TOWARDS EFFICIENT COLLABORATION FOR E-SERVICE TRANSACTION MANAGEMENT: Semantics, Security and Reliability

By

Jiangang Ma

A Thesis Submitted to School of Engineering and Science

in Fulfillment of the Requirements for the degree of

DOCTOR OF PHILOSOPHY

Supervisor: Professor Yanchun Zhang

Australia

2010

DECLARATION

I, Jiangang Ma, declare that the PhD thesis entitled *Towards Efficient Collaboration for e-Service Transaction Management: Semantics, Security and Reliability* is no more than 100,000 words in length including quotes and exclusive of tables, figures, appendices, bibliography, references and footnotes. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work.

Signature **Date**

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to thank many people who have contributed to the successful completion of this thesis.

First, I would like to express my sincere gratitude and thanks to my principal supervisor, Professor Yanchun Zhang, both in my academic research and in other aspects of life during my PhD research. I have learned much how to do research from him. Our regular discussions and meetings have immensely contributed my research. This research work would have not been completed without his valuable guidance, encouragement and suggestions. My thanks also go to my co-supervisor, Dr. Fuchun Huang, for his valuable comment, discussion, and assistance throughout my research.

Second, I wish to thank postgraduate officers, Ms. Natalie Gloster, Ms. Lesley Birch and Ms. Elizabeth Smith for their organizing all academic research activities for my PhD research. I highly appreciate their excellent work for assisting PhD students. My thanks also go to the School of Engineering and Science at Victoria University for providing me with an excellent research environment.

Furthermore, I would like to thank all the staff members and my classmates at the School of Engineering and Science. In particular, my thanks go to A/Prof. Xu Yi, A/Prof. Hao Shi, A/Prof. Yuan Miao, Ms. Anne Venables, Dr Alasdair McAndrew, Dr Gitesh Raikundalia and Prof. Pietro Cerone, for their encouragement, suggestions and assistance.

Finally, I would like to thank my wife Yan for her loving encouragement, patience and great support throughout my PhD research.

ABSTRACT

With advances in Web technologies, business and scientific communities are increasingly depending on collaboration to support a variety of business domains, such as enterprise production management and e-business integration, and to assist scientists in their researches. These business and scientific explorative activities are typically carried out in multi-organizations, run over long periods of time and involved in different business processes and huge amounts of data. In these situations, data resources (e.g. business databases and medical testing data), individual computational tasks (e.g. gene comparison) and user-created workflows (e.g. experimental steps) are likely to be distributed over various locations, connected over the Internet. In addition, these distributed resources can be further wrapped as Web services, which are published and deployed on the Web, so that other business partners are able to share these valuable computational resources for facilitating their business and scientific activities. These collaborations in distributed environment allow business partners to transform a complicated business or scientific problem into a series of simpler, and more easily handled ones.

There are a number of unique challenges that remain unaddressed in a collaborative Web services-based transaction context. The first challenge is the lack of a set of semantic descriptions for Web services. As Web services are usually created by different providers that employ different phrases and models for presenting and describing services' characteristics and capabilities, it is difficult to correctly understand the relevant semantic information hidden in the service descriptions for effective discovery and composition of Web services.

The second challenge is security, which is currently ignored in most of Web services-based transaction systems. This is particularly important in a Web services context because the messages transmitted over the Web could be accessed and altered by impostors. For security reasons, it is important that confidential information like private business processes and sensitive data (e.g. patients' private data in medical research) should not be visible or released to unauthorized users.

Another challenge to overcome in this collaborative environment is that efficient fault tolerant mechanisms are needed for ensuring reliable execution of collaborative systems. A basic requirement for reliable collaboration in distributed experiments is that the business processes or experiments should be allowed to proceed without violating business process specifications and data consistencies even when the failures occur for some reasons. This requirement is particularly applicable to distributed transactions because the collaboration may consist of hundreds of separate processes or computing steps and may involve lots of data objects whilst the computing steps may fail. In such cases, giving up the overall collaboration would be too expensive.

In this thesis, we address the challenges mentioned earlier by presenting an access control framework and models for supporting semantic, secure and reliable collaboration. The proposed approaches depend on workflow technologies by integrating control flows and data flows into a workflow model, and they employ a relaxed transaction concept for fault tolerance, in order to maintain the processes consistencies, as well as, data consistencies even in presence of failures. To achieve these objectives, we first locate relevant Web services by a semantic approach. The key idea of our

approach is to indirectly associate the intention of users to the advertisements in Web services by applying Probabilistic Latent Semantics Analysis (PLSA). Then, we model the internal activities within an institution by employing workflow technology whilst the individual workflow models of participating institutions can be mapped to process views, which are further described by Web services. Based on the process view model, a two layered access control architecture is proposed to protect computing resources. Finally, the atomicity concept is relaxed by integrating transactions and exception handling models in order to ensure the reliable collaborations on the Web.

The main contributions of this thesis are threefold. First, we proposed a novel approach to find relevant services through the combination of keyword technique and the semantics extracted from the services' descriptions by using a probabilistic semantic approach in order to handle poor scalability and lack of semantics. Second, a novel two layered access control model based on the general principles of the order of security priority in an organization is proposed. With the model, various roles are associated with views, so that right users can only see necessary parts of workflows exposed to them. Third, the atomicity sphere model is relaxed by considering two levels of atomicity abstraction for supporting reliable collaboration at the level of process, as well as, at the level of data to maintain the process consistency and data consistency in case of failures.

PUBLICATIONS BASED ON THIS THESIS

Refereed Journal Papers

[1] Jiangang Ma, Jinli Cao and Yanchun Zhang**.** Efficiently Supporting Secure and Reliable Collaboration in Scientific Workflows. In Journal of Computer and System Sciences (JCSS), Volume 76, Issue 6, September 2010, Pages 475-489, 2010.

Refereed International Conference Papers

[2] Jiangang Ma, Yanchun Zhang and Jing He. Web Services Discovery based on Latent Semantic Approach**.** In the proceedings of The IEEE International conference on web services (ICWS2008), September 23-26, China, 2008.

[3] Jiangang Ma and Yanchun Zhang. Efficiently Finding Web Services Using a Clustering Semantic Approach. In the proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection Integration and Adaptation: organized with the 17th International World Wide Web Conference (WWW 2008), CSSSIA 2008, Beijing, China, April 22, 2008.

[4] Yanchun Zhang and Jiangang Ma. Discovering Web Services based on Probabilistic Latent Factor Model. In the proceedings of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management, China, June16-18, 2007.

[5] Jiangang Ma, Jinlin Cao and Yanchun Zhang. A Probabilistic Semantic Approach for Discovering Web Services. In the proceedings of 16th International World Wide Web Conference (WWW 2007) Banff, Canada, May 8 -12, 2007.

[6] Jiangang Ma, Yanchun Zhang and Minglu Li. OMWSC-An Ontology-Based Model for Web Services Composition. In the Proceedings of Fifth International Conference on Quality Software (QSIC 2005). Australia, 2005.

[7] Guandong Xu, Yanchun Zhang, Jiangang Ma and Xiaofang Zhou. Discovering User Access Pattern Based On Probabilistic Latest Factor Model**.** In the Proceedings of the Sixteenth Australasian Database Conference, Australia, 2005.

1. Chapter 1

Introduction

Fierce competition and quickly changing markets in today's global economy demand business enterprises to provide high-quality products or services, with short life cycles in order to meet increasing expectations of customers. This implies that enterprises need to possess a broad range of knowledge, techniques and skills, ranging from developing capabilities at the logistics layer and the transaction layer [46, 121] to provide competitive products and services. However, it is difficult for a single enterprise, in particular, for small and medium-sized enterprises (SME), to have such comprehensive skills and capabilities either for suffering high cost or for infeasibility. This, together with the advances in Information and Communication Technologies (ICT), particularly with Service-Oriented Architecture (SOA), has motivated the evolution of collaborative techniques to manage it.

Business collaborative activity is a dynamic process in which a set of independent partners (organizations, institutions, or specialized individuals) work together to form a temporary business alliance, called a virtual organization or virtual enterprise [58, 110], where each member of business alliance contributes parts to the overall virtual enterprise in order to exploit an apparent market opportunity. Currently, business enterprises and scientific communities are increasingly depending on collaborations to support a variety of business and scientific activities, such as enterprise production management, e-

business integration and scientific research experiments. In particular, small and medium-sized enterprises (SMEs) represent the major parts of the collaborations. For example, SMEs comprise 99% of all companies in the European Union [110]. As demonstrated by a recent survey [131] from AT&T, about 75% of firms in a worldwide expect the number of collaborative relationships with third parties overseas to grow.

Effectively collaborating with business partners in a distributed environment poses a number of significant challenges. First, the collaboration needs to take into consideration semantics that has an impact on finding proper business partners. These business partners often integrate existing applications, including data, business logic and other resources in terms of Web services, which are published and deployed on the Web, so that other trading partners are able to discover and share these valuable computational resources for facilitating their collaborations. As Web services are usually created by different providers that employ different phrases and terms for presenting and describing services' characteristics and capabilities, it is difficult for trading partners to correctly understand the relevant semantic information hidden in the service descriptions for effective discovery and composition of Web services [76, 123].

Second, security has also impact on the effective collaboration among different trading partners. In Web-based collaborative systems, the messages transmitted over the Web could be accessed and altered by impostors. For security reasons, it is important that confidential information like private business processes and sensitive data (e.g. patients' private data in medical research) should not be visible or released to unauthorized users.

Finally, reliability is another challenge to be overcome in a distributed collaborative environment. A basic requirement for reliable collaboration in distributed experiments is

that the business processes or scientific research experiments should be allowed to proceed without violating business process specifications and data consistencies even when the failures occur for some reasons. This requirement is particularly applicable to distributed transactions because the collaboration may consist of hundreds of separate processes or computing steps and may involve lots of data objects whilst the computing steps may fail. In such cases, giving up the overall collaboration would be too expensive. At the same time, Service-Oriented Architecture (SOA) has emerged as an effective means to support cooperating applications that seamlessly join together heterogeneous software modules from diverse organizations in the form of Web services. Our research is motivated by the need to facilitate the effective collaboration in a semantic, reliable and secure manner.

This chapter is organized as follows. In Section 1.1, we outline the research problems that will be tackled in this thesis. In Section 1.2, we summarize proposed solutions and we present the structure of this thesis in Section 1.3.

1.1 Motivation and Research Problems

SOA is an emerging software engineering paradigm for developing distributed Internet applications. In this paradigm, Web services are application components that are supported by a set of emerging standards such as the Web Service Description Language (WSDL) [132] for service description, the Universal Description, Discovery and Integration (UDDI) [134] for service discovery, the Simple Object Access Protocol (SOAP) [133] for service communication, and the Web Service Business Process Execution Language (WS-BPEL) [135] for workflows. With SOA, business companies

are able to collaborate with other counterparts for a given goal, by assembling these application components with little effort to create business processes. These business processes can be further improved via automated techniques such as Workflows in order to enhance overall efficiency and effectiveness of business collaboration.

Typical examples in a distributed collaboration where semantics, reliability and security can be of great importance are in the fields of e-business and e-research. Consider the first case related to supply chain management and manufacturing processes that involve a high level of complexity and many business trading partners with complex interrelationships. For example, the completion of whole business transaction involves selecting goods remotely, managing orders with an electronic cart, paying electronically and tracking the shipment, and so on. Consequently, any failure during the collaboration among trading partners may result in unreliability in the distributed software systems, and may have a significant commercial impact. For example, a customer may have paid the bill whereas the supplier fails to deliver the ordered products.

The second example is concerned with scientific applications in bioinformatics like DNA analysis. These scientific explorative activities are increasingly the result of collaborative efforts among scientists who make use of each other's experimental results. In these situations, data resources (e.g., DNA databases and medical testing data), individual computational tasks (e.g., gene comparison) and scientist-created workflows (e.g., experimental steps) are likely to be distributed over various locations, connected over the Internet. In addition, these distributed resources can be further wrapped as Web services, which are published and deployed on the Web, so that other scientists are able to share these valuable computational resources for facilitating their

scientific researches. As a result, these collaborations in a distributed environment allow scientists to transform a complicated scientific research problem into a series of simpler, and more easily handled steps.

Although current Web technologies have achieved a certain level in facilitating applications mentioned above, there are a number of unique challenges that remain unaddressed in distributed collaborations. These challenges include: (i) how to effectively discover Web services based on semantics, (ii) how to securely access Web services in a distributed context, and (iii) how to reliably collaborate trading partners in the presence of failures in a distributed transaction context.

1.1.1 Discovering Web Services Based on Semantics

Web services discovery process mainly involves locating desired Web services either published in a registry like UDDI or scattered in P2P systems, matching users' requirements to a set of Web services and returning relevant ones to the consumers. One of the current discovery approaches is based on keyword strategy. The keyword-based discovery approach consists of two main steps. A service user first types keywords into a Web service search engine to look for the corresponding services. The user then selects the desired Web services returned.

Web service discovery has played an important role in the overall cycle of the business collaborative process, including process design phase and process running phase [121, 11]. The process design is to formally model whole business activities by describing different aspects among collaborative activities, such as data flows, control flows and transaction dependencies among tasks making up a business process, and so on. After process design phase is done, Web services must be discovered for the tasks deigned.

Here Web service discovery refers to a process of locating the appropriate collaborative participators (Web services and Web service providers). In Service-Oriented Architecture, a set of Web services might be found for individual tasks in a local repository or located across different organizations over the Internet. During the run-time phase, the Web services discovered are invoked through enabling data flows and control flows among tasks. When individual Web service completes its application invocation on behalf of the tasks, the collaborative process proceeds to the next tasks until the whole collaborative objective is realized. In this collaborative application, service consumers and service providers need to agree on the descriptions on Web services in terms of functional and non-functional properties. The functional properties specify what Web services can do whilst non-functional properties mean the quality of services, such as reliability, security and transaction.

Efficient discovering Web services is a challenging issue for several reasons. Firstly, with the ever-increasing number of services published on the Internet, it is difficult for service consumers to select the best ones to complete their tasks because a large number of Web services offer similar functionalities. Secondly, keywords are insufficient in expressing semantic concepts, which is partially due to the fact that keywords are often described by natural language, being much richer in terms of diversity. As a result, finding desired Web services is akin to looking for a needle in a haystack [43].

Existing discovering mechanisms like UDDI already enable much of the Web service discovery process. However, they concentrate largely on syntactic approaches, such as by using keyword-based search and category-based matching, which is inefficient and

time-consuming. Therefore, there is a need to have effective approaches to semantically discovering Web services for fascinating distributed collaborative applications.

1.1.2 Securely Accessing Web Services

The second challenge is security, which is currently ignored in most current collaboration applications. The existing collaborative applications like workflow systems are based on the assumptions that the collaboration among trading partners is conducted within a dedicated infrastructure whilst the integration of different information and securely accessing Web services are under control of a trusted centralized coordination mechanism. On such an assumption, participants involving in a collaborative process have a prior expectation of what applications will be accessed and how clients will invoke Web services.

Nevertheless, modern collaborative applications are more complicated than traditional applications because of heterogeneous domains of collaborative applications and highly dynamic environments [95, 121]. Here heterogeneous application domains make it difficult to enforce security collaborations because different participants belonging to separate security domains may use various security mechanisms, such as different access control policies and different authorization approaches. In addition, as Web resources involving collaborations are increasingly exposed to end users in a dynamic environment, they are more likely accessed and altered by impostors. Therefore, for security reasons, there is a need that confidential information like private processes and sensitive data (e.g. patients' private data in medical research) should not be visible or released to unauthorized users

1.1.3 Reliable Collaboration in Distributed Transaction Management

A robust business application should be reliable. In other words, it should be able to continue to complete its work, and capable of adapting itself to some failures. There are a variety of sources that are responsible for failures. A failure may be due to hardware reasons, such as computers crash or Internet communication interruption, or it may come from software reasons, for instance, an incorrect commuting outcome caused by executing a task in a workflow.

Reliable business collaboration in a Web service context is even more complex. First, it requires loosely coupled connections between different distributed systems so that business partners involving a collaboration can cooperate more freely across the Internet. Second, the collaboration is established in a highly dynamic fashion and on-demand basis. For example, new participants (e.g., new services) may join the collaboration whilst existing services may be removed at any time. This requires efficient adaptive collaboration techniques in order to ensure reliable execution of business processes. Here, a reliable distributed collaboration system refers to one that can continue to process users' requests even when some errors occur. In other words, even if components of distributed transaction management system fail, a reliable distribute transaction system should be able to continue executing users' requests without violating process consistency and data consistency.

It is a challenging work to build a reliable and secure collaboration across independent organizational boundaries and their systems. The challenge is how to link the elements of collaborative business processes together into a cohesive whole as well as how to maintain the consistency of the collaboration as a whole in the face of failures.

1.2 Proposed Solutions

In this thesis, we address the challenges mentioned earlier by presenting a transaction framework and models for supporting semantic discovery of Web services and maintaining secure and reliable collaborations. The proposed process-oriented infrastructure of collaboration for e-service transaction management will be referred to as acronym CSTM (towards efficient collaboration for e-service transaction management: semantics, security and reliability) in the remainder of this thesis. The proposed approaches depend on workflow technologies by integrating control flows and data flows models, and they employ a relaxed transaction concept for fault tolerance, in order to maintain the process consistencies, as well as, data consistencies even in presence of failures.

1.2.1 Collaboration through Web Services Discovery Based on Latent Semantics Analysis

To address the first challenge, we propose a novel approach for discovering and matching Web services for distributed collaboration [76, 77, 122]. The proposed semantics-based approach is based on our observation of uncertainty on the usage of Web services in the Web environment. This uncertainty is reflected in two aspects. On the client sides, a service user may not have a specific goal in his/her mind while he/she browses Web service categories on the Web. For this reason, the query the user selects may not fully represent his/her real intention. Second, it is difficult for users to choose appropriate words to indicate semantic concepts because of the dictionary problem [38]. In our approach, the key idea is to indirectly associate the intention of users to the

advertisements in Web services by applying latent semantics analysis. As a result, Web services can be matched against a query at the concept level. The salient features of our approach are:

- Complying with currently dominating industrial techniques for Web services discovery. At present, the dominating industrial techniques for Web services discovery and description are to use UDDI registry and WSDL description. Our method is compatible with the existing standards through combining the keyword technique and the semantics extracted from the service WSDL descriptions, which is different from traditional keyword-based technique for Web service discovery.
- Supporting scalability for Web service discovery. We propose a divide and conquer semantic discovery approach in order to transform a complex problem into a series of simpler ones, which can be handled more easily. The objective is to handle the poor scalability for Web service discovery in the Web environment.

1.2.2 A Layered Access Control Architecture Based on Process View

We propose a layered access control architecture to securely access Web services [75]. The proposed layered security architecture is based on the general principles of the order of securing priority in an organization. Accordingly, the higher securing requirements in an organization denote the trust objectives of the organization and specify the security of the organization's external interface to its environment. Based on this general principle, this security requirement should be put at the first level. On the other hand, the security requirements for the components within a system can be put at the second level. Such security requirements focus on the securing integrity of the components, indicating that

resources such as data and file should not be modified by unauthorized executing entities. The basic features of our approach are:

- Balancing security and information choice for users. We use process views as the abstraction of the specifications for original business processes, which can be achieved by defining parts of the specifications. These parts of specifications are only visible and accessible to authorized users. The objective of our approach is to minimize what collaborative partners need to know during their collaboration with each other.
- Associating various roles with process views. We extend the Role-Based Access Control (RBAC) [102] model to securely access Web services. Based on the process view model, various roles can be associated with views, so that users can only see necessary parts of processes they are entitled to see.

1.2.3 Reliable Collaboration Based on Relaxed Atomicity Sphere Model

We propose a transaction approach [75] in order to support reliable collaboration in a distributed context. The main objective of the approach is to reduce the cost of recovering processes in case of failures, and ensure data consistency at various levels of granularity. For that purpose, the atomicity sphere concept is relaxed by integrating transactions and exception handling models so that data consistency as well as process consistency will be maintained. The features of the proposed approach are:

- Maintaining data consistency. We apply the notion of atomicity concept to a single task level, ensuring the consistency of data in the case of failures. We model the

execution of a task as a run. Based on the run model, data flows are manipulated in the right way in the execution of a business process to ensure data consistency.

- Maintaining process consistency. We extend the notion of atomicity sphere by combining atomicity sphere with exception handling approach, aiming to maintain the consistency of computing processes. During the design phase, a collaborative application system is constructed by creating spheres properly. For example, a set of spheres may form a hierarchy structure where the outermost sphere represents a transaction whilst other spheres below the root indicate sub-transactions. Furthermore, we combine a backward error recovery and forward error recovery methods in order to ensure reliable collaboration.

1.3 Thesis Outline

This dissertation is organized in three parts. In the first part, we deal with semantics introduced in Web services discovery for collaborative processes. In the second part, we discuss security issues by introducing an access control framework and process view model. Finally, we focus on reliability by presenting a relaxed transaction model.

The underlying research issues constitute grand challenges in service-oriented computing, and need the integration of concepts and techniques from various disciplines, including SOA, business process management, workflows, database transaction and security.

Chapter 2 introduces research background related to reliable collaboration in a distributed environment, including basic technologies, ranging from service-oriented architecture (SOA), business process, and workflows to transaction management.

Furthermore, e-service transaction management of collaboration through Web services composition is reviewed and related frameworks are also discussed in this chapter.

Chapter 3 describes basic security and reliability requirements for distributed collaborations. In this chapter, we define basic models used for distributed collaborations. The basic models include task models, a Web service model and a workflow model.

Chapter 4 deals with semantic Web services discovery. The proposed novel discovery approach combines keywords techniques with the semantics extracted from the services' descriptions by employing a divide and conquer semantic discovery approach, in order to handle poor scalability and lack of semantics. The key idea of our approach is to indirectly associate the intention of users to the advertisements in Web services by applying Probabilistic Latent Semantics Analysis (PLSA). The chapter also provides the experimental results.

Chapter 5 discusses the security issue. In this chapter, we present an approach to support securely access Web services. Our approach is based on the process views, which place emphasis on information hiding, minimizing what trading partners need to know during their collaboration with each other.

Chapter 6 presents the design of enabling transactional support for failures handling. To this end, the atomicity sphere concept is extended by considering two levels of atomicity abstraction for supporting reliable collaboration.

Chapter 7 reports a framework to exemplify the usage of the approaches presented in this thesis for supporting semantic, reliable and secure collaborations.

Chapter 8 summarizes the contributions of the thesis and discusses future research

work.

2. Chapter 2

State of the Art

The need for the coordination of trading partners and the importance of transaction management in a distributed environment has been evident for long time. As traditional business approaches are mainly based on central control mechanisms by which trading partners are working together with closely coupled, it is quite difficult to facilitate the interoperability among trading partners and may have a significant impact on providing competitive products and services. As we discussed in the previous Chapter, effective collaborative approaches can cope with those issues, but they also pose a number of significant challenges such as semantics, security and reliability. Fortunately, service-oriented technologies are deemed as an effective means to support cooperating applications and have significant impact on expanding service-based economy [96, 123]. As demonstrated by a recent survey [108], about 75% of economic output in industrialized countries comes from the service-based economy.

In this chapter, we examine the background knowledge, introduce literature reviews and describe the evolutions from doing business via traditional approaches to doing business via modern approaches, which are related to semantics, security and reliability in distributed collaborations. The literature review presented here is more than just a survey of the literature of semantic, secure and reliable enabled e-service transaction management. As in the proposed thesis, it will reveal the key gaps in the current theory

and practice of collaborations in distributed contexts. It does this by first presenting background knowledge on business collaborations and describing basic terminologies, concepts and techniques, such as web services, business process and workflows, and so on. We then give brief overview of some research work relevant to web service transaction management frameworks and address their limitations.

This chapter is organized as follows: Section 2.1 introduces some background knowledge, concepts and terminologies used in this thesis. Section 2.2 provides a survey on Services-Oriented Architecture (SOA). Section 2.3 reviews some e-service transaction management prototypes. In Section 2.4, we discuss e-service collaborative transaction management through Web services compositions. Research problems in this thesis are provided in Section 2.5. Finally, a summary is reported in Section 2.6.

2.1 Background

This section introduces some background knowledge, concept and terminologies relevant to the research field in a semantic, reliable and secure collaboration.

2.1.1 Collaboration in Distributed Environment

In this section, we first introduce a simple car manufacturing scenario that will be used to illustrate the traditional approach to do business and its limitation, and then discuss business collaboration and its characteristics.

Doing Business via Traditional Approaches

Figure 2.1 illustrates a simple scenario of car manufacturing business. The starting point is a demand for car from customers. This demand, in the form of an order, will be placed

to an automobile dealer who further sends the order to an auto manufacturer through some kinds of methods. After receiving the order for car, the vehicle manufacturer commences in ordering the parts by contacting the parts suppliers, who in turn order raw materials in order to create the parts. Meanwhile, the suppliers may calculate the price of the ordered parts and the fee for shipping the parts to the manufacturer, and send an invoice to request payment. Once receiving the all the parts from the suppliers, the manufacturer begins to produce the car and send it the dealer. Normally, this business activity is a long transaction process that may last for weeks or months.

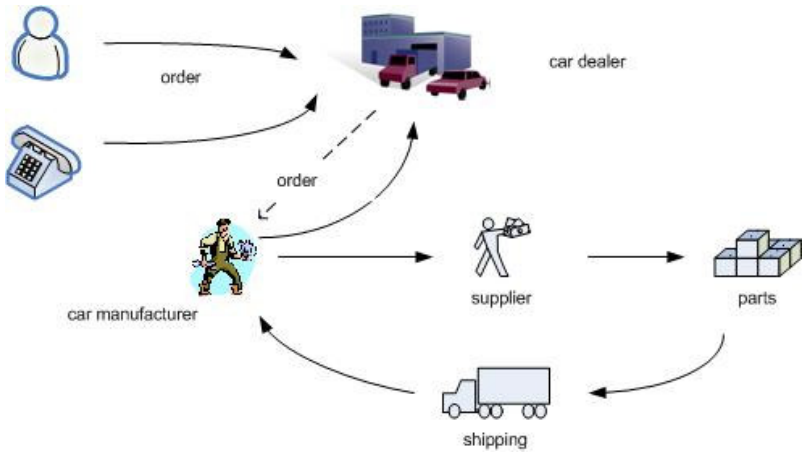

Figure 2.1: Traditional business approach

From this example, we can lead to several observations. First, whole business activity is a complex network that involves different actors related to a business transaction, such as manufacturers, suppliers, banks, transportation, warehouses and customers, and so on. Second, this business activity involving various actors can be further categorized by two layers: transaction layer and logistics layer [46]. Here transaction layer refers to the setting of the structure for actual business transaction, or more generally, denoting the exchange of goods, services or money. Logistics layer refers to general transportation

services like road transport, distribution and shipping, etc. Finally, the objective of a business is to be cost-effective from transaction layer to logistics layer.

We can further summarize the basic characteristics of traditional business approaches based on the car manufacturing scenario:

- Tightly coupling. Traditional business approaches emphases on intimated relationships among business partners. For example, actors in the scenario above mainly focus on direct relationship with their partners rather than have the end-to-end management of processes running across many different organizations. Furthermore, business information is shared with direct business partners by using traditional business approaches [46].
- Centrally controlling. Traditional business approaches are mainly based on a central decision making control mechanism. As mentioned above, in a traditional business activity, a business is viewed as a sequential chain of events (a value chain) by which business actors are working together with closely coupled. In such an architecture, the coordination among partners is adopted by a hierarchical mechanism by which a single top level activity defines the overall business goal and regulates the detailed transaction steps [95, 64].

Doing Business via Collaborative Approaches

As traditional business approaches show some disadvantages such as lack of flexibility and inability to rapidly meet a specific objective, and associated high costs, modern enterprises are seeking relatively effective collaborative approaches. Figure 2.2 shows an example [35].

In this scenario, seller company (SC) is deemed as a product or service provider whilst buyer company (BC) as a product requestor. To begin with, SC looks for collaborative specification and using scenarios in a registry (step 1). After familiarizing with the use-cases and related regulations, SC will build its own business applications that comply with the specification by using different software solutions such as using Web services to wrap existing applications (step 2). After that, SC will describe the functionality of its applications in terms of business profile, and then publish it to a registry (step 3). If BC intends to do business with SC, it will look for the functionality specification supported by SC in the registry (step 4). Next, the two companies will negotiate each other to reach a collaborative agreement on different aspects of a business like documents changing and security requirement, etc (step 5). At this point, BC sends a request to SC (step 6). Finally, the two companies are already to do business according to the reached collaboration agreement.

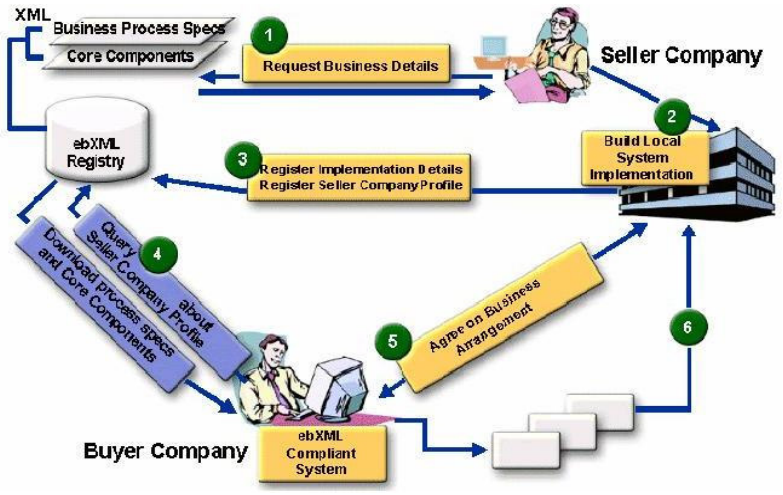

Figure 2.2: Business collaboration [35]

The detailed collaborative strategy and collaborative patterns will be discussed in the following section.

2.1.2 Business Collaboration Approaches

In the previous section, we have described the interaction of two business companies doing a business by using ebXML. In this section, we introduce some of basic collaborative approaches and patterns in a distributed context whilst some detailed mathematics models related to collaborations will be discussed throughout the remaining chapters.

.

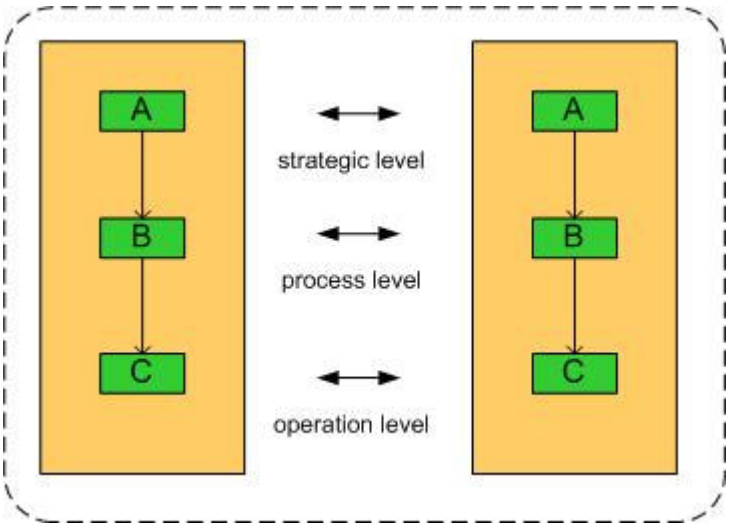

Figure 2.3: Business collaborative strategy

Collaborative Strategy

Based on SOA, modern enterprises do business by adopting collaborative approaches, which normally span a large spectrum of different trading partners' boundaries. In addition, the collaborative relationships among trading partners are built based on a multiple levels' strategy, from a high-level strategic level through the concept level to

the concrete operational level [123]. Figure 2.3 shows the general idea of doing business by using a collaborative approach.

- The strategic level deals with high-level decisions that involve different trading partners. This often includes general decisions regarding achieving a particular, potential common business objective, such as if two trading partners agree to design a software application together. As a result, a virtual organization will be formed according to the strategic decision made. Also at this level, each partner within the virtual organization expects to complete a task, for instance, one partner providing loans service and the other for insurance, and so on.
- The concept level describes business processes according to a value chain model. These concepts are generally well-defined coarse processes so that multiple scenarios are enabled to reflect the flexibility of a collaboration.
- The operational level focuses on the technical aspects of a collaboration to realize the scenario of a collaboration. At this level, each partner may develop its own applications, integrate existing applications as services and further wrap them as Web services. Furthermore, with Web services, the well-defined coarse processes can be realized through discovering and composing Web services to enable control flows and data flows among the partners.

Collaborative Patterns

The collaborative strategy above describes general cooperating ideas of doing business. Based on the basic principle, the concrete interaction of trading partners in a

collaboration can be categorized into three patterns: chained pattern, nested pattern and synchronized pattern [56], shown in Figure 2.4.

In a chained collaborative pattern (Figure 2.4 (a)), a business process triggers a collaboration by invoking a task in another business process. During the collaboration after the process triggering, the two collaborative processes have an independent relationship, that is, triggering process can independently continue to proceed or terminate without being subject to the triggered process. In a nested collaborative pattern (Figure 2.4 (b)), on the other hand, a task in business process triggers a collaboration by invoking a task in another business process. However, the triggering process has to wait until the completion of the triggered process. Finally, in a synchronized collaborative pattern (Figure 2.4 (c)), the two collaborative processes have a dependent relationship. At a collaborative point, two processes execute in parallel through exchanging synchronizing information and they can proceed to work after the completion of both processes.

