level solutions is that users might unaware of the functionality of those underlying referred semantics, since there is no predefined structure to specify the detailed annotating rules. WSMO-Lite tries to resolve this problem by filling SAWSDL with concrete semantic service descriptions in WSMO style according to the predefined minimal representation of the semantic information. In WSMO-lite, quality aspects are part of the non-functional information of a Web Service description and are simply defined as: *Accuracy, Availability, Financial, Network-related QoS, Performance, Reliability, Robustness, Scalability, Transactional and Trust*. However, such QoS definition is neither expressive nor flexible enough for QoS properties to distinguish functionally-similar services or operations for service discovery and composition [Wang et al., 2006].

QWSMO-Lite, developed in our system, is an extension of WSMO-Lite with devoting to providing expressive representation of semantics on the one hand, and facilitating the automation of Web Service discovery and composition on the other hand. Besides dealing with Web Services functional semantics, QWSMO-Lite enables to annotate QoS semantics with a predefined QoS ontology where QoS characteristics are modeled in a three layered modular framework. This modular structure makes the QWSMO-Lite more flexible to extend and add any concrete QoS models. Moreover, users are allowed to specify units for each QoS dimension. QoS priority can also be customized by assigning weights. Regarding the mechanism to annotate semantics, inspired by the minimal Web Service Model in WSMO-Lite, a simplified version of Web Service annotation model is defined in QWSMO-Lite, which is also a bottom-level extension of WSDL.

Table 7-1 A comparison of existing approaches to QWSMO-Lite

Criterion		OWL-S	SAWSDL	WSMO	WSMO-Lite	QWSMO-Lite
How to model semantics?						
Functional Semantics	Data Semantics	Process Model	Ontology	WSMO ontology	WSMO ontology	WSMO ontology
	Operational Semantics	Process Model	Classification schema Capability ontology	WSMO Web Service	Classification schema Capability ontology	Classification schema Capability ontology
	Execution Semantics	-	Ontology (implicitly)	Choreography Orchestration	Ontology (implicitly)	Ontology (implicitly)
QoS Semantics	Unit Support	- OWL-Q (OWL-S variant)	-	- WSMO-QoS (WSMO variant)	-	Support
	Concrete QoS	- QoS-MO (OWL-S variant) onQoS (OWL-S variant)	External ontology	Dublin core WSMO-QoS	Dublin core	Dublin core Domain QoS
	QoS Priority	- onQoS	-	-	-	QoS weight
	Flexibility	- OWL-Q	-	-	-	Modular
How to annotate a Web Service with semantics?						
Semantics Formalism		Top Level	Bottom Level	Top Level	Bottom Level	Bottom Level
Semantics Awareness		Support	-	Support	Minimal RDF	Modified Minimal RDF

7.2 Evaluation of *SEwsDM*

In order to evaluate our approach, we compare *SEwsDM* against those existed Web Service discovery and ranking methods introduced in Section 2.3. The comparison is shown in Table 7-2.

Table 7-2 A comparison of SEwsDM with existing approaches

System	OWLSM	OWLS-iMatch	OWLS-MX	WSMO-MX	SAWSDL-MX	SEwsDM
How to match relevant operations?						
Language	OWL-S	OWL-S	OWL-S	WSMO	SAWSDL	QWSMO-Lite
Exploited Information	-Input -Output -Category -Custom parameter	-Input -Output	-Input -Output	-Input -Output -Pre. -Effect	-Input -Output	-Input -Output -Pre. -Effect
QoS Semantics	Custom parameter	-	-	-	-	QWSMO-Lite
Logic-based	-Fail -Unknown -Subsumes -Equivalent	-	-Exact -Plug-in -Subsumes -Subsumed-by -Nearest-neighbor	-Exact -Plug-in -Subsumes -Subsumed-by	-Exact -Plug-in -Subsumes -Subsumed-by	-Exact -Plug-in -Subsumes -Siblings
Non-logic based	-	Name and text similarity	-BOW similarity - Vector similarity	-BOW similarity -Vector similarity	-BOW similarity -Vector similarity -Structural WSDL Matching	TOPSIS based multiple attribute matching
Logic DoM	Fixed numeric score	-	Fixed numeric score	Fixed numeric score	Fixed numeric score	Concept similarity
Non-logic DoM	-	Non-logic similarity	Non-logic similarity	Non-logic similarity	Non-logic similarity	TOPSIS similarity
QoS Match	String similarity	-	-	-	-	TOPSIS similarity
Asymmetric Match	No	No	No	No	No	Yes
How to rank matched operations?						
Ranking Criterion	sum of logic based and non-logic based scores	sorted by DoM	sorted by DoM	sorted by DoM	sorted by DoM	TOPSIS similarity

The comparison falls into two part:

- How to match relevant operations
- How to rank those matched operations.

In most of the algorithms, matching capabilities are described as sets of inputs and outputs. The category information is only dealt by OWLSM, and preconditions and effects information are concerned only by WSMO-MX with WSM rules. To enhance the matching capacities, hybrid matching is supported by almost all approaches. They share one thing in common that the logic-based **Degree of Matching** (DOM) is identified by the predefined logic-based filters with fixed numeric numbers, and the non-logic DoM depends on selected text similarity measures. As for QoS semantics, none of the existing systems explicitly attempts to handle them. OWLSM among others tries to manage such semantics with a set of user-defined parameters. QoS based matching is performed by simple string similarity. Regarding the ranking mechanism, in real application to obtain the reasonable ranking list multi-attributes semantics including functional and non-function aspects should take into account. However, most of the systems rank those matched operations only in terms of the DoM.

Referring to Table 5-3 in Section 5.3, assumed that six candidates of matched operations with different levels of matching degree are retrieved in terms of the requirements. This example will be used to evaluate our system.

Existing matching efforts determine the DoM with fixed numeric values of correspondence logical filters. Logical DoMs of those matched operations are computed in different systems shown in Figure 7-1. Here, since both category matching and IO matching in OWLSM fall into *subsume* matcher which is assigned with the value of 2, the similarity is calculated as follows: $Match_{cat} + Match_{OI} = subsume + subsume = 2 + 2 = 4$. In OWLS-MX, SAWSDL-MX and WSMO-MX, IO matching falls also into *subsume* matcher which is also assigned with the value of 2. Therefore the similarity equals 2 in these three systems. Different to the systems mentioned above, *SEwsDM* computes the similarity more precisely by applying the similarity model defined in Definition 5-2 at the concept level. By doing so, different relatedness of each pair of concepts can be recognized even when they fall into the same logical filters. Furthermore, the predefined logical filters are extended with sibling relationship which has been ignored by most of existing solutions. In addition, our solution makes an asymmetric match between every two concepts, which conveys more semantic to distinguish different logical closeness behind concepts.

Regarding the ranking mechanisms, as we figured out from the comparison in Table 5-3, they mainly depend only on the non-logical DoM. For instance, based on the three groups of decision requirements listed in Table 5-4, the DoM is computed in OWLS-iMatch using simple text similarity. For example, $Sim(O_1) = 0.64 * 0.4 + 0.59 * 0.1 + 0.57 * 0.2 + 0.43 * 0.2 + 0.38 * 0.1 = 0.55$. In OWLS-MX, SAWSDL-MX and WSMO-MX, the intentional loss of information (LOI) is applied to calculate non-logical DoM. Taking O_1 as an example, $Sim(O_1) = 1 - \frac{\frac{23-3}{16+10} + \frac{10}{14}}{2} = 0.26$. However, due to the reason that QoS information has not been taken into account in all these systems, there is no difference to rank the related services in terms of various criteria as shown in Table 7-3. In *SEwsDM*, the similarity is based on multi-attribute semantics including concept closeness at the service interface level, user-defined weighted QoS information and construction heuristic attributes as well. A modified

TOPSIS algorithm is implemented to calculate the similarity score between two examined operations. By doing so, the system is capable of obtaining various values of DOM when applying different decision making criteria as shown in Table 7-3.

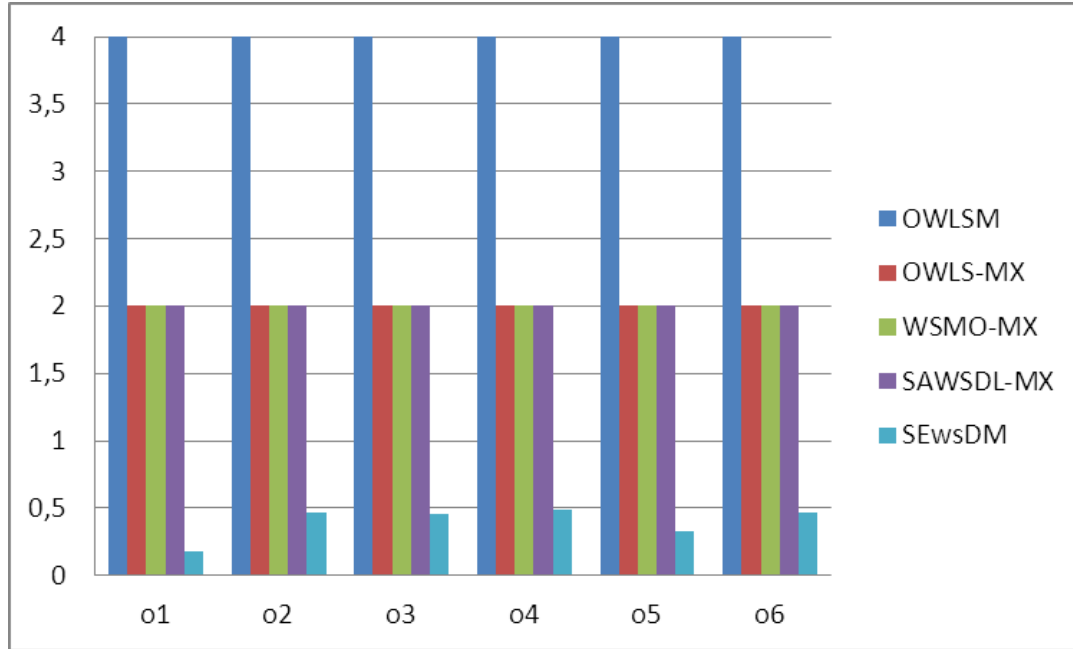


Figure 7-1 Logical DoM using different systems

Table 7-3 Non-Logic DoM of two criteria

Oper.	OWLS-iMatch		OWLS-MX		SAWSDL-MX		WSMO-MX		SEwsDM	
	DM1	DM2	DM1	DM2	DM1	DM2	DM1	DM2	DM1	DM2
O_1	0,55	0,55	0,26	0,26	0,26	0,26	0,26	0,26	0,130	0,138
O_2	0,62	0,62	0,31	0,31	0,31	0,31	0,31	0,31	0,148	0,153
O_3	0,48	0,48	0,34	0,34	0,34	0,34	0,34	0,34	0,142	0,152
O_4	0,42	0,42	0,35	0,35	0,35	0,35	0,35	0,35	0,146	0,154
O_5	0,52	0,52	0,48	0,48	0,48	0,48	0,48	0,48	0,143	0,141
O_6	0,38	0,38	0,52	0,52	0,52	0,52	0,52	0,52	0,142	0,161

7.3 Evaluation of SEwsPL

In this section, we will evaluate our modified GraphPlanning algorithm according to those discussed requirements with the test sets from the Web Service Challenge 2010²¹. The details of the test set are shown in Table 7-4. To simplify the composition, all the services here have only one single available operation.

