Lifetime Service Level Agreement Management for Service Composition

by

Qiang He

B.Eng.   (HUST)

A thesis submitted to

Swinburne University of Technology

for the degree of

Doctor of Philosophy

March, 2009

To my parents

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Qiang He

March, 2009

Acknowledgements

It had been a great pleasure working with the staff and students at the Swinburne University of Technology (SUT), during my candidature as a doctoral student.

My foremost thank goes to my coordinate supervisors, Prof. Yun Yang at SUT and Prof. Hai Jin at HUST; associate supervisors, Dr. Jun Yan and Prof. Ryszard Kowalczyk – they are also my friends and mentors. Guidance and keen insight from Yun and Jun in particular, along with their thirst for new knowledge and excitement had made this time fun, and always stimulating and inspiring.

I thank Prof. Hai Jin, who first introduced me to SUT, opening the door for my academic journey in Australia.

The Faculty of Information and Communication Technologies (FICT) is made up of dedicated, insightful and fun people. My thanks go to Ke Liu, Xiao Liu and Dong Yuan – my colleagues in the Workflow Technology Program – for their advice and support. Sean Hunter, a smart, diligent and helpful friend, had been an enormous help to my research. I will not forget the many times I asked him to work overtime in order to meet deadlines. Playing basketball, working out and surfing with him and his girlfriend Matilda had been fun and wonderful. Rui Zhou and Jiajie Xu, two basketball fans, had watched National Basketball Association (NBA) games with me, laughing, screaming, applauding and complaining. My thanks go to Nigel Fisher for teaching me rock climbing and scuba diving, which I had never tried before. To Ke Liu, my gym buddy, who had been an interesting partner and a source of stimulation. To the members of our badminton club, who had given me great fun and relaxation. To other postgraduates in the faculty, for all the fun we had had, some of whom had worked with me until late (5:30am the latest, wow). To all the staff in the faculty, for helping me figure out how to use the coffee machine and photocopy machine (they are more complicated than a lot of people would think), how to apply for my trips to conferences, how to relieve myself from awkward moments, and so much stuff that I wound not have known without their instruction and help. To all my friends, both in and out of the university, thanks for the music, movies, phone calls, lunches, dinners and all the fun.

My parents, Bifen Chen and Xizhuo He (Mom and Dad), had always been there. Their love and support had been a foundation for my life. To them, I owe a huge debt of gratitude. Their love, understanding, moral support, care and very existence are my keystone. I can honestly say that without them, I would not be able to complete this work.

Finally, I express my sincere appreciation to SUT, who provided the scholarship to financially support my research work.

Abstract

This thesis presents a set of technologies supporting service level agreement (SLA) management for service composition in the service-oriented computing (SOC) environment. Its purpose is to tackle some of the existing problems in the research on quality of service (QoS) aware service composition. The QoS guarantee throughout the process of service provision is a critical and challenging issue, especially in the service composition scenarios where a group of component services compose a composite service.

As services are considered merchandises on the Internet, the quality of a service is as important as, if not more than, the functionality of the service. SLAs can be used to specify service consumers' QoS requirements and service providers' QoS promises. A set of supporting technologies are proposed in this thesis to address the major issues of QoS-aware service composition that arise at different stages of the lifetime of SLAs: establishment, enforcement and completion. Specifically, an innovative SLA negotiation approach is proposed to support SLA establishment at build time based on combinatorial auctions; a new SLA adaptation approach is proposed to support SLA enforcement at runtime; and a novel trust system is proposed to support reputation-oriented service selection based on SLA profiling upon SLA completion at completion time. Experimental results shows that our approaches 1) effectively improve the quality of the composite service through SLA negotiation; 2) significantly improve the satisfaction rates of service consumers' QoS requirements for the composite services through SLA adaptation; and 3) remarkably improve the success rate of composite services and well resist the unique threats in the open SOC environment through SLA profiling.

The major contribution of this research is to provide a comprehensive and integrated solution to lifetime SLA management for service composition in the open SOC environment. With the new approaches developed, the quality of the composite services can be guaranteed at different stages of the lifetime of the composite services: build time, runtime and completion time.

The Author's Publications

Published:

[1] He, Q., Yan, J., Kowalczyk, R., Jin, H., Yang, Y. "Lifetime Service Level Agreement Management with Autonomous Agents for Services Provision," *Information Sciences*, Elsevier, 179(15):2591-2605, July 2009.

[2] He, Q., Yan, J., Jin, H., Yang, Y. "Adaptation of Web Service Composition Based on Workflow Patterns," *Proc of 6th International Conference on Service-Oriented Computing (ICSOC2008)*, Sydney, Australia, 2008, Lecture Notes in Computer Science, pp. 22-37.

[3] He, Q., Yan, J., Yang, Y., Kowalczyk, R., Jin, H. "Chord4S: A P2P-based Decentralised Service Discovery Approach," *Proc of IEEE International Conference on Services Computing (SCC2008)*, Honolulu, Hawaii, USA, 2008, IEEE Computer Society, pp. 221-228.

[4] He, Q., Yan, J., Yang, Y., Kowalczyk, R., Jin, H. "Towards Collaborative Service Level Agreement Negotiation," *Proc of 4th International Conference on Grid Service Engineering and Management (GSEM2007)* Leipzig, Germany, 2007, pp. 123-134.