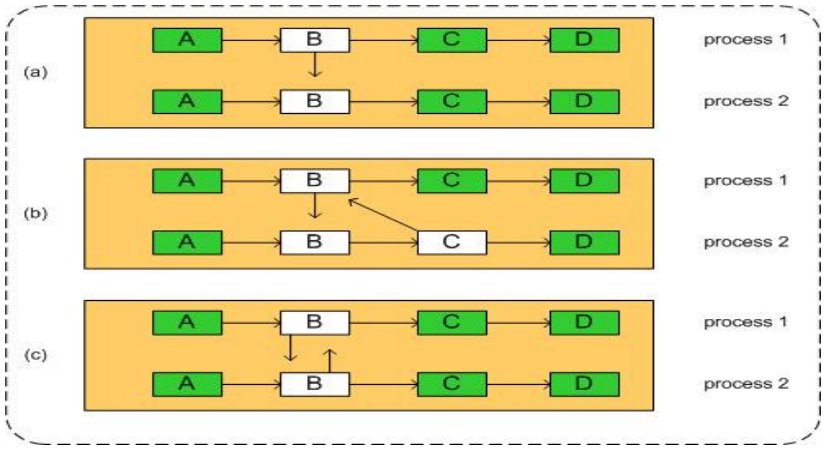

Figure 2.4: Collaborative patterns [56]

Forming Virtual Organization

Due to the increasing pressure of competition and market changing, business enterprises employ collaborative approaches to seek new business opportunities. This means that it may not effective for a single business enterprise to perform all of business activities alone. For example, it may be cheaper for an enterprise to select new business partners to do parts of its work because these partners may have highly specific expertise related to the work, so that acquiring another partner to do the job could lower business costs. What is more, business enterprises may change their business processes to further improve work efficiency by merging their processes with a partner's ones to run in a competitive manner. As a result, this outsourced running manner and the process changing results in forming a virtual organization. We summarize the above process of forming a virtual organization as follow [95], shown in Figure 2.5:

- Starting from an agreement on collaborations
- Reducing some repeating work units
- Developing independent work unit based on common objective
- Integrating relationship between suppliers and customers
- Forming virtual organization via repeating some above steps

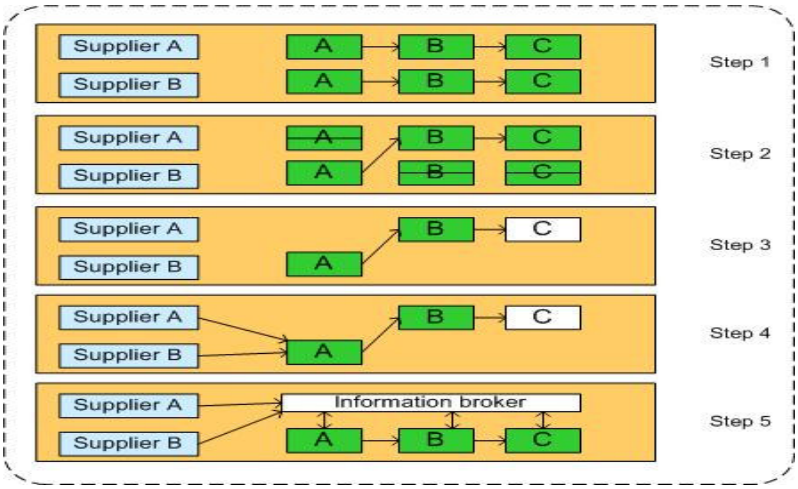

Figure 2.5: Form virtual organization [95]

Based on the description above, we can further summarize the basic characteristics of do business via a collaborative business approaches. The main objectives of such collaboration are to share resources, improve a business enterprise's working efficiency and provide better products or services. Furthermore, effective process collaboration provides business companies with more flexible means for the choice of the services offered by other independent service providers. For example, a business company is able to focus on its own important work, outsourcing some routine work to other partners in order to reduce the costs of production.

Figure 2.6 shows an example of car manufacturing business via a collaborative approach. This scenario involves four business partners: a manufacturer, two component suppliers and a shipper. Both supplier 1 and supplier 2 are supposed to provide car engines or components to the car manufacturer. With collaboration techniques, the car manufacturer may produce qualified cars whilst the cost of production may be reduced

by selecting more appropriate business partners. For example, if the quotes offered from supplier 1 are over the expected budget, the manufacturer may select supplier 2.

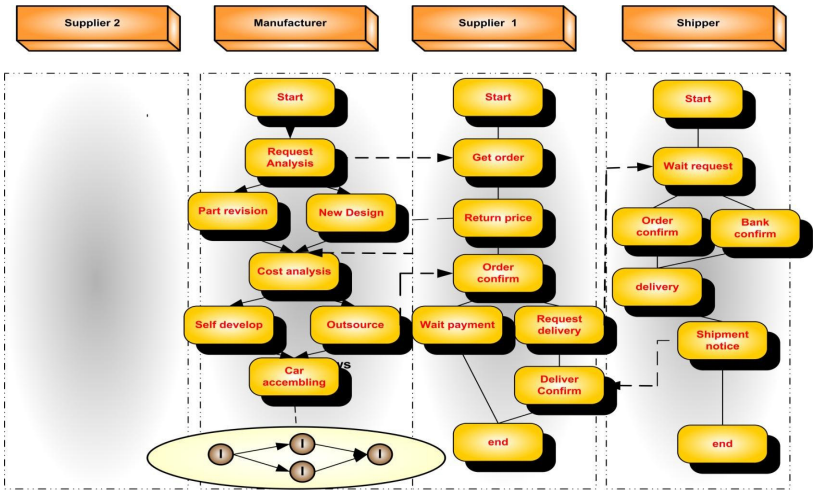

Figure 2.1: A scenario of distributed collaboration

In the following section, we will present some terminologies relevant to the business collaborations introduced in the above scenario.

2.1.3 Business Process

A business process in an enterprise refers to a set of tasks that describe how its products and services are designed, manufactured and marketed. These basic tasks consisting of a business process are performed collaboratively to realize a business objective, such as achieving a particular profit of production. Depending on the specific applications, a business process can belong to an intra-enterprise, but can also span over different organizations (inter-enterprise). For example, in the PC maker scenario of supply-chain management, the task of ordering motherboards might be conducted by a motherboard provider at a place A; the task of loaning process for CPU purchase may be carried out

by a CPU supplier at place B; and the claiming management process may be implemented by an insurance company at place C. Finally, these tasks in a business process could also be sub-processes that may be realized by combing various Web services, which correspond to a business transaction.

Business Process Design Approaches

Two basic strategies have been used to design business processes: the top-down approach and the bottom-up approach. The top-down process design approach refers to the procedure of analyzing requirements, and decomposing a business process into sub-processes or tasks, which can be easily handled and realized with individual Web services. More specifically, the top-down approach starts with a requirements analysis, which specifies what a business company expects to achieve with respect to the objectives of the company, such as whole performance of business process, reliability of business activities and security, and so on. After the decomposition of a business process, the dependency among tasks needs to be examined and determined. This includes the description of each task in the business process, such as functional features (e.g., input and output) and non-functional features (e.g., reliability, transaction), and the relationships between tasks in the process, including control flows and data flow among the tasks. Top-down design is a suitable approach when a business process is being designed from scratch.

Commonly, however, a number of processes or Web services may already exist. In this case, the design task involves integrating the existing services into one business process. The bottom-up approach is suitable for this type of situation. The starting point of

bottom-up design is the individual services, and final result of this approach would be a global process model. The advantage of this approach is to reduce the cost of developing business processes because of the reuse of existing resources. The downside is the lack of an integrated view of the entire system. Therefore, in real application in developing business processes, the two approaches may need to be applied to complement one another.

Web Service Discovery and Business Process

Before the execution of a business process, a set of Web services needs to be located to fulfill the tasks included in the business process. In this case, the assignment of tasks to appropriate services is referred to Web service discovery. In this situation, how to effectively select appropriate Web services for tasks is considerably challenging issue in distributed context because there may exist a number of solutions.

2.1.4 Workflows

Workflows refer to the automation of business processes, in whole or part. The Workflow Management Coalition(WfMC) [136] defines workflow as "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" [136]. In the following, we introduce some basic concepts about workflow.

Tasks and Collaboration

A workflow consists of a set of tasks. Each task can be a primitive task or a complex one. A primitive task represents a basic business activity unit that cannot be decomposed

into simpler tasks whilst, a complex task include a subset of tasks and the ordering relationships of their execution. The execution of a task in workflows is performed by an agent. Depending on the specific processes and applications, an agent can be a human, a program, a physical device or a Web service. If a task is assigned to people, the assignment of people to tasks in a business process is often performed via roles.

The tasks in a workflow need to coordinate each other for achieving a business objective. More precisely, the coordination of tasks involves how to schedule tasks available, that is, which tasks to select, what order of task to follow and what data dependency among task to regulate. This relationship among the tasks in the course of coordination is called workflow patterns, and can be further described via control flows and data flows. For example, in a sequence invocation of tasks, some tasks have to be scheduled one after the completion of the other. In service-oriented environment, a complex and long-running business process is generally performed in a highly distributed and heterogeneous environment. With workflow technology, the enterprise business activities can be structured as a set of tasks that are executed in a specified partial order.

Workflow Description and Workflow Enactment

Generally, a workflow includes two basic components: workflow description and workflow enactment. A Workflow description, also called workflow schema, formally models a business process and the relationships between tasks in the process, which include basic tasks, control flows and data flows. This basic workflow description may be based on a graph theory, where a node denotes a task and the directed links between

tasks reflect their dependency. A task could be corresponding different business activities such as transaction and sub-process. Control flows denote *the potential execution sequence* of various tasks. For example, a directed edge pointing from A to B, called control connectors, regulates the flow of control from task A to task B, which means that task B can only start after task A has completed. On the other hand, data flows reflect *real execution sequence* of various tasks. Based on specific business rules, real data determines which path (consisting of real execution sequence of tasks) should take. These control and data flows are normally called *business logic*, which is necessary to realize the desired processes.

Workflow Management System

A workflow management system (WfMS) provides an effective environment to support for both defining workflow and run, schedule and monitor its execution. Formally, a WfMS is a software system that completely defines, manages and executes workflow through the execution of workflow whose order of execution is driven by a computer representation of the workflow logic [136].

First, for the definition of workflows, tasks and the relationships among tasks in the workflows need to be modeled as related processes through using a modeling approach, such as *metamodel* [65]. The metamodel provides build-time functions and constructing units by which a business process is able to be translated from the real world into a formal, computer processable process metadata. What is more, workflow definition language, such as Web Services Flow Language (WSFL)[125] (e.g., BPEL4WS[135]) , scripting language like MQSeries Workflow, and together with some editors, can further

assist designers to model workflow processes. Second, for running and scheduling workflows, a workflow engine (a software service) will take as input the description of the workflow description, interpret the process definition, and control and manage the process instances.

2.1.5 Transaction Management

e-service transaction management is a comprehensive logical architecture that uses e-services to develop, deploy, discover, compose, and manage computing resources. Normally, the transaction management involves complex business processes that may include several transactions. The transactions often involve various activities that may invoke and span in multiple e-services such as payment processing, shipping, coordinating and managing marketing strategies, and so on. Over the years, organizations have been seeking closer collaboration, application reusability and cost cutting in the face of tougher competition. Therefore, an effective and efficient e-service transaction management has become a critical factor for the organizations'' business success.

Basically, the existing approaches, standards and transaction protocols are needed to consider more complex transactions, such as in the scenarios of reliable distributed collaboration and dynamic composition of e-services. First, the existing approaches lack semantic support for description, discovery and composition of Web services. Second, traditional database transactions demand to satisfy strict ACID properties (atomic, consistent, isolated, durable) [40], and extended transaction models must also comply

with the weaker specifications of the ACID properties. However, to e-services transaction management, the strict ACID properties may not be desirable.

Then, traditional transactions are effective in a homogenous and centralized database. In this case, the data in traditional transactions is raw (no meaning) and known (with fixed value). Nevertheless, in a distributed transaction management of service-oriented structure, Web services have long running behaviors [73], which may not be precisely known when they are invoked. In addition, traditional transaction models and analyses fail to include the performance aspects of e-services transactions (e.g. QoS, scalability, etc.)[5, 23, 74], nor do they consider composite business transactions. Imagining ordering goods on the Internet as an example, it needs long running business activities such as checking price, sending order and shipping package, and so on.

Therefore, to achieve the objectives of these complex business transactions, an efficient transaction management should understand the different services' syntactic and semantic descriptions, have efficient approaches and models for coordinating business interactions, and have proper frameworks for managing computing resources that may span different administrative domains.

2.2 Services-Oriented Architecture (SOA)

In this section, we will introduce the essential protocols and components that are related to constructing the CSTM framework.

2.2.1 Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is a XML based protocol that can be used to exchange structured information in a decentralized, distributed environment [133]. SOAP provides an XML format for sending messages, which is independent of programming language and computer platforms. In particular, SOAP works on existing network transport protocols, such as HTTP, SMTP, FTP, etc, rather than defining a new transport protocol. The transporting elements in SOAP are a SOAP message, which is an XML document with the body element and header element. The body element includes the transmitted documents whilst a header element can contain optional routing and security information. Both the header and body elements are themselves XML nodes. In addition to the basic message structure, the SOAP specification also defines a model that dictates who should process messages and how recipients should process SOAP messages. For business collaborative applications in distributed environment, SOAP can be used as the common communication protocol for constructing the CSTM framework.

2.2.2 Web Service Description Language (WSDL)

Web Service Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedural-oriented information [132]. With WSDL, a Web service can be described as a collection of network endpoints. The description consists of two main parts: the abstract definition of interfaces and the concrete implementations of network. In the abstract definition, interfaces and a set of operations are defined by *portType* element and *operation* element respectively. Besides, each operation may contain input/output messages that are defined by *message* element.

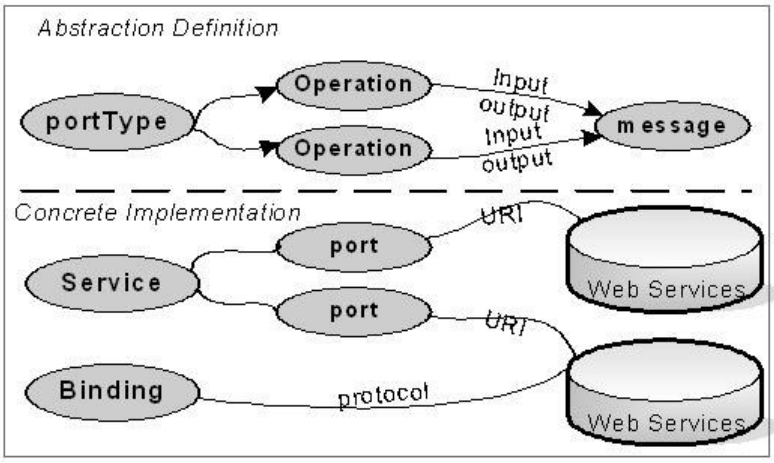

Figure 2.7: The specification of web service

On the other hand, the concrete implementations specify how the abstract interfaces are mapped to the specific bindings, which may include particular binding protocols like SOAP and network address. Similarly, a set of elements such as *service, port* and *binding* are used to define these deployment details. For business collaborative applications in distributed environment, WSDL can be considered as the common language that defines service interfaces.

The key advantages of this mechanism adopted in Web services lie in the separating the interface definition from the network implementation and making it possible to multiple deployments on the identical interface. Moreover, it would facilitate the reuse of the software in the Web service community. Figure 2.7 shows the specification of Web services and Figure 2.8 shows an example of WSDL file for a CargoShipping service.

```
<message    name="ListOfDeliveryDestination">
        <part name="destinations" type=
              " tns:Vector"/>
</message>
<message name="DeliveryDestinaiton">
        <part name="destinaiton" type=
              " xsd:String"/>
</message>
<message name="DeliveryPrice">
        <part name="price" type="tns:String"/>
</message>
<portType name="CargoShippingService_port">
        <operation name="getDestinationList">
            <output message="
                 ListOfDeliveryDestination"/>
        </operation>
        <operation name="getDeliveryPrice">
            <input message="
                 DeliveryDestinaiton"/>
            <output message="DeliveryPrice"/>
        </operation>
</portType>
```

Figure 2.8: An example of WSDL file for cargoshipping web service

2.2.3 Universal Description, Discovery and Integration (UDDI)

Universal Description, Discovery and Integration (UDDI) [134] is a service discovery specification that is proposed by an industry consortium led by IBM, Microsoft. With the UDDI, Web services are registered and described as core type of information: white pages with their contacting details, yellow pages containing their classification information based on standard taxonomies and green pages providing the specification of interface for Web services. The UDDI is also a platform-independent, open framework for describing services, discovering businesses, and integrating business services. It proposes a unified and systematic way to publish and discover Web services. At present, UDDI has become one of the dominating industrial techniques for Web services discovery. UDDI also allows syntactically search and category-based match Web services. In addition, a service requester can use the Inquiry API provided in UDDI for retrieving services via submitting instructions like *find_service().* For business

collaborative applications in distributed environment, UDDI can be considered as a service broker.

2.2.4 Business Process Execution Language for Web Services (BPEL4WS)

At present, BPEL4WS [135] is an industrial standard for describing Web services. In particular, it can serve not only as a platform for modeling an abstract process but as the implementation language for executable processes. In an abstract process, BPEL4WS describes a business protocol, specifying roles between different parties and their relationship without considering internal behaviors for parties. On the other hand, in an executable process, BPEL4WS specifies an execution order among a set of activities constituting the process. Currently, BPEL4WS engine is one of the main execution environments, which provides a full package of Web service processes, including development, deployment and execution.

2.2.5 Ontology

Ontology refers to a formal specification of a conceptualization. Based on a theory on existence, such a technique organizes a domain knowledge into the categories by virtue of objects' components and their semantic connectives, aiming at capturing the information on the structure and semantics of a domain. In other words, an ontology describes concepts and relations between domain concepts.

Ontology defines basic concepts and terminologies that are agreed by all participants in a community. In particular, service ontology specifies a domain like healthcare, a set of synonyms, which is used mainly to facilitate flexible search for the domain. In addition,

an ontology also describes service classes, which are used to define the property of the services. Because of its properties, ontology is used to discover Web services.

2.3 e-Service Transaction Management through Workflows

The approach of workflow-based composition is first achieved by definition of an abstract process of service model that includes a set of tasks and their data dependency, and then in runtime, each task is used to search the real atomic Web service to fulfill the task. Normally, a business process is a set of one or more interconnected activities to realize a business objective while a workflow is an automation of business process during which information or tasks are passed from one participant to another, according to a set of predefined rules. Thus, we can model workflow using a graph in which its nodes represent activities or tasks and its edges represent control and data flows.

2.3.1 eFlow

eFlow [22] is a workflow-based services composition platform that supports the specification, deployment and management of composite e-Services. In the eFlow, a composite service is modeled by a graph, which defines overall flow of services execution and data exchange among the services. In the graph of a composite model, a service node represents the invocation of an atomic service or a composite service while arcs in the graph denote the dependency among the nodes. In order to dynamically create process definitions for composite services, a customer can specify the services he/she needs by accessing a module called generic service node. Once the customer submits a

form, a new instance of the process is started. When a service node is invoked, a search recipe is executed to select a reference to a specific service.

However, most approaches of these process-based workflows are static and predefined. In other words, an abstract process model should be created before the composition planning starts. Obviously, this feature reflects that this kind of workflow method lacks flexibility and is not user-friendly. The main problems include:

- these kinds of workflow-based approaches have been mainly defined by users manually
- the degree of automation of producing composite service is low
- there is shortage of semantics

2.4 e-Service Collaboration Transaction Management through Web Services Composition

To tackle the problems mentioned above, semantic Web and AI technologies point the way to the future. Recently, academic researchers of the Web have been making unrelenting efforts towards automatic, semantic Web enabled e-services transaction management. The basic ideas and methods are to annotate Web services with semantic information, develop different mathematical models and apply various automatic algorithms. Some examples are summarized as following.

2.4.1 Quality-Driven Web Services Composition Transaction

Basically, the performance and properties of a Web service can be represented or evaluated by the Quality of Service (QoS) that includes Execution Price, Execution

Duration, Reliability, Availability and Reputation. As multiple Web Services may provide similar functionalities with different properties (e.g. different price), the QoS is used to be a standard for selecting proper candidate services and producing an optimizing plan for a composite service. An interesting approach that describes the composition process of e-workflows is put forward in [23]. The approach relies on the use of ontology to describe workflow tasks and Web services interfaces. In [23], service composition consists of two steps. First, a client creates a service template (ST) to indicate his/her intention, and then the ST will be employed later to find appropriate Web services. Second, the ST is sent to a discovery module that returns a set of service objects (SO). Consequently, the task of service composition can be completed by matching ST with SO in terms of syntactic similarity, operational similarity, and semantic similarity.

2.4.2 Ontology-Driven Web Services Composition Transaction

One of the methods to empower Web services with semantics is to use ontology. Ontology is a kind of knowledge representation describing a conceptualization of some domain. As ontology specifies a vocabulary that includes the key terms, services' semantic interconnections, and some rules of inference, it is now used to compose and manage Web services. One of the examples of using ontology can be found in [22]. In [22], Web services are first semantically described through an ontology-driven Web services composition platform, and then a composite service is generated. In addition, by using Web services ontological descriptions and relationships to other services, possible automatic compositions are obtained through checking semantic similarities between

interfaces of individual services. After that, these compositions are ranked and an optimum composition is selected.

Another example of composing e-services is to combine states with atomic process of Ontology Web Language for Services (OWL-S) [137]. In OWL-S, a state is viewed as a database instance, and an atomic process is similar to one of OWL-S. In fact, this approach regards service composition problem as mediator synthesis problem, that is, the approach results in building a mediator service that simulates the behavior of an objective Web service. Finally, other related works on automatic compositor of OWL-S services can be found in [39].

2.4.3 Rule-based Web Service Composition Transaction

A rule-based approach to compose Web services is applied in SWORD [94]. The SWORD is a toolset that allows service developers to quickly compose basic Web services to realize new composite Web services. In SWORD, a service is firstly represented by a rule that expresses that given certain inputs, the service is capable of producing particular outputs. A rule-based expert system is then used to automatically determine whether a desired composite service can be realized using existing services. If so, this derivation is used to construct a plan by which SWORD will instantiate the composite service when the plan is executed. In particular, SWORD does not deploy the emerging service-description standards such as WSDL and DAML-S, instead, it uses an Entity-Relation(ER) model to specify the inputs and the outputs of Web services. Therefore, the ER model provides a basis on which reasoning can be carried out on the entity and attribute information.

While being promising in their theories and applications, no single approach can provide a solution that efficiently and successfully supports all business transaction process. For example, an expert system is needed in [94] to realize e-services transaction management, but such an expert system is not always available and not always practical in service-oriented computing.

2.5 Research Problems and Challenges

Due to its complexity, the problems of the semantics and the reliable collaboration in Web-based context have attracted much attention in service-oriented computing. The complexity can be mainly summarized as following:

- As Web services are usually created by different providers that employ different phrases and models for presenting and describing services' characteristics and capabilities, it is difficult for service consumers to correctly understand the relevant semantic information hidden behind the Web service descriptions.
- As the number of available Web services on the Web is ever increased, it is very difficult or beyond the human capability to discover and compose Web services manually.
- As the Web services can be created and updated dynamically, it is very difficult for the e-service transaction systems to handle the exception raised at the execution of business processes and make a correct decision on the fly.
- Traditional transaction models are mainly based on the database transaction models such as ACID properties, which require that a transaction can not share its data with other concurrently running transactions until it is completed. However this strict

atomicity and isolation policy is inappropriate for enabled e-service transactions management in a service-oriented application.

In this thesis, we address the challenges mentioned earlier by presenting an access control framework and models for supporting semantic, reliable and secure collaboration. The proposed approaches depend on workflow technologies by integrating control flows and data flows models, and they employ a relaxed transaction concept for fault tolerance, in order to maintain the process consistencies, as well as, data consistencies even in presence of failures.

To achieve these objectives, we first locate relevant Web services by semantic approach. The key idea of our approach is to indirectly associate the intention of users to the advertisements in Web services by applying Probabilistic Latent Semantics Analysis (PLSA). Then, we model the internal activities within an institution by employing workflow technology whilst the individual workflow models of participating institutions can be mapped to views, which are further described by Web services. Based on the view model, a two layered access control architecture is proposed to protect computing resources. Finally, the atomicity concept is relaxed by integrating transactions and exception handling models in order to ensure the reliable collaboration on the Web.

2.6 Summary

In this chapter, we have examined the traditional approaches to doing business, introduced new collaborative approaches and discussed related research issues. Traditional business approaches are mainly based on a central decision making control mechanism by which trading partners are working together with closely coupled. As a

result, these enterprises lack the flexible ability to meet a specific business objective and respond the market change rapidly because it is difficult for them to provide relatively complex, add-value and quickly delivered products and services to markets. The new approaches are based on process-oriented collaborative strategy [121, 95, 64]. The key idea is that the collaboration among trading partners is based on sharing business logic of whole collaborative process. Under this strategy, each partner can focus on its core businesses, flexibly select trading partners and join the collaborative process according to the common objective.

Some questions often raised are the semantics, security and reliability in a distributed collaboration. These issues require the use of analysis and model, a hybrid semantics extracting approach and transaction-based approach, which can solve effective collaboration among trading partners in a distributed environment. For a detailed discuss, see the following chapters. In the next chapter we will first present basic models for semantic, secure and reliable transaction management applied to this thesis.

.

3. Chapter 3

Basic Models for e-Service Transaction Management

3.1 Introduction

In the previous Chapter, we introduce a scenario of doing business by using new collaborative approaches, which involves different actors, such as manufacturers, suppliers, banks, transportation, warehouses and customers, and so on. In a distributed environment, the collaboration among these trading actors may show the complex behaviors in the setting of the structure for doing actual business transaction and the exchange of goods, services or money. Once this initial observation is done, how do we derive some essential features from these complex collaborative phenomena, so that various aspects of effective collaboration like semantics, security and reliability can be examined?

The process used to address this issue is known as models. This is typically done by identifying main actors in a collaborative system, describing the interactions among them and analyzing the main characteristics and behaviors they show in collaboration.

In this Chapter, we will discuss main elements in a distributed collaboration and analyse the relationship among the elements. We start with introducing basic requirements in a

distributed environment, including security requirements, reliability requirements and semantics requirements. We then present some basic terminologies used in security model and transaction model. Finally, we describe a workflow model

This chapter is organized as follows. In Section 3.2 we first present basic modelling concept. In Section 3.3, we identify the basic requirements about security, reliability and transaction in a distributed application. Section 3.4 gives a role based access control model and a distributed workflow model is discussed in Section 3.5. Finally, Section 3.5 reports the summary of this chapter.

3.2 Introduction to Modeling for Distributed Collaboration

Business collaborative applications often involve long-running computations, loosely coupled systems and components, which are linked together to form a cohesive whole. In particular, the process-based collaboration with Web services is emerging as a promising approach to automate business process across organizational boundaries. With this approach, individual Web services are federated into a cohesive whole whose business logic is expressed as a process model. Furthermore, workflow management systems (WFMSs) can be employed to automate business process by choreographing its component services.

However, business collaboration across independent organizational boundaries and their systems is complex for several reasons. First, Web environments are highly dynamic. A partner joining collaboration with Web services may leave or join the collaboration again at any time. Second, organizations are changing constantly by restructuring their business process or forming a virtual organization alliance for introducing new products

or seeking new business opportunities, in order to survive in the increasing competitive pressure of the globalization of economies.

Thus, there is a need for requirement analysis, models and abstraction in order to help us understand what attributes of the key components involved in a distributed collaboration system. In this chapter, we describe basic security, reliability and transaction requirements and models that are used in this thesis.

The aim of developing models is to provide a basic framework in which the various aspects of a distributed collaboration such as structure, reliability and fault tolerance can be examined. A model is an abstract description of the important features of a system under study. The feature description often includes the attributes of the system under study and a set of rules, which are used to demonstrate how these attributes interact with one another. Instead of listing all characteristics related to a system in a model, an effective model should consider only the essential attributes in order to help designers to understand the systems' behaviors, ignoring some minor aspect of the system.

3.3 Basic Requirements for Distributed Collaboration

In this section, we consider basic requirements for distributed collaboration. The design of a distributed collaborative application, particularly in a loosely coupled collaborative environment, should not only take into account functional requirements needed for the distributed applications, many non-functional yet important aspects, such as security and reliability, scalability and service quality, etc, are also needed to be handled properly. Based on the fundamental properties and requirements, we are able to present models to make generalizations concerning what is possible or what is impossible.

Normally, requirements are an informal description of what a collaborative system is supposed to achieve. The description is really indicating what a system can do with it, rather than how the system exactly do with it.

3.3.1 Security Requirements

Compared to traditional collaborative applications, modern distributed collaboration applications put forward new security requirements. These requirements include: decentralized security management, capability-based access control and multiple domain access control interaction.

Decentralized Security Management. Distributed business applications often involve coordinating the flow of processes and information across several domains and linking their support and information systems together into a cohesive whole. In this case, each participant joining a collaborative application may have domain-specific security policies and requirements that are confidential. Thus, each domain should have full autonomy of specification, management and enforcement of its security policies. This is contrast to the situation in traditional collaboration applications, where a trusted centralized coordination strategy is needed to manage collaborative applications execution. Moreover, in decentralized security management, there is no existent any prior knowledge about the security policies governing the collaborative partners because the collaborations are built dynamically. As a result, in some unpredicted execution environments, and business partners are not expected to reveal their security policies to other business partners.

Capability-Based Access Control. As business integrations in distributed context take place in a loosely coupling manner, they can interoperate more freely across the Internet. In addition, the collaboration in distributed context is established in a highly dynamic fashion and on-demand basis, and interacting users and Web service providers may even know little about each other. In this situation, access control methods based on the identity of each user is unsuitable for dynamic distributed collaboration. For example, accessing scalability based on the identity control will be decreased when the number of users and services increase, especially when the population of users and services is highly dynamic. Therefore, capability-based access control, rather than a partner's identity, is of importance [60].

3.3.2 Reliability Requirements

The importance of service-oriented computing for collaboration in business applications and software engineering continues to grow. The growth however, is impeded by the increasing complexity of distributed applications and the lack of reliability and flexibility of applications that use Web services. Here, reliability refers to the *continuity of the service* delivered by a system. In other words, a system could successfully complete its tasks in the face of failures.

A challenge to overcome in this collaborative environment is that efficient fault tolerant mechanisms are needed for ensuring reliable execution of business processes. For example, a basic requirement for reliable collaboration in distributed scientific experiments is that the experiments should be allowed to proceed without violating process specifications and data consistencies even when the failures occur for varying

reasons [4, 45, 41]. This requirement is particularly applicable to scientific workflows because the experiment may consist of hundreds of separate computing steps and involve lots of data objects, whilst the computing steps may fail. In such cases, giving up the overall experiment would be too expensive [4, 29].

In a distributed system, both collaborative processes and communication connections may fail. In this case, real business collaborative behaviors may depart from the behaviours regulated in a specification, which is considered to be correct or desirable behaviours. One key requirement for a distributed collaboration system is to ensure that the outcome reached by each partner in the collaboration maintains consistency, and keep consistency for the collaboration as a whole during the long running business process in the presence of failures. More precisely, the following basic reliable requirements should be met:

- **Hybrid Fault Tolerance Mechanism**. Fault tolerant approaches have been employed to ensure reliable execution of distributed applications. The objective of fault tolerance is to prevent the faults from leading to system failure so that the system is able complete its tasks assigned. There is a long history of efforts to make distributed applications reliable. The two fundamental approaches for constructing a reliable system are fault tolerance and a transaction based approach. Fault tolerance refers to a system design approach which recognizes that faults will occur; and it tries to build mechanisms into the system so that the faults can be detected and removed, or compensated for before they can result in a system failure [45, 107]. One of the basic techniques in implementing fault tolerance is to utilize error recovery. The goal of error recovery is to transform the current erroneous system state into a well-

defined and error-free state, from which normal system operation can continue. Specifically, there exist two basic ways to deal with the recovery: forward error recovery and backward error recovery.

- **Minimum of Failure Handling Cost.** A complex business collaborative application may integrate many different processes across organizations and link their support and information systems together into a cohesive whole. This requirement is particularly applicable to scientific workflows because the experiment may consist of hundreds of separate computing steps and involve lots of data objects, whilst the computing steps may fail. In the traditional approaches to dealing with failures, long-running computing steps can be structured to form a so-called a global transaction. If any of the computing steps fails, the whole application will be aborted and all effects of implemented components constituting the application will be undone. This would be too expensive. Thus, one of basic requirements is the lower cost of error recovery in distributed collaboration.

The proposed approaches are based on the combination of atomicity sphere and exception handling. The concept of sphere was originally used to refer to the sphere of control in the traditional database [40, 28]. A sphere of control logically defines the boundaries around a collection of operations performed on resources. As a unit of work composing of set of operations, a sphere is atomic if all its composed operations are committed or aborted unilaterally. This property can be used to create a fault-handling mechanism for reducing the cost of recovering processes in case of the failures, and for ensuring data consistency at various levels of granularity. For example, when a sphere included in a process is found to be in error, recovery can be made by undoing or

compensating for the parts of tasks included in the sphere, rather than undoing the whole process.

3.3.3 Transactional Requirements

To support the reliable collaborations in a distributed environment, effective transaction mechanisms should satisfy different transaction requirements needed in different contexts. These transaction requirements are mainly reflected in the collaborating participants in various application scenarios. For example, orchestrating a computational step A and a computational step B may be considered to be reasonable, but combining the step A or step C without step B might be not acceptable.