Table 7-4 Test set from WSC 2010

	Test 1	Test 2	Test 3	Test 4
Web Service	10	20	30	40
Initial states	10	10	10	10
Goal states	4	4	4	4

7.3.1 Scalability Analysis

First of all, the scalability problem has been improved in our system by reducing the size of the SOPG graphs with a bidirectional expansion strategy (ref. R3 in Section 6.1). The details of the expansion algorithm have been presented in Section 6.5.2.

Let (A, S_0, G) be a planning problem, where A is a set of available actions, S_0 is initial conditions with p propositions and G are a set of expected goals involving q propositions. Suppose that the planning problem has m actions and $l = \max_{a \in A} \{|a. eff_a|\}$ is the maximum value of the number of effects of any action. Let $k = \max_{a \in A} \{|a. A_{in}|\}$ be the largest number of inputs in any action. Since the number of different propositions that can be created by an action is no more than $O(l)$, the maximum number of nodes in any proposition layer of F-SOPG is $O(p + ml)$. Considering that any action can be invoked in at most $O(p^k)$ distinct ways, the maximum number of nodes in any action layer of F-SOPG is $O(mp^k)$. Similarly, in B-SOPG, the maximum number of nodes in any proposition layer is $O(mk)$, since a new proposition layer is created by only the inputs of newly added actions. Owing to the facts that an action can be added if any of its effects is available, therefore an action can be invoked in at most $O(pl)$ ways. Then the maximum number of nodes in any action layer of B-SOPG is $O(mpl)$. Accordingly, since k is constant, the total size of the F-SOPG and B-SOPG is polynomial in n, m, p, q, l .

We examine our SOPG graphs with the graphs generated by **B**idirectional-**P**aralleled **G**raphPlan Algorithm (BPGP) [Gu et al., 2004], **S**implified **P**lanning **G**raph (SPG) [Yan and Zheng, 2008] and **W**eighted **P**lanning **G**raph (WPG) [Li et al., 2010]. Expansion of the SPG and WPG is based on the classic GraphPlan algorithm using a forward-chaining strategy. BPGP, on the other hand, allows the graph to expand backwards from the goal set to the initial state and forwards from the initial set to the goal states. Specifically, the expansion begins with the comparison of initial states and goal states. The proposition nodes which exist in both states will be removed from the goal states. Then, the action layers will be created forwards and backwards by adding actions whose inputs and outputs are present in the corresponding proposition layers respectively. Such procedure will continue until the graph is leveled off.

²¹ Web Services Challenge 10: <http://ws-challenge.georgetown.edu/wsc10/>

The comparison shows in Figure 7-3 and Figure 7-3. Here, we randomly create four groups of services containing ten, twenty, thirty and forty Web Services respectively for testing. Owing to this randomness, each group of services might contain its own plans for the composition. Therefore, it makes no sense to do the lateral comparison among groups. Instead, the vertical analysis which compares the results generated by different approaches within a certain group is concentrated in our evaluation.

Let us take a close look at the Figure 7.2. As we pointed out before, since the forward-chaining expansion applied in SPG/WPG depends only on the initial states, the size of a graph dramatically grows by adding irrelevant nodes to the existing graph. Thus, in the figure, we can easily notice that the size of SPG/ WPG is larger than the other two approaches. Moreover, it also indicates that the SOPG graph generated by *SEwsPL* is smaller than the graph created by BPGP. The main reason behind that is in our approach the graph expands forwards and backwards with different paces. Specifically, the forward expansion is slower than the backward expansion. With such strategy, the size of a graph will be reduced by reducing the number of F-SOPG layers. Take the third group which has thirty services as an example. In SPG and WPG approaches, the graph grows forward from the initial states. The forward expansion halts on the third proposition layer where all goal conditions are present. Accordingly, the Planning Graph generated using SPG/WPG contains totally 108 nodes including four proposition layers and four action layers. BPGP reduces the size of the Planning Graph with a bi-directional expansion algorithm in the graph expansion phase, the forward graph generated by BPGP has 56 nodes containing three proposition layers and three action layers and its backward graph contains 14 nodes including two proposition layers and two action layers. Totally, the Planning Graph here contains 70 nodes. In our *SEwsPL*, owing to the different rates used in the expansion phase, the backward graph grows faster than the forward graph. In this test, the forward graph contains 26 nodes with two proposition layers and two action layers, while the backward graph has 32 nodes including three proposition layers and three action layers. Totally, it has 58 nodes.

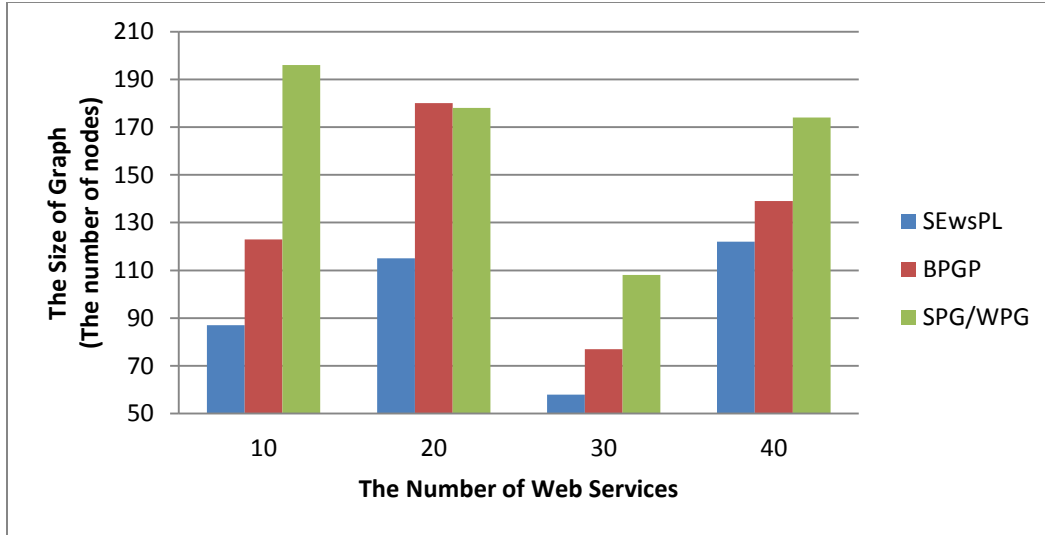


Figure 7-2 Graph size comparison among *SEwsPL*, BPGP and SPG/WPG

The time required to compose a graph can be broken down into:

- *Construction of a proposition layer.* In *SEwsPL*, the proposition layer is updated by inserting associated proposition nodes of the newly added action nodes in the previous action layer. In F-SOPG, since the maximum number of nodes in any action layer of F-SOPG is $O(mp^k)$, the time to build a new proposition layer is $O(p^k)$. Likewise, in B-SOPG, the time required to create a new proposition layer is $O(mpl)$.
- *Selection of actions for an action layer.* In F-SOPG, an action is added to a new action layer if and only if all its inputs are present in the current proposition layer. Therefore, the time required for finding all action candidates is $O(mkp + m^2lk)$. In B-SOPG, the selection of the action is performed by comparing its effects. The time needed is $O(m^2lk)$.
- *Calculate the similarity of each action node and sort them.* Here, the quick sort algorithm is selected to arrange the action nodes. Due to that the maximum number of nodes in any action layer of F-SOPG is $O(mp^k)$, the time complexity for this task is $O(mp^k + m^2p^{2k})$. Similarly, since the maximum number of nodes in any action layer of B-SOPG is $O(mpl)$, the time for arranging nodes is $O(mpl + (mpl)^2)$.

Since k is constant, it is clear that the time for building the graph is also polynomial in n, m, p, q, l . Figure 7-3 shows the time complexity comparison. It indicates that *SEwsPL* is capable of composing the graph in the shortest possible time. The reason behind is that the execution time highly depends on the number of proposition nodes and action nodes in the graph. Since from the comparison of the space complexity, it shows that our solution can provide the graph with minimum size, thus it can proof that our solution is better than the other three in building the smallest Planning Graph in a minimal time.