[5] He, Q., Yan, J., Kowalczyk, R., Jin, H., Yang, Y. "An Agent-based Framework for Service Level Agreement Management," *Proc of 11th International Conference on CSCW in Design (CSCWD2007)*, Melbourne, Australia, 2007, pp. 412-417.

Submitted:

[1] He, Q., Yan, J., Jin, H., Yang, Y.: "A Quantitative Trust System for Reputation-Oriented Service Selection", submitted to *IEEE Transactions on Services Computing*, 2009.

[2] He, Q., Yan, J., Jin, H., Yang, Y.: "A Decentralized Service Discovery Approach on Peer-to-Peer Overlay Network", submitted to *IEEE Transactions on Services Computing*, 2009.

[3] He, Q., Yan, J., Jin, H., Yang, Y.: "Dynamic Service Selection for Service Composition Based on Iterative Multi-Attribute Combinatorial Auction", submitted to *IEEE Transactions on Software Engineering*, 2008.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis addresses the issue of service level agreement (SLA) management for quality-of-service (QoS) aware service composition in the service-oriented computing (SOC) environment. The major contribution of the research is to develop a set of technologies to support SLA management for service composition at different stages of the lifetime of SLA, including establishment, enforcement and completion. To support SLA establishment, an SLA negotiation approach based on combinatorial auction is designed which allows service consumers and providers to flexibly exchange offers and counter-offers for QoS. To support SLA enforcement, an SLA adaptation approach is designed which adapts a subset of the composite service in a confined scope when SLA violations occur in component services. A service trust system based on SLA profiling upon SLA completion is designed which evaluates service consumers' trust over service providers based on their historic performance over past SLA enforcement.

The background, motivations and key issues of this research are introduced in this chapter. First, an introduction to service composition is given in Section 1.1. Then the key issues of this research are introduced in Section 1.2 . At last, an overview of the structure of this thesis is presented in Section 1.3.

1.1 Introduction to service composition

Service-oriented computing (SOC) is the computing paradigm that utilises services as fundamental elements for developing applications [83]. In the SOC environment, service-based applications are developed as independent sets of interacting services offering self-describing standardised interfaces to potential service consumers. Services perform functions – from simple requests to complicated business processes. Organisations can expose their core competencies in the form of services over the Internet (or intranet) using standard (XML-based) languages and protocols [82].

Web Services are the current most promising technology based on the concept of SOC, offering significant benefits in flexibility, ease of use, and reuse as well as providing a way to develop an SOA incrementally, although an SOA contains more than Web services. Web services technologies enable making connections among heterogeneous software that each performs a discrete function. By wrapping these softwares with Web service interfaces, they can communicate and interoperate to perform a complex business process. Technically, the term "Web services" describes a standardised way of integrating Web-based applications using the Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) [36], Web Service Description Language (WSDL) [20, 22] and Universal Description, Discovery and Integration (UDDI) [23] open standards over an Internet protocol backbone. For example, if a programmer wants to make a Java program accessible to other applications deployed over the Internet, the programmer can publish the program as a Web service in a UDDI server, with its interface (some of the method names and associated parameters) detailed with a service description written in WSDL. Then the application builders who need the Java program can find it in the UDDI server and invoke it through sending SOAP messages typically conveyed using HTTP. Figure 1.1 presents the architecture of Web services [57]. By adopting the Web services technology, heterogeneous applications distributed across the Internet can be integrated in a uniform and standardised manner because Web services provide the following benefits:

- Decoupling of service interfaces from implementations and platform considerations;
- Enablement of dynamic service binding; and
- Increase in cross-language and cross-platform interoperability.