A transaction in a distributed environment is characterized by some distributed features: long-running, heterogeneous, and loosely coupled [73, 107]. Firstly, long-running computational tasks can be executed over a long period of time duration, so that it is impractical to lock all data used in a computing process for extended period of time. Next, heterogeneous features may involve multiple participants from various organisations whose scientific computing processes may run independently. These organisations may have different transaction models developed, managed, and run independently. In addition, loosely coupling indicates that the collaborating relationships between partners are established in a highly dynamic fashion and in an on-demand basis. From these characteristics, it is clear that the overall transactional behaviours associated with a transaction depend on the transactional capabilities and behaviours of individual computing processes. Therefore, an effective transaction model, suitable for a distributed environment, should support different transactional semantics in the same model [45].

To sustain such transactional semantics, we integrate transaction properties into the task model and the Web service model by extending the approaches proposed in [45, 5, 81, 107]. The main transactional properties include retriable, compensatable and pivot properties.

- **Retriable Property:** This property indicates that an entity (e.g., a task) with this property is allowed to repeat itself for completing its work successfully after a finite number of tries.
- **Compensatable Property:** This property implies that an entity with this property is allowed to undo the effect. In addition, the result produced by the entity could be rolled back with transaction.
- **Pivot Property:** This property regulates that an entity with this property cannot repeat itself or to undo the job because the cost of these behaviours may be expensive.

3.4 Role Based Access Control

As described in the previous section, a distributed collaboration consists of different actors like manufacturer, suppliers and customers, etc. In this section, we will first present some terms involved in the design of a secure systems, and then present a role-based access control model.

- An object is a protected entity such as data, files or a method in a Web service. An object is often subject to a domain, and is intended to be accessed by different users (subjects). Normally, an object may have a set of rules to evaluate whether or not this access is allowed.

- A subject is an active entity that intends to access an object by performing an operation on the object. A subject may be a user, a program or a process.
- Access rights refer to if a subject is allowed to access an object. If the access is permitted, this subject is called an authorized subject.
- A role represents an abstract job function.

A traditional approach to denote access rights is to use an access matrix, where the elements in the matrix denote whether a subject is permitted to perform an operation on an object. Another access control policy that is used to collectively determine whether a subject is allowed access to an object is the role-based access control [2]. We summarise this policy as follow:

U: users; R: roles; P permissions

$UA \subseteq U \times R$: user assignment;

$PA \subseteq R \times P$: permission assignment;

$R \to 2^P$: map role r to a set of permissions

$\text{Permission}(r_i) - \{p \in P \mid (\exists r \leq r_i)[(p, r) \in PA]\}$

3.5 A Distributed Workflow Model

In this section, we present basic models based on the fundamental requirements and properties described in the previous section. Theses basic models will provide a design basis on which more complex business processes and collaborative contexts could be built. In addition, this model allows one to concentrate on the reliable collaboration. In

the following, we start with briefly introducing a number of terms, and then present the detailed descriptions of terms.

- A process consists of a collection of tasks that can be described by a graph theory.
- A task is an atomic work item, but can also be a composite task.
- A task can be implemented by different entities, such as humans, Web services, or a program. Such entity is called an actor.
- A role refers to a job function, assigned to authorised persons.

3.5.1 Basic Task Models

Tasks are main components for a workflow and display a variety of characteristics. A task first is the unit of activity within a workflow. A unit of activity could be an atomic work item, with the implication that any further internal structure of the item cannot be discerned, or is not of importance. On the other hand, a compound task could contain its components. A task interacts with other task through context it exists in, such as enabling control flows and data flows between tasks. Meanwhile, the availability of an input incoming a task may trigger the execution of the task if the input satisfies certained constraint conditions.

Second, if a task is started by a human, the task is called a manual task t_i^m. In a distributed collaboration, there are a number of scenarios where business activities involve human endeavor. For example, in the processing of an insurance claim, a client first fills out an application form, followed by the form checking. A staff may then exam the filled form to make sure its completeness. After that, the form may be verified and approved by an auditor or through invoking other related processes.

Instead of directly assigning a task to a human for implementing, a task is more often assigned to role. A role describes an entity's ability or function. For an actor representing a human, a role reflects the actor's skills to perform some job function regulated in an organization, that is, it represents the access rights that could be assigned the actor. However, if an actor refers to a program, machine or a Web service, a role reflects this program's computing capabilities. In distributed workflow-based applications, a role often refers to a group of tasks. Thus, a human actor can only execute the assigned task activities if it is given related role associated with that task.

According to the above description, we summarize the basic features of tasks as follows:

- Types of tasks. A task may be an atomic task or a composite task. An atomic task cannot be further broken down into smaller ones while a composite task may contain some other tasks or sub-processes.
- Execution of tasks. Tasks are performed by agents (e.g., humans, or computer programs) involved in a process enactment by providing the tasks with designed input data.
- States of tasks. The implementation of tasks can result in the changes of these tasks' states. The main states for a task may include initial, completed, failed, aborted, cancelled and terminated states. For example, a task enters into the terminated state when the work performed has stopped and related exit conditions have not evaluated.

In the following paragraph, we first formulate the definition of a basic task.

Definition 3.1 (Basic task model) A task T_i is a tuple $T_i = (n, S, A, T_{input}, T_{output}, \phi)$, where

. n is the name of the task T_i,

. S = {*initial, completed, failed, aborted, canceled, terminated }* is the set of task T_i ' possible states,

. $A = \{compensatable, retriable, pivot\}$ is the set of the transactional properties of the task T_i,

. T_{input} and T_{output} are input and output types for the execution of the task T_i , and $\phi \subseteq (T_{input} \times T_{output})$ is a transition relation on the T_{input} and T_{output} .

◊

Based on Definition 3.1, we associate transaction properties with tasks to distinguish different tasks in a computing process.

Task and Transaction

Definition 3.2 (Transactional Task) A task with transactional properties defined in the Section 3.3.3 is called a transactional task. In this case, a task is implemented as a transaction.

◊

A task *t* is compensable if its execution effect can be reversed by another task t^c . In this situation, task t^c is called a compensating task whilst its associated task *t* is called a compensable task. For example, a task that reserves hotel is compensable because the room booked can be cancelled by its associated compensating task *cancel*.

Definition 3.3 (Compensatable task T_i^c) Let T^* be the set of all tasks defined by a process. A task $T_i \in T^*$ is compensatable if there exists a task $T_i^c \in T^*$ such that T_i^c can undo the effect produced by T_i, that is, task T_i and T_i^c form a pair $p = (T_i, T_i^c)$, and p satisfies the following properties:

(1) Let $P_T = \{T_i \mid \exists (T_i\ T_i^c) \in T^*\}$ be the set of tasks in T^*, then $\exists T_i \in P_T$ such that $\varphi(T_i^c) \bullet output \Rightarrow \varphi(T_i) \bullet output = null$;

(2) For each $T_i \in P_T$, $\phi(T_{i-1} \times T_i \times T_i^c \times T_{i+1}) \bullet output = \phi(T_{i-1} \times T_{i+1}) \bullet output$.

Thus, task T_i^c is called the compensating task of T_i.

◊

The property (1) and property (2), defined in Definition 3.3, indicate that the computing work performed by compensatable task T_i can be undone by its compensating task T_i^c.

Accordingly, we can define other kinds of tasks such as retriable task T^r and pivot task T^p.

However, in some situations, some tasks may be neither retriable nor compensable for various reasons such as because of violating business policy or resulting in high cost. Such tasks are often referred to as pivot tasks. For example, online shopping task for auctioning a computer is such an example. Due to business policies, a delivered computer is non-refundable and not retriable.

3.5.2 Data Flow Model

Different applications naturally have different kinds of dependability requirements.

Basically, the connection between two tasks represents their dependencies, which imply some constraints imposed on the occurrence of the two tasks. More precisely, such dependencies can arise from data flows.

Data dependencies describe the dependencies of tasks, which regulate how data produced by one task can be routed to multiple downstream tasks. This dependency can also be represented as a graph, with nodes denoting tasks (computational steps or processes) and edges representing the flow of data. Based on the description presented in [65], each task $T \in \boldsymbol{T}$ has associated input and output containers, denoted as $in(T)$ and $out(T)$ respectively. If the input container $in(T_2)$ of task T_2 receives (consumes) the data that are produced from the output container $out(T_1)$ of another task T_1, we say that T_2 depends on T_1, denoted as $d(T_1, T_2)$. Formally, for a given workflow model *W*, let set T denote the set of all tasks, then all data dependencies among two tasks $T_1 \in T$ and $T_2 \in T$ can be represented as a connector map which has been adapted from [65]:

$$\Delta : T \times T \rightarrow \bigcup_{T_1 \in T, T_2 \in T} \wp(out(T_1) \times in(T_2))$$

Where, $\wp(T_1)$ denotes the powerset of T_1 and the following conditions should be satisfied:

(1) $\Delta(T_1, T_2) \in \wp(out(T_1) \times in(T_2))$,

(2) $\Delta(T_1, T_2) \neq \phi \Rightarrow T_2$ is reachable from T_1.

The condition (2) indicates that there exists a control connector $k(T_1, T_2)$ between two tasks T_1 and T_2. In what follows this paragraph, we consider control dependency.

3.5.3 Control Flow Model

Control dependencies describe the partial order of execution of the tasks inside a workflow. For example, in a sequence dependency of tasks, control linking from task *A* to task *B* means that task *B* could be implemented after task *A* has been completed successfully, denoted as $q = (A, B, c)$, where *c* denotes transition condition. In this thesis, we use a projection map to represent the dependencies between the tasks.

Let set *T* denote the set of all tasks within a workflow model *W*, and set *Q* consist of all possible sequences of tasks associated with the model *W,* thus, a control link is a triple $(t_1, t_2, c) \in Q \subseteq T \times T \times C$, where *C* denotes transition conditions, then

. the set of all control connectors pointing to task $t_s \in T$ can be denoted by $\triangleright t_s := \pi_3(\{q \in Q \mid \pi_2(q) = t_s\})$.

For example, in Fig. 3.1(a), $q_1 = (T_1, T_2, c) \in T \times T \times C$ and $\pi_2(q_1) = T_2$.

. The set of all control connectors leaving to task $t_s \in T$ can be denoted by $\triangleleft t_s := \pi_3(\{q \in Q \mid \pi_1(q) = t_s\})$.

Here π denotes the projection map between Cartesian products.

Based on the above description, we can represent the dependencies between the tasks in terms of BNF syntax:

$$\text{DT} ::= T_1 \rightarrow T_2 \mid T_1 \| T_2 \mid T_1 \oplus T_2 \ ,$$

where $T_1 \rightarrow T_2, T_1 \| T_2$ and $T_1 \oplus T_2$ denote sequential, parallel and choice relationship operators respectively.

Definition 3.4 (Sequential relation $T_i \rightarrow T_j$) Two tasks T_i and T_j are defined as the sequence pattern $T_i \rightarrow T_j$ if and only if they satisfy the following conditions:

1) $\pi_3(\{q \in Q | \pi_1(q) = T_i\}) = \pi_3(\{q \in Q | \pi_2(q) = T_j\})$,

2) $\forall (T_i, T_j) \in T : i \neq j \Rightarrow T_i \cap T_j = \phi$ for example, $T_1 \neq T_2$.

◊

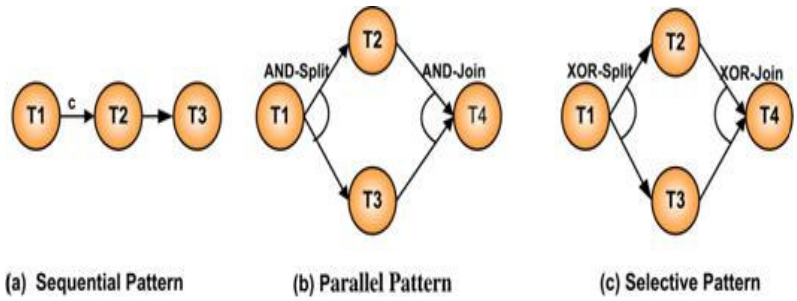

Figure 3.1: Control dependency

The condition (1) indicates that the outgoing edge of task T_i is the ingoing edge of task T_j. The condition (2) expresses that the two tasks are distinct. The sequence operator $T_i \rightarrow T_j$ indicates that task T_j can only be started after task T_i completes its work. Accordingly, we can define other dependency between tasks such as parallel relation $T_2 \parallel T_3$ and choice relation $T_2 \oplus T3$.

3.5.4 Transactional Web Services Model

According to Web Services Description Language (WSDL), a Web services can be described as an abstract interface that consists of collections of operations. The interaction of a Web service with its environment (or other services) takes place through

the operations by receiving input message and likely responding with output message. Based on the description of the interaction, we model a Web service by adopting a states/transitions approach.

Definition 3.5 (Web service) A Web service *WS* is defined as a transition system with the form of a tuple $WS = (\Sigma, S, R, s_0, \delta, F)$, where Σ is a nonempty finite set of elements including input elements and output elements; *S* is a nonempty finite set of elements called states; *R* is the set of transition preconditions; $s_0 \in S$ is an initial state; δ is a state transition partial function regulated by $\delta: S \times \Sigma \times P \rightarrow S$ and *F* is a set of final states given as a subset of $S: F \subseteq S$.

◊

Definition 3.6 (Transactional Web service) A Web service *WS* is defined as a transactional service if it satisfies the transactional requirements defined in Section 3.3.

◊

According to the transactional requirements defined in Section 3.3, we use similar method introduced in Section 3.4 to distinguish between compensable, retriable and pivot services.

3.5.5 A Workflow Model

A workflow model (or specification) defines its basic components and regulates how its components interact with one another. More precisely, the main components of consisting a workflow normally include tasks (or actors), data flows and control flows. As mentioned before, a task represents a basic unit of work (step) that performs a specific function within a workflow. For example, a task may merely implement a

simple mathematics computation, but can also complete other complicated function such as coordinating the execution of other processes. In addition, the control flows express the potential sequence by which different tasks can be implemented whilst the data flows describe the data dependency between tasks.

We define a workflow by using the flow model [72, 64, 65, 125], which describes the connection between tasks, the executing order of tasks, the data exchanging between tasks and the decision point. The flow models play a role in the collaboration between partners in the context of Web services. For example, the tasks in a scientific workflow can be implemented as operations of Web services. Thus, a Web service provider can publish a scientific workflow (flow model) as a Web service that can be invoked by other collaborative partners. In this case, the collaborative partners play the role of service requestors. Formally, a flow model can be represented as a directed graph in which nodes correspond to tasks and edges indicate possible control dependencies or data dependencies between tasks. Let *G* be a graph, with a set of nodes *V* and with a set of edges *E*; and *V(G)* and *E(G)* represent the set of all nodes and the set of all edges in G respectively. Thus, a scientific workflow is a simple graph $G=(V,E,f)$, where $b \in V(G)$ and $e \in V(G)$ denote the beginning node and the end node in *G;* the edges can be expressed as $E \in V \times V$, and *f* is a function mapping each node onto unique label.

3.6 Summary

In this chapter, we discussed many of the important aspects that can be employed in the design of an e-service transaction management system. We started off by discussing security, reliability and transaction requirements needed in an effective e-service

collaborative context. Based on the requirements, we presented related models that allow us to concern primarily with the important feature of a collaborative system under study. The objective of the abstraction is to provide a basic framework through which the various aspects of collaboration in a distributed context like semantics, security and reliability can be examined.

Next, we examined the role-based access control model by which trading partners coordinate the efforts of their collaboration based on the roles.

Finally, we described a distributed workflow model, including a data flows model and a control flows model. In the following chapters, we will apply these models to create an effective collaboration across the various organizations' boundaries.

4. Chapter 4

Collaboration through Web services Discovery and Composition Based on Semantics

Web service technology is an important enabler of effective collaborations across organization boundaries. Much of the current interest in business collaborations is motivated by employing Web services to reduce transaction costs and to enhance the efficient collaboration among trading partners [95, 121, 64]. In addition, promising characteristics entailed by Web services, such as flexible connections with unknown transaction partners, platform independence and easy composition of other Web services, have also increased the interest in a distributed collaboration among trading partners.

In the Chapter 3, we have described the design of basic models applied to distributed collaborative transaction managements, including different actors such as manufacturer, supplier and shipper, task models, process model and workflow model, and so on. In the context of Web services, these actors are represented as Web services and the business process models can also be realized with Web services. As the first step of realizing a collaborative process, Web services needs to be discovered to complete an assigned task in the process. Therefore, our discussion of effective collaborative transaction management will include discovery of Web services.

In addition, a distributed collaboration typically spans different organizations, communicates with various trading partners, and coordinates the flow of processes and information among organizations by linking their support and information systems together into a cohesive whole. Thus, after discovering proper Web services, in this chapter we will also discuss Web service composition.

The effective discovery and composition of Web services based on semantics typically poses a significant challenge. This is because Web service infrastructure itself can not fully understand the business context of an e-business collaborative application [121], although the infrastructure provides a good foundation to build a flexible collaborative business information system. For example, current Web service discovery approaches are based on syntactic information required to access the Web services that are described in WSDL. However, these approaches fail to contemplate the semantic concepts hidden behind the words in a query and the descriptions in Web services, which motivates this research. Specifically, in this chapter we will discuss the following questions:

- How to effective discover Web services based on semantics?
- How to compose Web services based on the semantics?

This chapter is organized as follows. In Section 4.2, we first summarise some Web service discovery approaches. In Section 4.3, we present Web services discovery based on Singular Value Decomposition (SVD). Section 4.4 employs a probabilistic semantic approach to extract semantic concepts hidden in services descriptions. Section 4.5 deals with Web service composition with ontology. Finally, we review related work in Section 4.6 and present summary of this chapter in Section 4.7.

4.1 Introduction

The distributed collaboration often needs to integrate different business applications. This requires coordinating the flow of processes and information among organizations and linking their support and information systems together into a cohesive whole. In this situation, the collaborative business logic can be specified in terms of a workflow in the design time, thus the collaboration in distributed context can be modeled as an ordered collection of tasks.

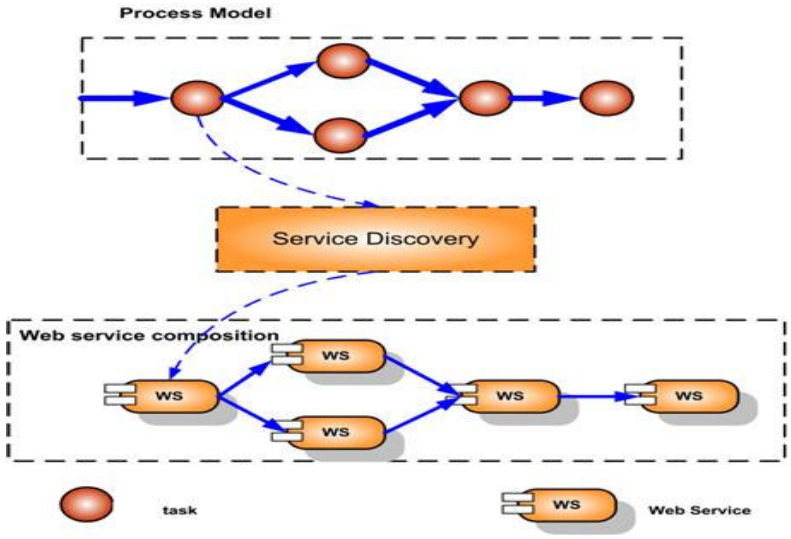

Figure 4.1: Service discovery and composition

As described in the previous chapter, each task represents a basic unit of work that realizes parts of whole collaborations' objective. These individual tasks can be deployed on individual collaborative partners, each owning its own workflow engine. In service-oriented architecture, a Web service is discovered and assigned to a task in order to complete the task. On the other hand, during the runtime, a set of Web services might be found for individual tasks in a local repository or located across organizations

boundaries. When individual Web service completes its application invocation on behalf of the tasks, the whole collaboration objective is realized. Thus reliability collaboration in a distributed environment is a process of discovering and matching the appropriate Web services, and composing them at running time. This process is illustrated in Figure 4.1.

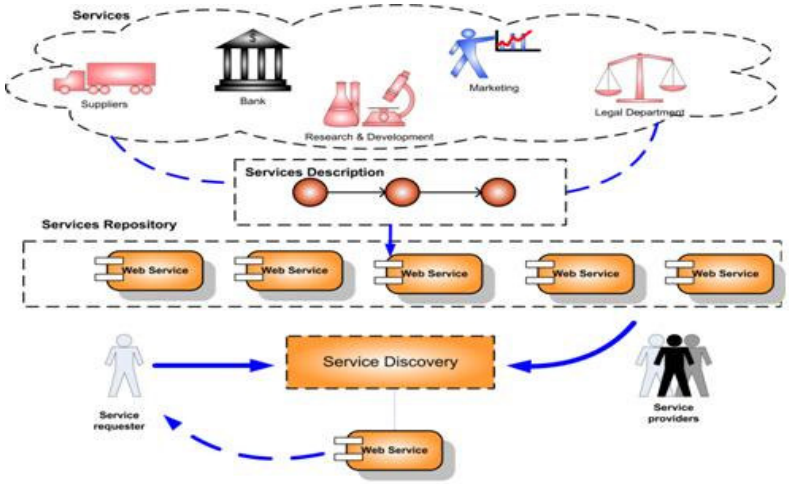

Figure 4.2: Web service discovery

Web service discovery is one of key open issues in service-oriented computing. It includes locating relevant services published on the Internet, matching the requirements of consumers against the services returned, and recommending desired services to the consumers. Efficient discovering Web services is a challenging issue for several reasons. First, as Web services are starting to get deployed on the Web, a large number of Web services would offer similar functionalities. As a result, it is difficult for service consumers to select the desired ones to complete their tasks. Second, current discovery mechanism like UDDI already enable much of the Web service discovery process, however they concentrate largely on syntactic approaches such as by using keyword-

based search and category-based matching, which is inefficient and time-consuming. In this chapter, we describe a service discovery framework, where mathematical approaches such as Probabilistic Latent Semantic Analysis approach (PLSA) are used to extract the semantic concepts hidden in service descriptions.

4.2 Web Service Discovery Approaches

Web services have emerged as one of distributed computing technologies and sparked a new round of researches. Web services are actually self-contained, self-describing and modular applications. Because Web services adopt open standard interfaces and protocols, they are increasingly used to integrate and build business applications on the Internet. With Web services, business organizations can build their applications by outsourcing some other services published on the Internet. As an ever-increasing number of Web services published and deployed on the Internet, it is critical for service users to discover desired services that match their requirements.

Web service discovery is the process of locating effective approaches to find, match and access the Web services published in a local repository or located across the Internet. To effectively discover and match Web services, it is a necessary to establish some kinds of the correlations between a user and potential services available, which can be achieved through two steps. On the client side, a service user can express his/her requirements described in the form of nature language and then use a service search engine to interact with a set of potential Web services. On the side of services providers, on the other hand, they advertise services' capabilities through some descriptions such as the Web services' names, the operations' descriptions and the operations' names described by WSDL. In

this situation, they also assume that clients would agree on the words used to describe the Web services. However, the problem of how to deal with the agreement and how to associate the users' requirements to the advertisements of Web services would have a critical impact on discovering Web services. Therefore, locating desired services might be difficult.

Web service discovery has been widely used in business collaboration and integration. In a distributed collaboration environment, whole business objectives can be realized by firstly locating basic Web services deployed on the Web. These services discovered are then integrated into new composite services that can provide more sophisticated functionality and create add-on values. For example, a travel booking composite service can provide a high-level travel transaction management service that uses individual car rental, hotel reservation and payment Web services as components. It is also likely that the interaction and collaboration among Web services that make up the composite Web service is regulated via a business process model. Thus the dependency between tasks in the process model can be further described by control flows and data flows. In this situation, it is necessary to discover appropriated Web services and assign them to a task in order to complete the task.

In addition, participants who request Web services through the Web in distributed collaboration need to apply appropriated approaches to access provided Web services that meet their requirements. Therefore, efficient collaboration in distributed context is achieved by automatic Web services discovery and binding Web services in order to satisfy their requirements.

Current Discovery Approaches

A commonly used approach on discovering and matching Web services is to *directly* associate a user's requirements to the advertisements of Web services (solid line 1 shown in Fig. 4.3). For example, a user types keywords in a Web service search engine [126, 127, 128, 129] to look for the desired Web services. If the typed words are included in or identical to the descriptions of some services, the return services might be relevant to his/her need. Nevertheless, this approach based on the term frequency analysis is insufficient in the majority of cases. For one thing, syntactical different words may have similar semantics (synonyms), which results in low recall. For another, semantically different concepts could possess the identical representation (homonyms), thus leading to low precision. In short, this discovery mechanism fails to contemplate the semantic concepts hidden behind the words in a query and the descriptions in Web services.

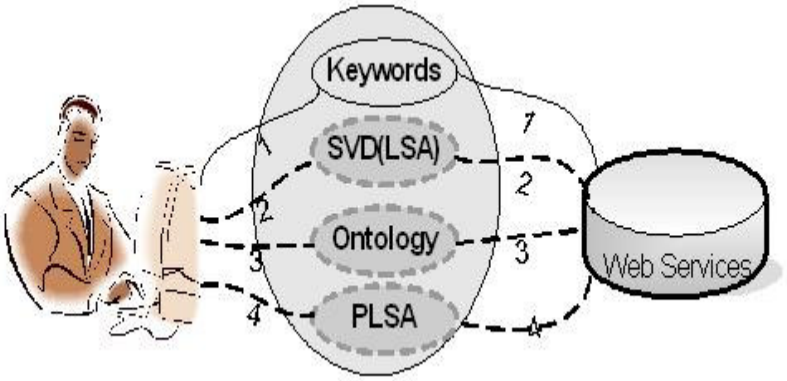

Figure 4.3: Approaches of Web service discovery

An alternative to the keywords-based approach is to *indirectly* associate a user's requirements to the advertisements of Web services (dashed lines 2, 3, 4 shown in Fig. 4.3). This relies on finding common semantic concepts between the terms in a query and

services' advertisements. Then the similarity between a query and services can be compared at concept level. More recently, ontology-based approaches [92, 105] have been seeking to use ontology to annotate the elements in Web services. Such techniques, based on a theory on existence, organize domain knowledge into the categories by virtue of objects' components and their semantic connectives so that the suggested approaches aim to not only capture the information on the structure and semantics of a domain, but facilitate software agents to make inference at concept level.

Main Problems of Current Discovery Approaches

- **Lacking Semantics Support.** Existing discovering approaches and standards such as UDDI can be used to enable the discovery process of Web services. However they concentrate mainly on syntactic service descriptions. In addition, the success of such approaches is based on the fundamental assumptions that the words used to describe Web services are predefined and known for both service requesters and service providers. For example, when service requesters want to locate a Web service, they express search requirements described in the form of nature language. In this situation, the service requesters can be assumed to know the service's semantics, that is, what kind of service they need. On the side of service providers, on the other hand, it is assumed that service requesters would agree on the words used to describe the Web services capabilities through some descriptions such as the input and output names, the operations' description. However, the assumption is impractical in reality because different customers may use different ways (for instance, using different words) to express their requests and also because different services providers may use

different phrases to describe their services capabilities. It is thus clearly that the problem of how to deal with the agreement and how to associate the users' requirements to the advertisements of Web services would have a critical impact on discovering Web services. Therefore, locating desired services based on semantics might be difficult.

- **Poor scalability**. Current centrally maintained repositories like UDDI are poor at supporting scalability in the Web context. In the situation where keywords are used to find services, the excessive number of the services returned has made it difficult for users to browse, to look at and to choose the interesting services.

In this chapter, we present the design and implementation of a Web services discovery framework to effectively discover Web services. We argue that efficiently locating Web services needs to consider the issues of semantic concepts and scalability.

The salient features of our approaches are:

- Complying with the current dominating mechanisms of discovering and describing Web services. We use keywords to first retrieve Web services, and then extract semantic concepts from the natural language descriptions in WSDL and UDDI, which are currently dominating mechanism of discovering Web services. This is different from keyword-based discovery approach.
- Supporting scalability for discovering and selecting Web services. We propose to use the divide and conquer methodology to handle the poor scalability in the Web environment and the issue of lacking semantics.

4.2.1 Keyword-Based Web services Discovery

Keyword-based approaches are widely used in traditional information retrieval systems. An information requester submits the system with a query that consists of a number of keywords in order to retrieve the desired documents. The retrieval system returns stored documents in answer to the information requester based on the similarity between the query and the stored documents. Here similarity means that the documents contain particular keywords from the requester's query or those documents prove similar enough to the corresponding the query, and those documents are returned to the information requester.

Currently, keywords-based mechanism is one of the techniques for Web services discovery and matching [34, 37, 43, 59]. As described in the previous section, service discovery is performed by comparing a query with the descriptions of services to figure out which provided services are relevant to a specific request. This comparison is also referred to as matching. As its core functionality, a discovery framework must specify how the matching of capability descriptions is carried out.

Keyword-based discovery approach is based on two observations: term frequency (TF) and inverse document frequency (IDF).

- Term frequency (TF). TF refers to counting number of occurrences of each term in a document. For example, if a term "computer" occurs with high frequency in an particular document, it indicates that the corresponding document are high related to computer subject, In this case, term "computer" will be assigned to be a high frequency weight

- Inverse document frequency (IDF). Inverse document frequency measures how often the terms occurs in all documents, and its inverse value indicates the relative rarity of a term. It is defined as:

$$IDF_t = \log\left(\frac{N}{n_t}\right),$$

Where N is the number of documents in the collection, and n_t is the number of documents in which term t appears. The logarithm function is used to limit the range of data.

Formally, for a given a set of documents, we can measure their similarity by applying the term frequency and inverse document frequency (TF-IDF) mechanism.

Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of documents. For each term t_j, let n_{ij} denote the number of occurrences of term t_j in the document d_i. Also let n_j be the number of documents that contain term t_j at least once. Thus the TF-IDF weight of the term t_j in the document vector d_i is given by

$$w_{t,d} = TF_{t,d} \times IDF_{t,d}$$

Where $TF_{t,d}$ indicates how many times term *t* occurs in the documents *d*:

$$TF_{t,d} = \frac{n_{ij}}{|d_i|}$$

$IDF_{t,d}$ is inverse document frequency:

$$IDF_{t,d} = \log\left(\frac{n_j}{n}\right)$$

Based on the representation of term weight above, information retrieval can be completed by computing the similarity between two documents d_1 and d_2:

$$sim\ (d_1, d_2) = \frac{\sum_t w_{t,d1} \bullet w_{t,d2}}{\sqrt{\sum_t w_{t,d1}^2} \bullet \sqrt{\sum_t w_{t,d2}^2}}$$

Current UDDI Supporting Keyword-Based Approach

At present, UDDI standard and most of discovery approaches support keyword-based discovery mechanism. The core of the UDDI architecture is a central business registry that servers as a directory service. In a UDDI registry, Web services are registered and described as three main type of information:

- *white pages* containing addresses and contacting details for an organization,
- *yellow pages* containing classification information based on standard taxonomies and,
- *green pages* providing the technique information about the interface of Web services.

This main type of information allows users and enterprises to discover and share information with regard to the Web services and other electronic and non-electronic services that are registered in a registry. A UDDI registry service is a Web service that manages information about service providers, service implementations, and service metadata.

The UDDI also allows syntactically search and category-based match Web services. In addition, a service requester can use the Inquiry API provided in UDDI for retrieving services via submitting instructions like *find_service ().* For example, a service requester submits the UDDI a query that includes keywords. The query is matched against the service description stored in the UDDI registry. The matched Web services are then returned as a candidate answer set and the service requester then browsers them in order

to find which one of them really suits theirs needs. Unfortunately, UDDI registry does not support semantic descriptions and semantic searching on functionality. Searches, as a result, can only be based on keywords, such as a service's name, provider, location, or business category.

4.3 DC-SVD: Divide and Conquer Semantic Discovery Approach

In this section, we first introduce the basic matching problems in Web service discovery. We then discuss the semantic matching approach based on Singular Value Decomposition (SVD), matching algorithms, followed by the preliminary experimental evaluation.

Currently, Web services discovery depends on the WSDL for describing Web service. With WSDL, a Web service can be described as an abstract interface in natural language. This syntactically-based description such as a service name, method names and some descriptions included in a service implicitly indicates information about the service's corresponding functionality and domain. Based on the standard description of Web services, various methods can be used to find services on the Web, such as using Web search engines [128, 126], service portals and UDDI, etc.

Research Issues

We argue that efficiently locating Web services needs to consider the issues of semantic concepts and scalability. Existing discovery approaches heavily rely on the keyword-based finding mechanism. Unfortunately, keywords are insufficient in expressing semantic concepts, which is partially due to the fact that keywords are often described by natural language, being much richer in terms of diversity. For example, syntactically

different words may have similar semantics (synonyms) which results in low recall. In addition, semantically different concepts could possess identical representation (homonyms), leading to low precision. As a result, the retrieved services might be totally irrelevant to the need of services' consumers. One possible solution to this problem is to describe services capabilities by using the Semantic Web. For example, some research uses ontology to annotate the elements in Web services for finding common semantic concepts between the query and services' advertisements.

Scalability is another issue needed to be addressed. Current centrally maintained repositories like UDDI are poor at supporting scalability in the Web context. In the situation where keywords are used to find services, the excessive number of the services returned has made it difficult for users to browse, to look at and to choose the interesting services. One way to deal with this issue is to compress data for reducing the number of services returned to service requesters. However, conventional techniques such as Singular Value Decomposition (SVD) [13] may not be suitable for dealing with a large-scale data collection due to the high cost of computing and storing of SVD.

We address these two problems by presenting a novel two-phase approach for efficiently finding Web services. Given a query, in the phase one, the proposed approach first retrieves a set of samples of Web services from a services database to form an initial dataset. The dataset is then divided into a set of smaller clusters by using the Divide and Conquer approach [55], aiming to reduce the number of services returned. This phase focuses on analyzing the syntactical correlations between the query and service descriptions. After finding the right cluster that is relevant to the query, in the phase two, the SVD technique is applied to the cluster to capture the semantic concepts hidden

behind the words in a query, and the advertisements in services, so that services matching against the query is expected to be carried out at an advanced concept level. We call our approach DC-SVD. Broadly speaking, our method combines keyword-based syntactical analysis with semantic concepts extracted from service WSDL files.