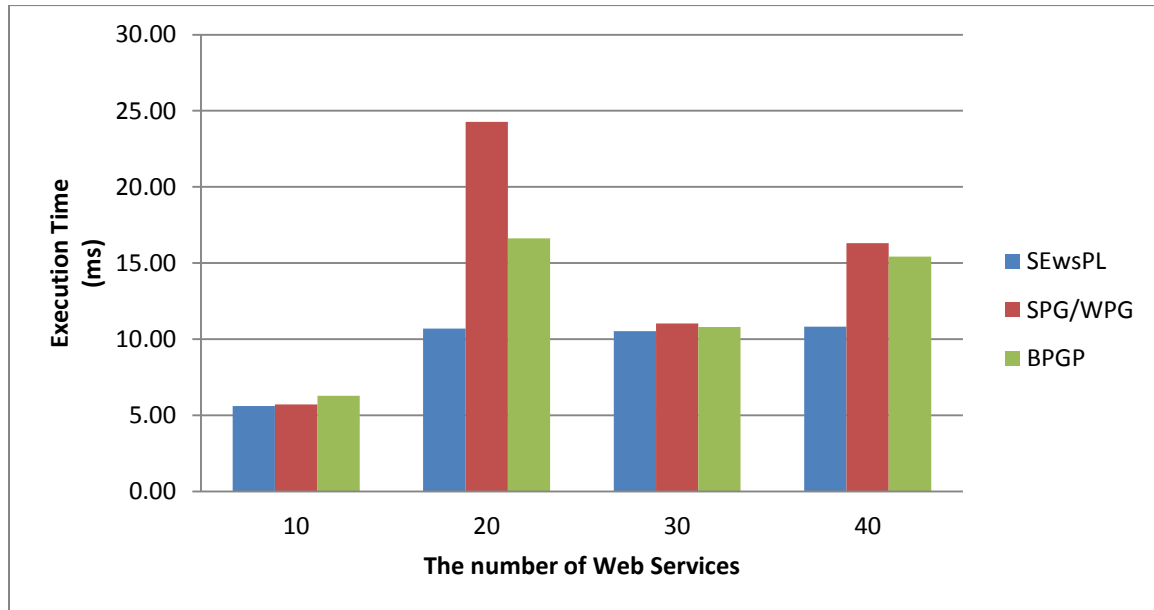


Figure 7-3 Execution time comparison among *SEwsPL*, BPGP and SPG/WPG

7.3.2 QoP Analysis

Another concern about the Web Service Composition is quality awareness (ref. R1 in Section 6.1). Recently, a number of researches have been carried out for the service discovery with QoS-aware matchmaking algorithms. However, such semantics has been largely ignored in the case of AI Planning. Multiple combination of different Web Services may provide similar functionalities but with different quality properties. **Quality of Plan** (QoP) is actually the aggregation of **Quality of Service** (QoS). Thus, to find the best or near best solution is converted into the question how to efficiently aggregate the sequence of services to meet the user's requirements.

The evaluation of QoP is also based on the WSC 2010. Assume that two quality dimensions are defined for each service, namely Response Time (RT) and Throughput (TP). Users expect to get two kinds of plans with lowest response time and highest invocations per minute respectively. We compare our system with the other three GraphPlan based methods mentioned above.

The bar charts in Figure 7-4 reveal that plan generated by *SEwsPL* is the closest solution to the optimal one compared with the other three. Regarding BPGP and the composition algorithm proposed in [Yan and Zheng, 2008] compose the final plan by choosing the action nodes randomly without taking any QoS aspect into consideration. Accordingly, their results only satisfy user's functional requirements. Due to the randomness, the probability that they get the plan with the best QoS attributes is relative slim. In the WSC-WPG [Li et al., 2010] method where the weight indicating similarity is introduced to improve the functional quality of the plan, the non-functional quality is still missing. Therefore, its results shown in the following figure maybe are the most functionally related solutions but without considering user's QoS requirements. In our approach, besides computing functional similarity, we deal with the QoP by aggregating of QoS of each service component. That means, the QoP is regarded as a combination of a sequence of local optimal actions to compose the final plan. However, our solution is not always the optimal one, since the local optimal and the global optimal results are not equal in certain cases. For instance, in the fourth group which contains forty services, the optimal response time for the final plan should be 850ms, while the response time of the plan generated by *SEwsPL* is 950ms.

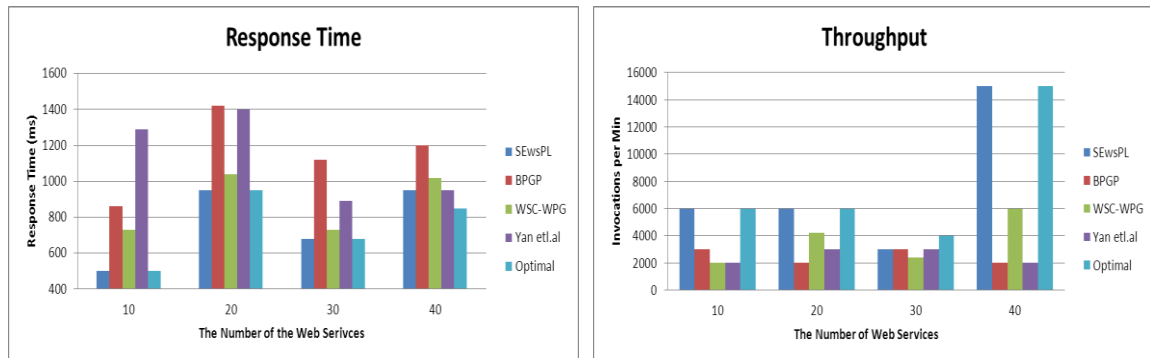
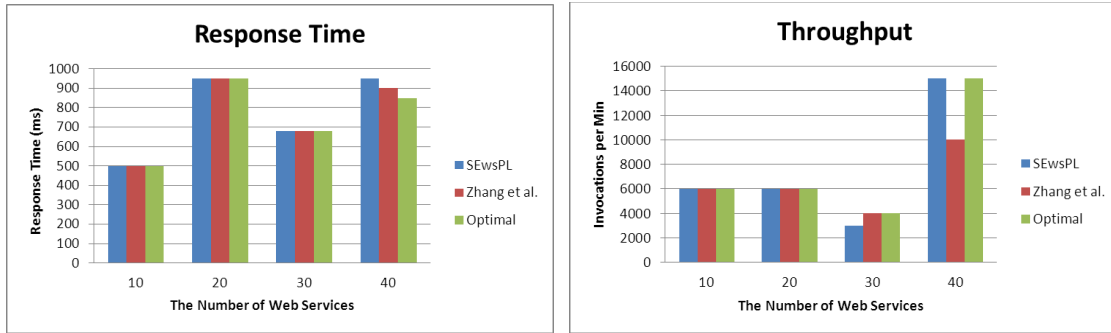
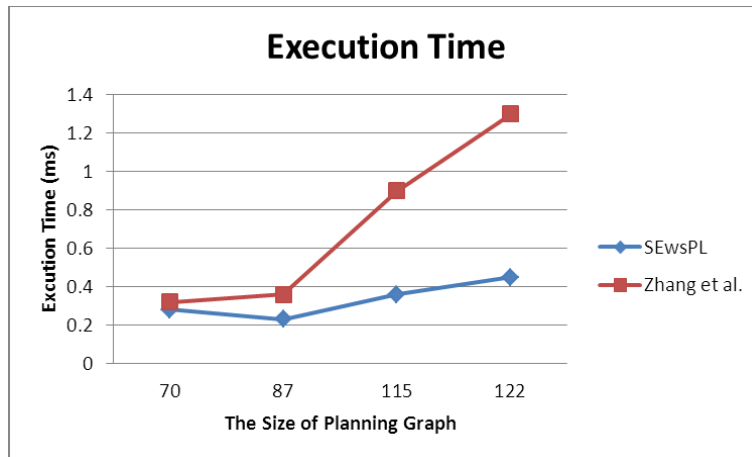


Figure 7-4 QoP comparison

On the other hand, after surveying related work concerning the QoS aspect of composition, we noted most of the work sharing one thing in common. The optimization is

done by decomposing a Planning Graph into a set of parallel execution paths. And then different global criterions are used to measure the QoS levels of each execution path. For instance, in [Zhang et al., 2010], the **Multi-Objective Optimization** (MOP) model is applied for this purpose. In [Pop et al., 2009], the immune-inspired algorithm has been adopted to this end. Different to those approaches, in our *SEwsPL*, QoP is achieved by integrating the similarity into the Planning Graph. Specifically, the action nodes in the action layers are arranged as a sorted list according to their TOPSIS based multi-attribute similarity. Details are available in Section 5.3. As a result, the most appropriate action which is at the front of the list will be selected to compose the plan in the first place.

We examine *SEwsPL* with one of the mentioned methods called MOP based optimization in [Zhang et al., 2010]. The result depicted in Figure 7-5 shows that both methods can provide the near-optimal solutions. However, as is shown in Figure 7-6, MOP based method will take more time to get the result. Because the time complexity of those methods depends on the number of the parallel execution paths which are related to the number of the action nodes in each layer. Suppose that the largest number of action nodes is m and the number of action layers is n . In the worst case, the number of parallel execution paths will be m^n . It means m^n paths should be measured using the pre-defined global criterions. In this context, it is clear that the performance of those methods will dramatically decrease when the size of the graph grows.

Figure 7-5 QoS comparison between *SEwsPL* and MOPFigure 7-6 Execution comparison between *SEwsPL* and MOP

7.3.3 Dynamicity Analysis

The dynamicity is also a requirement that has not been adequately explored in AI Planning techniques (ref. R2 in Section 6.1). To enable a dynamic composition, in *SEwsPL*, a kind of Plan Repair algorithm is developed. Details are presented in Section 6.6.

To better compare our solution with other Plan Repair approaches; we reuse the example shown in [Yan et al. 2010 a] and [Yan et al. 2010 b]. The principles of Plan Repair have been introduced to Web Service Composition domain in the work of [Yan et al., 2010 a], the Planning Graph based plan repair is achieved by a backward chaining strategy starting from the unsolved goal states to the available initial states. The original Planning Graph is shown in Figure 7-7 and the modified Planning Graph generated from this algorithm is presented in Figure 7-8.