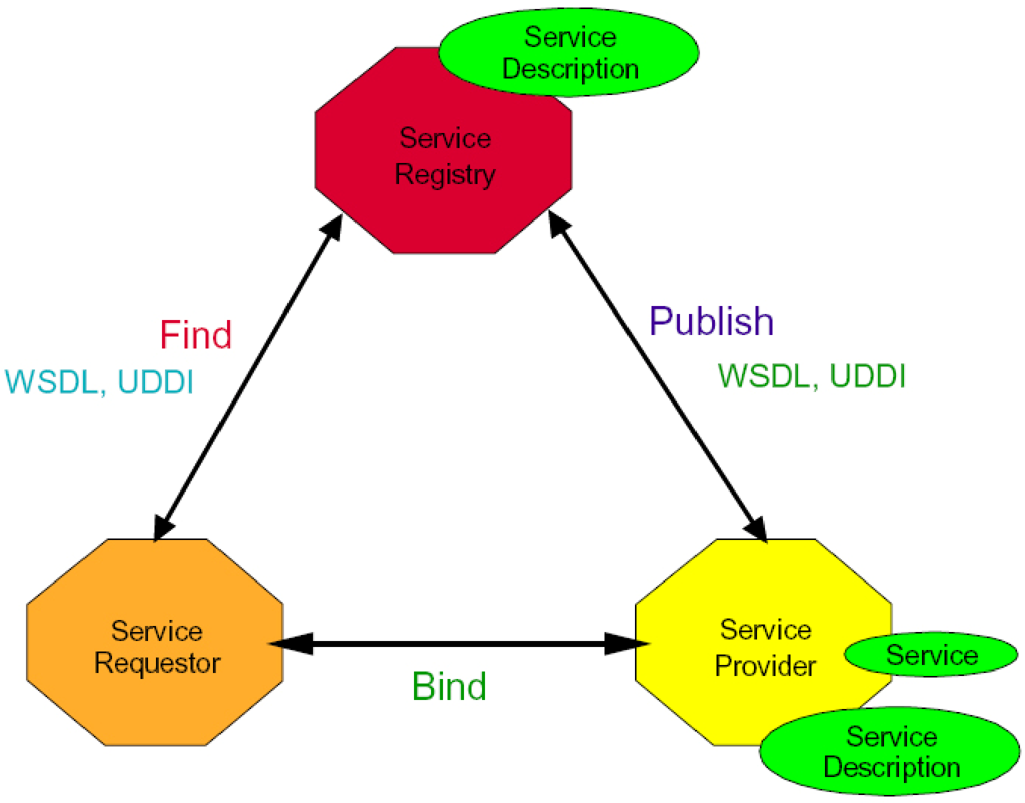


Figure 1.1 Architecture of Web services

A great advantage of SOC is its support for service composition which encompasses the functionality for the consolidation of multiple services into a single new value-added service. Such a service is referred to as a composite service while its constituent services are called component services. The resulting composite services can be used as a component service in further service compositions or be utilised as applications/solutions by service consumers [83].

Service composition provides a number of benefits:

- Service composition allows service providers to minimise the amount of coding work for building new applications;
- Service composition reduces the cost and risks for building new application on top of the existing reusable Web services; and
- Service composition reduces the complexity of building new applications by separating application development and implementation.

In the SOC environment, services are delivered as merchandises from service providers to consumers and hence expose both functional properties (i.e. what they do) and non-functional properties (i.e. the way they are provided). There are many various non-functional properties of services, including all properties that are not directly related to the provided functions. QoS is a most important subset of non-functional properties. QoS is traditionally used to refer specifically to network performance and reliability characteristics. In the context of SOC, QoS refers to a wider variety of service properties. QoS plays an important role in all service-related tasks, especially in service selection. Imagine a scenario in which multiple services (provided by different service providers) provide the same functionality that can fulfil a service consumer's request, the ability of the service consumer to differentiate between the services as well as the service providers depends on the quality of the services that they can provide.

Service consumers' QoS requirements for requested services usually vary from one to another. For example, a service consumer with a sufficient budget may want a service delivered rapidly regardless of price while another service consumer with limit budget may want the service delivered at a low price by sacrificing the delivery time. How to guarantee QoS in order to meet service consumers' requirements at different stages of the service provision is a very important problem.

In a service composition scenario, component services are composed to create a composite service that meets the service consumer’s functional and non-functional requirements. In this research, we focus on the non-functional aspect.

In the service composition scenarios, the issue of service composition with QoS constraints, namely QoS-aware service composition, is complicated and challenging. Before the service provision, the service consumer needs to select a service provider from several candidates for each of the component services that compose the composite service. In this case, the component services not only jointly offer the functionalities, but also collectively fulfil the service consumers’ requirements for the quality of the composite services. During the service provision, exceptions may occur in component services. The faulty component services may result in violations of service consumers’ requirements for the quality of the composite services. After the service provision, service providers’ performance – at what extent they fulfil service consumers’ requirements – needs to be collected to profile the service providers for future analysis of their reputation and capability.

The fact that a service consumer usually has requirements for more than one QoS property of the service turns the problem of optimising QoS-aware service composition with multiple QoS constraints into NP-complete, as proved by the work presented in [11]. A lot of efforts in the area of QoS-aware service composition have been devoted to addressing the computational issue of the optimisation of QoS-aware service composition before actual service provision based on the static quality of the component services. However, considering the dynamic nature of the Internet and the SOC environment, the issue of QoS-aware service composition must be addressed from a more comprehensive perspective – not just before but also during and after the service provision. As an important part of SOC, Service Level Agreement (SLA) offers service consumers contractual guarantee that service providers and the services they provide will operate within pre-specified bounds – particularly with regards to the QoS. At the same time, SLAs serve a role for the service providers in planning resource allocation and avoiding unexpected legal wrangles. Before the service provision, an SLA must be established between the service consumer and the service provider, in order to unambiguously specify the service consumer’s expectations and the service provider’s promises of the service.

During the service provision, the established SLA must be enforced, which includes monitoring the status of the service and checking whether the service provision is compliant with the SLA. If SLA violations occur, the faulty service must be adapted – updated or replaced – to ensure that the service consumer's requirements are met. After the service provision, the results of the SLA enforcement – at what extent the SLA is enforced – must be collected and profiled for future analysis of the service provider's reputation and capability of enforcing SLAs.