Our key contributions are as follows:

- We proposed a novel approach to find relevant services through the combination of clustering technique and Singular Values Decomposition to handle poor scalability and lack of semantics.
- We described the preliminary experiments over the real service collections to evaluate the effectiveness of our approach, and results show improvements over precision.

4.3.1 Overview of DC-SVD Approach

Our Divide and Conquer semantic approach (DC-SVD) is dependent on the combination of the keyword technique and the semantics extracted from the service descriptions. The objectives of DC-SVD are to handle the poor scalability in the Web environment and the issue of lacking semantics. To realize these goals, a large service collection is first partitioned into a set of smaller clusters by using a modified clustering algorithm. After finding the right cluster related to a query, the SVD technique is applied to the cluster so that service matching against the query can be carried out at the concept level. Figure 4.4 shows DC-SVD discovery approach.

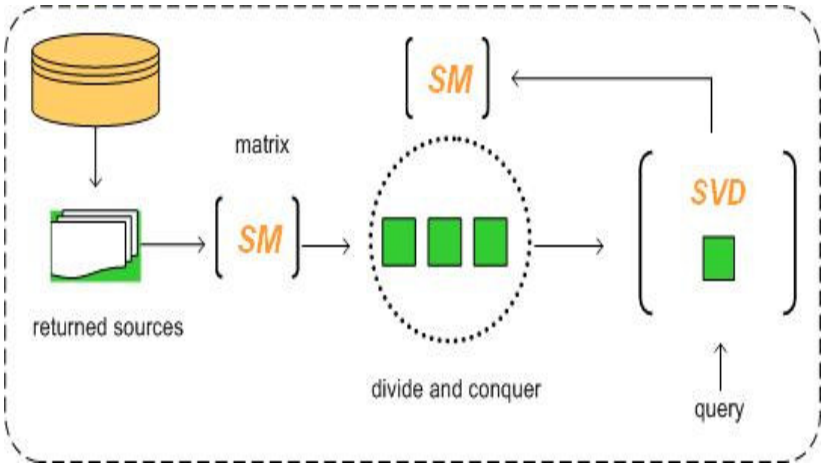

Figure 4.4: Decomposing search results

The DC-SVD approach is based on the assumption that the efficiency of finding services can be improved if relevant service cluster can be located before the extracting semantics algorithm is implemented. To begin with, given a query, the proposed approach retrieves a set of samples of Web services from a source of Web services to form an initial data set. Instead of directly applying the SVD to this large initial data set, we partition it into a set of smaller clusters by using a clustering algorithm, aiming to reduce the number of services retrieved. This phase focuses on analyzing the syntactical correlation between the query and service descriptions. As a next step, we filter out those Web services whose contents are not compatible with a user's query via finding relevant cluster. Finally, SVD technique is applied to the right cluster to capture semantic concepts hidden behind the words in a query and the advertisements in services, so that services matching against the query is expected to be carried out at an advanced concept level.

4.3.2 Decomposing Search Result Collection

This section considers the divide and conquer semantic approach for efficiently finding Web services. As we noted earlier, the initial data set of services retrieved may be huge, so we first use a modified k-means algorithm to partition the dataset into a set of clusters. After finding the right cluster related to the query, SVD is used to capture semantic concepts hidden behind the words in a query and the advertisements in services.

As the first stage of efficiently finding Web services, we decompose a large dataset of the services into smaller groups. Given a query, the size of the data set of services returned may be very large, therefore the computational cost of directly applying SVD to the dataset may be expensive. A possible solution to dealing with this issue is to reduce the size of the large dataset to a reasonable magnitude. In this project, we utilize the Divide and Conquer approach to handle such complex information decomposition.

The Divide and Conquer approach is a methodology that transforms a complex problem into a series of simpler ones, which can be handled more easily. The approach has been often used in computer science, as in database design and in software engineering, etc. In our case, we use it to partition an initial query-related document collection. In the beginning of decomposition processing, the size of the data collection is assumed to be big. When the decomposition processing is completed, the original dataset is divided into different groups, so SVD can be applied to specific cluster to capture semantic concept. In the following definition, we first formulate the problem of decomposing search result collection.

Definition 4.1 Given w documents $S = \{s_1, s_2, ..., s_w\}$ returned with respect to a query, partition S into k groups, $C = \{c_1, c_2, ...c_k\}$.

◊

Basically, several methods can be used to split a large data set based on the above definition. In this thesis, we use a Bisecting k-means, a simple variance of k-means algorithm, to reduce the size of a data set. Before introducing the Bisecting k-means partitioning algorithm, we first briefly outline the principle of k-means.

The k-means algorithm partitions a data set A into k clusters c_j, each cluster c_j including its centre denoted as:

$$cm_j = \frac{1}{|c_j|} \sum_{a_i \in c_j} a_i \tag{4.1}$$

Where $|c_j|$ is the number of data points in cluster c_j

Based on the Euclidean distance measure, the distance between a data point a_i and a cluster centre cm_j can be represented as:

$$dis(a_i, cm_j) = \|a_i - cm_j\|_2 = \sqrt{\sum_{d=1}^{n} (a_{i,d} - cm_{j,d})^2} \tag{4.2}$$

, and the following objective function is used to represent quality of a cluster:

$$O = \sum_{j=1}^{k} \sum_{a_i \in c_j} dis(a_i, cm_j)^2 \tag{4.3}$$

K-means algorithm continues until the objective function reaches minimum.

Compared with the basic k-means algorithm, the Bisecting k-means algorithm shows some advantages [101]. Firstly, it can produce consistent clusters with relatively uniform sizes. In this case, every sub-cluster may include a similar number of documents. Secondly, the computation on partitioning a data set is simplified. With the Bisecting k-

means algorithm, for a new data point, the bisecting algorithm only needs to compute the distance between the point and two clusters' centroids. Obviously, it is in contrast to basic k-mean algorithm where it is necessary to compute the distance between a point and every cluster centroid.

The Bisecting k-means algorithm starts with dividing a single cluster of the whole data set into two sub clusters with k-means algorithm. After that, the partitioning processing continues until the desired number of clusters is acquired. The details of the variant k-means clustering algorithm are described as follows.

Algorithm 1: DataDecomposition

```
1:  DataDecomposition (S, T, K)  {
2:  input: data_corpus(S);T: number_of_trial bisectionr
3:  K: number of clusters
4:  output:  a set of clusters MC;
5:  MC ← φ
6:  Select a cluster  c to split (initially, c is  S)
7:  Divide selected cluster into two clusters using 2-means
8:  Repeat step 7  T times, take dividing with the best similarity
9:  Repeat steps 6, 7 and 8 until k clusters are found
10: End
```

Evaluating Quality of Cluster

After partitioning a large data collection into a set of smaller clusters, we use commonly used entropy and purity to evaluate a cluster's quality. Suppose m classes represent

partitioned services (service categories) and k clusters produced by our clustering algorithm, then the following definitions apply [82]:

For a cluster c_j , its entropy is defined as:

$$E(c_j) = -\sum_{i=1}^{m} \frac{n_j^i}{n_j} \bullet \log\left(\frac{n_j^i}{n_j}\right) \tag{4.4}$$

Where $n_j = |c_j|$, representing the size of cluster c_j , and n_j^i indicates the number of services in cluster c_j that belongs to class *i*.

Entropy expresses a cluster's consistence. If the members of a cluster come from different classes, the value of the entropy is high.

The purity of a cluster c_j is defined as:

$$P(c_j) = \frac{1}{n_j} \sum_{j=1}^{k} \max_i \{n_j^i\} \tag{4.5}$$

Where *i* varies over all classes

Alternatively, one can build a centroid for each cluster through taking the average of all documents in that cluster:

$$C_i = \frac{1}{|C_i|} \sum_{j \in C_i} d_j \tag{4.6}$$

Constructing Service Matrix

Based on the approach introduced in the previous section, we can find the right cluster related to a query. After finding relevant cluster, SVD is applied to the cluster to capture semantic concept hidden behind the service documents.

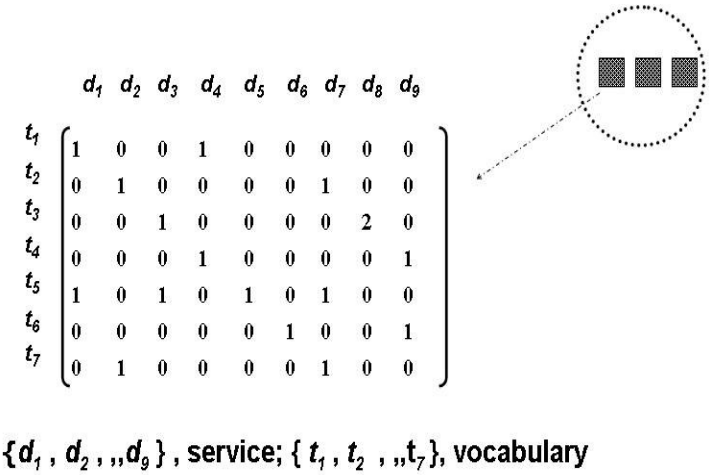

Figure 4.5: An example of matrix

Formally, we exploit the Vector Space Model (VSM) to describe each document in the cluster. In VSM, each document is described by bag of words or terms, which means that the frequency of the words in a document is considered while the positional relationship between terms is ignored. In addition, all terms in a data collection form a vocabulary, which spans a high dimensional vector space in which documents are represented as a set of points. According to VSM, each document can be represented as a vector:

$$\vec{a}_i =< w_{1,i}, w_{2,i}, \ldots, w_{v,i} >$$

Where v is the size of the vocabulary

The key advantages of VSM adopted in Web services lie in the flexibility to the selection of vectors dimensions and weights. For example, the values of entries $w_{i,j}$ in the document vector $\vec{a}_i$ can be determined through different schemes such as with counting the time of co-occurrence of a word appearing in a document.

In this thesis, the commonly used term-frequency and inverse-document-frequency (TF-IDF) [97] is used to denote the entries $w_{i,j}$ in the document vector $\vec{a}_i$. More precisely, the weight w_{ij} is defined as the TF-IDF weight of the word j in documents i as following:

$$w_{ij} = f_{ij} \bullet \log(\frac{n}{n_i}), \tag{4.7}$$

Where $f_{ij} = \frac{n_{ij}}{|a_i|}$ denotes word frequency, that is, the number of times word j appears in service i, and n_i is the number of services that contain word j.

Thus, the similarity between two documents can be compared by computing the cosine of the angle between the two vectors d_1 and d_2:

$$Sim(d_1, d_2) = \frac{\sum_{i=1}^{k} d_{1,i} * d_{2,i}}{\sqrt{\sum_{i=1}^{k} d_{1,i}^2} * \sqrt{\sum_{i=1}^{k} d_{2,i}^2}} \tag{4.8}$$

Where $d_1 =< d_{1,1}, d_{1,2}, ..., d_{1,k} >$ *and* $d_2 =< d_{2,1}, d_{2,2}, ..., d_{2,k} >$

With the above description, the documents in a cluster can be represented as a matrix:

$A=[a_1, a_2, ...a_n] \in R^{m \times n}$ *with* $a_i \in R^m$

4.3.3 Matching Web services in Latent Semantic Space

In this section we describe how to find similar Web services in a semantic space. We first introduce the basic principle of SVD, and then present a Latent Semantic Indexing (LSI) approach to find similar services in semantic space.

Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD) is a linear algebra technique that deals with the transformations and the decomposition of a matrix. With the approach of orthogonal transformations used by SVD, a matrix can be decomposed into the products of three sub-matrices in which the information contained the original matrix is rearranged. This decomposition technique is commonly utilized for producing a low-rank approximation matrix to the initial matrix, and for indexing semantics in traditional Information Retrieval (IR).

Formally, given a word-document matrix $A \in R^{m \times n}$, decomposition of matrix A can be represented as:

$$A = U \Sigma V^T \tag{4.9}$$

Where $U \in R^{m \times m}$, $V \in R^{n \times n}$,*and*

$$\sum = diag \ (\sigma_1, \sigma_2, ..., \sigma_w) \tag{4.10}$$

In formula 4.9, sub-matrix U is an $m \times m$ orthogonal identity matrix, that is, $U^T U = I_m$, and sub-matrix V is an $n \times n$ orthogonal identity matrix with $V^T V = I_n$. In a similar way, matrix $\sum$ is a $m \times n$ diagonal matrix whose values are called singular values. The matrix $\sum$ also indicates some interesting features. First, all off-diagonal entries in $\sum$ are zeros. Next, singular values along the main diagonal are sorted in decreasing order numerically. Furthermore, the number of the nonzero values in the diagonal in Σ represents the rank of matrix A. All these characteristics can be used to discover the latent semantics hidden in a data collection.

Finding Similar Services in Latent Semantic Space

Efficiently finding similar services in a semantic space involves several steps: approximating original matrix *A*, representing documents and query in dimension-reduced semantic space and matching services with respect to a query.

At first, the technique of SVD decomposition can be utilized to produce a low-rank approximated matrix to the initial matrix. The approximation relies on reducing the dimension of the initial matrix so that the main information in the initial matrix is kept while minor information is removed. It is believed that the semantic concept corresponding main data in a dataset is merged within the minor data called noise. With SVD, the latent semantic concept can be discovered by omitting all smaller singular values, corresponding noise data in the dataset.

Specifically, according to the equation 4.9, a proper parameter k can be chosen to construct an approximated matrix A_k with rank k as:

$$A_k = U_k \sum_k V_k^T \qquad k \le rank(A) \tag{4.11}$$

Where $U_k = < u_1, u_2, ..., u_k >$, $\sum_k - diag < \sigma_1, \sigma_2, ..., \sigma_k >$ and

$V_k = < v_1, v_2, ..., v_k >$

The theorem [36] of SVD ensures that matrix A_k is the best rank-k approximation of *A* with respect to Frobenius norm.

Before representing query and documents in latent semantic space, we summarize some important properties of SVD:

- The rows of matrix U correspond to the rows of A, and the columns of matrix v correspond to the columns of A.

In k-dimensional semantic space, the coordinates of terms can be represented by the rows of $U_k \sum_k$.

- Similarly, the coordinates of documents can be represented by the columns of $\sum_k V_k^T$.

As we know, the matrix $A_k A_k^T$ includes all term by term dot products and can be further expressed as:

$$A_k A_k^T = U_k \sum_k^2 U_k^T = U_k \sum_k (U_k \sum_k)^T \quad (4.12)$$

This equation indicates that the dot products of the two elements i and j of $A_k A_k^T$ can be obtained by taking the dot product between the row i and j of $U_k \sum_k$. Similarly, matrix $A_k^T A_k$ contains the dot products among documents. Thus, we have the properties of SVD mentioned before.

Next, the query and the documents can be represented as vectors in dimension-reduced semantic space. Based on the properties of SVD, the coordinates of a query q in k-dimensional space are defined as [12, 27]:

$$q' = q^T U_k \sum_k^{-1} \quad (4.13)$$

In the similar way, each document can be represented as:

$$d_i = < \sigma_1 v_1, \sigma_2 v_2, ..., \sigma_k v_k >$$

With the help of the representation in semantic space, the semantic concepts of a word can be deduced from its correlations with other words. In addition, the service documents sharing the similar correlations tend to be semantically similar.

Finally, the query vector can be compared to all document vectors, which produce a ranking vector as:

$$r =< q^T U_k \sum_k^{-1} >< \sum_k V_k^T > \quad (4.14)$$

To measure the similarity between the query and documents, the commonly used similarity measurement such as the cosine scheme can be used to recommend the relevant service documents to a user if the score based on the similarity exceeds a predefined threshold.

Summary of DC-SVD Approach

The DC-SVD semantic matching approach uses a dynamic algorithm that partitions a large service collection into smaller pieces. It includes the two main phases: decomposing service collection and semantic matching services. A service collection is first decomposed into a set of smaller clusters. Note that at this stage, no semantic similarity is involved because our main objectives are to reduce the initial size of service collection and also to diminish the cost of computing a large data set. Once the service collection is divided into a set of smaller services groups, SVD is applied to the right cluster for capturing semantic concepts. If the service results returned are not compatible with a user's query, the second best cluster will be chosen and the computing proceeds to the next iteration. The semantic matching process continues until the matched results are compatible with user's request.

The pseudo code for DC-SVD algorithm is given as following:

Algorithm 2: SummaryOfAlgorithm

```
1:  Retrieving initial service collection
2:  Partitioning the collection into a set of clusters
3:  Finding relevant cluster
4:  Applying SVD to the cluster
5:  Semantic Matching service against query
6:  if the results match the query then goto step 10
7:  else   choosing next cluster
8:         goto step 4
9:  end if
10: end
```

4.3.4 Experiments Evaluation

We now present our preliminary experiments to show the effectiveness of the DC-SVD approach. We first describe two experimental datasets and the evaluation metric, and then present the experimental results.

Experimental Datasets

We implemented experiments over the two real data sets. The first one is the collection of Web services that can be downloaded from the Web site [138]. The collection

includes 424 Web service descriptions covering the 25 categorises such as Zip code finder and weather information, etc. The reason that we select the service collection in our experiments is that the Web services in the collection are gathered from real-world service sites like SALCentral and XMethods. In addition, the Web services are artificially classified into different categories so that the services provide a basis on which testing and comparison can be made on a variety of situations.

As the extensive datasets of real services are not available and as the number of services in some service collections is very limited, the second data set we use is

MED data set [139] with 1033 documents companied by 5831 terms. The data set is used to evaluate whether DC-SVD approach can improve the scalability of efficiently discovering Web services. The two datasets are shown in the Table 4.1.

Table 4.1: Experiment datasets

	Documents	Terms
Dataset 1	240	735
Dataset 2 (MED)	1033	5831

Data Processing

The data processing mainly consists of transforming raw Web service information into appropriate the format of data suitable for the model learning, which involves several steps. We first extract the text information from the Web services provided in the first data set. As the text information in a service such as its description and its operations' names are likely concatenated by a sequence of strings where individual word starts by uppercase letter, for instance, getCityWeather, the names and description should be

separated so that each token conveys some meaning. Secondly, after extracting the keywords, each Web service document is represented as a vector in which the values of its entries are set by the TF-IDF scheme. Other methods of data processing include the word stemming and the stopwords removing. The former intends to remove common term suffixes while the latter eliminates very frequently used words.

Evaluation Metric

In order to evaluate the effectiveness of the proposed approach, we use the widely adopted standards in IR: precision, recall and accuracy to measure overall performance. Given a query *q*, let C be the total number of services retrieved, *A* be the total number of relevant services in the service collection and B be the number of relevant services retrieved. The recall of our approach is defined as

$$\text{Recall} = \frac{B}{A} \tag{4.15}$$

The precision of the approach is defined as

$$\text{Precision} = \frac{B}{C} \tag{4.16}$$

We also define the retrieval accuracy as follows:

$$\text{Accuracy} = \frac{number\_of\_identified\_serivces\_j}{number\_of\_service\_in\_Category\_j} \tag{4.17}$$

Experimental Results

We now turn to the evaluation of the experimental results. Given a query, a large set of the services returned is first divided into a set of clusters by using a variant k-means algorithm. On the next step, SVD is applied to the relevant clusters via computing the similarity between a query and the centroid of each cluster.

The first experiment was implemented over the dataset 2 to deal with the scalability. Given a query, we retrieve an original data set returned from the MED, and the initial dataset is then divided into k different groups. Intuitively, the choice of value k in such division is correlated to the prior knowledge on a training data set, such as the knowledge on the number of the categories in the dataset. In our experiment, we set k to be 10, 20, and 30. After locating related cluster, SVD is applied to the cluster. The results for precision and recall are displayed in Figure 4.6.

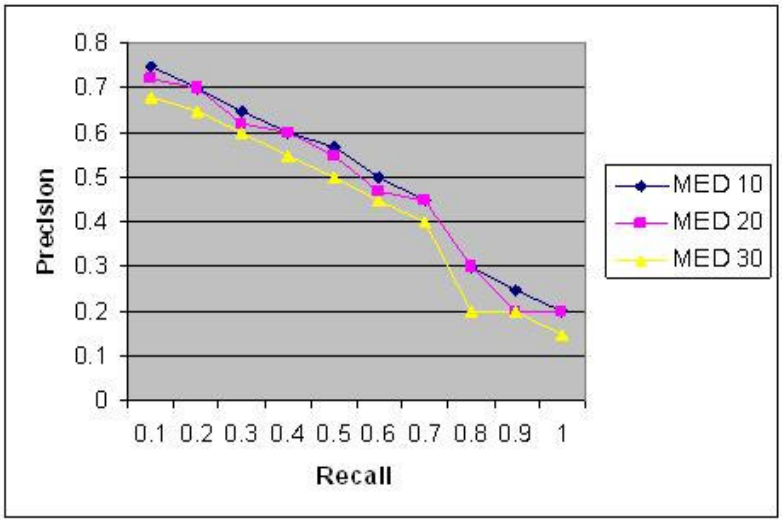

Figure 4.6: Precision and recall

From the Figure 4.6, we can see that the different sizes of dataset have some effect on the performance of service retrieval. As shown in Figure 4.6, the performance of MED 10 outperforms MED 30. The reason for this is that in the MED 30 situation, the increase of the number of clusters would lead to small cluster size, which is insufficient in representing the correlations among different terms.

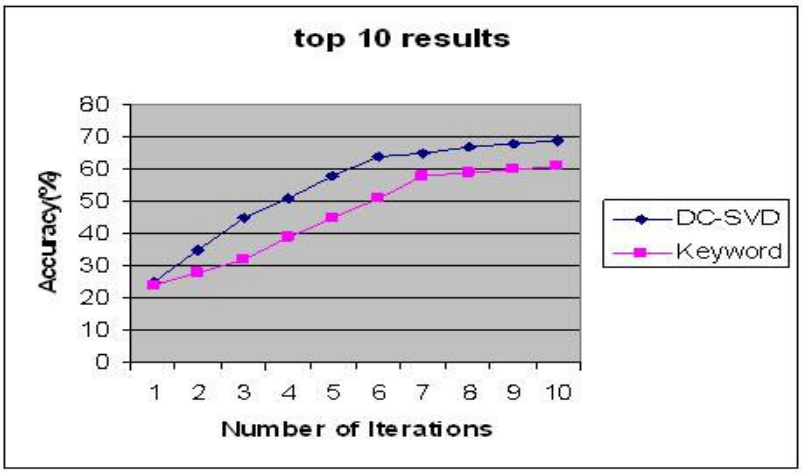

Figure 4.7: Comparisons of DC-SVD with keyword

To see how effective DC-SVD approach using Divide and Conquer strategy described in the previous section is, we compared the performance of DC-SVD approach with that of the keyword-based approach. In this experiment, the experimental parameter is first fixed, e.g., setting the number of clusters to be 10. The comparison was made on the top n=10 results retuned by each method. During the experiment, we run 10 times at each situation. The results of the comparison on the accuracy are shown in the Figure 4.7. As can be seen from this figure, the performance of keyword-based technique only based on the text description in Web services is poor. However, the accuracy of service retrieval is improved when DC-SVD approach is introduced.

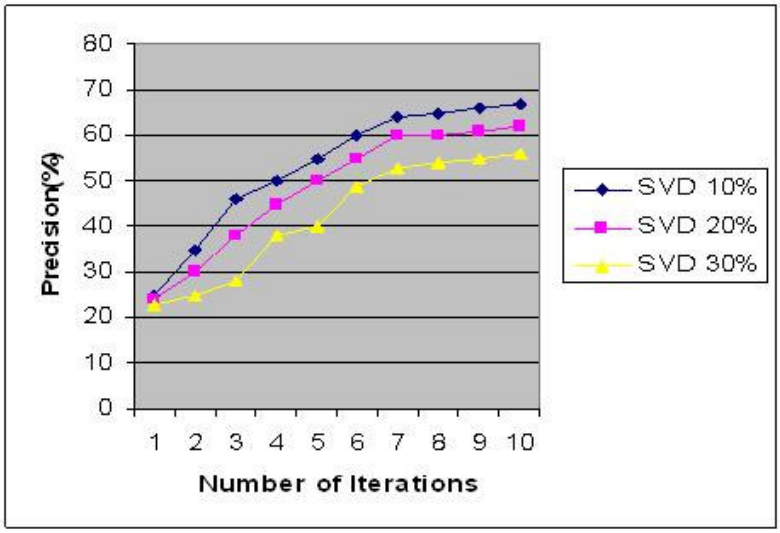

Figure 4.8: Performance in different size of SVD

Another experiment was performed to evaluate the performance of the services matching in the latent semantic space with different dimensions. In this case, the size of a dataset is varied from 10% to 30% of the dataset to observe the performance. Figure 4.8 shows the experimental results.

The results show that the dimension of a semantic space has impact on the precision of service matching. This is because the rank in SVD represents the number of semantic concept classes in a dataset. Therefore, lower values below the optimizing rank would cause information loss, leading to decreasing the service retrieval accuracy. On the other hand, higher values over the rank would result in too much noise in computation.

4.3.5 Related Work

In this section we briefly discuss some of the research work related to locating Web services. Although Web services adopt standard interfaces and protocols, it is difficult to enumerate various approaches for Web service discovery because the notion of service

discovery can be explained in different ways. In this thesis, we generally identify the discovering approaches as two categories: source-based discovery approach and capability-based matching approach. In the first category, there are three major approaches [2, 37] for discovering Web services based on the source of services: UDDI, Service Portals and Web search engines. These approaches normally provide users with some basic functions such as by using keyword-based search and category-based browsing Web services. The second category emphasizes on service capability matching at the level of syntactical analysis by directly using keywords as well as at the level of semantic matching by indirectly employing Ontology and by using Information Retrieval (IR) techniques [76].

One of IR techniques is latent semantic indexing (LSI), which uses the Singular Value Decomposition (SVD) to deal with the transformations and decomposition of matrices. SVD has been widely used in the compression of data and traditional information retrieval [33]. However, the main drawback is the high cost in computing and storing of the SVD, particularly for large-scale datasets [13, 14].

More recently, LSI is used for discovering Web services [104]. Sajjanhar [106] designs an algorithm for Web service matching based on SVD. Paliwal [91] presents an approach for service discovery, which combines ontology linking with LSI. Based on the features extracted from the selected WSDL files, the approach first gets a training set by using LSI and then expands the query by using Ontology.

Dong [34] puts forward a valuable similarity search approach to find Web services based on the keyword strategy. The search consists of two main steps. A service user first types keywords into a service search engine to look for the corresponding services. The

approach then extracts semantic concepts based on the initial services returned. With the help of the co-occurrence of the terms appearing in service inputs and outputs, names of operation and description in services, the similarity search approach employs the agglomerative clustering algorithm to cluster these terms into different concept groups. As a result, the similarities of Web services can be compared at the concept level. Other similar methods also intend to learn semantic concepts from the natural language descriptions provided in Web services [9]. For example, Balke [9] proposes a personalized approach to select Web services. The method is based on a keyword search, and expands user query with the user profile for the purpose of enhancing the catalogue concept in UDDI.

Our DC-SVD approach has similarities to approaches [106, 92, 9] in that we also use keywords to first retrieve Web services, and extract semantic concepts from the natural language descriptions in the Web services. However, our work differs from these works in several ways. Firstly, we extend our previous approaches [74] by utilizing the Divide and Conquer approach to handle the issue of scalability by partitioning a large service collection into a set of smaller groups. Secondly, SVD is applied to the specific service cluster related to a query, aiming to match services at the concept level. Furthermore, the preliminary experiments were implemented over the real service collection.

4.4 Efficiently Selecting Web services Using a Clustering Semantic Approach

In this section, we propose a clustering semantic approach to select Web services. Our clustering semantic approach is dependent on combination of the keyword technique and the semantics extracted from the services' descriptions by using, Probabilistic Latent

Semantic Analysis approach (PLSA). The objectives of the proposed approach are to diminish the cost of computing a large dataset and to match services at the semantic concept level.

4.4.1 Probabilistic Latent Semantic Analysis (PLSA)

In this section, we first investigate main specifications of WSDL and then briefly introduce our probabilistic latent factor discovering approach.

Services Description and Specification

Since the WSDL and UDDI are currently the dominating mechanism for Web services description and discovery, we focus on discovering and matching Web service in this context, rather than using ontology to annotate elements in Web services.

Normally, a Web service can be described by WSDL as a collection of network endpoints. The description consists of two main parts: the abstract definition of interfaces and the concrete implementations of network. In the abstract definition, interfaces and a set of operations are defined by *portType* element and *operation* element respectively. Besides, each operation may contain input/output messages that are defined by *message* element. On the other hand, the concrete implementations specify how the abstract interfaces are mapped to the specific bindings, which may include particular binding protocols like SOAP and network address. Similarly, a set of elements such as *service, port* and *binding* are used to define these deployment details.

The key advantages of this mechanism adopted in Web services lie in the separating the interface definition from the network implementation and making it possible to multiple

deployments on the identical interface. Moreover, it would facilitate the reuse of the software in the Web service community.

Overview of CPLSA Approach

Our clustering semantic approach is dependent on combination of the keyword technique and the semantics extracted from the services' descriptions. Given a query, we first filter out those Web services whose contents are not compatible with a user's query via a clustering algorithm to acquire an initial working dataset. As a next step, Probabilistic Latent Semantic Analysis approach (PLSA) [50, 51, 52]] is applied to the working dataset, which is further clustered into a finite number of semantically related groups. In this phase, we use a Probabilistic Latent Semantic Analysis approach (PLSA) to capture semantic concepts hidden behind the words in a query and the advertisements in services, so that services matching is expected to be implemented at an advanced concept level. We call our approach CPLSA. Broadly speaking, our method combines syntactic analysis with a clustering semantic approach which is based on the current dominating mechanisms of discovering and describing Web services with UDDI and WSDL.

The objectives of CPLSA are to diminish the cost of computing a large dataset and to match services at the semantic concept level. To realize these goals, we first eliminate irrelevant Web services with respect to a query by using a modified clustering algorithm. After acquiring an initial service dataset, we use Probabilistic Latent Semantic Analysis to find a common semantic concept between Web services and query so that service matching against the query can be carried out at the concept level.

The CPLSA approach is based on the assumption that the efficiency of finding services can be improved if irrelevant data can be eliminated before the extracting semantics algorithm is implemented. In this thesis, the analysis of the proposed approach focuses on the scenario of discovering public Web services on the Web environment, which consists of following main procedures. Given a query, the proposed approach first retrieves a set of samples of Web services from a source of Web services. As the samples returned may include irreverent services with respect to the query, we then filter out those Web services whose contents are not compatible to a user's query via using a clustering algorithm to obtain an initial working dataset. Next Probabilistic Latent Semantic Analysis approach (PLSA) is applied to the working dataset, which is further clustered into a finite number of semantically related groups. This phase focuses on capturing semantic concepts hidden behind the advertisements in services. Finally, the semantic similarity of a query and Web services is measured within the related semantic cluster. Figure 4.19 illustrates the outline of the proposed clustering semantic probabilistic approach.

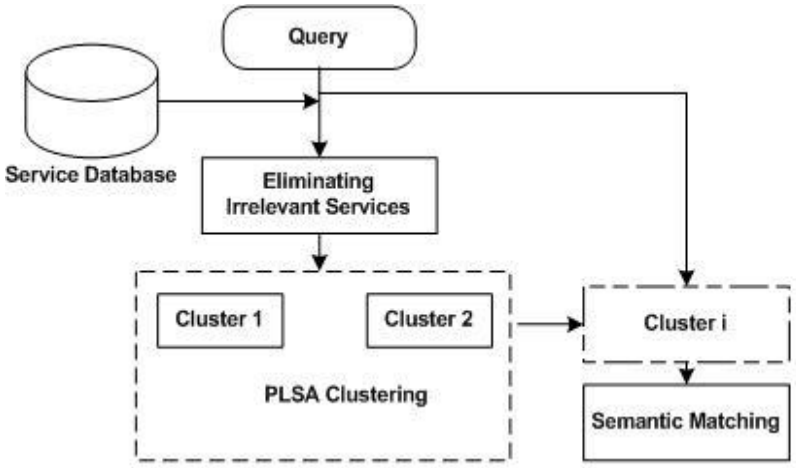

Figure 4.9: Outline of the matching

4.4.2 Eliminating Irrelevant Services from Service Collection

In this section we propose our clustering probabilistic semantic approach (CPLSA) for efficiently finding Web services. As we noted earlier, the samples returned may include irreverent services with respect to a query, so we first filter out those Web services whose contents are not compatible to a user's query to form a working dataset. Then we apply PLSA to the working dataset for further clustering the dataset into a finite number of semantically related groups.

We first retrieve a set of samples of Web services from a source Web services. Given a query q, a source of services would return a set of services based on some kind of similarity. To calculate the similarity, we use the Vector Space Model (VSM) to represent Web services as points in syntactic space. Based on VSM, we can measure the similarity between a query q and a service s in the samples by computing the cosine of the angle between query vector q and service vector s as:

$$Sim\ (q, s) = \frac{|q \bullet s|}{\|q\|^2 \bullet \|s\|^2} \tag{4.18}$$

Using the above similarity computation, we can acquire an initial set of samples of services through selecting a predefined threshold.

Considering the possibility that the initial set of services may contain the services whose contents are not compatible with a user's query, we eliminate them accordingly from the sample set to improve the efficiency of service discovery, and also to reduce the cost of computation. Intuitively, these irrelevant data may have some negative impact on efficiently finding Web services; for one thing, the data may diminish the accuracy of the learning algorithms; for the other, they would increase the computational load.