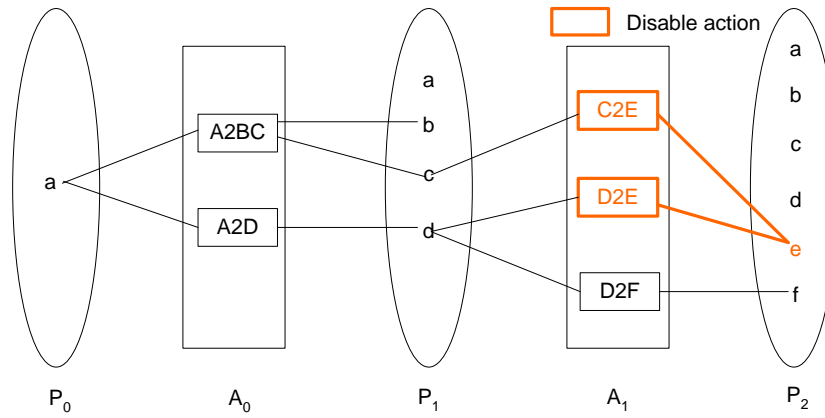


Figure 7-7 The original Planning Graph

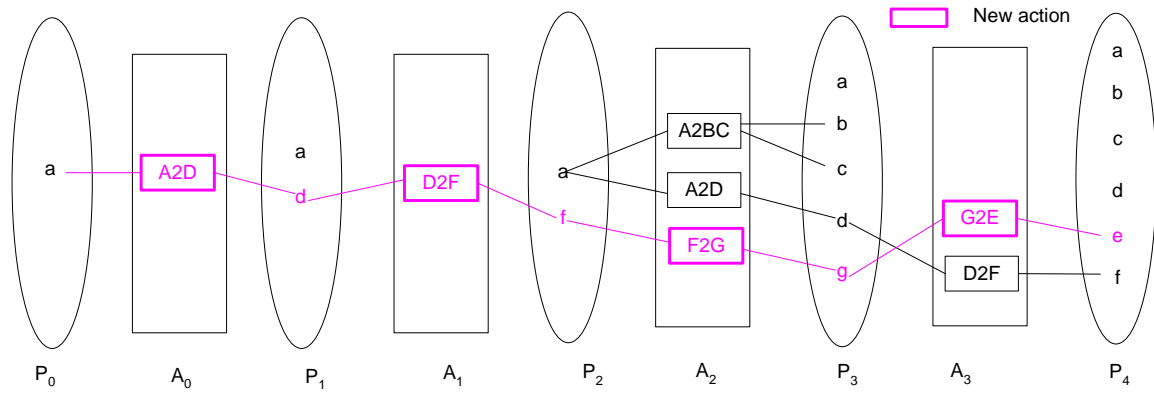


Figure 7-8 The Planning Graph grown by [Yan et al., 2010 a]

To better reuse the original planning graph structure, in Yan et al.'s later work [Yan et al., 2010 b], authors developed a greedy search process starting from the highest level of the graph where unsatisfied proposition firstly occurs. Rather than adding actions which can satisfy the unsolved propositions in the following layers, in their work, a new action layer is created above the current level. The result plan is shown in Figure 7-9.

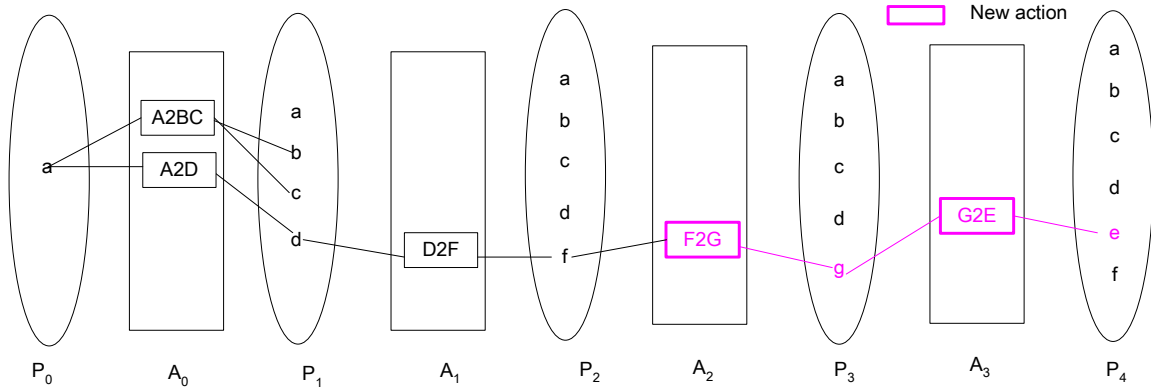


Figure 7-9 The Planning Graph generated by [Yan et al., 2010 b]

After comparing the results generated by our algorithm shown in Figure 6-17 with the updated plan created by different Plan Repair approaches shown in Figure 7-8 and Figure 7-9, we can find our SOPG based adaptive algorithm has the following advantages:

- *More similar to the original graphs.* Since the adaptive algorithm is a kind of re-extension of the original algorithm, it is similar to the original graphs. However, the new graph created by the first method is completely separated from its original one. The second method which aims to improve the reuse of the original graph by inserting new layers for unsolved proposition nodes with predefined rules. Unfoundedly, this insertion procedure is time-consuming, owing to their adding rules, once it inserts a new layer, all the added layers should be updated.
- *Easy to get the near-optimal results.* Due to the facts that in the SOPG graphs, all action nodes are arranged according to their similarity, the updated composition results generated by our algorithm is closer to the optimal one than using other solutions which choose the action randomly from the related action set.
- *Reduce the search space.* Again, because the SOPG graphs used in our algorithm, the newly updated graph is also capable to reduce the search space. For instance, in this example, the final graph generated by the first algorithm contains sixteen proposition nodes and 7 action nodes. The updated result created by the second algorithm involves 21 proposition nodes and 5 action nodes. And the results built by our system consists of only 9 proposition nodes and 5 actions nodes.

Chapter 8

Conclusion and Future Work

“Today is not yesterday. We ourselves change. How then can our works and thoughts, if they are always to be fittest, continue always the same?”

- Thomas Carlyle (1795-1881)

This chapter summarizes this dissertation and provides potential avenues for future work based on the contributions in this work. The conclusions about this work will be drawn firstly in Section 8.1. Finally, discussion and direction of future work extended from this dissertation is pointed out in Section 8.2.

CONTENTS

8	CONCLUSION AND FUTURE WORK	98
8.1	SUMMARY	99
8.2	OUTLOOK ON FUTURE WORK.....	102
8.2.1	<i>Improvement for the Semantic Enhancement Layer</i>	<i>102</i>
8.2.2	<i>Enhancement of the WSD/WSC Layer.....</i>	<i>102</i>
8.2.3	<i>Enrichment of the Adaptation Layer</i>	<i>103</i>

8.1 Summary

This dissertation is motivated by the requirements to enhance the current **Service-Oriented Computing (SOC)** models with semantic awareness. Our work provides a methodology for the automated Web Service composition system by leveraging semantics involving both functional semantics and QoS in order to raise the accuracy and assess the quality of the composition results. Several key concepts of the methodology are summarized in the following paragraphs.

Definition of service ontology for semantics annotation: The first contribution of this dissertation is a service ontology called QWSMO-Lite, which is an extension of WSMO-Lite ontology with QoS attributes. WSMO-Lite [Kopecký and Vitvar, 2008], in turn is an integration of two recent W3C standards for semantic Web Service, namely, SAWSDL [Kopecký et al., 2007] and WSMO [Roman et al., 2005]. However, owing to the lack of QoS characteristics, it fails in the task of efficient service discovery and service composition, when choosing among functionally similar services. Accordingly, QWSMO-Lite defined in this work comprises both functional and non-functional required semantics. Specifically, considering that a service may contain various operations, Web Services are allowed to be annotated at the operation level with:

- QoS attributes using an extensible QoS model which involves a set of generic and domain specific QoS dimensions
- Concepts related to the inputs/outputs together with their associated domain properties.

Moreover, at the service level, generic attributes, such as the author name, create date, etc. can also be annotated.

In comparison to other existing approaches shown in Table 7-1, it indicates that QWSMO-Lite is able to annotate Web Services with more expressive functional and non-functional properties. Regarding functional annotation, besides input/output semantics, categorization information is also added to both operations and services. This aids in service discovery by narrowing the range of candidate services. A pre-defined QoS model is used to annotate non-functional properties. Its modular structure makes it more flexible to extend and add any concrete QoS models. Moreover, users are allowed specify units and weights for each QoS dimension. The annotated semantics can be further explored for service discovery, selection and composition.

Development of a multi-attribute matchmaking algorithm: Alongside the aforementioned QWSMO-Lite ontology, this dissertation contributes a novel multi-attribute matchmaking approach called *SEmsDM*. The first step is to determine functionally similar services which can be done by measuring semantic distance between two concepts. In *SEmsDM*, an ontology based single attribute matchmaking algorithm called *SAMatch* is presented (see Section 5.2) for this purpose. The next step is to rank the discovered services according to their similarity, which is conducted in *SEmsDM* by a TOPSIS based multi-attribute services ranking algorithm, namely *MAMatch* (see Section 5.3). Moreover, to avoid out of range problem, instead of classic normalization functions adopted in TOPSIS, a logistic function which performs the “softmax scaling” [Pyle, 1999] is defined in *MAMatch*.

According to the comparison shown in Table 7-2, we figure out that our matchmaking algorithm differs from other similar approaches [Algergawy et al., 2010][He et al., 2008] in the following aspects:

- Allowing for an asymmetric match between concepts by defining five logical filters, namely exact match, plugin match, subsume match, siblings match and fail match regarding the relation between two concepts.
- Making the matching more precisely by means of a semantic score.
- Ranking the candidate operations with a multi-attribute similarity.

Implementation of a bidirectional service discovery approach: The third contribution of this dissertation is a bidirectional service discovery algorithm (see Section 5.4) which is based on the matchmaking approach *SEnsDM* presented above. It supports the service discovery with either forward matchmaking or backward matchmaking. That means users are allowed to find a sequence of services in terms of the given inputs and outputs respectively. If there is at least one service showing in both service sets, it will mean that at least one directly matched service is available for the given query. Otherwise, the service discovery problem will be converted into the service composition problem by continuing the matchmaking process backwards and forwards.