In the service composition scenarios, the issue of SLA management is more complicated and challenging. Before the provision of the composite service, the service consumer must establish SLAs with each of the service providers that provide the component services. The quality of the component services must be specified, collectively fulfilling the service consumer's requirements for the quality of the composite service. Moreover, during the provision of the composite service, when a faulty component service needs to be adapted, the SLA attached needs to be adapted. However, adapting the faulty component service only might not be able to guarantee the quality of the composite service as required by the service consumer. Other component services, as well as the attached SLAs, might need to be adapted accordingly. Therefore, it is imperative to determine the scope of adaptation and then the adaptation solution when SLA violations occur. Finally, the service providers' reputation and their capability of enforcing SLAs must be evaluated effectively through SLA profiling in order to enhance future SLA management. With the support of SLA profiling, the trustworthiness of the service providers can be evaluated based on their reputation, which can improves the effectiveness of SLA establishment and enforcement.

Therefore, SLA management for service composition is complicated and challenging, yet will be of much importance in the filed of SOC.

1.2 Key issues of this research

The following key issues need to be addressed to provide a comprehensive solution to SLA management for service composition:

- **For SLA establishment:** A lot of efforts have been devoted in research on selecting service providers, whose capacities for providing the QoS are static, to fulfil QoS requirements for composite services [5, 8, 61, 107, 108]. However, service providers' capacity of providing QoS can change dynamically due to different reasons, e.g. schedule of the service providers, status of the market, macroeconomic policy, etc. Also, service consumers may have acceptable ranges for the QoS instead of mandatory and unnegotiable requirements. During the process of deciding the QoS acceptable to both parties, i.e. service consumer and provider, the dynamicity referred above needs to be captured. Service consumers and providers should be allowed to flexibly express their preferences and capacities for the QoS.

- **For SLA enforcement:** When an SLA violation of a component service occurs, the component service needs to be adapted with satisfying QoS to meet the service consumer's requirements for the quality of the composite service. Sometimes the adaptation of the faulty component service may trigger the need of the adaptation of other component services, hence expanding the scopes of the adaptation. Therefore, the confines of the adaptation should be determined automatically and dynamically. In addition, when selecting the adaptation solution, e.g. selecting which service provider to provide which service, not only the benefit but also the cost that applies during the adaption process should be taken into account.

- **For SLA completion:** When selecting service providers, service consumers usually prefer those that are most likely to successfully enforce the SLAs. Unfortunately, the success rate of the service transaction – the probability that the service provider is likely to enforce the SLA successfully – cannot be provided by the service provider. Generally speaking, the service providers that have higher success rates of past service transactions are more trusted by the service consumers in having the capability of enforcing SLAs successfully. Therefore, a service trust system is needed to identify trustworthy service providers. A promising way to achieve this is to evaluate service consumers' trust over service providers based on their historic performance because the service providers with better reputation – attained from better past performance – are believed to be more trustworthy. Therefore, the results of SLA enforcement, as the data needed for SLA profiling, must be collected and analysed upon the completion of the SLAs.

1.3 Overview of this thesis

We propose a set of technologies to support SLA management for service composition in the open and dynamic SOC environment throughout the lifetime of SLA (corresponding to the process of service provision).

In Chapter 2, some basic concepts related to service, service composition and SLA are presented. The research problems are described and discussed in detail. Then the work related to SLA negotiation, adaptation and profiling are presented. Finally, the requirements for the abovementioned topics are analysed.

In Chapter 3, we present the lifetime of an SLA. In particular, we introduce the major SLA operations performed at different stages of the lifetime of the SLA, including SLA negotiation, adaptation and profiling. Finally, the interactions among the SLA operations are discussed.

In Chapter 4, we present an innovative SLA negotiation approach to support SLA establishment for service composition based on combinatorial auction. We first give some preliminary ideas about combinatorial auctions. Then we illustrate the procedure of the proposed combinatorial auction. We also provide the details of the mechanisms supporting the combinatorial auction for service composition, including bidding constraints, QoS model, bid evaluation, winner determination, ask QoS generation and completion criteria. Finally we demonstrate some experimental results to evaluate the proposed approach.

In Chapter 5, we describe a new SLA adaptation approach to support SLA enforcement for service composition based on workflow patterns. In particular, we introduce some basic ideas related to our approach, including value of change (VOC) and workflow patterns. Then we present an adaptation approach that takes into consideration the internal logic of the composite services and the impact of adaptation for a single component service on other component services during the adaptation process. Finally, we present some experimental results to demonstrate the effectiveness of the proposed approach.

In Chapter 6, we describe ServiceTrust – a novel service trust system to support reputation-oriented service selection based on SLA profiling upon SLA completion. In particular, we introduce the calculation of local transactional ratings, local trust, global trust and transactional trust. We also assess the effectiveness of ServiceTrust and ServiceTrust's resistibility against the threats of malicious reputation manipulation and QoS abuse.

The final chapter, Chapter 7, summarises this thesis, the major contribution of this research and discuss the further research directions.

Chapter 2

Literature review and requirements analysis

In this chapter, we give an introduction to the research fields of service composition. We present some major approaches and standards related to our research, and analyse the research problems and requirements to help the readers of this thesis gain a better understanding of the work described in this thesis.

This chapter is organised as follows. Section 2.1 introduces some basic concepts related to this thesis, including service, service composition and service level agreement. Section 2.2 analyses the research problem in this thesis. Sections 2.3, 2.4 and 2.5 present related work about SLA negotiation, adaptation and profiling respectively. Finally, Section 2.6 analyses the research requirements using a motivating example and Section 2.7 summaries this chapter.