Therefore, as the first step towards efficiently locating Web services, these irrelevant services should be eliminated before the clustering semantic algorithm is implemented. Sseveral ways can be used to remove unrelated data from a dataset. One of the possible solutions is based on the feature selection, as indicated in [82]. This approach first sets a numerical threshold, and then computes the number of times a data object appears in a collection. If the number of times an object appearing in a collection is less than the predetermined threshold, the object is regarded as unrelated data and should be removed. We use a different approach to eliminate the unrelated services from the dataset. The method consists of two main steps. Given a query, the initial sample set of services retrieved is first divided into different groups by using a clustering algorithm, each group gathering related services and including a cluster centre. On the next step, the distance between a data object and each centre of each cluster is computed. If the distance between a data object and every cluster's centre is higher than a predefined threshold *u*, the object is regarded to be irrelevant to query, and should be eliminated. We first formulate the problem of irrelevant service elimination as follows.

Definition 4.2. Given w returned services $S = \{s_1, s_2, ..., s_w\}$ with respect to a query, cluster S to k groups $C = \{c_1, c_2, ...c_k\}$ and remove service s_i such that

$$\left\| s_i - c_j \right\| \geq \varepsilon \quad , k < w \ , \ j \in (1, 2, ..., \ k) \tag{4.19}$$

Where ε is a predefined threshold and c_j is the centre of a cluster.

◊

Specifically, a k-means algorithm is used to clean the initial sample set of services retrieved. With k-means algorithm, a service set S is divided into k clusters c_j, each including a centre denoted as:

$$cm_j = \frac{1}{|c_j|} \sum_{a_i \in c_j} a_i \tag{4.20}$$

Where $|c_j|$ is the number of data points in cluster c_j

Based on the Euclidean distance measure, the distance between a data point a_i and a cluster centre cm_j can be represented as:

$$dis(a_i, cm_j) = \|a_i - cm_j\|_2 = \sqrt{\sum_{d=1}^{n} (a_{i,d} - cm_{j,d})^2} \tag{4.21}$$

, and the following objective function is used to represent quality of a cluster:

$$O = \sum_{j=1}^{k} \sum_{a_i \in c_j} dis(a_i, cm_j)^2 \tag{4.22}$$

K-means algorithm continues until the objective function reaches minimum.

The algorithm of services elimination based on k-means is shown as follow:

Algorithm 3: EliminatingIrrelevantService

```
1.  ServiceElimination (S, k, μ)  {
2.  input: services_corpus(S); k: number_of_cluster
3.  μ : similarity _threshold
4.  output:  a set of clean services  MC;
5.   MC ← φ
6.  Assigning initial values to means cm1, cm2,...,cmk
7.  begin
8.    for  service s ∈ S  do
9.        Finding s cluster centre cmi , assign s into the cluster
10.       Computer new centre
11.    end for
12.       /* service elimination */
13.   for  each cluster ci  ∈ C = {c1, c2,...,ck} do
14.        for   each service  s ∈ ci  do
15.             if   distance (s, ci • center) ≤ μ  then do
16.                  MC = MC ∪ s
17.             end if
18.        end for
19.    end for
20.   return MC
21. end
```

Constructing Service Transaction Matrix

As described in the previous section, an initial working dataset is obtained by eliminating irrelevant services from the sample set of services retrieved with a k-means

algorithm. In this section, we further consider the relationship amongst services and construct a service matrix to be used as the input for our cluster-based algorithm introduced in the next section.

In traditional distributed databases, the relationship between the words in a dataset and service documents can be represented as a transaction matrix, where each column corresponds to a Web service document; each row represents a word (transaction). Meanwhile, the entries in the transaction matrix represent the frequency of occurrence of a word appearing in a service document. A service transaction matrix is shown in Table 4.2.

Table 4.2: An example of service transaction matrix

	Service 1	Services 2	Service 3	Service 4
Transaction 1	2	4	1	5
Transaction 2	0	1	2	2
Transaction 3	2	0	0	2
Transaction 4	3	2	2	1

To construct such a matrix, we exploit the Vector Space Model (VSM) to describe each service document in the working dataset. In VSM, each document is described by bag of words or terms, which means the frequency of the words in a document is considered while the positional relationship between terms is ignored. In addition, all terms in a data collection form a vocabulary, which spans a high dimensional feature space in which documents are represented as a set of points. According to VSM, each document can be represented a vector as

$$\vec{a}_i = (w_{1,i}, w_{2,i}, ..., w_{v,i})$$

Where v is the size of the vocabulary

The values of entries $w_{i,j}$ in the document vector $\vec{a}_i$ can be determined through different schemes such as with counting the time of co-occurrence of a word appearing in a document.

In our case, the term-frequency and inverse-document-frequency (TF-IDF) are used to denote the entries $w_{i,j}$ in a document vector $\vec{a}_i$. The weight w_{ij} is defined as the TF-IDF weight of the word j in documents i, denoted by formula 4.7.

Using formula 4.8, we can denote the similarity between two documents by computing the cosine of the angle between the two document vectors.

Based on the above description, the service documents in the working dataset can be represented as a matrix:

$$A = [a_1, a_2, ..., a_n] \in R^{m \times n} \quad with \quad a_i \in R^m$$

In the above descriptions of service elimination and service matrix construction, the focus is put on analyzing the syntactical correlation between the query and services at a basic level, aiming to improve the performance of service discovery. In the following sections, we shift our attention to the analysis of semantic concept.

4.4.3 Web services Discovering based on PLSA

Our probabilistic approach is based on the PLSA model that is called *aspect model* [52]. PLSA utilizes the Bayesian Network to model an observed event of two random objects with a set of probabilistic distributions. In the text context, an observed event

corresponds to occurrence of a word *w* occurring a document *d*. The model indirectly associates keywords to their corresponding documents through introducing an intermediate layer called the hidden factor variable $Z = \{z_1, z_2, ..., z_k\}$ with each observation of a word *w* in a document *d*. In our context, each observation corresponds to a service user accessing to services by submitting a query for locating desired Web services. Thus, the generative probabilistic model is expected to infer the common semantic concepts between a query and services. PLSA model works like this:

- Select a document d_i from a corpus of documents with probability $P(d_i)$
- Select a latent factor z_f with probability $P(z_f | d_i)$
- Generate a word w_j distribution with probability $P(w_j | z_f)$

Based on the assumption that a document and a word are conditionally independent when the latent concept is given, the joint probability of an observed pair (d_i, w_j) obtained from the probabilistic model is shown as following:

$$P(d_i, w_j) = P(d_i)P(w_j \mid d_i), \tag{4.23}$$

Where

$$P(w_j \mid d_i) = \sum_{f=1}^{K} P(z_f \mid d_i)P(w_j \mid z_f), \tag{4.24}$$

Now we face the task on fitting the model from a set of training data. The basic principle is to maximize probability $P(training-data \mid parameters)$ by finding a set of parameters. To simplify the computing procedure, an iterating approach is adopted. First of all, initial estimation can be used to update the model, and then the updated model presents a new

estimation on the previous iteration. In our context, an objective function based on the whole data collection is:

$$\prod_{j=1}^{N} \prod_{i=1}^{M} P(w_i \mid d_j)^{m(w_i, d_j)} \tag{4.25}$$

Thus, a log likelihood function of an observation is defined as following:

$$\ell = \sum_{j=1}^{N} \sum_{i=1}^{M} m(w_i, d_j) \bullet \log P(w_i \mid d_j) \tag{4.26}$$

Where $m(w_i, d_j)$ denotes the frequency of a word w_i occurring in a document d_j .

PLSA uses the Expectation-Maximization (EM) [51] algorithm to learn the model. The whole learning process starts with randomly assigning initial values to the parameters $P(z_f)$, $P(d_i \mid z_f)$ and $P(w_j \mid z_f)$, then is followed by alternative two steps: E-step and M-step. In E-step, based on the current estimation of the parameters, the posterior probabilities for latent variables are computed for all observed pairs (d_i, w_j) In M-step, the parameters are updated based on the probabilities computed in the previous E-step.

- This learning process can be summarized as following:
- Parameter $P(z_f)$, $P(d_i \mid z_f)$ and $P(w_j \mid z_f)$ are randomly assigned an initial value
- In E-step, the posterior probability over the latent variable conditioned on the occurrence of w_j occurring in d_i is computed as:

$$P(z \mid d, w) = \frac{P(z)P(d \mid z)P(w \mid z)}{\sum_{z_i \in Z} P(z_i)P(d \mid z_i)P(w \mid z_i)} \tag{4.27}$$

- And in M-step, according to the previous values, the parameters are updated for the conditional likelihoods of the observation:

$$P(d \mid z) = \frac{\sum_{w_i \in W} P(z \mid d, w_i) m(d, w_i)}{\sum_{w_i \in W} \sum_{d_j \in D} P(z \mid d_j, w_i) m(d_j, w_i)} \tag{4.28}$$

$$P(w \mid z) = \frac{\sum_{d_j \in D} P(z \mid d_j, w) m(d_j, w)}{\sum_{w_i \in W} \sum_{d_j \in D} P(z \mid d_j, w_i) m(d_j, w_i)} \tag{4.29}$$

$$P(z) = \frac{\sum_{d_j \in D} \sum_{w_i \in W} P(z \mid d_j, w_i) m(d_j, w_i)}{\sum_{d_j \in D} \sum_{w_i \in W} m(d_j, w_i)} \tag{4.30}$$

PLSA was originally used in text context for information retrieval and now has been used in Web data mining. In this thesis, we utilize PLSA for discovering and matching Web services.

The overall process of discovering Web services includes information collecting, data processing, data representation and similarity matching.

The information collecting: The main consideration on the information source in our discovering approach is based on the current specification of WSDL description and UDDI discovery mechanism. As each Web service has its associated WSDL file describing its functions, we firstly extract the overall service interface information such as name and textual description in the WSDL file. This kind of information will be used to decide whether a Web service's category is relevant to a user's query.

For this purpose, the commonly used approaches for the words processing are applied. As descriptions and names are likely concatenated by a sequence of strings where individual word starts by uppercase letter, for instance, getCityWeather, the names and descriptions are separated so that each token conveys some meaning. Other methods of data processing include the word stemming and the stopwords removing. The former intends to remove common term suffixes while the latter eliminates very frequently used words.

4.4.4 PSMA -- Probabilistic Semantic Matching Algorithm

Our discovering approach is first to cluster services to a group of learnt latent variables, then the similarity of a query in respect to the services in its relevant group can be computed in a smaller size of collection of Web services.

The learnt latent variables can be used to characterize Web services. From formula 4.24 introduced in the previous section, one can obtain some interesting interpretations. First, the right-hand side of the formula indicates a matrix decomposition, that is, the aspect model expresses dimensionality reduction by mapping a high dimensional term document matrix into a lower dimensional one(k dimension) in a latent semantic space. Second, $P(z_f \mid d_i)$ implies the association of the service d_i and its hidden factors z_f . For example, for a latent factor such as z_x, if the probability $P(z_x \mid d_i)$ is very high, the concept implied in the hidden factor z_x is regarded as to be highly correlated to the service d_i. So, based on a group of hidden variables, we can compute $P(z_f \mid d_i)$ over each hidden factor z_f, $f \in \{1,2,...,k\}$. With the k computing values, we can find a

maximum value $P_{\max}(Z_f \mid d_i)$, which can be used as the class label for this service. What is more, in a dimension-reduced semantic space, each dimension represents a semantic concept and the services with similar semantic concepts are projected to be close each other. Based on the discussion, we employ the following formula to infer the relationship between a Web service and hidden factors.

$$P(z \mid d_{new}) = \frac{P(d_{new} \mid z)P(z)}{P(d_{new})} = \frac{P(d_{new} \mid z)P(z)}{\sum_{z_f \in Z} P(d_{new} \mid z_f)} \tag{4.31}$$

An algorithm of the category matching is shown as following:

Algorithm 4: **Categorizing Services**

1. **CategorizingServices** (SM, K) {
2. **Input:** services matrix SM =$\{ sm_1, sm_2, ..., sm_n \}$
3. k: the number of latent factors
4. μ : threshold
5. **Output:** k service communities: SC = $\{ sc_1, sc_2, ..., sc_k \}$
6. $sc_f \leftarrow \phi$, $sc_f \in SC$
7. **begin**
8. **for** *each service* sm_i *in* SM **do**
9. **for** *each hidden variable* hv_j ,*j = 1,…,k* **do**
10. $hs_{ji}[j]$ = calculate_P($hv_j \mid sm_i$)
11. **end for**
12. $f \leftarrow$ find_max(probability_$hs_{ji}[j]$)
13. $sc_f = sc_f$ .append(sm_i)
14. **end for** *each service* $s \in c_i$ **do**
15. **return** SC **if** distance $(s, c_i \bullet center) \le \mu$ **then do**
16. **end**

The learned latent variables can be used to cluster Web services. As already mentioned, a Web service can be described as a multinomial probability distribution $P(z_f \mid d)$ over the latent variables $z_f \in Z = z_1, z_2, ..., z_k$. This representation of a service with these factors reflects the likelihood that the service belongs to certain concept groups. If a probability distribution over a specific factor z_f when given a Web service d_i is high, the Web service d_i can be clustered to the aspect z_f. This fact indicates that the PLSA model can function as a soft clustering approach that maps the observed object corresponding to natural concepts. In other words, if the objective of a service user, represented by a query, is closely associated to some Web services, the query and the services are expected to be mapped to some given factors with higher probability, compared to others with lower probability. In order to locate memberships that are associated with the latent factors, we can compute mixing coefficients:

Thus, for each hidden factor, we can compute $P(z_f \mid d_i)$ and get a maximised value for a specific Web service $d_i : Z_{\max}(d_i)$ that can be used as the class label for this service. In this way, all Web service documents are clustered to different categories in which all Web services indicate similar types.

The key to our approach is to cluster the services into a group of learned latent variables, which can be achieved by computing probability $P(latent - \text{var}iable \mid service)$ for each latent variable using formula 4.31. The rationale for this is that in the dimension-reduced semantic space, each Web service can be represented as a mixture of latent variables and the services with similar semantic concepts are projected to be close to each other. With

the maximum value of the computation used for the class label for a service, we can categorize services into their corresponding group.

The outline of clustering services based-on PLSA is shown as following:

Input: service matrix

Output: k service communities

Step1: choosing a service, compute probabilistic

with respect to each hidden variable using formula (4.31)

Step2: find the maximum value of the probability for the Service

Step3: put the service to its corresponding to group and select next service

As a query may be outside the model, we use Expectation Maximization [52] algorithm to fold the query in the model. Finally, we use the following formula for computing the similarity.

$$sim_{PLSA}(d_i, q) = \frac{\sum_{z_f \in Z} P(z_f \mid q) P(z_f \mid d_i)}{\sqrt{\sum_{z_f \in Z} P(z_f \mid q)^2} \sqrt{\sum_{z_f \in Z} P(z_f \mid d_i)^2}} \quad (4.32)$$

Algorithm 5: SimilarityMatching

1. **SimilarityMatching** (SM, q, μ_s) {
2. **Input:** *services matrix(SM); q*: query; μ_s : *similarity _threshold*
3. **Output:** *Matched Services, MC;*
4. MC $\leftarrow \phi$
5. **Begin**
6. SC $\leftarrow$ CategorizingServices(SM, K)
7. /* add new query to model */
8. P (z | q) $\leftarrow$ *fold _in _query*()
9. $sc_m \leftarrow$ find_ matched_ category for query
10. **for** *each service* sm_i *in MatchedCategory* sc_m **do**
11. $S_{sm_i\_QoS} = calculate\_S_{sm_i\_QoS}$
12. /*compute similarity using formula (10)*/
13. $sm_{i\_score} = calculate\_sim_{PLSA}(sm_i, q)$
14. $sm_i\_FinalScore = S_{sm_i\_QoS} + sm_{i\_score}$
15. **if** $sm_i\_FinalScore > \mu_s$ **then do**
16. $MC \leftarrow MC.append\ (sm_i)$
17. **end if**
18. **end for**
19. **return MC**
20. **End**

4.4.5 Experimental Evaluation

In this section we present our preliminary experiments to evaluate the effectiveness of our clustering semantic approach CPLSA. We first describe the experimental dataset, data processing and the evaluation metric, and then present the experimental results.

Experimental Dataset

Our preliminary experiments were implemented over the real dataset of Web services whose WSDL files can be accessed via [138]. The collection of services includes 424 Web service descriptions covering the 25 categorises such as Zip code finder and weather information, etc. We selected the dataset of Web services for several reasons. Firstly, up to now, there are no extensive datasets of real services available. Secondly, the Web services in the collection are gathered from real-world service sites like SALCentral and XMethods [128], and artificially classified into different categories so that these Web services provide a basis on which testing and comparison can be implemented based on a variety of situations.

For the experimental comparison, we particularly choose four categories: Business, Communications, Converter and Money.

Data Processing

The goal of the data processing is to transform raw Web service information into an appropriate data format suitable for model learning. We extracted keywords from service description, names of operation, etc., and applied commonly used approaches for word processing. One of methods for data processing included word stemming and stopwords removing. The former removes common term suffix while the latter eliminates very frequently used words. In our experiment, we used the Porter stemmer to parse the 320

Web services documents. All these processes were expected to improve the performance of matching Web services. An example of service is shown as following:

Converts between different currencies in the Euro zone

and the Euro.

After extracting the keywords, we obtain a service collection consisting of 320 services which are divided into two data sets: training data and test data.

Performance Measure

In order to evaluate the effectiveness of the proposed approach, we use the standard accuracy, precision and recall to measure overall performance. After training the model with PLSA, a set of hidden semantic variables, each of which indicates a service category is given. With the learned hidden semantic features, a query's or a new service's related category by computing the probability using formula 4.31 was determined. To evaluate the performance of the clustering algorithm, the accuracy is defined by using formula 4.17; and the recall and precision can be defined by using formula 4.15 and formula 4.16.

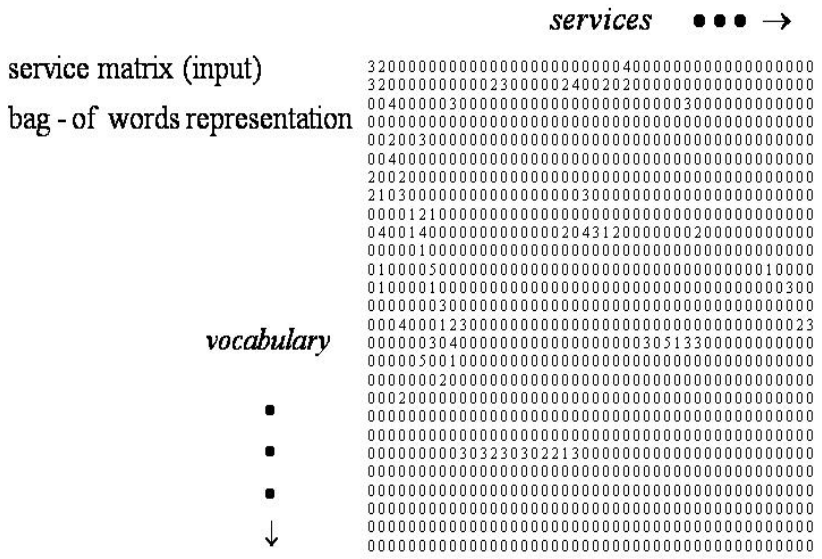

Figure 4.2: An example of data processing

Experimental Results

In this experiment, we evaluated the performance using our semantic matching service approach (CPLSA). We first trained the probabilistic semantic model through setting the different numbers of latent semantic variables ranging from 2 to 20, in order to observe the performance of the retrieval. As data used in this work is artificially classified into different categories, the results were observed in various service categories. In particular, we calculate and compare the recall and precision by selecting four hidden semantic variables, which represent four service categories: Business, Communications, Converter and Money.

In addition, CPLSA performance was investigated by comparing our probabilistic semantic approach with a keyword approach.

The first experiment was implemented over the dataset to observe the performance of probability model learning. Table 4.3 lists the 10 extracted latent aspects and their

corresponding categories. An example of the likely words for four hidden semantic concept is shown in Table 4.4.

Table 4.1: Aspects and their service

Aspect	Service categories
1	Business
2	Web
3	News
4	Money
5	Developers
6	Finder
7	Converter
8	Games
9	Mathematics
10	Country

To see how effective our clustering semantic approach described in the previous section is, we compared the performance of the CPLSA approach with the keyword-based approach. In this experiment, we first fixed the parameter for the experiment, e.g., setting clusters' number to 4. The comparison was made on the top n=10 results retuned by each method, and the experiments were repeated 10 times. The result of the comparison on the accuracy is shown in the Figure 4.11. As can be seen from this figure, the performance of keyword-based technique only based on the text description in Web

services is poor. However, the accuracy of service retrieval is improved when CPLSA approach is introduced.

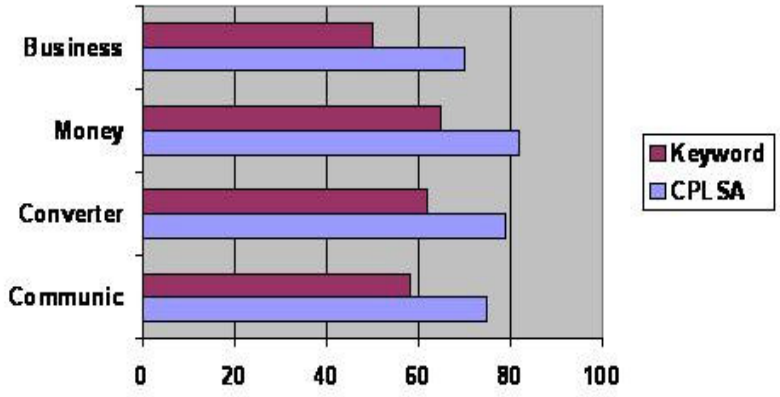

Figure 3: Accuracy of CPLSA and keyword for four categories

The next experiment implemented was to inspect the recall and precision by comparing CPLSA and keyword-based approach. The results are illustrated in Figure 4.12. From the Figure 4.12, we can see that CPLSA performed better than keyword-based approach in the selected four categories. For example, for the query containing "conversion", keyword-based approach may retrieve only those service that include the word conversion, but PLSA-based approach can get services including words "conversion" and "translation".

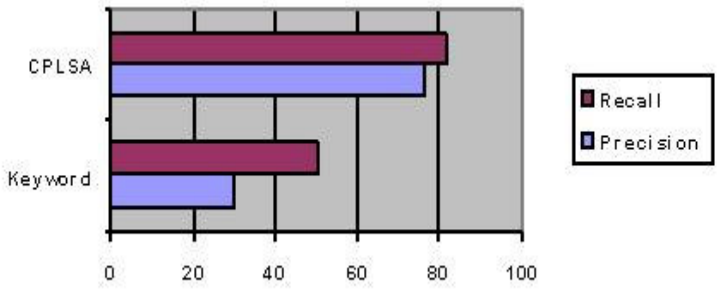

Figure 4.4: Precision and recall of PLSA and keyword

Table 4.4: Examples of the most likely words for 4 hidden concepts

	P(word | aspect)	Most likely words
Aspect 1	0.0835	Information
	0.0683	Address
	0.0612	Telephone
	0.0607	Service
Aspect 2	0.1368	Translate
	0.0879	Convert
	0.0684	State
	0.0586	English
Aspect 3	0.1093	Address
	0.1054	Zip
	0.0828	Place
	0.0753	Name
Aspect 4	0.2052	Service
	0.1758	Web
	0.0892	Fax
	0.0803	Address

4.4.6 Related work

In this section we briefly discuss some of the research work. Clustering approaches are used for discovering Web services [34, 3]. Dong [34] puts forward a clustering approach to search Web services where the search consisted of two main stages. A service user first types keywords into a service search engine, looking for the corresponding services. Then, based on the initial Web services returned, the approach extracts semantic concepts from the natural language descriptions provided in the Web services. In particular, with the help of the co-occurrence of the terms appearing in the inputs and outputs, in the names of the operations and in the descriptions of Web services, the similarity search approach employs the agglomerative clustering algorithm for clustering these terms to the meaningful concepts. Through combination of the original keywords and the concepts extracted from the descriptions in the services, the similarity of two Web services can be compared at the concept level so that the proposed approach improves the precision and recall.

Abramowicz [3] proposes an architecture for Web services filtering and clustering. The service filtering is based on the profiles representing users and application information, which are further described through Web Ontology Language for Services (OWL-S). In order to improve the effectiveness of the filtering process, a clustering analysis is applied to the filtering process by comparing services with related the clusters. The objectives of the proposed matchmaking process are to save execution time, and to improve the refinement of the stored data. Another similar approach [88] concentrates on Web service discovery with OWL-S and clustering technology, which consists of three main steps. The OWL-S is first combined with WSDL to represent service semantics before a

clustering algorithm is used to group the collections of heterogeneous services together. Finally, a user query is matched against the clusters, in order to return the suitable services.

Other approach focuses on service discovery based on a directory where Web services are clustered into the predefined hierarchical business categories. In this situation, the performance of reasonable service discovery relies on both service providers and service requesters having prior knowledge on the service organization schemes.

Our approach CPLSA has similarities to approaches [34, 3, 88] in that keywords are used to first retrieve Web services, and extract semantic concepts from the natural language descriptions in the Web services. However, our work differs from these works in several ways. Firstly, we eliminate irrelevant service via exploiting a clustering algorithm to diminish the size of services returned; this approach shows some potential applications like over mobile uses. Secondly, based on the characteristics of Web services with a very limited amount of information, we regard the extraction of semantic concepts from service description as a problem of dealing with missing data. Therefore, we utilize Probabilistic Latent Semantic Analysis (PLSA) [50, 51], a machine learning method, to capture the semantic concept hidden behind the words in a query and the advertisements in services.

4.5 Web service Composition Based on Ontology

In this section, we study the issue of composing Web services with combination of Ontology, Web services and agent technology. We present a goal-driven and ontology-based architecture in which (1) user's goal is decomposed to subgoals; (2) the

information in the goal and Web services are annotated with domain specific ontology; (3) AI technology and theory of reasoning about action are used to compose Web services. We also present a composing algorithm to show an application.

An overview of Ontology-Based System Architecture

In our Ontology-Based Model for Web services Composition (OMWSC), we incorporate ontology, Web services and agent technologies. Ontology is a kind of approach for knowledge representation, which is used to describe the information in the goal and annotate Web services. An agent is a software component and works as information broker. It takes instruction from user's goal and performs some tasks on the behalf of the owner of the goal.

The OMWSC architecture shown in Figure 4.13 consists of four modules. The first module Goal captures user's objectives. A user's goals usually express a user's intention that s/he expects to get the results from the system. The second module consists of a group of agents. If an agent is assigned a goal, it will analyze information in the goal and perform tasks on the behalf of owner of the goal. For example, an agent, according to the intention and objectives of user or other components, may look for corresponding service in service databases. The next module is a service database (SD) that stores Web services. And these services are annotated with OWL-S ontology and provide a solution for implementing a task. The fourth module is a composition component. For the automated generation of the composed Web services, The OMWSC uses the AI planning approach. This approach is based on Theory of reasoning about action to produce a plan of composing services.

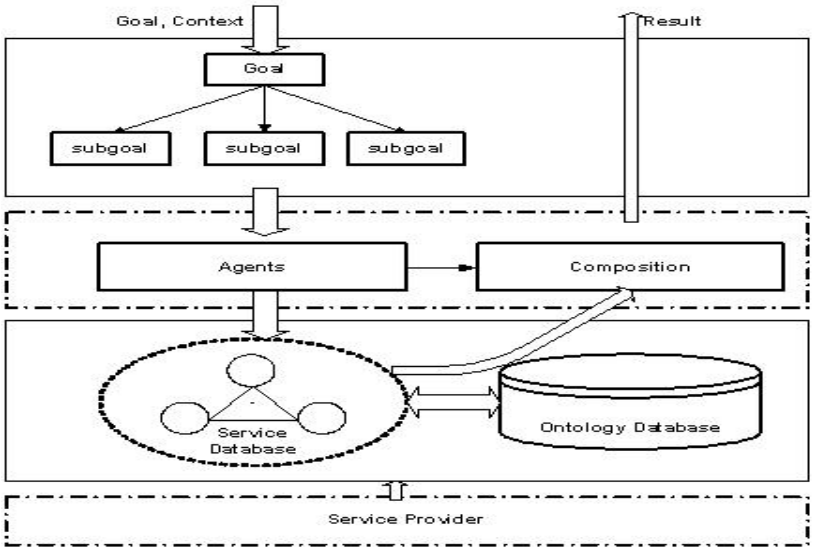

Figure 4.5: Ontology0based system architecture

Goal

A goal expresses user's desires. For instance, Peter wants to rent a car. In order to effectively realize a goal, the goal is usually decomposed to some subgoals because such smaller goals require fewer service resources and less power agents and it is also easy to implement them. In a practical application, this decomposition of goal can be expressed and organized as an AND-OR graph. In the graph, AND means that realizing all subgoals of a goal is a condition for realizing the goal. On the other hand, OR indicates that realizing one of subgoals is a condition for realizing the goal.

The information in goal represents functionalities and non-functionalities that a particular application domain offers. For achieving a goal, an agent needs to make use of this kind of information to link the goal to the corresponding service resources. For this purpose, the information in the goal is needed to be described semantically. In this

model, we use ontology to describe information and Web services. The information in goal is semantically described by the following elements and properties:

- *Ability* refers to the functionality provided by the information in goal
- *Non-functional property* refers to the quality of service, location, title , etc
- *Precondition* refers to permission for a goal to achieve the goal
- *Postcondition* refers to some state and condition to be satisfied to achieve the goal

4.5.1 Web service Representation

We use OWL-S to describe Web services. The upper level Ontology at the top in the Figure 4.14 includes the three modules: (1) a *serviceProfile* that provides functional attributes and a specification of functionalities for a service (input/output, effect and parameter name, for example). An agent can make use of this information to discover a service; (2) a *serviceModel* that represents a service's internal structure. A service's behaviors can be described in terms of its process model. Further, the instance of the model can be used for invoking, matching and composing services; (3) a *serviceGround* that handles the issues of accessing Web services such as accessing protocol and message formats etc.

Our domain ontology derives from the upper ontology, therefore it inherits features, classes from the upper level ontology. As the concepts from a particular domain and relations among them are precisely specified, it enables reasoning on service matching and service composition. In particular, we assume that service users and service providers have shared common understanding by using ontology so that it facilitates the exchange of knowledge. In our model, we identify following service properties:

- *Input* refers to requirements needed to get expected result, also refers to concept denoted by ontology
- *Output* refers to possible result and concept denoted by ontology
- *Precondition* refers to condition for invoking a service
- *Effect* refers to impact resulting from executing a service
- *Service type* refers to taxonomy of service

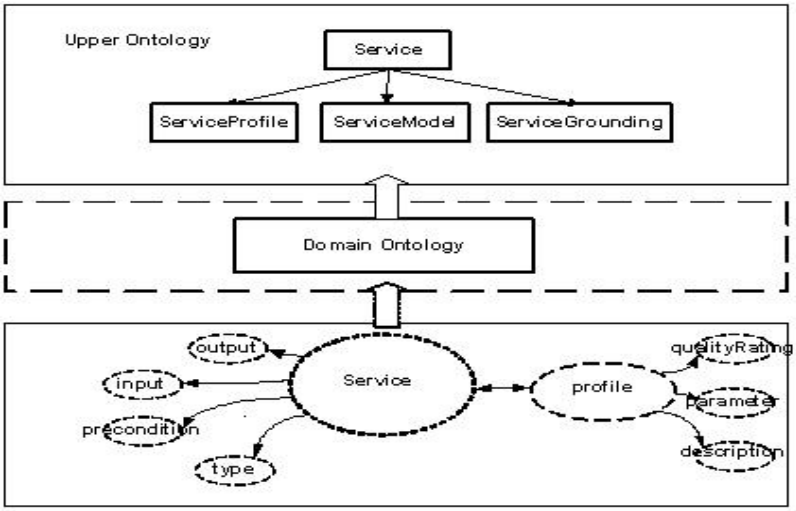

Figure 4.6: Ontology for Web services

In this section, we'll present a scenario that shows an example of Web service application and then briefly discuss ontology design's approach.

As a working example, we create a virtual shipping company CargoShip like a travel agency. It provides clients with shipping services: sending client's cargos from one place to another one. In reality, shipping services may occur locally, for example, using truck or train to send cargo from Brisbane to Melbourne or may occur internationally, sending cargo by sea or by air from Australia to Singapore. When a client makes a query to CargoShip for transferring his or her cargo, the CargoShip accepts the client's

requirements, processes the related information and sends the cargo to the destination for the client if everything is ok. But CargoShip does not own its truck, train, and shipping facilities; therefore it has to outsource other Web services such as LandShip, SeaShip, Payment and Delivery, etc. First, we introduce ontology design.

4.5.2 Ontology Design

We construct a framework by creating an ontology model for the scenario so that agent is able to reason.

Design Method of Cargoship Ontology

A domain ontology usually includes related classes, properties, relations between entities and inference rules, etc. In order to develop ontology, one can define the ontology of a particular application field or reuse existing ontologies. Currently, there are different kinds of methods of developing ontology. One is to borrow the idea from developing class used in Object-Oriented technology. In developing classes in Object-Oriented technology, the nouns can be used to identify the classes in an application field; the descriptive words can be used to identify the class's attributes; in addition the verbs can be used to represent class's behaviors or operations. In defining ontology of a domain, one concerns more about classes' structure and relations between the classes. Although there are different methodologies in developing ontology, following objectives are consensus: (1) in the end, the instances of ontology of a domain forms a knowledge base where other services and agents can know something; (2) ontology represents the practical application of a domain.

Cargoship Ontology

Cargoship ontology denotes our application domain's concepts and their relations in a machine-understandable way. These concepts are used to describe different types of services in the application.