The first three contributions help to develop a semantic-aware bidirectional service composition approach called *SEnsPL* which is the main result of this dissertation. The benefits of this approach can be listed as below:

- *PDDL 3 based planning problem model for Web Service Composition.* PDDL 3 [Gerevini and Long 2006] extends the previous version of PDDL language with new constructs increasing the expressive power with respect to the plan quality specification. In *SEnsPL*, inspired from it, a PDDL3 based data model has been specified to encode the Web Service Composition problem as a planning problem (see Section 6.3). Particularly, PDDL 3 has been enhanced to support the QoS specification. Moreover, in order to improve the quality of the plan, we distinguish soft goals called preferences which may not be satisfied but are desired, from strong goals called constraints which must be achieved.
- *Simplified Ordered Planning Graph (SOPG).* To grasp the information of semantically connected services, a directed layered graph model called SOPG is defined in *SEnsPL* involving F-SOPG and B-SOPG for forward Planning Graphs and backward Planning Graphs respectively (see Section 6.5.1). Different from the traditional Planning Graph, besides the service connection information, the SOPG also presents how these services are related to the current available propositions by ranking discovered action nodes in terms of the similarity which is calculated in *SEnsDM*.
- *Bi-directional graph expansion algorithm.* A bi-directional graph expansion algorithm for building the SOPG is presented (see Section 6.5.2). We put a particular focus on reducing the search space by trading off between backward and forward chaining. Due to the fact that F-SOPG grows much faster than B-SOPG, in our algorithm forward and backward graphs are allowed to be expanded at different paces, that is, the forward expansion will be slowed down, while the backward expansion will be speeded up. Additionally, the above mentioned bidirectional discovery algorithm has been integrated to enable semantic awareness of the connected services.

- *Workflow based plan extraction approach.* Traditionally, to extract the appropriate plan, a two-step approach is applied involving the extraction of plan candidates and the selection of the final plan. The coupling of these two procedures into a single approach has been addressed in our work by providing a workflow based plan extraction approach (see Section 6.5.3). The SOPG is represented as a workflow by marking every redundant action node as an alternative node of the used node. Furthermore, the quality of the final plan encoded as a workflow is improved by combining the first available nodes in each action layer, since in SOPG, all action nodes have been ordered based on their similarity.
- *Plan repair approach for the self-adaptive composition.* Supporting self-adaptation is becoming more and more important due to the dynamic nature of a Web Service environment, which is realized in our work with a plan repair approach (see Section 6.5). Rather than building the plan from the scratch, our plan repair based solution tries to reuse most of the original plan as possible.

We examine our algorithm with some related work in this field. The comprehensive evaluation illustrated in Section 7.3 shows that our *SEwsPL* has the following advantages:

- Due to the bi-directional expansion mechanism applied in the planner, it is capable of creating a relative smaller Planning Graph.
- The composition results can be extracted in a short time by representing the graph into a scientific Workflow.
- The quality of plan can be obtained by aggregating the QoS of each component service.
- It supports for self-adaptive composition and enables to get the updated result in a short time.

The applicability of *SEwsMining*: Our semantics enhanced composition system is suitable for any applications that need to manage Web Services. Presented here are some examples of possible applications of *SEwsMining*:

- *Investigations and integration of BioMoby²² based services.* As biology becomes an increasingly computational science, it is critical for the biologists to reuse the vast and complex data-sets in their experiments. BioMoby based services are developed with the goal of facilitating greater interoperability between Web-based bioinformatics and biological resources. It now boasts greater than 50 independent host providers spanning five continents and offering more than 800 data retrieval and analysis services [Kawas et al., 2006]. Therefore, our system can be used to aid in building BioMoby workflows to facilitate the reuse of the existing data for end-users by pipelining BioMoby resources.
- *Composition of SoapLab²³ services.* SoapLab is a tool for wrapping command-line executable programs and legacy programs automatically as Web Services. Considering that, in some cases, especially, scientific experiments, the coordinated use of computational resources from various institutes is highly needed. A growing number

²² BioMoby is available in: <http://biomoby.open-bio.org/index.php/what-is-moby/>

²³ More information of SoapLab is available in: <http://soaplab.sourceforge.net/soaplab2/>

of these resources are being made available in the form of Web Services with SoapLab API. Accordingly, the orchestration of these services in workflows can also be made using our systems.

8.2 Outlook on Future Work

SEwsMining is a prototype for composition of Web Services with semantics awareness. It still leaves various possibilities for further improvement and enhancement. This section envisions three directions of future work that could follow, according to the architecture provided in Section 2.2.

8.2.1 Improvement for the Semantic Enhancement Layer

In *SEwsMining*, QWSMO-Lite is defined as a service ontology to annotate the semantics information of Web Services. Future work includes the following aspects:

Modeling more semantics in QoS: QoS properties should comprise multiple layers from the SOC stack. In this dissertation work, we concentrate on the elementary QoS attributes only in the service foundation layer, such as performance, dependency, security and cost of a service. However, to implement a more flexible service-oriented system, the QoS aspects from the other two layers should also be concerned. For instance, in the service composition layer, QoS refers to QoS policies which define QoS guarantees for various partners. In the service management layer, QoS is expressed on a high level in form of Service Level Agreement (SLAs) between two partners, which is guaranteed by service advertisements.

Adding semantics to other distributed resources: QWSMO-Lite is defined to add semantics awareness to the Web Services which are emerging as a major technology for deploying automated interactions between distributed and heterogeneous applications. Nowadays, use of the “light-weight” approaches to services, especially for Web applications is increasing [Benslimane et al., 2008]. Therefore, adding semantics to those “light-weight” approaches will be an exciting prospect in the near future:

- *Semantic annotation of RESTful services.* In [Sheth et al., 2007], the authors propose a so-called SA-REST description for the annotation of RESTful services by borrowing the idea from SAWSDL. As an extension work, we can consider annotating the RESTful services with our pre-defined service ontology, QWSMO-Lite.
- *Semantic annotation of Web APIs.* Many Web 2.0 based applications like Facebook, Google, Flickr, and Twitter offer easy-to-use Web APIs. However, there is no widely accepted structure language for describing Web APIs. SWEET presented in [Maleshkova et al., 2010] enables users to create machine-readable descriptions and semantic annotations. The Authors pointed out that the annotation procedure provided in SWEET is quite time-consuming including a number of manual tasks for users to complete. We conjecture that, the annotation of Web APIs can be simplified by adapting the QWSMO-Lite ontology.

8.2.2 Enhancement of the WSD/WSC Layer

In this dissertation, *SEwsPL*, an AI-Planning based planner is developed for automated service composition. To improve the quality of the plan, *SEwsDM*, a multi-attribute decision making

engine is integrated into the planner. In this part, we will discuss the possible items that may remain on the research agenda of this domain.

Handling structural heterogeneity between services: Traditionally, the heterogeneity is classified into four types [Nagarajan et al., 2007]:

- *Syntactic heterogeneity*: difference in the used specification languages.
- *Model/Representation heterogeneity*: difference in the underlying models or their representations.
- *Structural heterogeneity*: difference in types and structures.
- *Semantic heterogeneity*: difference in semantic interpretations of the same real world entity.

The first two types of heterogeneity have been addressed by using XML. In our research work, the semantic heterogeneity is reduced by the semantic annotation using QWSMO-Lite. However, big challenges for structure heterogeneities still remain. In our previous work [Leng et al., 2009], a framework called **S**emantically **E**nriched **I**ntegration **S**ystem (SEIS) is presented. It provides OGSA-DAI services for the semi-automatic generation of the necessary data transformations between two sources under different structures. In this context, the further development could be devoted to integrate SEIS to *SEwsPL* for the manipulation of such structural heterogeneity between services.

Adaptation for the generation of services mashups and the cloud service composition: Recently, with the development of Web 2.0 technologies, one noteworthy trend over the Web is the rapid growing services mashups which combines existing services, such as Web APIs, RESTful services into a single integrated service. Most service mashup solutions are semi-automatic which assume that all available Web resources are known and available on the Web²⁴. **U**ser **G**enerated **S**ervices (UGS) enable users to build mashups by finding, combining and reusing Web resources manually. However, to facilitate the creation of service mashups, an automated composition way is highly needed. Regarding cloud services which are developed as self-contained component, how to compose services in cloud environments and achieve high resource utility which is customized for client requests has become an important research issue. In order to realize such composition, multi-attribute semantics involving functional and non-functional semantics should be taken into account for the service selection and planning procedure. To this end, our *SEwsPL* and *SEwsDM* can be further developed for the service mashups and the cloud service composition by parsing the semantics from annotated Web resources and modifying the problem modeling component for the further reuse.

8.2.3 Enrichment of the Adaptation Layer

In this dissertation, a plan repair based approach has been developed for the adaptation of the composition. Some future research is needed to make the adaptation procedure more robust.

Improvement of plan repair based solution: In *SEwsPL*, we have designed and implemented a plan repair based approach for the adaptation which aims to reuse most of the original plan as possible. However, in the case that users change their goals or some new services are becoming available for the composition, the plan repair solution is converted to a

²⁴ For instance, users can share, find and reuse Web APIs in www.programmableweb.com

re-planning procedure. Therefore it remains an open question how to make the modification more efficiently in such cases.

QoS Monitoring: Our system lacks a mechanism for QoS monitoring. The dynamic changes are obtained by a user-defined external file. In real applications, it is also important that QoS attributes can be monitored continuously by using non-intrusive monitoring mechanism. Some efforts are available in this domain. For instance, QUATSCH which is a novel client-side QoS monitoring approach for Web Service is presented in [Artaiam and Senivongse, 2008]. It allows to monitor performance-specific QoS attributes such as response time, latency or throughput continuously from a client-side perspective without requiring access to service provider. Such existing monitoring system can be integrated into this layer as part of future work in this direction.