2.1 Background

This section gives an introduction on service and service composition in order to help the readers of this thesis understand the background of this research.

2.1.1 Service

A generic definition for a *service* in SOA is an application function packaged as a reusable component for use in a business process [24]. From the service requester's perspective, a service has the appearance of a software component that provides self-contained function and can be invoked through defined communication protocols. The actual service implementation may involve many steps executed on different computers within one enterprise or across a number of enterprises. As Web services are the most popular paradigm of service in the field of SOC, we introduce the Web service technologies in this section to help the readers of this thesis better understand some foundation concepts and knowledge.

Existing definitions for Web services range from very generic to very specific. Web services can be defined as software identities, technologies and even platforms.

A *Web service* is defined by the World Wide Web Consortium (W3C) as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards" [37]. This definition puts Web services in the confines formed by specific standards, i.e. WSDL and SOAP.

IBM defines Web services as a set of emerging standards that enable interoperable integration between heterogeneous IT processes and systems. They can be thought of as a new breed of Web applications that is self-contained and self-describing, and can provide functionality and interoperation ranging from the basic to the most complicated business and scientific processes. This definition is proposed form the business perspective instead of the technical perspective since IBM is a corporation that aims at providing business solutions to meet service consumers' goals.

Microsoft sees XML Web services as the platform for application integration where applications are constructed using multiple XML Web services from various sources that work together regardless of where they reside or how they were implemented. And the individual Web services are defined as the fundamental building blocks in the move to distribute computing on the Internet [103]. The term "XML Web services" exhibits the importance of XML in Web services. The standards and protocols used by Web services, specifically WSDL and SOAP, are XML-based.

So many definitions for Web services have been proposed because there are companies building them, but all the definitions have the following things in common:

- Web services expose useful functionalities for service consumers to invoke with messages that conform to a standard Web format, in most cases the SOAP format.

- Web services provide a way to describe their interfaces in enough detail to allow service consumers to build client applications to access them. The descriptions are usually provided in the form of WSDL documents.

- Web services are registered so that potential service consumers can find them easily. This can be done using UDDI.

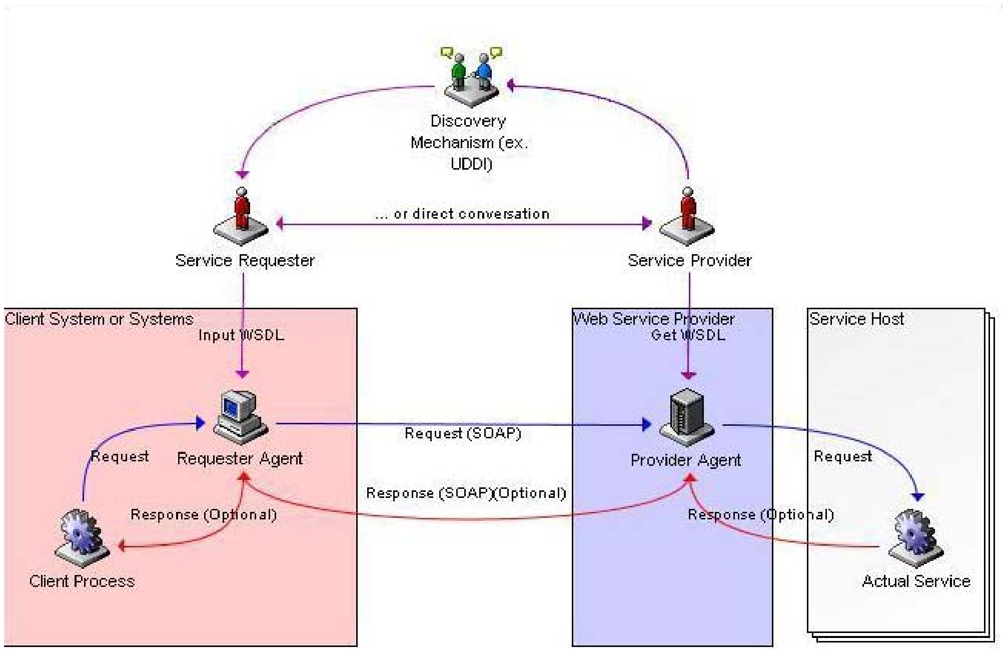


Figure 2.1 Web service interaction model

Figure 2.1 presents the Web service interaction model [7]. From the figure we can see there are two main roles involved: service requestor and service provider. Web services are published in a UDDI service registry as WSDL service descriptions by the service providers. Service requesters (in most cases the service consumers) discover the Web services in the UDDI service registry and then invoke the Web services through sending SOAP messages according to the WSDL service descriptions.

The three main standard specifications for Web services are WSDL, UDDI and SOAP. From Figure 2.1 we can see what roles WSDL, UDDI and SOAP play in the interaction model of Web service. To help readers from other areas better understand the work presented in this thesis, we give a brief introduction to WSDL, UDDI and SOAP.

1. Web Service Description Language (WSDL) [22]: WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.

   A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

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

2. Universe Description, Discovery, and Integration (UDDI) [23]: The focus of UDDI is the definition of a set of services supporting the description and discovery of (1) businesses, organisations, and other Web services providers, (2) the Web services they make available, and (3) the technical interfaces which may be used to access those services. Based on a common set of industry standards, including HTTP, XML [14], XML Schema [9, 31, 94], and SOAP, UDDI provides an interoperable, foundational infrastructure for a Web services-based software environment for both publicly available services and services only exposed internally within an organisation.