Figure 15 shows an ontology model of cargo shipping. Every rectangle in the Figure denotes a class. The Cargoship is a top class that contains a set of properties and subclasses. Customer, Location, Weight, Price, and Constraints represent Cargoship's properties while Action, ByLand, BySea and ByAir are subclasses that derive from class Cargoship. These subclasses can also have their own properties and operations. For example, class BySea may have its information about seaport's URL. Cargoship's properties not only express class's common attributes such as customer, but also denote relations between the concepts:

- *Customer, Location, Weight, and Price*: they describe common properties of class Cargoship. These properties' instances can be used to denote concrete service information about Cargoship. For instance, a customer, peter, may have Australia citizenship or a concrete price may be paid with US dollars, etc.
- *Constraint:* this property is used to describe constraints and relationships between the concepts. It has three subclasses: ObjectCon, SubjetCon and ActionCon. ObjectCon denotes the Objective constraints for a Web service such as QoS used for selecting a service. SubjectCon denotes the subjective constraints for a Web service. For example, a constraint may denote client's specific interest or client's location. ActionCon describes constraints and relations between operations. For example, Operation getPayment() must be executed before operation delivery () is executed.

With these relations, an agent is able to reason, for example, if executing action Payment () is successful, the agent could take action Delivery ().

Context information plays an important role in selecting a service. These constraints are defined as *Subjective* constraints. Therefore, in our application, the process of selecting services involves two steps:

- Step 1: calculate service's objective constraint.
- Step 2: match service's subjective constraint.

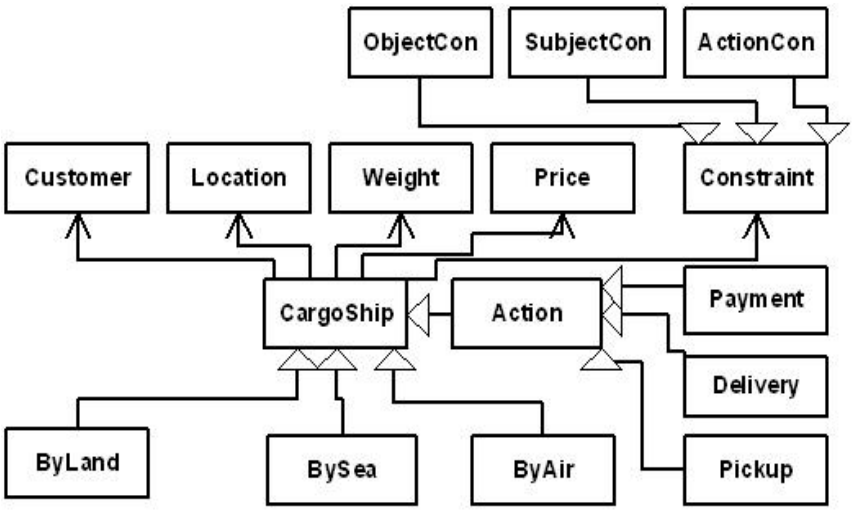

Figure 4.7: Cargoshipping ontology

Definition 4.3. A Web service's Quality of Service (QoS) is defined as the service's Objective constraint.

◊

For example, Let $WS = \{ws_1, ws_2, ..., ws_n\}$ be a set of Web services and $OC = \{oc_1, oc_2, ..., oc_n\}$ be a set of Web services *Objective* constraints. Where oc_i denotes Web services ws_i's QoS such as response time, cost, reliability, security and

availability, etc. According to these constraints, an agent can roughly determine whether a service meets its needs.

In our case, we only consider three QoS parameters: response time, service availability and service reliability:

Response time (RT): the amount of time required to send a request and get the result of the service.

Service availability (SA): the probability that the service is available.

Service reliability (SR): the probability that a request is correctly responded.

According to these QoS parameters, we can use following formula to find a suitable service:

$$S_{QoS} = w_1 * SA + w_2 * SR + w_3 * (1 - \frac{RT}{M_t}) \quad (4.33)$$

$$w_1 + w_2 + w_3 = 1 \quad (4.34)$$

Where

SA: Service availability

SR: Service reliability

RT: Response time

M_t: Maximum response time

w_1, w_2, w_3: Weights. Their values reflect the importance or priority for corresponding parameter. For example, if response time is more important than service availability and service reliability, it has a greater value. According to formula (4.33), an agent can roughly determine whether a service meets its needs and then it checks a service's subjective constraints.

Definition 4.4. User's current situation (user's interests, user's location, etc, for example) is defined as a Web service's Subjective constraints.

◊

Table 4.2: Two services QoS parameters

WS	SA/w1	SR/w2	RT/w3	S_{QoS}
ws1	0.72/0.25	0.95/0.25	0.615/0.5	0.61
ws2	0.9/0.25	0.85/0.25	0.427/0.5	0.724

4.5.3 Service Selection

Service selection should comply with the standard of service's QoS. We define QoS as *Objective* constraint for selecting a service because the service provider stipulates the constraint. We also believe that a client's interests and context information play an important role in selecting a service. These constraints are defined as *Subjective* constraints. Therefore, in our application, the process of selecting services involves two steps:

- Step 1: calculate service's objective constraint.
- Step 2: match service's subjective constraint.

Definition 4.5. A Web service's Quality of Service (QoS) is defined as the service's Objective constrain*t.*

◊

For example, Let $WS = \{ws_1, ws_2, ..., ws_n\}$ be a set of Web services and $SC = \{sc_1, sc_2, ...sc_n\}$ be a set of the Web services *Subjective* constraints. Where sc_i denotes a user's interest for Web service ws_i or user's context characteristics such as location information.

Thus, a selected service must satisfy its objective constraints and subjective constraints.

Example 1. In this example, we assume that two services, ws_1 and ws_2, have similar functionalities. In addition, ws_1 and ws_2's QoS parameters are showed in Table 4.5. For ease of explanation, let weights w_1=0.25, w_2=0.25, w_3=0.5 and maximum response time M_t=1. According to Formula 4.33, for ws_1, we have S_{QoS} = 0.61; for ws_2, S_{QoS}=0.72. Therefore, an agent roughly selects service ws_2 because it has a greater value of S_{QoS}.

4.5.4 Web service Composition Algorithm

The goal of composing service algorithm is to produce a plan of executing actions based on user's goal and a set of services available. This can be done in following steps.

First of all, user's goal and a set of Web services are available to the system. In our application, user's requirement is expressed as the goal of the composed Web service and a component service is described as an action or program [1, 84, 85]. According to the theory of the situation calculus [103], each action has its preconditions and possible effects resulting from executing the action. The second step will find a group of matched services for a goal. For each goal, agent will search for appropriate services in database for matching. If the goal's precondition is compatible to the precondition provided in a service, and the goal's other properties satisfy corresponding properties in the service,

the match is successful. At the next step, the agent will select a matched service that satisfies its standard of quality of services.

For simplicity, we make following assumptions:

- We consider a limited set of component services and also assume that these limited services are available.
- A component service will provide a single functionality and the number of services that may provide similar functionality is also limited.

Algorithm 6: **Compositing Services**

```
1.  ServiceComposition (Goal, WS )
2.  input: a set of goals;Goal = { g_1, g_2,..., g_n };a set of services;Service= { s_1, s_2,..., s_m }
3.  output: plan, a sequence of actions
4.  begin
5.      /*match goal with services */
6.      for each g in Goal do
7.          for each s in Service do
8.              if match_precondition(g, s ) ∧ match_postcondition(g, s ) ∧
9.                 match_postcondition(g, s ) ∧ match_ability(g, s ) ∧
10.                match_precondition(g, s ) then
11.                GoalMatchedService.append( s )
12.             end if
13.         end for
14.         /* select service for goal g */
15.         if GalMatchedService.number !=0 then
16.             for each service in GolmatchedServic do
17.                 QoS_i = calculate_QoS ( service )
18.                 serviceQoS.append (QoS_i )
19.             end for
20.             servicSelected = getService(Max(serviceQoS))
21.             plan= plan.append(servicSelected)
22.         end if
23.         Goal = Goal.remove(g)
24.         until Goal = φ
25.     end for
26.     return plan
27. end
```

4.5.5 Related Work

Web services composition has been one of hot research topics these days. There are a number of works related to it. Some recent researches deal with composition of Web services with automatic program synthesis and others use software agents to compose Web services through auctions and delegation. There are also works that are based on AI planning, reasoning about actions in AI [84, 85] and using ConGolog to compose Web services. All these will be dealt with in turn as follows.

[86] addressed the composition of Web services by applying the software synthesis's methods. The method is based on the idea that a Web service can be represented as a software component and Web services composition is similar to software synthesis through using the software component's pre-/post-condition and reasoning methods. If there is a service's specification in the repository of Web services that matches the requirements, the service is selected. Finally, a Structural Synthesis Program (SSP) is used to build a new composite service. In short, the SSP-based synthesizer makes a plan of executing these single services. However, this approach relies heavily on SSP that is often not available in the contexts we use.

[87] has focused on a software agent-based approach that supports Web services composition. In [87], there are two types of agents: user-agent and provider-agent. User-agents act on behalf of users and provider-agents act on behalf of Web service providers. Selection of a component service is based on two criteria: execution cost and location of computing hosts. The process of the composition of Web services can be divided into two phases: firstly searching for provider-agent and secondly selecting a provider-agent. It also presents an algorithm whose purpose is to produce a short list of provider-agents. What's more, user-agent can delegate a part of compositing work to a third party- a

delegate agent so that composition of services could be executed concurrently. Nevertheless, this approach, selecting a component service based on its domain, is inappropriate in our context because we must consider a service's QoS.

Turning now to works that are based on planning and reasoning about actions in AI. [85] addressed the issue of automated Web services composition and used ConGolog [26] as a natural formalism for this problem. In [86], the process of composing Web services is regarded as producing a plan of actions and executing a service is regarded as executing an action. Every action is a program that has input and output parameters and these parameters act as precondition and effects in reasoning about actions.

4.6 Summary

Web services discovery and composition are playing an important role over the Internet applications. It is a challenging work to effectively find the desired Web services that conceptually match a user's needs. In this chapter, we first studied two main problems related to the keyword-based search approach: poor scalability and lacking semantics. To overcome the problems, we used Divide and Conquer semantic approach (DC-SVD) to handle the issues of poor scalability and lacking of semantics. A large service collection is first partitioned into a set of smaller clusters by using a modified k-means clustering algorithm. After finding the right cluster that is relevant to a query, the SVD technique is applied to the cluster so that service matching against the query can be carried out at the concept level.

We also performed several experiments over the real datasets to evaluate the effectiveness of the proposed approach. The results show that our approaches improve over precision.

In this chapter, we also studied the issue of Web services composition with combination of Web service, Ontology and agent. Using the theory of reasoning about action, we represented a Web service as an action that has precondition and the effects resulting from executing the action so that we are able to compose Web services with agent technology. In addition, we presented a novel notion that a selected service must satisfy its objective constraints and subjective constraints. Finally, these notion and approaches introduced in this chapter can be used to compose Web services.

5. Chapter 5

Secure Collaboration in Scientific Workflows

In the Chapter 2, we introduced a scenario of doing business through a collaborative approach, which can provide a wealth of additional opportunities with trading partners such as reducing cost of transaction, adding value to their products and services and enhancing market access, and so on. At the same time, the information exchange among collaborative partners in a distributed context can give rise to some potential security problems to be aware of.

Confidentiality in a business collaboration among trading partners can become an issue. As discussed in the previous chapter, one of a virtual organization's objectives is to share valuable information and resources among the trading partners within the organization. More precisely, this information sharing means that one participant's confidential information like its business process, private data and technique strength might be provided to its competing business partners, which may potentially weaken the participant's competing power. For this reason, each partner in a virtual organization may have its own security mechanisms to protect its valuable resources. However, these protecting mechanisms may impede the effective collaborations among trading partners.

Therefore, it is important to effectively manage the collaborative conflict as well as to maintain the confidentiality of each trading partner.

In this chapter, we present an approach to support secure collaboration. Our approach is based on the concept views, which place emphasis on information hiding, minimizing what partners need to know during their collaboration with each other. We also propose a novel two layered access control model based on the general principles of the order of security priority in an organization. With the access model and process views, various roles can also be associated with views, so that users can only see necessary parts of workflows that they are entitled to see.

This chapter is organized as follows. Section 5.1 presents introduction and Section 5.2 introduces security requirements of cross-organizational collaboration. Section 5.3 derives a consistency view based on a workflow model. Section 5.4 gives a two layer access control model for secure collaboration. Section 5.5 discusses related work, and Section 5.6 summarizes the work introduced in this chapter.

5.1 Introduction

Over the past decade, workflow techniques have been successfully developed to automate business processes, which support a variety of business domains, such as enterprise production management and e-business integration. Additionally, workflow techniques have been increasingly employed in scientific communities where, scientists make discoveries by conducting complex sets of scientific computations and data analyses. These scientific explorative activities in biology, chemistry engineering, geosciences, medicine and physics [29, 54, 90, 115] are typically carried out at multiple

sites, run over long periods of time and involve multidisciplinary processes and huge amounts of data. For example, a typical biological experimental scenario may consist of hundreds of computational steps; each step may be distributed over the Web by taking input data from various databases and disseminating the results obtained from one step to multiple downstream steps in a distributed environment. In such data-oriented application systems, workflows are often referred to as scientific workflows. Scientific workflows assist scientists in their research by orchestrating compute-intensive tasks, analyzing large data sets, as well as, integrating distributed computing processes.

As an example of a scientific workflow, consider lymphoma DNA classification research scenario as shown in Fig. 5.1(a), which has been adapted from [72]. The objectives of this experiment are to select valid gene data from differing databases for the use of scientists. During these experiments, DNA samples are first collected from the Gene Collection Laboratory before being sent to the Data Validation Centre for data processing. The processed DNA samples, matching anticipated lymphoma symptoms, are considered as valid data and in turn are sent to the Lymphoma Research Lab for further classification, based on specific gene characteristics. Otherwise, the results of the classification are assumed to be negative, which may result in the process of DNA classification analysis being rolled back for a new round of experiment.

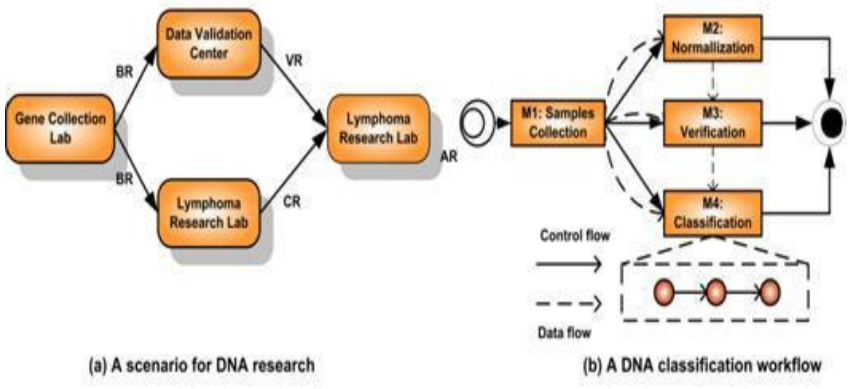

Figure 5.1: Scenario of lymphoma DNA classification

With advances in Web technologies, scientific research like the above DNA analysis is increasingly the result of collaborative efforts among scientists who make use of each other's experimental results. In these situations, data resources (e.g., DNA databases and medical testing data), individual computational tasks (e.g., gene comparison) and scientist-created workflows (e.g., experimental steps) are likely to be distributed over various locations, connected over the Internet. In addition, these distributed resources can be further wrapped as Web services, which are published and deployed on the Web, so that other scientists are able to share these valuable computational resources for facilitating their scientific researches. These collaborations in a distributed environment allow scientists to transform a complicated scientific research problem into a series of simpler, and more easily handled steps. Examples of scientific workflow management systems that have developed to support scientists for tackling complex scientific computing problems, include work done by Taverna [90], Triana [109], Pegasus [31] and Kepler [67].

The second challenge is security, which is currently ignored in most scientific workflows systems. This is particularly important in a Web services context because the messages transmitted over the Web could be accessed and altered by imposters. For security reasons, it is important that confidential information like private processes and sensitive data (e.g. patients' private data in medical research) should not be visible or released to unauthorized users. Also selecting the relevant information for specific scientific experiments can be problematic. Typically, scientific experiments are characterized by their exploratory nature, that is, scientists conduct experiments in a trial-modification manner, by which procedures of experiments may be often modified, whilst tasks may need to be replaced with alternative ones. As a result, a lot of provenance information about the creation of data objects, such as program version number, data resources and related computing steps, could be produced, which can make it difficult for scientists to browse, look at and choose the proper computing services for their experiments. It is therefore desirable to provide an effective mechanism to enable authorized users to see and to access the relevant part of a scientific workflow.

One such effective mechanism for balancing security and information choice for users is the concept of a view. Views represent an abstraction of original workflow specifications by defining parts of the specifications, which are only visible and accessible to authorized users. Like the notion of the views in databases[40], a view in a business process or scientific workflow places emphasis on information hiding, minimizing what collaborative partners need to know during their collaboration with each other[7, 61, 62, 72, 115]. Various roles can also be associated with views, so that users can only see necessary parts of workflows that they are entitled to see. Furthermore, only the relevant

data products and provenance information need to be provided to users based on their different requirements in cooperative scientific researches.

In this chapter, we address the challenges mentioned earlier by presenting an access control framework and models for supporting reliable collaboration. The proposed approaches depend on describing scientific workflows by integrating control flow and data flow models, and they employ a relaxed transaction concept for fault tolerance, in order to maintain the process consistencies, as well as, data consistencies even in the presence of failures. To achieve these objectives, we first model the internal scientific activities within an institution. Then, the individual workflow models of participating institutions are mapped to views, which are further described by Web services. Based on the view model, a two layered access control architecture is proposed to protect computing resources. Finally, the atomicity concept is relaxed by integrating transactions and exception handling models in order to ensure the reliable collaboration on the Web.

5.2 Security Requirements of Cross-Organizational Collaboration

A business collaborative activity often operates in a highly dynamic and open distributed environment as new participants may become available at any time, and existing partners may be unavailable due to different reasons.

The security is obviously an important issue for any organization and for any serious application of system. Within the scope of computer system or more precisely within the scope of business transactions executing over the Internet, the security issue looms to be even more critical because the messages transmitted over the Web could be altered by

imposters; confidential information and valuable Web resources in an organization might be accessed by unauthorized users, etc. In order to deal with these threats, the basic strategies are based on the general principles: the classification and needs-to-know. For example, approaches commonly used to handle the security issues include authentication, authorization and encryption. This methodology can effectively establish the identity of a participant, determine the accessing mode by which a client is allowed to access and protect the information from being accessed by unauthorized users.

Nevertheless, the mechanisms of Web Service-based transactions put forward further requirements for security. First, due to massive interconnection of heterogeneous Web Services in the distributed environment, it is necessary to establish a mutual level of trust (security) to before being able to interact with each other. Second, in Web Service-based distributed applications where invokers are not known, access control decisions are usually made based on the attributes of the entity requesting access to a particular resource, rather than its identity, so it is impractical to request a user login name and some other credentials. In addition, as Web Services are designed as one of computing paradigm by which a service application may drill through the network-level (e.g. an enterprise's firewall), a client may access the confidential and valuable information assets in an organization. As a result, securing the business interactions among the business partners at the application-level is important.

In this chapter, we focus on the effective mechanism for balancing security and information choice for users by employing the concept of views. Based on the view model, the individual workflow models of participating institutions introduced in the previous chapter are mapped to views, which are further described by Web services.

Finally, a two layered access control architecture is proposed to protect computing resources.

The main features of our framework are:

- Balancing security and information choice. We propose a view-based security model for balancing security and information choice. Views represent an abstraction of the original business process specifications by redefining parts of the specifications. With the view-based security model, various roles are associated with views, so that right users can only see necessary parts of workflows exposed to them.
- Layered security architecture. We propose a two layer security architecture, which is based on the general principles of the order of securing priority in an organization. According to the principle, the higher securing requirements in the organization specify what needs to be done, which includes the trust objectives of the organization and the security of the whole system' external interface to the environment. On the other hand, the security requirements within the components internal to a system can be put at the second level.

5.3 Deriving Consistency View Based On Workflow Model

Based on the basic models introduced in the previous section, in this section, we describe reliable collaboration across organisations based on a view model. We start with deriving a view based on the workflow-based model presented in Section 3.4.5, and then describing an access control model. After introducing the notion of atomicity sphere, we focus on maintaining data consistency and process consistency for reliable collaboration across organisations.

5.3.1 Consistency Views

This section considers deriving a consistency view from a computing workflow model. We first summarize ordering reservation rules proposed in [61, 62, 72], and then derive a consistency view by using a projection approach.

Views represent an abstraction of the original workflow specifications by redefining parts of the specifications. Let a workflow specification be $G = (V, E, f)$, where $V = V_1 \cup V_2 \cup \ldots \cup V_n$ is the set of nodes, then views deduced from the specification *G* can be represented as $G^v = (V^v, E^v, f^v)$. In G^v, nodes V^v indicate the parts of nodes in V, that is, $V^v = V_1 \cup V_2 \cup \ldots \cup V_m$, where $m \leq n$ and f^v maps each node onto a new unique label.

Views can be used to improve trust and security in collaboration among scientists in distributed environment. This is because a view hides the internal private process within an organization from other partners. Like the notion of the views in a database, a view places emphasis on information hiding, minimizing what collaborative applications need to know during coordinating with other computing processes. With views, the information only necessary for process enactment of the collaboration can be made available to both partners, in a fully controlled manner. For example, during design time, a workflow designer can determine various views based on the roles of participants by exposing the information that those participants only need to know, whilst concealing the core information within an institution. As a result, different views of a workflow can be presented to individual scientists or different organizations according to their specific requirements.

We now consider the consistency of the views. Intuitively, for a computing process, a view derived from it should be coherent in both structure and semantics. Basically, the following requirements should be satisfied:

- If a virtual task t_v^1 precedes another virtual task t_v^2 in a process view, then the actual task t_a^1 contained by t_v^1 also precedes t_a^2 contained by t_v^2 in an actual process. That is, a process-view preserves the original structure ordering relations of an actual process.
- If a virtual task t_v^1 begins to run, then at least one of actual tasks contained by t_v^1 has started. This means that the behavior of a virtual task in a view is dependent on that of actual tasks in workflow.

Formally, we let $T_V = \{ t_v^1, t_v^2, ..., t_v^m \}$ be the set of all virtual tasks in the view, and let $T_A = \{ t_a^1, t_a^2, ..., t_a^n \}$ be the set of all actual tasks in a workflow model. In some cases, a virtual task may include actual tasks as well as other virtual tasks, thus we represent all tasks contained by a virtual task $t_v^i = (\bigcup_{j \in n}^{i} t_a^j) \bigcup (t_v^h)$, where $h \in m$, *and* $h \neq i$. As a virtual task t_v^i is determined by its contained actual task t_a^j, we say t_v^i is dependent on t_a^j, denoted as $t_a^j \rightarrow t_p^i$. In the following definition, we use symbol $\prec$ to express the ordering relation between two tasks.

Definition 5.1 Given two virtual tasks t_v^1 and t_v^2, we say that t_v^1 is before t_v^2, denoted as $t_v^1 \prec t_v^2$, if and only if $\forall t_a^x \in t_v^1 \; \exists t_a^y \in t_v^2$ such that $t_a^x \prec t_a^y$, $x \neq y$.

◊

Next, we derive a consistency view by using a projection approach. In a relational database, a projection operation is used to perform a vertical decomposition of the input relations in such a way that each tuple in a table becomes part of the output, but only the

attributes of related selections are preserved. Similarly, the proposed projection approach can be used to derive views. The projection method will take as input a workflow-based specification and produce as output a view. In particular, the specification of workflow-based process is specified in a path expression. Thus, we first define a path expression.

Definition 5.2 (Path) Let V be the set of control connectors of a process model $P, v_1,...,v_n \in V$, and let $T_A = \{t_a^1, t_a^2,...,t_a^n\}$ be the set of all actual tasks. The sequence $v_1,...,v_n$ is called a path starting from the beginning task $t_a^1 \in T_A$ to the ending task $t_a^n \in T_A$.

Definition 5.3 (Deriving consistency view). Given a process model $P=(P_1, P_2,..., P_n)$ and a path expression V, a view $PV=(PV_1, PV_2,..., PV_m)$ is a division of P and the projection operator $\pi_V(P)$ will return a view $PV = PV_1 \cup PV_2 \cup ... \cup PV_m$.

Fig. 5.2 shows an example of views where a back-end process consists of five tasks, starting from t_a^1 to t_a^5; a path expression can be denoted as $\mu = v_1 v_2 v_3 v_4$; and the process model is $P=t_a^1 \times t_a^2 \times t_a^3 \times t_a^4 \times t_a^5$. Thus, virtual task t_v^1 can be described as: $\pi_{1,2}(P)= t_a^1 \times t_a^2$, and virtual task t_v^2 is shown as $\pi_{3,4,5}(P)= t_a^3 \times t_a^4 \times t_a^5$.

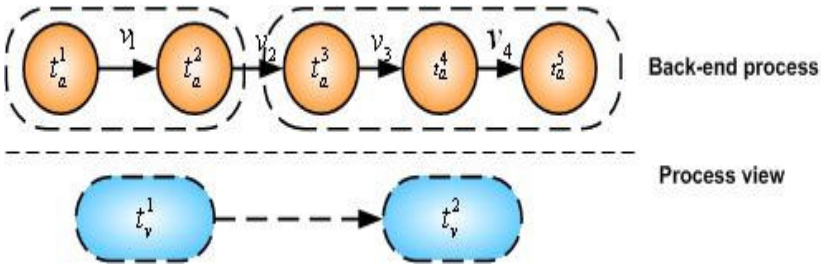

Figure 5.2: An example of deriving views

5.4 A Layered Access Control Model for Secure Collaboration

In this section, we consider the security issue on accessing Web Services. We address this problem by presenting a novel two layered model for securely accessing Web Services. At the level one, the abstraction of security emphasizes on the individual tasks by associating each task with security features through defining security attributes in the definition of a task, so an entity such as a user or a Web Service implementing a task must satisfy the security requirement. Second, at the level two, the abstraction of security focuses on the individual Web Services. At this level, the invocation of each Web Service operation is controlled by access control policies, which will verify what credentials an entity or an invoker Web Service must posses in order to be able to invoke the operation. Furthermore, we model behavior of different candidate Web Services by using a finite transition system where the access control policies of different candidate Web Services are treated as the precondition for the transition system.

5.4.1 Secure Collaborative System

In this section, we will introduce a domain concept and give a definition on secure collaborative system.

In the previous chapter, we have described that an inter-enterprise collaboration adopts a layered collaborative architecture, from high-level conceptual layer to concrete technique operation layer. At the conceptual level, trading partners or entities create a strategic collaboration relationship by forming a virtual organization according to a value chain model. Based on the value chain, each partner within the virtual organization

complete its own tasks through providing its specific services and products, such as some partner providing loans service whilst the other with insurance, and so on. As a result, a high-level business objective can be achieved by collaborating these partners according to the reached high-level agreement. These business partners are often referred to as domains.

Although the different domains in a virtual organization agree to a common business agreement, share a common objective and are willing to burden some risks, they are still allowed to implement the agreement individually with their own ways. For example, each participant within a virtual organization may employ its business processes, security mechanisms and corresponding roles as an independent entity. For this reason, we present the definition of a domain shown as follow.

Definition 5.4 (Domain) A domains denotes an entity with specific capabilities to complete some certain tasks. A domain consists of a set of processes, security policies and corresponding roles. A domain D is described as a tuple $D=\langle BP, SP, R\rangle$, where $BP=\{bp_1, bp_2, ..., bp_n\}$is a set of business processes; $SP=\{sp_1, sp_2, ..., sp_m\}$ for security policies; and $R=\{r_1, r_2, ..., r_k\}$ for corresponding roles.

Based on the definition 5.4, we are able to define a secure collaborative system in a distributed collaboration environment, adapted from [44].

Definition 5.5 (Secure Collaborative System). A secure collaborative system is a simple graph G = (V, E), where V is a set of nodes, denoting domains and E is a set of edges, denoting a binary relation access on E. *V (G)* and *E (G)* represent the set of all nodes and the set of all edges in G respectively.

In addition, a binary relation is described as follow [44]:

Definition 5.6 (Permitted Access) Permitted access is a binary relation F on $\bigcup_{i=1}^{n} V_i$ where $\forall(u,v) \in F \ , u \in V_i, v \in V_j, and \ i \neq j$

Definition 5.7 (Restricted Access) Restricted access is a binary relation R on $\bigcup_{i=1}^{n} V_i$ such that $\forall(u,v) \in R \ , u \in V_i, v \in V_j, and \ i \neq j$

5.4.2 A Two Layer Security Architecture

Our layered security architecture is based on the general principles of the order of securing priority in an organization. According to the principle, the higher securing requirements in the organization specify what needs to be done, which includes the trust objectives of the organization and the security of the whole system' external interface to the environment. On the other hand, the security requirements within the components internal to a system can be put at the second level.

Generally speaking, service-based business collaboration involves the interactions among collaborating services, service access, and message handling. Therefore security policy needs to cover on these three aspects accordingly:

- At message level, access rules are specified with a proper indexing schema, which will be associated with service operations. Such access control model can support read and write privileges. The model includes various access types, user's right to change XML structure and access rules for different users. Finally, this mechanism can also support dynamic XML documents generation, which means that it is unnecessary to re-label even if the XML document needs to be updated.

- At service level, we will address the design issues of secure Web services. In role-based access control, users are assigned roles, and roles are associated with permissions or sets of operations. In the service oriented computing environments, user's access data or perform tasks via services invocations. Each service is associated with a number of operations on data elements.
- At service collaboration level: individual service may have its own authorization requirement. What is more, a coordinating service may need to exchange policy and credential information as well as managing the operation details. To deal with these issues, we consider solutions to realize reliable and secure collaboration, which mainly includes:
- Efficient description of security policy for a Web service. This description includes service security capability and security constraints. Security capability describes the security features of a Web service such as name of the service requestor, a set of credentials, or a set of particular parameters required to invoke the service or role performed by the service requestor. On the other hand, security constraints refer to a set of conditions that a Web service could impose on another Web service in order to cooperate with it. Based on these descriptions, we develop a method to check the security constraints of the individual Web service to determine whether they are compatible to the specified security requirements.

Our layered security architecture is based on the general principles of the order of securing priority in an organization. According to the principle, the higher securing requirements in the organization specify what needs to be done, which includes the trust objectives of the organization and the security of the whole system' external interface to

the environment. On the other hand, the security requirements within the components internal to a system can be put at the second level. To efficiently handle the issue of security in Web service transactional environment, we present a layered access control model, which consists of various levels of information such as *Level 1, Level 2, Level m*, where *Level i* is more sensitive or more restrictive than *Level (i-1).* In our layered securing model for Web service transaction management, the security requirement of what needs to be provided to perform a task within the business process will be captured at the level 1, the highest level. Once passing the security requirement at the task level, an entity must go to the next security level to be able to invoke concrete Web Service resources. Based on this information, we are able to select Web Services that meet the requirements of the security.

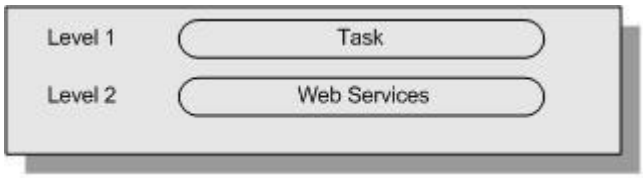

Figure 5.3: Model of a layered access control

We extend the Role-Based Access Control (RBAC) [102] to securely access Web Services. Generally speaking, access control refers to a process by which valuable information resource in an organization is not used in an unauthorized way. Pre-requirements for the access control are to identify a potential user of resource. RBAC goes further by introducing roles and permissions concept. A role describes a job function being performed in an organization, that is, it represents the access rights that could be assigned users. Permission, on the other hand, indicates whether the access rights can be granted to protected objects or not. By using role hierarchies and

constraints, a wide range of security policies can be expressed such as by using Discretionary Access Control (DAC), Mandatory Access Control (MAC), etc. In this thesis, roles and permissions are used for both Web services and workflow tasks so that an entity with the roles related to a task can be authorized to execute the task.

In order to implement a task, an entity or a user must be assigned at least one role. We first give the following the definitions of role, which has been adapted from [102].

Definition 5.8 (Role) A role is job function performed in an organization and can be described as a tuple r = (R, P, D), where R is the role name, P is a set of permission related the role and D is domain associated to the role.

◊

Roles are normally arranged in the form of hierarchies, which reflects the information about architecture, function and responsibility within an organization. From the mathematics point of view, a role hierarchy refers to a partial order relationship established among roles. A partial order is a reflexive, transitive, and antisymmetric relation.

Definition 5.9 (Role Hierarchy) Given a set of roles $R = \{r_1, r_2, ..., r_n\}$, a role hierarchy $\leq$ is defined as the partial order relation between roles r_i and r_j $:= < R, \leq >$, if and only if $r_j \leq r_i$ and $(r_i, r_j) \in R$, where r_i denotes a senior role while r_j is a junior role.

◊

For example, if role r_1 dominates r_2 denoted as $r_2 \leq r_1$, then $(r_i, r_j) \in R$. Thus, a user acquiring role r_1 can acquire permissions assigned to role r_2 by using the RBAC model permission inheritance properties. Here, inheritance is reflexive because a role inherits its own permissions, transitivity is a natural requirement in this context.