References

- [Algergawy et al., 2010] Algergawy, A., R. Nayak, N. Siegmund, V. Koppen, and G. Saake. „Combining Schema and Level-Based Matching for Web Service Discovery.“ *In: Proceedings of 10th International Conference of Web Engineering (ICWE 2010)*. Vienna, Austria, 2010. pp. 114-128.
- [Ardagna et al., 2007] Ardagna, D., M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani. „AWS: A Framework for Executing Adaptive Web-Service Processes.“ *IEEE Software*, vol.24(6) (2007): pp. 39-46.
- [Artaim and Senivongse, 2008] Artaim, N., and T. Senivongse. „Enhancing Service-Side QoS Monitoring for Web Services.“ *In: Proceedings of the 9th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2008)*. Phuket, Thailand, 2008. pp. 765-770.
- [Azar, 2000] Azar, F. S. „Multiattribute Decision-Making: Use of Three Scoring Methods to Compare the Performance of Imaging Techniques for Breast Cancer Detection.“ *Technical Reports (CIS)*, Dept. of Computer Science, University of Pennsylvania, 2000.
- [Baryannis and Plexousakis, 2010] Baryannis, G., and D. Plexousakis. „Towards Realizing Dynamic QoS-aware Web Service Composition.“ *In: Proceedings of the 8th IEEE European Conference on Web Services (ECOWS 2010)*. Ayia Napa, Cyprus, 2010. pp. 25-28.
- [Benslimane et al., 2008] Benslimane, D., S. Dustdar, and A. Sheth. „Services Mashups: The New Generation of Web Applications.“ *Internet Computing*, vol.12(5) (2008): pp. 13-15.
- [Bless et al., 2008] Bless, P. N., D. Klabjan, and S. Y. Chang. „Heuristics for Automated Knowledge Source Integration and Service Composition.“ *Computers and Operations Research*, vol.35(4) (2008): pp. 1292-1314.
- [Blum and Furst, 1997] Blum, A. L., and M. L. Furst. „Fast Planning Through Planning Graph Analysis.“ *Artificial Intelligence*, vol.90(1) (1997): pp. 281-300.
- [Brauers et al., 2008] Brauers, W. K., E. K. Zavadskas, F. Peldschus, and Z. Turskis. „Multi-objective Decision-making for Road Design.“ *Transport*, vol.23(3) (2008): pp. 183-193.
- [Cardoso, 2006] Cardoso, J. „Discovering Semantic Web services with and without a Common Ontology Commitment.“ *In: Proceedings of the 3rd International Workshop on Semantic and Dynamic Web Processes (SDWP 2006)*. Chicago, USA, 2006. pp. 183-190.
- [Chabeb and Tata, 2008] Chabeb, Y., and S. Tata. „Yet Another Semantic Annotation for WSDL.“ *In: Proceedings of LADIS International Conference (WWW/Internet 2008)*. Freiburg, Germany, 2008. pp.437-441.

- [Chan et al., 2006] Chan, M., J. Bishop, and L. Baresi. „Survey and Comparison of Planning Techniques for Web Services Composition.“ *Technical Reports*, Pretoria, South Africa: Department of Computer Science, University of Pretoria, 2006.
- [Chen et al., 2011] Chen, Y., K. W. Li, and S. F. Liu. „An OWA-TOPSIS Method for Multiple Criteria Decision Analysis.“ *Expert Systems with Applications*, vol.38 (2011): pp. 5205–5211.
- [Chiu et al., 2008] Chiu, D., S. Deshpande, G. Agrawal, and R. Li. „Cost and Accuracy Sensitive Dynamic Workflow Composition over Grid.“ *In: Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (GRID'08)*. Tsukuba, Japan, 2008. pp. 9-16.
- [Doan et al., 2002] Doan, A., J. Madhavan, P. Domingos, and A. Halevy. „Learning to Map between Ontologies on the Semantic Web.“ *In: Proceedings of the 11th International Conference on World Wide Web (WWW'02)*. Hawaii, USA, 2002. pp. 662-673.
- [Edelkamp and Hoffmann, 2004] Edelkamp, S., and J. Hoffmann. „PDDL2.2: The language for the Classic Part of the 4th International Planning Competition.“ *Technical Reports*, Freiburg, Germany: Institut fuer Informatik, Freiburg, Germany, 2004.
- [Eduardo et al., 2009] Eduardo, S., F. P. Luis, and S. V. Marten. „Supporting Dynamic Service Composition at Runtime based on End-user Requirements.“ *In: Proceedings of the 1st International Workshop on User-generated Services (UGS2009)*. Stockholm, Sweden, 2009. pp.464–471.
- [Eduardo et al., 2011] Eduardo, S., F. P. Luis, and S. V. Marten. „Towards Runtime Discovery, Selection and Composition of Semantic Services.“ *Computer Communications*, vol.34(2) (2011): pp. 159-168.
- [El-Gayyar et al., 2009] El-Gayyar, M., Y. Leng, S. Shumilov, and A. B. Cremers. „New Execution Paradigm for Data-Intensive Scientific Workflows.“ *In: Proceedings IEEE Congress on Services (SERVICE 2009)*. Los Angeles, USA, 2009. pp. 334-339.
- [El-Gayyar et al., 2010] El-Gayyar, M., Y. Leng, and A. B. Cremers. „Distributed Management of Scientific Workflows in SWIMS .“ *In: Proceedings the 9th IEEE Symposium on Distributed Computing and Applications to Business Engineering and Science (DCABES, 2010)* . Hongkong, China, 2010. pp. 327-331.
- [Evangelos, 2002] Evangelos, T. *Multi-Criteria Decision Making Methods: A Comparative Study*. Kluwer Academic Publishers, 2002.
- [Feng and Sun, 2010] Feng, P., and W. Sun. „Fast Goal-directed Graphplan based on Interfering Actions.“ *In: Proceedings of the 2nd International Workshop on Education Technology and Computer Science*. Wuhan, China, 2010. pp. 522-525.
- [Fikes and Nilsson, 1971] Fikes, R. E., and N. J. Nilsson. „STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.“ *Artificial Intelligence*, Vol.2(3) (1971): pp. 608-620.

- [Fox and Long, 2003] Fox, M., and D. Long. „pddl2.1 : An Extension to pddl for Expressing Temporal Planning Domains.“ *Journal of Artificial Intelligence Research*, 2003: pp. 61-124.
- [Fujii and Suda, 2009] Fujii, K., and T. Suda. „Semantics-based Context-aware Dynamic Service Composition.“ *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol.4(2) (2009): pp. 1-31.
- [Gabrilovich and Markovitch, 2007] Gabrilovich, E., and S. Markovitch. „Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis.“ *In: Proceedings of the 12th International Joint Conference for Artificial Intelligence*. Hyderabad, India, 2007. pp. 1606-1611.
- [Ganesan et al., 2003] Ganesan, P., G. H. Molina, and J. Widom. „Exploiting Hierarchical Domain Structure to Compute Similarity.“ *ACM Transactions on Information System*, vol.21(1) (2003): pp. 64-93.
- [Gerevini and Long, 2006] Gerevini, A., and D. Long. „Preferences and Soft Constraints in PDDL3.“ *In: Proceedings of Workshop on Planning with Preferences and Soft Constraints (ICAPS'06)*. Glasgow, UK, 2006. pp. 46-54.
- [Ghallab et al., 2004] Ghallab, M., D. Nau, and P. Traverso. *Automated Planning theory and practice*. Morgan Kaufmann Publishers, 2004.
- [Ghallab et al., 1998] Ghallab, M., et al. „PDDL: The Planning Domain Definition Language.“ *Technical Reports CVC TR-98-003/DCS TR-1165*, Yale Center for Computational Vision and Control, 1998.
- [Giallardo and Zimeo, 2007] Giallardo, E., and E. Zimeo. „More Semantics in QoS Matching.“ *In: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*. Newport Beach, USA, 2007. pp. 163-171.
- [Gruber, 2008] Gruber, T. „Ontology.“ *In Entry in the Encyclopedia of Database Systems*, Springer-Verlag, 2008.
- [Gu et al., 2004] Gu, W., L. Xu, X. Zhang, X. Li, and F. Ren. „Research and Implementation based on Bidirectional-paralleled Graphplan Algorithm .“ *In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (ICMLC 2004)*. Shanghai, China, 2004. pp. 239-243.
- [Gulla et al., 2009] Gulla, J. A., T. Brasethvik, and G. S. Kvarv. „Association Rules and Cosine Similarities in Ontology Relationship Learning.“ *Lecture Notes in Business Information Processing*, vol.19(4) (2009): pp. 201-212.
- [Guo et al., 2007] Guo, L., A. S. McGough, A. Akram, D. Colling, J. Martniak, and M. Krznaric. „Enabling QoS for Service-Oriented Workflow on Grid.“ *In: Proceedings of the 7th IEEE International Conference on Computer and Information Technology (CIT 2007)*. Fukushima, Japan, 2007. pp. 1077-1082.