   UDDI is a platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative, sponsored by OASIS [81], enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet. A UDDI business registration consists of three components:

- **White Pages**: address, contact, and known identifiers.
- **Yellow Pages**: industrial categorisations based on standard taxonomies.
- **Green Pages**: technical information about services exposed by the business.

3. Simple Object Access Protocol (SOAP) [36]: SOAP is a lightweight protocol for exchange of information in a decentralised, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

   SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralised, distributed environment using XML. SOAP itself does not itself define any application semantics such as a programming model or implementation-specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to remote procedure call (RPC).

SOAP consists of three parts:

- **SOAP Envelope Construct**: defining an overall framework for expressing what is in a message, who should deal with it, and whether it is optional or mandatory.

- **SOAP Encoding Rules**: defining a serialisation mechanism that can be used to exchange instances of application-defined datatypes.

- **SOAP RPC Representation**: defining a convention that can be used to represent remote procedure calls and responses.

Although these parts are described together as part of SOAP, they are functionally orthogonal. In particular, the envelope and the encoding rules are defined in different namespaces in order to promote simplicity through modularity.

In addition to the SOAP envelope, the SOAP encoding rules and the SOAP RPC conventions, e.g. how a SOAP message can be carried in HTTP [33] messages either with or without the HTTP Extension Framework [80].

2.1.2 Service composition

A great advantage of SOC is its support for service composition. Existing services can be dynamically composed in the form of business processes to offer new functions. Such a business process is known as a composite service while its constituent services are called component services. Business process execution language for Web Service (BPEL4WS, i.e. BPEL) [3, 53] is the de facto standard to specify the fashion in which Web services interact in a business process. Semantics [50, 55, 86] and artificial intelligence (AI) techniques [43, 85] are also popular in the area of service composition.

Research on service composition has been carried out for several years. A brief survey can be found in [72]. BPEL4WS, Semantics and AI are the three most popular approaches to service composition. Service composition based on BPEL4WS can be achieved in a centralised [6, 19, 53] or decentralised way [35, 78].

[53] examines the compositional aspects of BPEL4WS and explains what role it plays in service composition. In [6], the authors propose a non-intrusive platform, named ORQOS, on which the Web service composition is executed at both pre-deployment time and runtime, using a policy-based language, namely QoSL4BP, designed to express QoS behavioural logic specification. In [19], an approach is presented that uses XPath, WS-Policy and WS-Policy Attachment to enable non-intrusive specification of QoS requirements in BPEL4WS processes on both the messaging and process levels. To demonstrate the mechanisms, a prototype on top of the Colombo BPEL4WS engine is implemented and presented. In [35], the authors investigate some build time issues and runtime issues related to decentralised Web service composition. Build time issues include code partitioning and error handling while runtime issues include application server, messaging, potential deadlocks, error propagation and error recovery. In [78], a program dependence graph (PDG) based technique is proposed to partition a BPEL4WS business process into an equivalent set of decentralised business processes. Its goal is to maximise the throughput of the network of servers which execute the business processes.

Semantics and AI techniques are the other two popular approaches for service composition. Semantic Annotations for WSDL and XML Schema (SAWSDL), a recently announced World Wide Web Consortium (W3C) standard, provides a strong tool to automate semantics-based service composition [32, 56]. To name a few latest works in Semantic Web service composition, [55] describes Inter-Matching Automatic Composition (IMA), a technique for automatic generation of composite Web services. IMA uses Web service ontologies to find appropriate Web services through matching the input and output parameters. In [50], the authors consider the problem of semantic service composition on conceptual and practical levels. A “value propagation” semantics is used for service description. And a plug-in matching is adopted to find satisfactory or partially satisfactory component services in order to finally constitute the desired composite services. The authors in [86] propose a tractable greedy algorithm for automatic semantic Web service composition and implement the algorithm in a framework named jUDDI+ in

OWL-S [70]. In the AI community, authors in [85] represent Web services using transition systems [75] that communicate via exchanging messages. Symbolic model checking techniques are adopted to determine a parallel composition of all the available services. And then a controller is generated to make sure that the service composition meets the user-specified requirements. In [43], the authors analyse the planning tractability when new constraints can be introduced to limit the amount and form of outputs when formalising Web service composition. With the additional constraints, the set of possible states of the Web services becomes static and then can be modelled in terms of a standard notion of initial state uncertainty. Finally, scalable Web service composition can be realised with powerful background ontologies and modern planning techniques.

Some other solutions for service composition do not adopt either BPEL4WS. semantics or AI, [1, 2] propose a two-step, i.e. logical and physical, methodology for end to end composition of Web services by semantically annotating component Web services. In the first step, planning techniques are used to create desired functionality based on existing service types. In the second step, appropriate Web service instances are selected and bound together for deploying the newly created composite service. However, the two-step approach does not consider the issue of QoS. [16] proposes a service selection scheme that uses a broker to advertise and offer the composite service with a range of service classes. What makes it different from other approaches is that services are grouped requiring the same QoS level according to requesting user/organisation. Using SLA, statistical guarantee of QoS constraints can be met for provision of flows of requests.