5.4.3 Security Policy

This section discusses the security model for individual Web Services. Our access control approach is implemented at two levels, task level and service level. The task level ensures that the tasks within a business process can be carried out by only those entities that are authorized. Once passing the access control at task level, Web Service access control policy is applied so that the authorized entities or the transactions can access Web Services. In order to securely access services, we introduce security policies for services' operations and model a Web Service as a finite transition system. In particular, the security policies are represented as the preconditions for the transition system. For this reason, the transition system can be fired when the preconditions are evaluated as positive.

Usually, security policy represents an organization's valuable information assets and indicates how these assets are protected. It provides general principle that is used to restrict access of resources in the organization. More precisely, the policy specifies what the execution operations can be applied to perform on resource objects. In our case, security policies specify service security capability and security constraints. Security capability describes the security features of a Web Service such as name of the service requestor, a set of credentials, or a set of particular parameters required to invoke the service or role performed by the service requestor. Meanwhile, security constraints refer to a set of conditions that a Web Service imposes on another Web Service in order to cooperate with it. Based on these descriptions, we develop a method to check the security constraints of the individual Web Service to determine whether they are compatible to the specified security requirements.

Definition 5.10 (Policy) A Web Service policy is defined as a set of rules $R = \{r_1, r_2, ..., r_m\}$. Each rule includes a set of condition $C = \{c_1, c_2, ..., c_n\}$ and a set of actions, $A = \{a_1, a_2, ..., a_w\}$. The condition is described in terms of $C \leftarrow c_1 \wedge c_2 \wedge ... \wedge c_n$ or $C \leftarrow c_1 \vee c_2 \vee ... \vee c_n$. If a set of conditions are evaluated to be true, the corresponding actions related to the conditions will be implemented.

◊

For example, the billing task in the shipping process can be implemented by invoking the operations in the BilliingService if the invoker takes the role of a cashier and its credential category is compatible with that of the BillingServcies.

Policy Billing {

$if\ (((c_1 = Cashier) \wedge (c_2 = BillingService.credential)) = true)$

do BillingService()

}

End

5.5 Related Work

To protect the privacy of business processes and to facilitate Business-to-Business (B2B) interoperability, many researchers have focused on deriving process views from the back-end business processes for the collaboration across organizations [61, 62, 70, 72]. For instance, Liu [61] proposed an order-preserving process-view approach to protect B2B workflow interoperability, whereas Chiu [70] described an approach for the cross-organizational interactions by combining private workflows and workflow views together. Additionally, Shields [99] applied the workflow view approach to drive cross-

organizational workflows interoperability in the Web services environment. On the other hand, views have been also employed in scientific workflows for balancing security and information choice to users. For example, Li [72] introduced flow views for scalable scientific workflow process integration and for security whilst Biton [7] manages provenance in scientific workflows by using views.

5.6 Summary

Security is particularly important in a Web services context because the messages transmitted over the Web could be accessed and altered by imposters. For security reasons, it is important that confidential information like private processes and sensitive data (e.g. patients' private data in medical research) should not be visible or released to unauthorized users. Also selecting the relevant information for specific scientific experiments can be problematic.

In this chapter, we present an approach to support securing access Web services after they are located. Our approach is based on the concept views, which place emphasis on information hiding, minimizing what partners need to know during their collaboration with each other. Second, a novel two layered access control model based on the general principles of the order of security priority in an organization is proposed. With the access model and process views, various roles can also be associated with views, so that users can only see necessary parts of workflows that they are entitled to see.

6. Chapter 6

Ensuring the Reliable Collaboration in Distributed Environment

Many of the advances in collaborations for e-service transactional management are driven by advancing database theory. In Chapter 3 and Chapter 4, we considered the transactional requirements in a distributed context and web services composition respectively. In the current chapter, we focus in particular on reliable collaboration in a Web service collaborative environment.

Like many e-commerce applications, the reliability, in almost every case, is essential to collaborate and manage e-service transaction management effectively. Furthermore, the importance of service-oriented computing for collaboration in business applications and software engineering continues to grow. However, the growth is impeded by the increasing complexity of distributed collaborative applications, and the lack of reliability of applications that use web services [5, 119]. Here, reliability refers to the continuity of the service delivered by a transaction management system. In other words, a system could successfully complete its tasks in the instance of failure. As pervasive use of web

services, we need to ensure reliability in developing business collaboration infrastructure in services-oriented context.

To achieve the reliability, there are many aspects of reliable collaboration for transactional management system that can be analyzed and understood effectively, with the aid of database theory. It is exactly this type of assistance which reliable collaboration for e-service transaction management systems is designed to depend. For this reason, we will apply database theory to collaborative applications in a distributed context. In particular, we will, in this chapter, extend traditional transaction theory with the combination of failures handling approaches, in order to guarantee consistent outcome and correct execution of collaborative process.

This chapter is organized as follows: Section 6.1 is introduction and Section 6.2 introduces transactional models. Section 6.3 presents an extended atomicity sphere concept for reliable collaboration. Section 6.4 focuses on recovery approaches. Finally, we discuss related work in Section 6.5 and provide a summary of this chapter in Section 6.6.

6.1 Introduction

Business collaboration is a dynamic process in which a set of independent partners (organizations, institutions, or specialized individuals) work together to form a temporary *business alliance*, called a virtual organization or virtual enterprise [58, 95], where each member of business alliance contributes parts to the overall virtual enterprise in order to exploit an apparent market opportunity. The main objectives of such collaboration are to share resources, improve an organization's working efficiency and

provide better products or services, in order to survive in the increasing competitive pressure of the globalization of economies.

The importance of service-oriented computing for collaboration in business alliance and software engineering continues to grow. The growth however, is impeded by the increasing complexity of distributed applications and the lack of reliability and flexibility of applications that use Web services. Here, reliability refers to the continuity of the service delivered by a system. In other words, a system could successfully complete its tasks in the instance of failure. As pervasive use of Web services, there is a need for ensuring the reliability in developing business collaboration infrastructure in services-oriented context.

Typical examples in distributed collaboration where reliability and flexibility can be of great importance, are in the fields of e-business and e-research. Consider a case related to supply chain management and manufacturing processes that involves a high level of complexity and many business partners with complex interrelationships. For example, the completion of whole business transaction involves selecting goods remotely, managing orders with an electronic cart, paying electronically and tracking the shipment, and so on. Consequently, any failure during the collaboration among partners may result in unreliability in the software systems, and have a significant commercial impact. For example, a customer may have paid the bill whereas the supplier fails to deliver the products.

There is a long history of efforts to make distributed applications reliable. The two fundamental approaches for constructing a reliable system are fault tolerance and transaction based approaches [40]. Fault tolerance refers to a system design approach

which recognizes that faults will occur; it tries to build mechanisms into the system so that the faults can be detected and removed, or compensated for before they can result in a system failure. One of the basic techniques in implementing fault tolerance is to utilize error recovery. The goal of error recovery is to transform the current erroneous system state into a well-defined and error-free state, from which normal system operation can continue. Specifically, there exist two basic ways to deal with the recovery: forward error recovery and backward error recovery. Forward error recovery attempts to continue the execution of normal system operations by replacing the failed components or tasks with other tasks. In other words, the current erroneous state is manipulated by system to enable the system to move forward without failing. Conversely, backward error recovery aims to restore the system to a state which occurred prior to the manifestation of the fault.

The transactional support is widely used to address the reliability of systems. In traditional database systems and workflows [40, 41, 45, 107], the consistency of sharing data and administration among components can be achieved through implementing strict transaction semantics in terms of atomicity, consistency, isolation and durability (ACID) [40]. Although extremely reliable, traditional ACID transactions are not suitable for loosely coupled environments such as Web service-based business transactions [40, 73, 107, 119]. This is because fine-grained lock controls and full trustworthiness are not generally applicable in Web services-based transactions. Although a number of proposals[45, 107] are presented to address this issue, currently, the existing Web service frameworks still lack effective models and approaches for the reliable (fault-tolerant, transactional) execution of a group of Web services[5, 81].

In this chapter, we focus on the combination of transaction approach with exception handling in order to ensure the reliable collaboration on the Web. For this end, we present an extended atomicity sphere model by considering two levels of atomicity abstraction for supporting reliable collaboration. The basic features of our solutions are:

- Maintaining data consistency. At a single task level, we apply the notion of atomicity to ensure the consistency of data in the case of a failure.
- Maintaining process consistency. At a higher level of process, we propose an approach to maintain process consistency by combining atomicity sphere with exception handling, aiming to maintain the consistency of computing processes.

6.2 Transaction Models

In this section, we first take the view on some terms that are used to describe and define transactional capability, and then provide basic transactional model and reliable approaches.

6.2.1 Transaction and Transaction Properties

In traditional database, a transaction refers to a general computing activity, such as creating, updating or deleting data in a database. More precisely, a transaction is a running program that may contain a collection of operations. Although there are various descriptions for transactions, two key points on transactions are a collection of operations in a transaction and the *integrity* of a sequence of the operations. This integrity is embodied in the concept of transaction, that is, a transaction is a collection of operations to be performed as a single unit of work, either being completed successfully

or none of these operations being done. This all-or-nothing semantics of the collection as a whole can be described through implementing the transaction properties: atomicity, consistency, isolation and durability (ACID) [40].

- **Atomicity**. Either all of the operation of a transaction are completed successfully, or none of them are done. This is called the all-or-nothing semantics.
- **Consistency**. The unit of work maps an application from one consistent state to a new valid state. This is called the transaction's correctness.
- **Isolation.** The result of a running transaction is not revealed to other transactions until it commits. This is called partial results hidden.
- **Durability**. The committed updates cannot be erased. This is called permanency.

6.2.2 Advantages of Transaction

Transaction provides the designer of enterprise software systems with a useful conceptual tool for supporting mission-critical applications. First, transactions provide an effective approach for abstracting computations. For example, some important operations in an enterprise are grouped together to constitute a logic operation, which can be run as a transaction. As a result, a designer can focus on his correct program to execute the transaction because the consistency in the presence of concurrent users could be achieved via transaction semantics. Second, transactions offer an excellent fault-tolerance mechanism. The widely employed notion of atomicity on a transaction implies additional connotations such as *recovery properties*. As a collection of operations are arranged within a transaction, the benefits of all-or-nothing semantics are achieved despite of failures.

Although extremely reliable, traditional ACID transactions are not suitable for loosely coupled environments such as Web service-based business collaboration [75, 81]. Business collaboration using Web services may be complex, involving many parties, spanning many different organizations, and potentially lasting for hours or days. Typical examples in distributed collaboration where reliability can be of great importance are in the fields of e-business and e-research. Consider a case related to supply chain management and manufacturing processes that involves a high level of complexity and many business partners with complex interrelationships. For example, the completion of whole business transaction involves selecting goods remotely, managing orders with an electronic cart, paying electronically and tracking the shipment, and so on. Consequently, any failure during the collaboration among partners may result in unreliability in the software systems, and have a significant commercial impact. For example, the customer may have paid the bill whereas the supplier fails to deliver the products. Unfortunately, for a number of reasons, fine-grained lock controls that require to lock resources exclusively, and full trustworthiness are not generally applicable in Web services-based transactions. Therefore, there is a requirement for extended transaction models in the business collaboration.

6.2.3 Transaction for Collaboration

Firstly the transaction requirements for reliable collaborations in a distributed environment will be associated with tasks' models, Web services models and process models so that they are characterized by transactional functions.

To support the reliable collaborations in a distributed environment, effective transaction mechanisms should satisfy different transaction requirements needed in different contexts. These transaction requirements are mainly reflected in the collaborating activities among participants in various application scenarios. For example, orchestrating a computational step A and a computational step B may be considered to be reasonable, but combining the step A or step C without step B might be not acceptable.

A transaction in a distributed environment is characterized by some distributed features: long-running, heterogeneous, and loosely coupled. Firstly, long-running computational tasks can be executed over a long period of time duration, so that it is impractical to lock all data used in a computing process for extended period of time. Next, heterogeneous features may involve multiple participants from various organizations whose scientific computing processes run independently. These organizations may have different transaction models developed, managed, and run independently. In addition, loosely coupling indicates that the collaborating relationships between partners are established in a highly dynamic fashion and in an on-demand basis. From these characteristics, it is clear that the overall transactional behaviors associated with a transaction depend on the transactional capabilities and behaviors of individual computing processes. Therefore, an effective transaction model, suitable for a distributed environment, should support different transactional semantics in the same model.

6.2.4 Atomicity Sphere

The concept of sphere was originally used to refer to the sphere of control in the traditional database [28, 40]. A sphere of control logically defines the boundaries around

a collection of operations performed on resources. As a unit of work composing of set of operations, a sphere is atomic if all its composed operations are committed or aborted unilaterally. This property can be used to create a fault-handling mechanism for reducing the cost of recovering processes in case of the failure, and ensuring data consistency at various levels of granularity. For example, when a sphere included in a process is found to be in error, recovery can be made by undoing or compensating the parts of tasks included in the sphere rather than undoing the whole process.

Atomicity spheres go further by correlating related transaction properties with a process or a group of operations [65, 45, 119]. More precisely, an atomicity sphere represents a group of tasks with transactional properties. Each transactional task within a sphere can be implemented as a sub-transaction while the whole atomicity sphere forms a unit of work implemented as a global transaction. For instance, with a two-phase commit protocol (2PC), all tasks within an atomicity sphere are sure to be either committed or all aborted, thus the all-or-nothing semantics of the global transaction representing the atomicity sphere can be achieved.

6.3 Extended Atomicity Sphere

At a single task's level, we apply the notion of atomicity to ensure the consistency of data in the case of a failure. We model the execution of a task as a run. A run *r* on a task *t* can be triggered by different events. For example, the output data of a task *r* are arriving in the input container of task *h* may lead to the execution of its successor task *h*. We assume that these events and runs are recorded into log files as the history, which can be used to recover a scientific computing process in the presence of failures. In the

run model, execution of a task will influence the dependency between output data and input data. For instance, a run on a task will modify the data inputted to the task, change the state of the task and influence the next task to be executed.

6.3.1 Maintaining Data Consistency

Based on the principle of message passing in distributed computing [66], there is a separation between acceptation of message arriving at tasks and communication with other tasks by sending the computation results. For instance, in a data-oriented scientific workflow model, data availability would determine the related computational steps to run. For this reason, a run on a task starts with copying data from the task's input container to local variables, which may be associated with specific data types, so that it is possible to validate the data reaching the task. The run then carries out the computation based on some formula. If the computation completes successfully, the updated results will be put into the task's output container, ready to be sent to the next tasks. Therefore, for the sake of consistency, it is desirable to model *copy* and *update* operations performed on the data items in a run, as shown in Fig. 6.1 (a). Based on this model, each execution of an entity consists of copying incoming data items into the local variables, followed by updating the data items. If either of these operations fails due to a copying failure or due to the computation updating failure, the entire execution can be retried. Without loss of generality, we employ an entity to denote a task in the following definition.

Definition 6.1 (Run model) An entity can be invoked by applying the following two access modes to the entity's parameters: Copy(C) and Update (U). A run *r* of an entity e_i

is a tuple $R \bullet e_i = \{I_c, O_c, L, \theta\}$, where I_c is entity e_i's input container; O_c denotes e_i's output container, L represents local variables and $\theta \in I_c \times O_c$ represents the dependency of output on input with an execution of the entity.

◊

Intuitively, based on the run model, data flows should be manipulated in the right way in the execution of a workflow to ensure data consistency. The consistency mainly includes three aspects: proper connection of data flows between tasks, correct matching of data formats and atomicity of run a task. Firstly, the proper connection implies that for given two different nodes, there exists a feasible path by which data can flow from one to the other. Let a workflow specification be $G = (V, E, f)$, where $V = V_1 \cup V_2 \cup \ldots \cup V_n$ is the set of nodes, and let P_Σ be the set of all path among V. For two nodes $V_i \in V$ and $V_j \in V$, we say that data can move from v_i to v_j if the following conditions are satisfied:

(1) $\forall V_i, V_j \in V, \; if \; i \neq j \rightarrow V_i \cap V_j = \phi$ (e.g. V_i is different from v_j), and

(2) $\forall p \in P_\Sigma(V_1,...,V_n), \exists P_{V_i \times V_j} \in P_\Sigma(V_1,...,V_n). P = P_{V_i \times V_j} = d(V_i, V_j)$.

Secondly, data consistency should ensure correct data matching between two tasks. In other words, for two tasks T_i and T_{i+1}, data matching indicates that the output data of task T_i should be compatible with the input data of task T_{i+1}. In distributed environment, we assume that data transmitted over the Web are described by WSDL XML messages. Thus data matching can start with by extracting the data from containers by using XPath based on its expressions as well as the structure of the XML schema. The detailed data matching method is beyond the scope of this thesis.

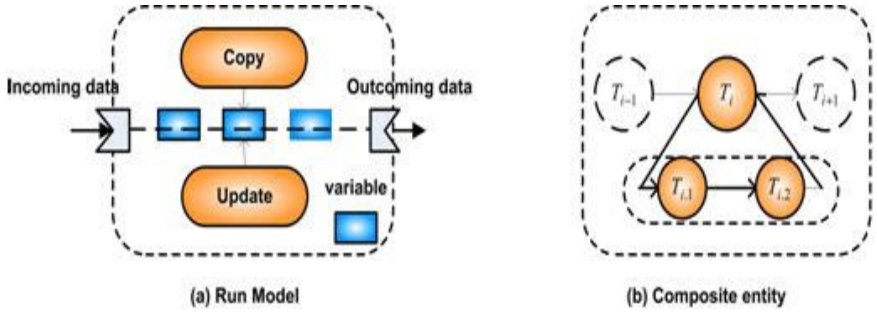

Figure 6.1: Run model and composite entity

Thirdly, the atomicity semantics of execution of an entity says that either execution of a task completes or none of runs depending on the execution has any effect. To achieve the atomicity, we model a run as a transaction.

Definition 6.2 (Transaction) A transaction T_j is defined as a partial ordering over its operations: $T_j = \{\Sigma_j, <_j\}$, where

(1) $\Sigma_j = \{copy_j[x], update_j[x]\} \cup \{abort_j, commit_j\}$; x denotes data item, and $<_j$ is ordering relation;

(2)for any two conflicting operations $h_j = \{C(x) or U(x)\}$ and $d_j = \{U(x)\}$, then either $h_j <_j d_j$ or $d_j <_j h_j$;

(3) for any operation $h_j \subseteq O_j$, $h_{j<_j}\{abort, commit\}$; O_j denotes all operation in T_j.

◊

Formally, we describe the atomicity of an execution of entity as follows. For task T_i, let its input container be $in(T_i) = \{v_1, v_2, ..., v_m\}$, output container as $out(T_i) = \{w_1, w_2, ..., w_q\}$, the set of local variable as $L_i = \{L_i^1, L_i^2, ..., L_i^m\}$, and its input and output instance as $in(T_i)^{inst} = \{v_1^i, v_2^i, ..., v_m^i\}$ and $out(T_i)^{inst} = \{w_1^i, w_2^i, ..., w_q^i\}$respectively. When data are available at T_i, the availability

would trigger T_i to run. If all conditions are satisfied such as data type matching, then the variables would be populated with the input instances, that is, $\forall L_i^g \mid g \in [1,2,...,m]: L_i^g = v_k^i \in in(T_i)^{inst} \mid k \in [1,2,...,m]$. Furthermore, if the update operation completes successfully at the end of execution of task T_i, then current value of data will be updated to $\cup_{i=1}^{m} L_i^{'}$ according to rules stipulated by task T_i. After the output container $out(T_i)$ of T_i is populated by the updated values $\cup_{i=1}^{m} L_i^{'}$, the local variables would then be cleared whilst the result of run would be sent to the next task. However, if for any reason an exception has been raised in one of the operations performed by a run, appropriate recovery measures have to be taken. In this case, a run on an entity provides a fault-tolerant approach based on exception handling for maintaining the consistency of data. For example, when a computation runs into trouble, a fault recovery can be made by employing a forward recovery mechanism, that is, run on the entity can be retried within the finite times. In this case, data consistency sub-system would remove the error and automatically restarts the run, so that regular computation can be resumed.

If an entity is a composite one, reliable data handling system needs to determine the scope affected by the failure to maintain the consistency of data in the case of failure. This can be achieved by identifying the children of the failed entity as shown in Fig. 6(b). In this case, the atomicity data handling sub-system detects the failure of an execution on a composite entity and decides to abort the execution by adopting the following steps. Firstly, an execution on a composite entity aborts if all executions depending the composite entities have aborted. Secondly, the abort of the execution on entity will remove all output data in its output container, and then recover all input data from the log files.

6.3.2 Maintaining Process Consistency

In the previous subsection, we extend the notion of atomicity in the lower level associated with executing an entity, aiming to maintain the consistency of data in the case of failure. When the atomicity handling system decides to abort the execution of an entity, for instance, if any of the components within an atomicity sphere does not have a fault handler for a raised exception, then all of the components in the sphere should indicate the sphere failure. In this subsection, we extend the notion of atomicity sphere in the higher level related to executing sets of entities by combining atomicity sphere with exception handling, aiming to maintain the consistency of computing processes.

As one failure may cause the entire collaborative computing process to be interrupted and as rolling back an entire process may be expensive, we use atomicity sphere to make the collaboration resumable because the approach can recover something smaller than the entire process. For this end, an application system can be constructed by nesting spheres properly. For example, a set of spheres can form a hierarchy construction where the outermost sphere represents a root and other spheres below the root indicate children. From the point of view of a transaction, the top-level sphere represents a global transaction that regulates the commits of nested sub-transactions (sub-spheres). If the failure of a sub-transaction is not recoverable inside its sphere, the failure can be propagated along the hierarchy construction until it can be treated by an ancestor transaction. One of main advantages in this sphere construction is that the failure of a sub-transaction can avoid aborting the entire computing process since such failure may be handled at a certain level.

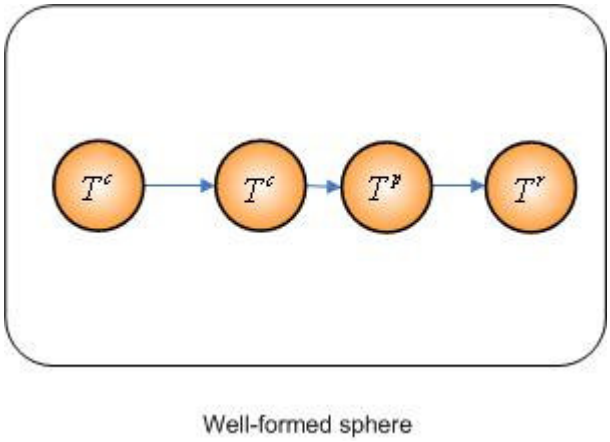

Figure 6.2: Example of process consistency

To ensure the consistency of the collaborative processes, an atomicity sphere should comply with the requirements of a well-formed structure. Let $T=\{T_1,T_2,...,T_n\}$ be the set of tasks; and atomic sphere $AP_{i.j}=\{T_{i.1},T_{i.2},...,T_{i.m}\}, m<n$, be the *ith* subset of set of *T*. Based on the requirements proposed in [45, 107], we summarize the basic requirements for a well-formed atomicity sphere as following:

(1) An atomicity sphere has only one pivot task.

The requirement indicates:

$$\exists T_{i.l} \in AP_{i.j} \mid l \in [1,...,m] : (T_{i.l}).type = pivot \wedge T_{i.j} \in (\bigcup_{j=1}^{m} T_{i.k} - T_{i.l}) \wedge (T_{i.j}).type \neq pivot$$

(2) Execution path pointing to the pivot task consists of only compensatable tasks.

Let T_i^p be a pivot task and all tasks $\prec T_i^p$ before T_i^p would be

$$\prec T_i^p := \{T_{i.j} \in AP_{i.j} \mid \exists e \in E : \pi_1 = T_{i.j} \wedge \pi_2 = T_i^p \wedge (T_{i.j}).type = compensatable\}$$

(3) Execution path leaving the pivot task consist of only retriable task.

Let T_i^p be a pivot task and all tasks $\succ T_i^p$ after T_i^p would be

$$\succ T_i^p := \{T_{i.j} \in AP_{i.j} \mid \exists e \in E : \pi_1(e) = T_i^p \wedge \pi_2(e) = T_{i.j} \wedge (T_{i.j}).type = retriable\}$$

The condition (1) and condition (2) indicate that a pivot task decides the outcome of the sphere. If the pivot task fails, the sphere has failed. In this case, the sphere has to be rolled back by undoing the executed task. And condition (3) expresses that after passing pivot task, undoing is possible only through the rollback method. In short, if satisfying these conditions, the process is able either to proceed until termination or to compensate all tasks executed so far. Fig. shows a well-formed example.

6.4 Recovery Approach

There is a long history of efforts to make distributed applications reliable. The two fundamental approaches for constructing a reliable system are fault tolerance and transaction based approaches [40]. Fault tolerance refers to a system design approach which recognizes that faults will occur; it tries to build mechanisms into the system so that the faults can be detected and removed, or compensated for before they can result in a system failure. One of the basic techniques in implementing fault tolerance is to utilize error recovery. The goal of error recovery is to transform the current erroneous system state into a well-defined and error-free state, from which normal system operation can continue. Specifically, there exist two basic ways to deal with the recovery: forward error recovery and backward error recovery.

Based on the basic models and notion introduced in the previous subsections, we combine a backward error recovery and forward error recovery methods in order to ensure reliable collaboration, aiming to maintain processes consistency as well as the data consistency. In particular, we adopt exception handling models proposed in [45, 121]. The basic actions of exception handling include retry and alternate methods.

Retry: it simply means that an interrupted computing process due to a failed task can be recovered by re-revoking the failed task in accordance with the following conditions. Firstly, the type of the task should be retriable. Secondly, the number of times of retrying is restricted as finite value.

Alternate: It allows the exception handler to replace a failed task with another one that is supposed to have the same function as the failed tasks.

In short, all these retry-based recovering approaches are characterized by a forward recovery mechanism in nature that a failed entity would be retried again.

6.5 Related Work

There are many transaction models [79, 97, 115] and protocols proposed to support long-running and distributed business processes, such as Saga model [80], Business Transaction Protocol (BTP) [125] and Web services coordination (WS-C) [126], etc. All these efforts have focused on the relaxation of the traditional ACID properties in order to meet the requirements for long running collaborative business applications.

Fault tolerant approaches have been employed to ensure reliable execution of scientific workflows. Basic fault tolerant mechanisms include checkpoint, log-based message approaches and transaction approaches. To reduce the cost of recovery from failures in distributed environments, one of the transaction-oriented recovery approaches used, is the concept of sphere [28]. Sphere simply means the control sphere that was originally used to recover from failures in the traditional database. Leymann [65] put forward the notion of atomicity sphere, which correlates related transaction properties with a process or a group of operations, whereas Hagen [45] proposed an advanced fault tolerance

mechanism to integrate both transactions and exception handling into workflow systems, for ensuring reliable implementation of workflow systems. In addition, a unified model of atomicity and isolation was presented by Schuldt [107] for the usage of transactional processes. There is also some work focused on applying fault tolerance and transaction approaches for Web services composition [5, 81, 119]. For example, Bhiri [5] proposes a transactional approach to effectively support reliable Web services composition. The method is based on the conception that a composite service is regarded as a structured transaction, whilst individual Web services are treated as sub-transactions.

6.6 Summary

In this chapter, we present an extended transaction model that supports reliable collaboration in the distributed context. To this end, we present an extended atomicity sphere model by considering two levels of atomicity abstraction for supporting reliable collaboration. The basic features of our solutions are:

- Maintaining data consistency. At a single task level, we apply the notion of atomicity to ensure the consistency of data in the case of a failure.
- Maintaining process consistency. At a higher level of process, we propose an approach to maintain process consistency by combining atomicity sphere with exception handling, aiming to maintain the consistency of computing processes.

In order to validate the model introduced in this chapter, a case study can be found in Chapter 7.

7. Chapter 7

A Framework for Supporting e-Services Transaction and Case Study

In this chapter, we present a framework to exemplify the usage of the approach presented in this thesis for supporting reliable collaboration. The proposed framework is a service-oriented architecture, aiming to collaborate and integrate a large number of distributed, autonomous services. In addition, case study shows that long-running scientific research activities can be structured as much short-duration transaction with atomicity sphere concept.

This chapter is organized as follows: Section 7.1 gives an overview of the framework for collaboration. Section 7.2 briefly describes user application layer. Section 7.3 overviews collaborative process model layer. Section 7.4 presents an evaluation and case study. Finally, summary of the chapter is provided in Section 7.5.

7.1 System Architecture

Based on the models introduced in the previous chapters, a securing framework (shown in Figure 7.1) for supporting reliable collaboration is proposed to deal with long-running computing processes that may include both human and automated tasks. The proposed framework is a service-oriented architecture, aiming to collaborate and integrate a large

number of distributed, autonomous services. In addition, the framework combines techniques from business process management, workflow technologies, process views and service-oriented computing, in order to facilitate general business users for partially automating process construction, manipulation of relationships among process and data, and the setup of business alliances.

The new challenge is to provide the benefits of flexible approaches tailored to end users at various abstraction layers, because there is no need for pre-established organisational relationships between the prospective partnering organisations. The proposed framework extends the traditional four-level model of collaboration among different enterprises by introducing virtual process alliances. The proposed prototype consists of three layers: User Applications Layer (UAL) (Section 7.2), collaborative process model layer (Section 7.3), and Web service layer. All components communicate through SOAP message.

7.2 User Application Layer (UAL)

A user application layer provides a basis on which a virtual process alliance for potential business partners could be created. At the virtual business alliance layer, different enterprises exchange high-level business strategies, for example, whether they want to form a business alliance relationship to develop a product together. Thus the purpose of UAL is to provide business partners with a loosely coupling collaborative management environment. Here the loosely coupling collaborative context means that a general and *strategy-level* agreement will be reached before the actual business actives among partners can be taken. It is on this layer that different companies are able to exhibit and

provide their particular services, products, and resources, which could be shared by one another.

More often, a modern business application is comprehensive in that, it is difficult for one single enterprise to solve all complicated business problems. Instead, it is more likely that the enterprise outsources parts of business activities to other business partners. For example, a computer maker of company A may contribute the virtual alliance by providing computer CPU components whilst company B supplies motherboard products. As a result, new business opportunities may be provided to business partners.

7.3 Collaborative Process Model Layer

Once different enterprises agree on their common strategic goal, form a virtual business alliance is created. Then a concrete business process specification will be regulated on the collaborative process model layer. On the process model layer, different enterprises decide how they can achieve their business goals defined at the level of process model, which is usually involving detailed operational steps. There are different tools which can be used as model the business process specification such as Business Process Modeling Notation (BPMN), and ebXML [35]. Finally, this model representation then can be further converted to a Business Process Execution language (BPEL).

Fig. 7.1 shows the basic structure of reliable Web service-based collaborative transaction management. The architecture consists of two layers: User Applications Layer (UAL) and Transaction Supporting Infrastructure Layer (TSIL). The purpose of UAL is to provide users with an accessing environment that consists of generic function templates. In UAL, users can use templates to register and describe their e-services. On the other

hand, TSIL provides a service transaction management environment for services discovery and transactional Web service composition. BPEL is used to describe process schemes and coordinate the interactions among Web services. In this framework, we separate the normal processes that are specified in BPEL and the reasoning of securing service transaction management. Other basic modules in the framework include the Plan Manager, the Access Control Evaluation Model (ACEM) and the Composition Policy Repository (CPR). The Plan Manager will monitor the behavior of the scientific computing process. The ACEM will evaluate the request issued by a service source via sending the request to CPR. If the request is compatible to the requirement of the security constraints, the requested action is permitted and ACEM will refer to the request to the target services.

Within this framework, developers can specify the access policy and associate services to the accesses authorizations, and generate access binding for collaborative services.

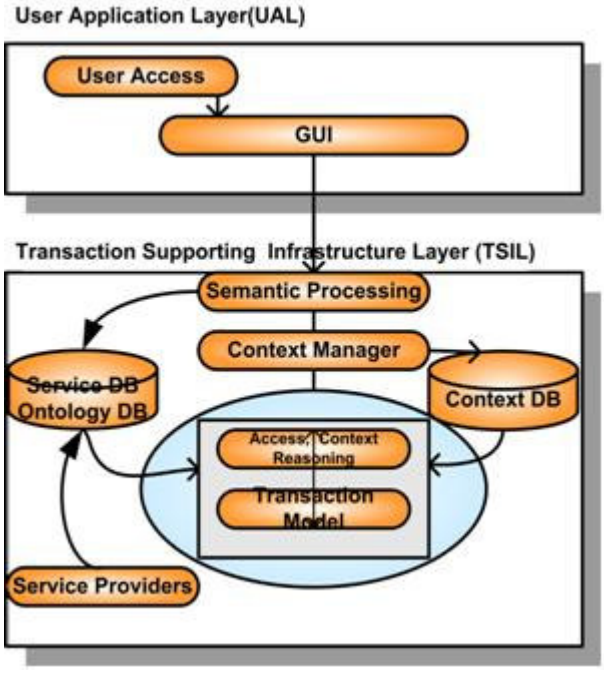

Figure 7.1: Structure of reliable collaboration

7.4 Evaluation and Case Study

Our case study is based on the scenario adapted from the CyberShake system [31], which focuses on earthquake science research conducted by the collaborations among scientists and different organizations (sites). In this scenario, for example, the participants involving the collaboration include hundreds of scientists from over 54 different institutes that form a virtual organization. The goal of the project is to produce hazard maps for some specific areas in order to forecast the probabilities at which earthquake would occurs in these areas.