- [Gupta et al., 2007] Gupta, M., J. Fu, F. Bastani, L. R. Khan, and I. L. Yen. „Rapid Goal-oriented Automated Software Testing Using MEA-graph Planning.“ *Software Quality Control*, vol.15(3) (2007): pp. 241-263.
- [Hang and Singh, 2010] Hang, W. C., and M. P. Singh. "Trustworthy Service Selection and Composition." *ACM Transactions on Autonomous and Adaptive Systems*, vol.5(4) (2010): pp. 1-18.
- [Hatzi et al., 2009] Hatzi, O., G. Meditskos, D. Vrakas, N. Bassiliades, D. Anagnostopoulos , and I. Vlahavas. „Semantic Web Service Composition Using Planning and Ontology Concept Relevance.“ *In: Proceedings of IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. Milano, Italy, 2009. pp. 418-421.
- [He et al., 2008] He, J. L., C. L. Liu, and C. Wu. „Asymmetric Web Service Operation Matching in Structural-Level Similarity Measure.“ *In: Proceeding of the 8th IEEE International Conference on Computer and Information Technology Workshops (CITWORKSHOPS '08)*. Washington, USA, 2008. pp. 688-693.
- [Helmert et al., 2008] Helmert, M., M. Do, and I. Refanidis. *Changes in PDDL 3.1*. 2008. <http://ipc.informatik.uni-freiburg.de/PddlExtension>. (Available on 8. 8 2011).
- [Hennig and Balke, 2010] Hennig, P., and W. Balke. „Highly Scalable Web Service Composition Using Binary Tree-Based Parallelization.“ *In: Proceedings of IEEE International Conference on Web Services (ICWS'10)*. Miami, USA, 2010. pp. 123-130.
- [Hwang and Yoon, 1981] Hwang, C. L., and K. P. Yoon. *Multiple Attribute Decision Making: Methods and Applications; a state-of-the-art survey*. Berlin: Springer-Verlag, 1981.
- [Jaeger and Rojec-Goldmann, 2005] Jaeger, M. C., and G. Rojec-Goldmann. "SENECA-Simulation of Algorithms for the Selection of Web Services for Compositions." *In: Proceeding of the 6th VLDB Workshop on Technologies for E-Services (TES'05)*. Trondheim, Norway, 2005. pp. 84-97.
- [Jaeger et al., 2005] Jaeger, M. C., G. Rojecgoldmann, C. Liebetrueth, G. Mühl, and K. Geihs. „Ranked Matching for Service Descriptions using OWL-S.“ *Kommunikation in Verteilten Systemen(KiVS 2005)*, (2005): pp. 91-102.
- [Jafarpour and Khayyambashi, 2010] Jafarpour, N., and R. M. Khayyambashi. „QoS-aware Selection of Web Service Composition based on Harmony Search Algorithm.“ *In: Proceedings of the 2th International Conference on Advanced Communication Technology (ICACT 10)*. Gangwon-Do, Korea, 2010. pp. 1345-1350.
- [Jahanshahloo et al., 2009] Jahanshahloo, G. R., F. H. Lotfi, and A. R. Davoodi. „Extension of TOPSIS for Decision-making Problems with Interval Data: Interval Efficiency.“ *Mathematical and Computer Modelling*, vol.49(5-6) (2009): pp. 1137-1142.

- [Jiang and Li, 2006] Jiang, Y. P., and B. Li. „An Approach to Determine the Attribute Weights Based on Different Formats of Evaluation Information.“ *International Journal of Computer Science and Network Security*, vol.6(10) (2006): pp. 285-289.
- [Kambhampati, 1999] Kambhampati, S. „Improving Graphplan's Search with EBL & DDB Techniques.“ *In: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*. Stockholm, Sweden, 1999. pp. 982-987.
- [Kambhampati et al., 1997] Kambhampati, S., E. Parker, and E. Lambrecht. „Understanding and Extending Graphplan.“ *In: Proceedings of the 4th European Conference on Planning: Recent Advances in AI Planning (ECP'97)*. Toulouse, France, 1997. pp. 260--272.
- [Kawas et al., 2006] Kawas, E., M. Senger, and M. D. Wilkinson. „BioMoby extensions to the Taverna workflow management and enactment software. “ *BMC Bioinformatics*, vol.7(523), (2006): pp. 16-29.
- [Kiefer and Bernstein, 2008] Kiefer, C., and A. Bernstein. „The Creation and Evaluation of iSPARQL Strategies for Matchmaking.“ *In: Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*. Tenerife, Spain, 2008. pp. 463-477.
- [Klusch et al., 2005] Klusch, M., A. Gerber, and M. Schmidt. „Semantic Web Service Composition Planning with OWLS-Xplan.“ *In: Proceedings 1st International AAAI Fall Symposium on Agents and the Semantic Web*. Arlington VA, USA, 2005. 52-59.
- [Klusch et al., 2009a] Klusch, M., B. Fries, and K. Sycara. „OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services.“ *Web Semantics: Science, Services and Agents on the World Wide Web*, vol.7(2), (2009): pp. 121-133.
- [Klusch et al., 2009b] Klusch, M., P. Kapahnke, and L. Zinnikus. „Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer.“ *In: Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*. Heraklion, Greece, 2009. pp. 550-564.
- [Klusch and Kaufer, 2009] Klusch, M., and F. Kaufer. „WSMO-MX: A Hybrid Semantic Web Service Matchmaker.“ *Web Intelligence and Agent Systems*, vol.7(1), (2009): pp. 23-42.
- [Kong et al., 2004] Kong, Y. C., C. L. Wang, and F.C. M. Lau. „Ontology Mapping in Pervasive Computing Environment.“ *Lecture Notes in Computer Science*, vol.3207 (2004): pp. 1014-1023.
- [Kopecký et al., 2007] Kopecký, J., T. Vitvar, and J. Farrell. „SAWSDL: Semantic Annotations for WSDL and XML Schema.“ *Internet Computing*, vol.11(6), (2007): pp. 60-67.
- [Kopecký and Vitvar, 2008] Kopecký, J., and T. Vitvar. „WSMO-Lite: Lowering the Semantic Web Services Barrier with Modular and Light-Weight Annotations.“ *In: Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008)*. Santa Clara, CA, 2008. pp. 238-244.

- [Kritikos and Plexousakis, 2009] Kritikos, K., and D. Plexousakis. „Mixed-Integer Programming for QoS-Based Web Service Matchmaking.“ *IEEE Transactions on services computing*, vol.2(2), (2009): pp. 122-139.
- [Krogt and Weerd, 2005] Krogt, R. V., and M. D. Weerd. „Plan Repair as an Extension of Planning.“ *In: Proceedings of the 15th International Conference on Automated Planning and Scheduling(ICAPS 05)*. Monterey, USA, 2005. pp.161-170.
- [Leng et al., 2010] Leng, Y., M. El-Gayyar, and A. B. Cremers. „Semantics Enhanced Composition Planner for Distributed Resources.“ *In: Proceedings of the 9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES 10)*. Hongkong, China, 2010. pp.61-65.
- [Leng et al., 2009] Leng, Y., M. El-gayyar, and S. Shumilov. „Semantically Enriched Integration System For Heterogeneous Web Services.“ *In: Proceedings of LADIS International Conference on WWW/Internet (WWW/Internet 09)*. Roma, Italy, 2009. pp.202-213.
- [Leymann, 2005] Leymann, F. „Combining Web Services and the Grid: Towards Adaptive Enterprise Applciaitons.“ *In: Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005)*. FEUP, Edicoes, 2005. pp. 9-21.
- [Li et al., 2010] Li, W. Q., X. M. Dai, and H. Jiang. „Web Services Composition based on Weighted Planning Graph.“ *In: Processing of the 1st International Conference on Networking and Distributed Computing (ICNDC 2010)* . Hangzhou, China, 2010. pp. 89-93.
- [Li et al., 2003] Li, Y., A. Z. Bandar, and D. Mclean. „An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources.“ *IEEE Transactions on Knowledge and Data Engineering*, vol.15(4) (2003): pp. 871-882.
- [Li et al., 2009] Li, Z., F. C. Yang, and S. Su. „Fuzzy Multi-Attribute Decision Making-Based Algorithm for Semantic Web Service.“ *Journal of Software*, vol.20(3) (2009): pp. 583-596.
- [Li and Zhou, 2009] Li, S. and J. Zhou. „The WSMO-QoS Semantic Web Service Discovery Framework.“ *In: Proceedings of International Conference on Computational Intelligence and Software Engineering (CiSE 2009)* . Wuhan, China, 2009. pp. 1-5.
- [Lin et al., 2008] Lin, N., U. Kuter, and E. Sirin. „Web Service Composition with User Preferences.“ *In: Proceedings of the 5th European Semantic Web Conference on the Semantic Web: Research and Applications (ESWC'08)* . Tenerife, Spain, 2008. pp. 629-643.
- [Liu and Shao, 2010] Liu, Y., and Q. Z. Shao. „The Similarity Calculation of Concepts from Different Ontologies Based on Cosine.“ *In: Proceedings of the 3rd International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII 2010)*. Kuming, China, 2010. pp. 130-134.
- [Lo et al., 2011] Lo, C. C., C. F. Tsai, and K. M. Chao. „Service Selection based on Fuzzy TOPSIS Method.“ *In: Proceedings of IEEE the 24th International Conference on Advanced*

- Information Networking (WAINA '10) and Applications Workshops*. Perth, Australia, 2011. pp. 367-372.
- [Maleshkova et al., 2010] Maleshkova, M., C. Pedrinaci, and J. Domingue. „Semantic Annotation of Web APIs with SWEET.“ *In: Proceedings of the 6th Workshop on Scripting and Development for the Semantic Web*. Crete, Greece, 2010. pp. 55-67.
- [Martin et al., 2007] Martin, D., et al. „Bringing Semantics to Web Services with OWL-S.“ *World Wide Web*, vol.10(3) (2007): pp. 243-277.
- [Martin and Domingue, 2007] Martin, D., and J. Domingue. „Semantic Web Services: Past, Present and Possible Futures (Systems Trends and Controversies).“ *IEEE Intelligent Systems*, vol.11(6) (2007): pp. 60-67.
- [Mayer et al., 2009] Mayer, W., R. Thiagarajan, and M. Stumptner. „Service Composition as Generative Constraint Satisfaction.“ *In Proceedings of the 2009 IEEE International Conference on Web Services*. Los Angeles, USA, 2009. pp. 888-895.
- [Menasce, 2002] Menasce, D. A. „QoS Issues in Web Services.“ *IEEE Internet Computing*, vol.6(6) (2002): pp. 72-75.
- [Nagarajan et al., 2007] Nagarajan, M., K. Verma, A. P. Sheth, and J A. Miller. „Ontology Driven Data Mediation in Web Services.“ *International Journal of Web Services Research*, vol.49(7) (2007): pp. 104-126.
- [Naseri and Towhidi, 2007] Naseri, M., and A. Towhidi. „Qos-Aware Automatic Composition of Web Services Using AI Planners.“ *In: Proceedings of the 2nd International Conference on Internet and Web Applications and Services (ICIW'07)*. Morne, Mauritius, 2007. pp. 29-35.
- [Nebel and Koehler, 1995] Nebel, B., and J. Koehler. „Plan Reuse Versus Plan Generation:a Theoretical and Empirical Analysis.“ *Artificial Intelligence*, vol.76(1-2) (1995): pp. 427-454.
- [Oh and Kumara, 2006] Oh, S., and R.S. Kumara. „A Comparative Illustration of AI Planning-based Web Services Composition.“ *ACM SIGecom Exchanges*, vol.5(5) (2006): pp. 1-10.
- [Olson, 2004] Olson, D. L. „Comparison of Weights in TOPSIS Models.“ *Mathematical and Computer Modelling*, vol.40(7-8) (2004): pp. 721-727.
- [Paolucci et al., 2002] Paolucci, M., T. Kawamura, T. R. Payne, and K. P. Sycara. „Semantic Matching of Web Services Capabilities.“ *In: Proceedings of the 1st International Semantic Web Conference on The Semantic Web (ISWC '02)*. Sardinia, Italy, 2002. pp. 333-347.
- [Papazoglou et al., 2007] Papazoglou, P. M, Traverso, P., S. Dustar, and F. Leymann. „Service-Oriented Computing: State of the Art and Research Challenges.“ *Computer* 40, vol.40(11) (2007): pp. 38-45.