2.1.3 Service level agreement

SLA is being utilised in many fields, including computer networks [63], grid computing [59, 62], electronic commerce [64] and SOC [44]. Briefly, an SLA is a contract between a provider and a customer that specifies, usually in measurable terms, the service and the level of service the service provider will furnish. SLAs between service providers and their customers will assure customers that they can get the service they pay for and will obligate the service providers to achieve their service promises [48]. Failing to meet SLAs could result in serious financial consequences for service providers.

2.1.3.1 Preliminary

An SLA usually has the following components [48]:

- **Purpose**: describes the reasons behind the creation of the SLA;

- **Parties**: describe the parties involved in the SLA and their respective roles (provider and consumer);

- **Validity period**: defines the period of time that the SLA will cover. This is delimited by start time and end time of the term;

- **Scope**: defines the service(s) covered in the agreement;

- **Restrictions**: define the necessary steps to be taken in order for the requested service levels to be provided;

- **Service level objectives**: specify the levels of service that both the service consumers and the service providers agree on, and usually include a set of service level indicators, like availability, performance and reliability. Each aspect of the service level, such as availability, will have a target level to achieve;

- **Penalties**: spell out what happens in case the service provider under-performs and is unable to meet the objectives in the SLA. If the agreement is with an external service provider, the option of terminating the contract in light of unacceptable service levels should be built in;

- **Optional services**: provide for any services that are not normally required by the service consumer, but might be required as an exception;

- **Exclusions**: specify what is not covered in the SLA;

- **Administration**: describes the processes created in the SLA to meet and measure its objectives and defines organisational responsibility for overseeing each of those processes.

In different fields, SLAs contain service level objectives regarding different service performance metrics (properties of services). For example, in the field of telecom, SLAs usually indicate properties of telecom services promised by the telecom companies, such as guaranteed uptime, technical response speed to faults, while in the field of computer networks SLAs often contain ISP's promises for bandwidth, delay packet loss rate and jitter. In the field of SOC, the range of properties that can be associated with services is much wider. Any non-functional property which affects the definition and execution of a service falls into the content of SLA, most notably, accessibility, integrity, reliability, regulatory and security [68].

2.1.3.2 SLA specifications

SLAs must be specified precisely and flexibly. For SLAs to be widely employed in the SOC environment, SLA specifications should remove all the ambiguities and inflexibilities.

Precision.

To understand the notion of precision in SLAs, consider the following example. Imagine a service for ordering supplies on the Internet. Assume that "order-supplies" is an operation supported by this service. In order to execute this operation, a customer should send a "purchase-order" document and should obtain a "confirmation" document in response. Now, consider the following guarantee by the supplier: *90% of the time, the time for executing "order-supplies" will be less than 15 seconds.* While this may seem like a reasonable specification on the first look, there are several ways to interpret this:

a. When? When should a service check for the compliance of this SLA? Here are some examples when the evaluation of this SLA can be triggered: (i) whenever a new "order supplies" execution is completed, (ii) after every 10 "order-supplies" executions, (iii) at the end of the day, or (iv) just before the completion of the SLA. It is trivial to note that if this guarantee holds true in one of these cases, it does not necessarily hold in the rest.

b. Which? Which inputs should be considered in evaluating the SLA? In order to check whether the above guarantee is upheld or not, one can consider (i) all executions of "order-supplies" since the establishment of the SLA, (ii) all executions of "order-supplies" since the beginning of that day, (iii) all executions of "order-supplies" no matter who the customer is, (iv) all executions of "order-supplies" initiated by the customer with which the SLA is established, or (v) the last 100 executions of "order-supplies", no matter when they happened.

c. Where? Where is the timing of the execution monitored? It can be monitored at the time of either the customer issuing the request or the service provider handling the request. Usually, customers are interested in SLAs that give guarantees from their perspective as opposed to the service provider's perspective. On the other hand, it is quite hard for the service provider to make service-consumer-side guarantees since that could be influenced by factors that are outside the control of the service provider (e.g., the documents between the service consumer and service provider often flow through other service providers such as ISPs).

d. What and How? In the above example, the metric of interest is the response time from the time when the purchase-order is sent out to that when a confirmation is received. The guarantee is on the 90th percentile. One way to interpret this is that out of every 100 executions, 90 will have a response time of less than 15 seconds. Another way to interpret this is that 90 out of every 100 executions will have a *mean* response time of less than 15 seconds. Metrics that are not as well defined as the typically well known metrics like availability, reliability, cost, and quality will further complicate how an SLA should be evaluated.

Flexibility.

There are many kinds of interactions between service consumers and providers – single request-reply interaction, multiple back-and-forth messages, short-lived interactions, or interactions that last for several hours or days. Services themselves are quite diverse. The metrics that are of interest in an SLA are, in many cases, quite specific to the service. For example, a bookseller would like to provide a guarantee on the number of days it takes to deliver a book. A credit card authorisation service provider would like to provide a guarantee on the security of the transmitted information. The vast diversity in the kinds of interactions as well as metrics that are of interest necessitates the need for a flexible SLA formalisation.