Fig. 7.2(a) shows an example of the collaborative earthquake science research involving three different sites (areas) around city M. For each site, scientist carry out the experiments for deducing a hazard curve that shows the probability for this specific site due to likely earthquake over some period of time. For example, a seismologist at a site may first stimulate an experimental plan by indicating required experimental steps, which may include the following seven experimental steps: selecting a site (SS), identifying ruptures (IR), computing rupture variations (CRV), computing tensors (CT), synthesizing synthetic seismograms (SSS), computing peak value (CPV) and computing hazard curve (CHC). These experimental steps can be modeled as tasks and the combination of the tasks forms a scientific workflow as shown in Fig. 7.2(b). The scientist then creates the data dependencies among steps though designing proper inputting data IDs to each step. During the run time, an engine would be in charge of invoking these tasks in an appropriate order. Finally, through the integration of the hazard curves created from the three sites, a hazard map for the city M is produced to be

able to indicate the ground motions, which may predicate a particular earthquake in city M. Normally, the number of sites used to produce a hazard map for an area in practice may be huge, therefore the computation and collaboration in this case may be complicated.

In order to depict the hazard map for city M, a scientist *S* would first run the scientific workflow by providing it with related information. The collaborative system would then send the requirement to different sites, where needed scientific research data are stored. Fig. 7.3 shows an example of atomicity sphere application.

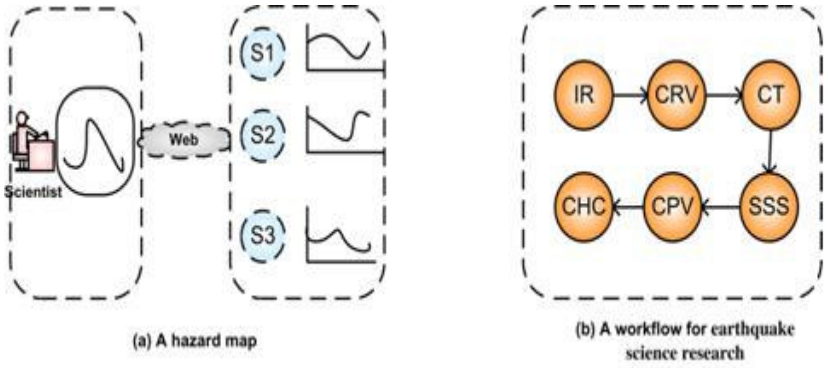

Figure 7.2: Scenario of earthquake science research adapted from [56]

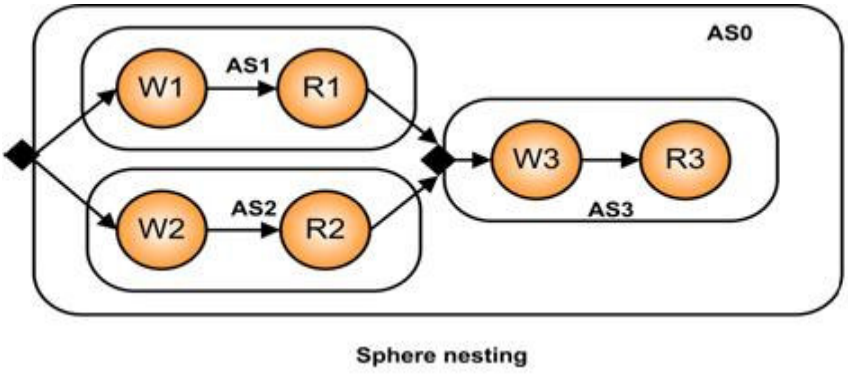

Figure 7.3: Example of atomicity sphere application

We assume that there exist two different spots at each site, for example, data warehouse W1 at spot 1 and R1 at spot 2 for site 1, similarly for W2 and R2 for the site 2, and W3

and R3 for site 3. To facilitate the interaction and collaboration, each computing step as well as the workflow in a site can be wrapped as Web services, which can in turn be associated with different back-end databases, so that the scientific computation can be carried out in a cooperative and standard way. Furthermore, for a specific requirement for producing a cure for a site, it may be desirable that data stored in W1 needs to be integrated with the data stored in R1, W2 to R2, and W3 to R3 respectively because the two spots in a site are closer each other, so that they are able to produce similar geographical data. As a result, W1 to R1, W2 to R2 and W3 to R3 form atomicity spheres AS1, AS2 and AS3 respectively, as shown in Figure 7.3. In addition, AS0 represents a top-level sphere that simulates the interaction between the scientists *S* and the collaborative system. In this case, AS0 can be implemented as a global transaction while nested spheres AS1, AS2 and AS3 are implemented as sub-transactions.

The during the execution of the workflow, if, for example, a failure would occur in invoking R1 for some reasons, R1 could be retired for finite times(e.g. two times) based on our run model for ensuring data consistency. On the other hand, if R1 were unavailable, it then could be replaced with another data warehouse that is close to R1, so that the computing process would be continued.

7.5 Summary

In this chapter, we present a framework to exemplify the usage of the approach presented in this thesis for supporting reliable collaboration. In order to validate the feasibility and benefits of the proposed approaches, a transaction model has been used to develop an application for reliable collaboration in a distributed context. The application

illustrates that the proposed approaches can efficiently ensure the reliable collaboration. In addition, case study shows that long-running scientific research activities can be structured as much short-duration transaction with atomicity sphere concept.

8. Chapter 8

Conclusion and Future Work

In previous chapters, the framework for supporting reliable collaboration has been presented based on the business process management, workflow technologies, process views and service-oriented technology. In this chapter, we summarize the contributions and identify directions for possible future work.

8.1 Contributions and Summary of This Thesis

The need for the coordination of trading partners and the importance of transaction management in a distributed environment has been evident for long time. As traditional business approaches are mainly based on central control mechanisms by which trading partners are working together with closely coupled, it is quite difficult to facilitate the interoperability among trading partners and may have a significant impact on providing competitive products and services. As we discussed in the previous Chapters, effective collaborative approaches can cope with those issues, but they also pose a number of significant challenges such as semantics, security and reliability. Fortunately, service-oriented technologies are deemed as an effective means to support cooperating applications and have significant impact on expanding service-based economy.

To support the business transaction, coordination across all or some of their involving

partners must be required. Such coordination needs a loose coupling context and must be imposed by a semantic, secure and reliable manner. Reliable interoperation with other companies is important for modern enterprises because it enables them to improve their working efficiency, provide better products or services, and exploit market opportunity in order to survive in the increasing competitive pressure of the globalization of economies. For this purpose, information and communication technologies have been identified as critical success factors for efficiently managing business alliances. However, current available collaboration technologies still lack concepts and approaches such as reliable integration of heterogeneous environments across partner sites, flexible support of interoperable business.

In this thesis, we propose an access framework and models for supporting semantic, secure and reliable collaboration. The major contributions of this work are:

- The first contribution is a novel approach to find relevant services through the combination of keyword technique and the semantics extracted from the services' descriptions by using a probabilistic semantic approach in order to handle poor scalability and lack of semantics.
- The second contribution is a novel two layered access control model based on the general principles of the order of security priority in an organization is proposed. With the model, various roles are associated with views, so that right users can only see necessary parts of workflows exposed to them.
- The third contribution is an extended transaction model. The proposed atomicity sphere model is relaxed by considering two levels of atomicity abstraction for

supporting reliable collaboration at the level of process, as well as, at the level of data to maintain the process consistency and data consistency in case of failures.

8.2 Possible Future Research Work

In this thesis, we propose a solution addressing secure and reliable Web service transaction management. This is a challenging work that is currently being investigated by both industries and research institutions. In the following future research work, several directions for possible future work are identified.

Develop process model with combination of control flows and data flows and workflow management. The new challenge is to provide a hybrid approach integrating control flows and data flows into a process model [4, 29, 99]. For this purpose, we will further investigate how to effectively model, verify process by extending our previous work on modeling workflows [75]. We will design new operators to analyze and manipulate process and process views introduced in the next task by employing *a process algebra* approach. These operators can flexibly map choreographies and orchestrations to services. For example, with these operators, a bioinformatics scientist is able to run, revise, and resume a workflow. This flexible ability to collaborate is currently absent in most business workflow systems.

Develop a protocol supporting personal business collaboration**.** We plan to propose related protocols to regulate the interaction of business processes in business alliance in a reliable and flexible manner. A protocol is a formal model, often represented by a set of rules, which govern software processing, decision making and communication tasks. A possible research work is to design a metadata, which can be used as fundamental

message units for participants to interact with each other to reach common goal. A typical message unit may include the requirement and response for collaboration.

9. Chapter 9

Bibliography

[1] A. Ankolekar, M. Burstein, J.R Hobbs, O.Lassila, D. Martin, D. McDermott, S.A McIlraith, S. Narayanan, M. Paolucci, T. Payne and K. Sycara. DAML-S: Web Service Description for the Semantic Web. In Proceedings of the 1st International Semantic Web Conference (ISWC), 2002.

[2] C. Atkinson, P. Bostan, O. Hummel and D. Stoll. A Practical Approach to Web Service Discovery and Retrieval. In Proceedings of IEEE International Conference on Web Services (ICWS2007), July 9-13, 2007, Utah, USA, 2007.

[3] W. Abramowicz, K. Haniewicz, M. Kaczmarek and D. Zyskowski. Architecture for Web Services Filtering and Clustering. In Proceedings of Internet and Web Applications and Services (ICIW '07), 2007.

[4] R.Barga and D. Gannon. Scientific versus Business Workflows. In book: Workflows for e-Science, Scientific Workflows for Grids. Pages: 9-16, Publisher: Springer London, 2007.

[5] S. Bhiri, O. Perrin and C. Godart. Ensuring Required Failure Atomicity of Composite Web Service. In the Proceedings of 14th International World Wide Web Conference (WWW2005), May 10-15, Chiba, Japan, 2005.

[6] Z. Bao, S. C. Boulakia, S. B. Davidson, A. Eyal and S. Khanna. Differencing Provenance in Scientific Workflows. In Proceedings of The 25th International Conference on Data Engineering (ICDE2009), pages: 808-819, March 29 - April 2, Shanghai, China, 2009.

[7] O. Biton, S.C. Boulakia, S. B. Davidson and C. S. Hara. Querying and Managing Provenance through User Views in Scientific Workflows. In Proceedings of The 24th International Conference on Data Engineering (ICDE2008), pages: 1072-1081, April 7-12, Cancún, México 2008.

[8] S. Bowers, B. Ludäscher, A. H. H. Ngu, T. Critchlow. Enabling Scientific Workflow Reuse through Structured Composition of Dataflow and Control-Flow. ICDE Workshops 2006, 2006.

[9] W. T. Balke and M.Wagner. Towards Personalized Selection of Web Services. In the 12th International World Wide Web Conference (WWW2003), 2003.

[10] D. Bachlechner, K. Sirpaes, D. Fensel and L. Toma. Web Service Discovery- A Reality Check. In Technical Report, 2006.

[11] B. Benatallah, M. Hacid, A. Leger, C. Rey and F. Toumani. On Automating Web Services Discovery. In VLDB Journal, Vol.14, 2005.

[12] M. W.Berry, S.T. Dumais and G.W.O'Brien. Using Linear Algebra for Intelligent Information Retrieval. In SIAM Rev. Vol. 37(4): pages: 573-595, 1995.

[13] M. W. Berry, S. A. Pulatova and G. W. Stewart. Computing Sparse Reduced-Rank Approximations to Sparse Matrices. In ACM Transactions on Mathematical Software, Vol. 31, No. 2, pages 252–269, 2005.

[14] J. Baliński and C. Daniłowicz. Re-ranking Method based on Inter-Document Distances. In Journal of the Information Processing and Management. Vol. 41, Issue 4, 2005.

[15] S. Bowers, B. Ludäscher. Actor-Oriented Design of Scientific Workflows. ER 2005: pages: 369-384, 2005.

[16] J. Brooke, S. Pickles, P. Carr and M. Kramer. Workflows in Pulsar Astronomy. In book: Workflows for e-Science, Scientific Workflows for Grids. Pages: 60-79, Publisher: Springer London, 2007.

[17] D.A. Brown, P.R. Brady, A. Dietx, J. Cao, B. Johnson and J. McNabb. A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis. In book: Workflows for e-Science, Scientific Workflows for Grids. Pages: 39-59, Publisher: Springer London, 2007.

[18] J. Chen and Y. Yang. Temporal Dependency Based Checkpoint Selection for Dynamic Verification of Fixed-Time Constraints in Grid Workflow Systems. ICSE 2008: pages: 141-150, 2008.

[19] J.Chen and W. M.P. van der Aalst. On Scientific Workflow. on Technical Committee on Scalable Computing. Available at: http://www.ieeetcsc.org/newsletters/2007-01/scientificworkflow.html.

[20] Q.Chen and M.Hsu.Inter-Enterprise CollaborativeBusiness Process Management. In Proceedings. of 17th International Conference on Data Engineering (ICDE), 2001.

[21] J. Crampton. XACML and Role-Based Access Control. In Presentation at DIMACS Workshop on Security of Web Services and e-Commerce. 2005.

[22] F. Casati and S. Ilnicki. EFlow: A Platform for Developing and Managing Composite-Services. In Research Challenges, 2000. Proceedings. Academia/Industry Working, 2002.

[23] J. Cardoso and A. Sheth. Quality of Service for Workflows and Web Service Processes. In Journal of Web Semantic. In Journal of Web Semantics, Vol.1, No.3, April 2004.

[24] W.C. Dou, J. Chen, S. Fan and S. C. Cheung. AContext- and Role-Driven Scientific Workflow Development Pattern. In Concurrency and Computation: Practice and Experience Vol. 20(15), pages: 1741-1757 , 2008.

[25] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The Next Step in Web Services. Communications of the ACM, Vol. 46. No. 10, October 2003.

[26] G. De Giacomo,Y. Lesperance, and H. Levesque.ConGolog, a Concurrent Programming Language Based on the Situation Calculus. In Artificial Intelligence, Vol. 1 – 2, No. 121, pages: 109 – 169, August 2000.

[27] S. Deerwester, S.T. Dumais. Indexing by Latent Semantic Analysis. In Journal American Society for Information Retrieval, pages: 391-407, 1990.

[28] C. Davies. Data Processing Spheres of Control. In IBM Systems Journal, Vol. 17(2), pages:179-198, 1978.

[29] E.Deelman, D.Gannon,M.Shields and I. Taylor. Workflows and e-Science: An Overview of Workflow System Features. In Future Generation Computer Systems. Vol. 25(2009), pages:528-540, 2009.

[30] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R.Graves, N. Gupta, V. Gupta, T.Jordan, C. Kesselman, P. Maechling, J. Mehringer, G. Mehta, D. Okaya, K. Vahi and L. Zhao. Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The Cyber-S-hake Example. e-Science 2006.

[31] E. Deelman, J.Blythe, Y. Gil, C. Kesselman, G. Mehta, K.Vahi, K. Blackburn, A. Lazzarini, A.Ar bree, R. Cavanaugh, and S. Koranda. Mapping Abstract Complex Workflows onto Grid Environment. In Journal of Grid Computing, Vo1.1, pages: 25-39, March 2003.

[32] S. Deerwester, S.T. Dumais. Indexing by Latent Semantic Analysis. In Journal American Society for Information Retrieval, pages: 391-407, 1990.

[33] S.C. Deerwester, S.T.Dumais, T.K. Landauer, G.W.Furnas and R. A. Harshman. Indexing by Latent Semantic Analysis. In Journal of the American Society of Information Science, Vol. 41(6), pages: 391-407, 1990.

[34] X. Dong, A. Halevy, J. Madhavan, E. Nemes and J. Zhang. Similarity Search for Web Services. In Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[35] ebXML(Electronic Business using eXtensible Markup Language),2006. http://www.ebxml.org/.

[36] C. Eckart and G. Young. The Approximation of One Matrix by Another of Lower Rank. In Psychometrika, Vol.1 (1939), pages: 211-218, 1939.

[37] J. Fan and S. Kambhampati. A snapshot of public Web Services. In ACM SIGMOD Record Archive, Vol. 34, Issue 1, 2005.

[38] G.W. Furnas, T.K. Landauer, L.M. Gomez and S.T. Dumais. The Vocabulary Problem in Human-System Communication. In Communication of ACM, Vol. 30(11), pages: 964-971, 1987.

[39] Y. Fu, Z. Dong. Modelling, Validating and Automating Composition of Web Services. In ICWE'06, July 11-14, 2006, Palo Alto, California, USA, 2006.

[40] J. Gray,A. Reuter.Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993.

[41] P.Grefen, J.Vonk and P. Apers. Global Transaction Support for Workflow Management Systems: from Formal Specification to Practical Implementation. In VLDB Journal, Vol.10, No.4, December 2001, pages: 3l6—333, 2001.

[42] Y. Gil, B. Deelman, M. H. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. A. Goble, M. Livny, L. Moreau and J. Myers. Examining the Challenges of Scientific Workflows. In IEEE Computer . Vol. 40 (12), pages: 24-32, 2007.

[43] J. Garofalakis, Y. Panagis, E. Sakkopoulo and A. Tsakalidis. Web Service Discovery Mechanisms: Looking for a Needle in a Haystack? In International Workshop on Web Engineering, August 10, 2004.

[44] L. Gong and X. Qian. The Complexity and Composability of Secure Interoperation. In Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy, 1994.

[45] C. Hagen , G. Alonso.Exception Handling in Workflow Management Systems. In IEEE Transactions on Software Engineering, Vol.26, No.10, October 2000, pages: 943-958, 2000.

[46] V. V. Heck, and P. Vervest. Smart Business Networks – How the Network Wins. Communications of the ACM, Vol. 50, No 6, pages: 29-37, 2007.

[47] J. Hendler. Agents and the Semantic Web. In IEEE, Intelligent Systems, Vol. 16, Issue: 2, March-April 2001.

[48] M. N Huhns. Software Agents: The Future of Web Services. Agent Technology Workshops 2002, LNAI 2592, pages: 1-18, 2003.

[49] R. Hull, M. Benedikt, V. Christophides and J. Su. E-services: A Look Behind the Curtain. In Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, June 2003.

[50] T. Hofmann. Probabilistic Latent Semantic Analysis. In Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval. Berkeley, California, pages: 50-57, ACM Press, August 1999.

[51] T. Hofmann. Probabilistic Latent Semantic Indexing. In Proceedings of the 22nd Annual International SIGIR Conference on Research and Development in Information Retrieval. 1999.

[52] T. Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. In Machine Learning. Vol. 42, No. 1-2, pages: 177-196, January 2001.

[53] V. van den. Heuvel, K. Leune and P. Papazoglou. EFSOC:A Layered Framework for Developing secure Interactions between Web services. In Distributed and Parallel Databases Archive. Vol. 18, Issue 2, pages:115-145, 2005.

[54] A. C. Jones.Workflow and Biodiversity e-Science. In book: Workflows for e-Science, Scientific Workflows for Grids. Pages: 80-90, Publisher: Springer London, 2007.

[55] A. K Jain,and M. N. Murty. Data Clustering: A Review. In ACM Computing Surveys, 1999.

[56] J. Y. Jung, W. Hur and S.H. Kang. Business Process Choreography forB2B Collaboration. In IEEE Internet Computing, Vol. 8(1), pages: 37-45, January-February 2004.

[57] O. Kipp, M.Wieland and F. Leymann. Towards Choreograph Transactions. In 1st Central European.

[58] E. C. Kasper-Fuehrer and N. M. Ashkanasy. The Interorganisational Virtual Organization: Defining a Weberian Ideal. In International Studies of Management & Organization, Vol. 33, No. 4, pages: 34-64, 2003.

[59] M. Klein and A. Bernstein. Toward High-Precision Service Retrieval. In IEEE Internet Computing, Vol 8, No.1, January-February 2004, pages: 30 – 36, 2004.

[60] H. Koshutanski. A Survey on Distributed Access Control Systems for Web Business Processes. In International Journal of Network Security, Vol. 9(1), pages: 61-69, 2009.

[61] D. Liu and M. Shen. Workflow Modeling for Virtual Processes: An Order-Preserving Process-View Approach. In Information Systems. Vol. 28, No.6, pages 505-532, September 2003.

[62] D. Liu, M.Shen, Business-to-BusinessWorkflow Interoperation Based on Process Views. In Decision Support Systems. Vol. 38 (2004), pages: 399–419, 2004.

[63] L.S. Larkey. Automatic Essay Grading Using Text Classification Techniques. In Proceedings of ACM SIGIR, 1998.

[64] F.Leymann, D. Roller and M. Schmidt. Web Services and Business Process Management. In: IBM Systems Journal: Web Services and Business Process Management. Vol. 41(2), IBM, 2002.

[65] F.Leymann and D.Roller. Production Workflow Concepts and Techniques, PTR Prentice Hall, 2000.

[66] L. Lamport. A Theorem on Atomicity in Distributed Algorithms. In Distributed Computing. Vol. 4(2), pages: 59-68, 1990.

[67] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J.Tao and Y. Zh-Ao. Scientific Workflow Management and The Kepler System. In Concurrency and Computation: Practice and Experience. Vol. 18(10), pages:1039-1065, 2006.

[68] B. Limthanmaphon and Y. Zhang. Web Service Composition with Case-Based Reasoning. In Proceedings of the Australasian Database Conference (ADC2003), Adelaide, Australia, 2003

[69] H.J. Levesque, R. Reiter, Y. Lesprance, F. Lin and R.B Scherl. Golog: A Logic Programming Language for Dynamic Domains. 1997.

[70] D. Chiu, S.Cheung, S. Till, K. Karlapalem, Q. Li, E. Kafeza, Workflow View Driven Cross Organizational Interoperability in A Web Service Environment. In Information Technology and Management. Vol. 5 (2004), pages: 221–250, 2004.

[71] L.J. La Padula, J.G. Williams. Toward a Universal Integrity Model. In IEEE Computer Security Foundations Workshop, June, 1991.

[72] Q. Li, Z. shan, P.C.K. Hung, D. K.W. Chiu and S.C. Cheung. Flows and Views for Scalable Scientific Process Integration. In Proceedings of the 1st International Conference on Scalable Information Systems, 2006.

[73] M. Little. Transactions and Web Services. In Communications of the ACM, October 2003, Vol. 46, No. 10, 2003.

[74] J. Ma, J. Cao and Y. Zhang. A Probabilistic Semantic Approach for Discovering Web Services. In the 16th International World Wide Web Conference(WWW2007). Banff, Alberta, Canada, May 8 -12, 2007.

[75] J. Ma, J. Cao and Y. Zhang. Efficiently Supporting Secure and Reliable Collaboration in Scientific Workflows. In Journal of Computer and System Sciences(JCSS), Vol. 76, Issue 6, pages: 475-489, September 2010.

[76] J. Ma, Y. Zhang and J. He. Web Services Discovery Based on Latent Semantic Approach. In the Proceedings of IEEE International Conference on Web Services (ICWS2008), September 23-26, China, 2008.

[77] J. Ma and Y. Zhang. Efficiently Finding Web Services Using a Clustering Semantic Approach. In the Proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection Integration and Adaptation: Organized with the 17th International World Wide Web Conference (WWW 2008), CSSSIA 2008, Beijing, China, April 22, 2008.

[78] J. Ma, Y. Zhang and M. Li. OMWSC-An Ontology-Based Model for Web Services Composition. In the Proceedings of Fifth International Conference on Quality Software (QSIC 2005). Australia, 2005.

[79] S. Mehrotra, R. Rastogi, A. Silberschatz, H. Korth.A Transaction Model for Multi-Database Systems. In Proceedings of the 12th International Conference on Distributed Computing Systems (ICDCS92), IEEE Computer Society Press, pages: 56-63, June 1992.

[80] H. Molina and K. Salem. SAGAS. In Sigmod Record, ACM Special Interest Group on Management of Data, Vol.16, No.3, pages;.249-259, December 1987.

[81] F. Montagut and R. Molva. Augmenting Web Services Composition with Transactional Requirement. In the Proceedings of International Conference on Web Services (ICWS'06), 2006.

[82] B. Mandhani, S. Joshi and K. Kummamuru.A Matrix Density Based Algorithm to Hierarchically Co-Cluster Documents and Words. In the 12th International World Wide Web Conference (WWW2003). 2003

[83] B. Medjahed, A. Bouguettaya andA.K Elmanarmid. Composing Web Services on the Semantic Web. In VLDB Journal. Vol. 12, Issue 4, November 2003.

[84] S.A. McIlraith, T.C Son and H. Zeng. Semantic Web Services. In Intelligent system, IEEE, Vol. 16, Issue: 2, pages: 46-53, March-April 2001.

[85] S. McIlraith and T.C Son. Adapting Golog for Composition of Semantic Web Services. In Proceedings of KR'02, 2002.

[86] M. Matskin, J. Rao. Value-Added Web ServicesComposition Using Automatic Program Synthesis. WES 2002, LNCS 2512, pages:213-224, Springer-Verlag, 2002.

[87] Z. Maamar, Q.Z Sheng and B. Benatallah. Interleaving web services composition and execution using software agents and delegation. In SiteSeer.IST –Scientific Literature Digital Library

[88] R. Nayak and B. Lee. Web Service Discovery with Additional Semantics and Clustering. In Proceedings of Web Intelligence, IEEE/WIC/ACM International Conference, 2007.

[89] M. Oussani and A. Bouguettaya. Efficient Access to Web Services. In IEEE Internet Computing, Vol. 8, Issue 2, pages: 34-44, March-April, 2004.

[90] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K.Glover, M.R. Po- cock, A. Wipat, and P. Li. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. In Bioinformatics Journal., Online, June 2004.

[91] A.V. Paliwal, N.R. Adam and C. Bornhövd. Web Service Discovery: Adding Semantics through Service Request Expansion and Latent Semantic Indexing. In 2007 IEEE International Conference on Services Computing (SCC 2007), 2007.

[92] M. Paolucci, T. Kawamura, T. Payne and K. Sycara. Semantic Matching of Web Services Capabilities. In Proceedings of the 1st International Semantic Web Conference (ISWC2002), 2002.

[93] C. Peltz. Web Services Orchestration and Choreography. In Computer, Vol. 36, Issue 10, pages: 46-52, October 2003.

[94] S. R. Ponnekanti and A. Fox. SWORD: A Developer Toolkit for Web Service Composition. In Proceedings of The Eleventh World Wide Web Conference(Web Engineering Track), Honolulu, Hawaii, USA, May 7-11, 2002.

[95] O. Prokein, T. Faupel and D. Gille. Web Services as An Enabler for Virtual Organizations. In Book: E-Commerce and V-Business: digital enterprise in the twenty-first century, pages: 245-269, Butterworth-Heinemann, 2007.

[96] O. Prokein, T. Faupel. Using Web Services for Intercompany Cooperation - An Empirical Study within the German Industry. In Proceedings of the 39th Hawaii International Conference on Systems Science (HICSS 2006), 2006.

[97] M. Rusinkiewicz, A. Sheth. Specification and Execution of Transactional Workflows. In Modern Database Systems: The Object Model, Interoperability, and Beyond. W.Kim Ed. ACM Press and Addison-Wesley, 1995.

[98] G. Salton. Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer. In Published by Addison Wesley Publishing Company. 1988.

[99] M. Shields. Control Versus Data-Driven Workflows. In book: Workflows for e-Science, Scientific Workflows for Grids. Pages: 167-173, Publisher: Springer London, 2007.

[100] M. Shehab, E. Bertino and A. Ghafoor. Workflow Authorisation in Mediator Free Environment. In International Journal of Security and Networks, Vol. 1, Issue 1/2, September 2006.

[101] M. Steinbach, G. Karypis and V. Kumar. A Comparison of Document Clustering Techniques. In KDD Workshop on Text Mining, 2000.

[102] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman.Role-Based Access Control Models. In IEEE Computer, Vol. 29, pages: 38-47, 1996.

[103] R. Reiter. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, 2001.

[104] S. Staab, W. Van der Aalst, V.R. Benjamins, A. Sheth, J.A. Miller, C. Bussler, A. Maedche, D. Fensel and D, Gannon. Web Services: Been There, Done That? In IEEE, Intelligent Systems, Vol. 18, Issue: 1, pages: 72-85, January-February 2003.

[105] K. Sivashanmugam, K. Verma, A.P and J.A. Miller. Adding Semantics to Web Services Standards. In Proceedings of the International Conference on Web Services ICWS'03, pages: 395-401, 2003.

[106] A. Sajjanhar, J. Hou and Y. Zhang. Algorithm for Web Services Matching. In Proceedings of the 6th Asia-Pacific Web Conference (APWeb 2004), Hangzhou, China, April 14-17, 2004.

[107] H. Schuldt, G. Alonso, C. Beeri and H. Schek. Atomicity and Isolation for Transactional Processes. In ACM Transaction on Database System (TODS). Vol. 27(1), March, 2002.

[108] J.Spohrer and D.Riecken. Special Issue on Services Science. In Communications of the ACM: Services Science, Vol. 49 Issue 7, July 2006.

[109] I. Taylor, I. Wang, M. S. Shields, and S. Majithia. Distributed Computing with Triana on The Grid. In Concurrency - Practice and Experience, Vol. 17(9), pages: 1197-1214, 2005.

[110] L.Thorburn. Knowledge Management and Innovation in Service Companies — Case Studies from Tourism, Software and Mining Technologies (Study for the Department of Industry, Tourism & Resources). 2004.

[111] K. Thompson. The Networked Enterprise. Competing for the Future Through Virtual Enterprise Networks. Meghan Kiffer, Tampa (2008), 2008.

[112] L.Thorburn, Knowledge Management and Innovation in Service Companies — Case Studies from Tourism, Software and Mining Technologies. (Study for the Department of Industry,Tounism & Resources). Innovation Dynamics 2006, htin://www.oecd.orpJdataoectll5V39/34698722.odf.

[113] R. K. Thomas and R. S. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. In Proceedings of 11th International Conference on Database Security, IFIP WG11, pages:166-181, California, USA, 1997.

[114] M. Wooldridge. An Introduction to Multiagent System. John Wiley and Sons, February 2002.

[115] L.Wang, S. Lu, X. Fei, A. Chebotko, H. V. Bryant and J. Ram. Atomicity and Provenance Support for Pipelined Scientific Workflows. In Journal of Future Generation Computer Systems. Vol. 25(5), pages: 568-576, 2009.

[116] Y. Wang, E. Stroulia. Semantic Structure Matching forAssessing Web Service Similarity. In Proceedings of the First International Conference on Service Oriented Computing, Trento, Italy, December 15-18, 2003.

[117] D. Woollard, N. Medvidovic, Y. Gil and C. Mattmann. Scientific Software as Workflows: From Discovery to Distribution. In IEEE Software 25(4), pages: 37-43, 2008.

[118] S.Weerawarana, F.Curbera, F.Leymann, T.Storey and D.F. Ferguson .Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice Hall, 2005.

[119] C. Ye, S. C. Cheung, W. K. Chan. Publishing and Composition of Atomicity-Equivalent Services for B2B Collaboration. In Proceedings of the 28th International Conference on Software Engineering (ICSE'06), pages: 351-360, 2006.

[120] J. Yu and R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. In Journal of grid Computing, Vol. 3(3-4), pages: 171-200, September, 2005.

[121] L. Zeng, H. Lei, J.Jeng, J. Chung and B.Benatallah. Policy-Driven Exception-Management for Composite Web Services. In Proceedings of the Seventh IEEE International Conference on B-commerce Technology (CEC'05), 2005.

[122] L. Zeng, B. Benatallah , M. Dumas.Quality Driven Web Services Composition. In Proceedings of the International World Wide Web Conference (WWW'2003), Budapest, Hungary, 2003.

[123] L.J. Zhang and M. Jeckle. The Next Big Thing: Web Services Collaboration. In the Proceedings of International Conference ICWS-Europe 2003, 2003.

[124] Y. Zhang and J. Ma. Discovering Web Services Based on Probabilistic Latent Factor Model. In the Joint Conference of the 9th Asia-Pacific Web Conference and The 8th International Conference on Web-Age Information Management APWeb/WAIM'07, Huang Shan, China, 2007.

[125] Business Transaction Protocol (BTP).Available at : http://www.oasis-open.org/committees/business-transactions/documents/.

[126] Web Services Coordination (WS-C). Available at: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os/wstx-wscoor-1.1-spec-os.html.

[127] Web Service Flow Language (WSFL). Available at :http://xml.coverpages.org/WSFL-Guide-200110.pdf.

[128] Google. Available at: http://www.google.com.

[129] Yahoo. Available at: http://www.yahoo.com.

[130] XMethods. Available at: http://www.xmethods.com/.

[131] Web Service List. Available at: http://www.Webservicelist.com.

[132] Web Services Coordination (WS-C). Available at: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os/wstx-wscoor-1.1-spec-os.html.

[133] AT&T Knowledge Ventures: Collaboration Across Borders. Available at: http://www.corp.att.com/emea/docs/s5_collaboration_eng.pdf.

[134] Web Services Description Language (WSDL). Available at: http://www.w3.org/wsdl.

[135] Simple Object Access Protocol (SOAP). Available at: http://www.w3.org/TRJSOAP.

[136] Universal Description, Discovery and Integration (UDDI) Specification. Available at: http://www.uddi.org.

[137] Business Process Execution Language for Web Services (BPEL4WS). Available at: http://www- 106.ibm.comldeveloperworks/library/ws-bpel/.

[138] The Workflow Management Coalition. Available at: http://www.wfmc.org/.

[139] OWL-S: Semantic Markup for Web Services. Available at: http://www.daml.org/services/owl-s/

[140] Web Service Data. http://www.andreas-hess.info/projects/annotator/ws2003.html.

[141] Data Sets. Available at : http://www.cs.utk.edu/~lsi/

[142] WS-Security Core Specification 1.1. OASIS, 2006. Available at: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[143] Web Services Policy 1.2- Framework (WS-Policy). W3C, 2006. Available at: http://www.w3.org/Submission/WS-Policy/

[144] WS-Trust 1.3. OASIS, 2006. Available at: http://docs.oasis-open.org/ws-sx/ws-trust/200512.

[145] Security Assertion Markup Language (SAML). OASIS. Available at: http://saml.xml.org/saml-specifications.