- [Parker, 1999] Parker, E. „Making Graphplan Goal-Directed.“ *In: Proceedings of the 5th European Conference on Planning: Recent Advances in AI Planning (ECP'99)*. Durham, United Kingdom, 1999. pp. 333-346.
- [Peer, 2005] Peer, J. „A POP-Based Replanning Agent for Automatic Web Service Composition.“ *Lecture Notes in Computer Science*, vol.3532 (2005): pp. 189-198.
- [Pop et al., 2009] Pop, C. B., V. R. Chifu, I. Salomie, and M. Dinsoreanu. „Optimal Web Service Composition Method based on an Enhanced Planning Graph and Using an Immune-inspired Algorithm.“ *In: Proceedings of IEEE the 5th International Conference on Intelligent Computer Communication and Processing (ICCP'09)*. Cluj-Napoca, Romania, 2009. pp. 291 - 298.
- [Pyle, 1999] Pyle, D. *Data preparation for data mining*. San Diego, USA: Academic Press, 1999.
- [Rijsbergen, 1979] Rijsbergen, C.J. van. *Information Retrieval*. Butterworth, 1979.
- [Roman et al., 2005] Roman, D., et al. „Web Service Modeling Ontology.“ *Applied Ontology*, (2005): pp. 77-106.
- [Russell and Norvig, 2002] Russell, S., and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2002.
- [Saghafian and Hejazi, 2006] Saghafian, S., and S. R. Hejazi. „Multi-criteria Group Decision Making Using a Modified Fuzzy TOPSIS Procedure.“ *In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-LAWTIC'06)*. Sydney, Australia, 2006. pp. 215-221.
- [Şenvar and Bener, 2006] Şenvar, M., and A. Bener. „Matchmaking of Semantic Web Services Using Semantic-Distance Information.“ *In Lecture Notes in Computer Science*, pp. 177-186. Berlin: Springer, 2006.
- [Sheshagiri et al., 2003] Sheshagiri, M., M. desJardins, and T. Finin. „A Planner for Composing Services Described in DAML-S.“ *In: Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering*. Melbourne, Australia, 2003. pp. 45-51.
- [Sheth et al., 2007] Sheth, A. P., K. Gomadam, and J. Lathem. „SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups.“ *Internet Computing*, vol.11(6) (2007): pp. 91-94.
- [Shumilov et al., 2008] Shumilov, S., Y. Leng, M. El-Gayyar, and A. B. Cremers. „Distributed Scientific Workflow Management for Data-Intensive Applications.“ *In: Proceedings of the 12th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'08)*. Kunming, China, 2008. pp. 65-73.
- [Simanaviciene and Ustinovichius, 2010] Simanaviciene, R., and L. Ustinovichius. „Sensitivity Analysis for Multiple Criteria Decision Making Methods: TOPSIS and SAW.“ *In:*

- Proceedings of the 6th International Conference on Sensitivity Analysis of Model Output (SAMO2010)*. Milano, Italy, 2010. pp. 7743-7744.
- [Sohrabi and Mcilraith, 2009] Sohrabi, S., and S. A. Mcilraith. „Optimizing Web Service Composition While Enforcing Regulations.“ *In: Proceedings of the 8th International Semantic Web Conference (ISWC '09)*. Chantilly, USA, 2009. pp. 601-617.
- [Steuer, 1986] Steuer, R. E. *Multiple Criteria Optimization: Theory, Computation and Application*. New York: John Wiley, 1986.
- [Syu et al., 2011] Syu, Y., Y. Fanjiang, J. Kuo, and S. Ma. „Towards a Genetic Algorithm Approach to Automating Workflow Composition for Web Services with Transactional and QoS-Awareness .“ *In: Proceedings of IEEE World Congress on Service (SERVICE 2011)*. Washington, USA, 2011. pp. 295-302.
- [Thiagarajan et al., 2008] Thiagarajan, R., G. Manjunath, and M. Stumptner. „Computing Semantic Similarity Using Ontologies.“ HP Laboratories, 2008.
- [Tondello and Siqueira, 2008] Tondello, G. F., and F. Siqueira. „The QoS-MO Ontology for Semantic QoS Modeling.“ *In: Proceedings of the 23rd ACM Symposium on Applied Computing (SAC 2008)*. Fortaleza, Brazil, 2008. pp. 2336-2340.
- [Tong and Zhang, 2006] Tong, X. H., and S. S. Zhang. „A Fuzzy Multi-attribute Decision Making Algorithm for Web Services Selection Based on QoS.“ *In: Proceeding of IEEE Asia-Pacific Conference on Service Computing (APSCC'06)*. Guangzhou, China, 2006. pp. 51-57.
- [Tversky, 1977] Tversky, A. „Features of Similarity.“ *In Psychological Review*, vol.84(2) (1977): pp. 327-352.
- [Vladislava, 2006] Vladislava, G. „Semantic Description of Web Services and Possibilities of BPEL4WS.“ *International Journal Information Theories and Applications*, vol.13(2) (2006): pp. 183-187.
- [Wang et al., 2006] Wang, X., T. Vitvar, M. Kerrigan, and I. Toma. "A QoS-aware Selection Model for Semantic Web Services." *In: Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC 2006)*. Chicago, USA, 2006. pp. 390-401.
- [Weld, 1994] Weld, D. „An Introduction to Least Commitment Planning.“ *AI Magazine*, vol.15(4) (1994): pp. 27-61.
- [Wen et al., 2006] Wen, G., L. Liang, and R. N. Shadbolt. „Ontology-Based Similarity Between Text Documents on Manifold.“ *In: Proceedings of the 1st Asian Semantic Web Conference*. Beijing, China, 2006. pp. 115-125.
- [Wu et al., 2007] Wu, X. Z., A. Ranabahu, K. Gomadam, A. P. Sheth, and J. A. Miller. Automatic Composition of Semantic Web Services using Process and Data Mediation. *Technical Reports*, Georgia, USA: LSDIS lab, University of Georgia, 2007.

- [Xiao et al., 2010] Xiao, Y., X. Zhou, and X. Huang. „Automated Semantic Web Service Composition Based on Enhanced HTN.“ *In: Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering (SOSE)*. Nanjing, China, 2010. pp. 59-63.
- [Yan et al., 2009] Yan, Y. X., B. Xu, Z. F. Gu, and S. Luo. „A QoS-Driven Approach for Semantic Service Composition.“ *In: Proceedings of the 12th IEEE Conference on Commerce and Enterprise Computing (CEC'09)*. Vienna, Austria, 2009. pp. 523-526.
- [Yan et al., 2010a] Yan, Y., P. Poizat, and L. Zhao. „Repairing Service Compositions in a Changing World.“ *Studies in Computational Intelligence*, vol. 296 (2010): pp. 17-36.
- [Yan et al., 2010b] Yan, Y., P. Poizat, and L. Zhao. „Self-Adaptive Service Composition Through Graphplan Repair.“ *In: Proceedings of IEEE International Conference on Web Service (ICWS)*. Miami, USA, 2010. pp. 624-627.
- [Yan and Zheng, 2008] Yan, Y., and X. Zheng. „A Planning Graph Based Algorithm for Semantic Web Service Composition.“ *In: Proceedings of E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services (CECandEEE 08)*. Washington, USA, 2008. pp. 339 - 342.
- [Yang et al., 2006] Yang, J. H., C. W. Lan, and J. Y. Chung. „Analyses of QoS-Aware Web Services.“ *In: Proceedings of International Computer Symposium (ICS 2006)*. Taipei, China, 2006. pp. 880-885.
- [Yoon and Hwang, 1995] Yoon, K. P., and C. L. Hwang. *Multiple Attribute Decision Making: An Introduction*. Sage Publications, 1995.
- [Zanakis et al., 1998] Zanakakis, H. S., A. Solomon, N. Wishart, and S. Dublsh. „Multi-attribute Decision Making: a Simulation Comparison of Select Methods.“ *European Journal of Operational Research*, vol.107(3) (1998): pp. 507-529.
- [Zavadskas and Turskis, 2008] Zavadskas, E. K., and Z. Turskis. „A New Logarithmic Normalization Method in Games Theory.“ *INFORMATICA*, vol.19(2) (2008): pp. 303-314.
- [Zeleny, 1982] Zeleny, M. *Multiple Criteria Decision Making*. McGraw-Hill, 1982.
- [Zeng et al., 2004] Zeng, L. Z., B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. „QoS-Aware Middleware for Web Services Composition.“ *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol.30(5) (2004): pp. 311-327.
- [Zhang et al., 2010] Zhang, W., C. K. Chang, T. Feng, and H. Y. Jiang. „QoS-Based Dynamic Web Service Composition with Ant Colony Optimization.“ *In: Proceedings of IEEE the 34th Computer Software and Applications Conference (COMPSAC'10)*. Seoul, Korea, 2010. pp. 493 - 502 .