Standards such as WSDL [20, 22] and Web Service Flow Language (WSFL) [60] create flexible and generic interaction models for services. For example, WSDL introduces concepts such as messages, operations, ports, and end points – which are useful for describing the operations of any Web service. Similarly, WSFL introduces the notion of activities and flows – which are useful for describing both local business process flows and global flows of messages between multiple services. So, one way to create a flexible SLA formalisation is to build upon these concepts. In other words, one can create a flexible SLA formalisation by associating "quality metrics" to the formalisations that are already defined in WSDL and WSFL. Here are some examples that show how such association can be done.

- Response time of a Web service operation.
- Response time of a flow.
- Security of an operation.
- Number of times an activity being executed in a flow.
- Cost of executing an operation.

- Availability of an end point.

Another way to guarantee flexibility is to identify generic components in typical SLA specifications, which in turn are extensible so that new components can be defined by extending these base components.

2.1.3.3 SLA specification standards

In light of precision and flexibility, Web Services Policy (WS-Policy) [96], Web Service Level Agreements (WSLA) [52, 66] and Web Services Agreement (WS-Agreement) [4, 65] are the three most recognised SLA specifications.

WS-Policy

Briefly, 'Web Services Policy 1.5 – Framework' defines a base set of constructs that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities [96].

'Web Services Policy 1.5 – Framework' defines a framework and a model for expressing policies that refer to domain-specific capabilities, requirements, and general characteristics of entities in a Web service-based system.

A policy is a collection of policy alternatives. A policy alternative is a collection of policy assertions. A policy assertion represents a requirement, capability, or other property of a behaviour. A policy expression is an XML Infoset representation of its policy, either in a normal form or in its equivalent compact form. Some policy assertions specify traditional requirements and capabilities that will manifest themselves in the messages exchanged (e.g., authentication scheme, transport protocol selection). Other policy assertions have no wire manifestation in the messages exchanged, yet are relevant to service selection and usage (e.g., privacy policy, QoS properties). 'Web Services Policy 1.5 – Framework' provides a single policy language to allow both kinds of assertions to be expressed and evaluated in a consistent manner.

‘Web Services Policy 1.5 – Framework’ does not cover discovery of policy, policy scopes and subjects, or their respective attachment mechanisms. A policy attachment is a mechanism for associating policy with one or more policy scopes. A policy scope is a collection of policy subjects to which a policy applies. A policy subject is an entity (e.g., endpoint, message, resource, interaction) with which a policy can be associated. ‘Web Services Policy 1.5 – Attachment’ [97] defines such policy attachment mechanisms, especially for associating policy with arbitrary XML elements [14], WSDL artefacts [20, 22], and UDDI elements [23]. Other specifications are free to define either extensions to the mechanisms defined in ‘Web Services Policy 1.5 – Attachment’ or additional mechanisms not covered by ‘Web Services Policy 1.5 – Attachment’, for purposes of associating policy with policy scopes and subjects.

WSLA

WSLA is developed by IBM and used to define assertions of a service provider to perform a service according to agreed guarantees for IT-level and business process-level service properties such as response time and throughput, and measures to be taken in case of violation and failure to meet the asserted service guarantees, for example, a notification of the service customer. The assertions of the service provider are based on a detailed definition of the service properties including how basic metrics are to be measured in systems and how they are aggregated into composite metrics. In addition, a WSLA expresses which party monitors the service, third parties that contribute to the measurement of metrics, supervision of guarantees or even the management of violations of service guarantees. Interactions among the parties supervising the WSLA are also defined.

The WSLA language is based on XML, defined as an XML schema.

WSLA can be used by both service provider and service customer to configure their respective systems to provide and supervise their services. This process is called deployment. Each organisation uses its own independent deployment function that interprets the WSLA and takes appropriate action. The deployment step includes creation and parameterisation of the relevant service implementing systems and WSLA supervising services. Also, parts of the WSLA (or derived information) can be passed on to third parties that support WSLA's supervision. After deployment, the WSLA can be enacted by supervising services.

An important aspect of WSLA is its capability to deal with specifics of particular domains and technologies. The language is extensible to include specific types of operation descriptions, (e.g., using WSDL to describe a Web services operation), measurement directive types for specific systems, special functions to compose aggregate metrics and predicates to evaluate specific metrics. The extension mechanism makes use of the ability to create derived types using XML schema. By design, the core of the WSLA language is very compact. To be of immediate use, the WSLA language encompasses a set of standard extensions that allow WSLA authors to define complete agreements that relate to Web services and include guarantees for response time, throughput and other common metrics. For other technical fields, for example, online storage based on the Network Attached Storage (NAS) technology, specific extensions may be defined by interested organisations.

WS-Agreement

WS-Agreement is a Web Services protocol for establishing an agreement between two parties, such as between a service provider and consumer, using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate the discovery of compatible agreement parties.

In a distributed SOC environment, service consumers like to obtain guarantees related to services they use, often related to quality of a service. Whether service providers can offer and meet guarantees usually depends on their resource situation at the requested time of service. Hence, quality of service and other guarantees that depend on actual resource usage cannot simply be advertised as an invariant property of a service and then bound to by a service consumer. Instead, the service consumer must obtain state-dependent guarantees from the service provider, represented as an agreement on the service and the associated guarantees.

Additionally, the guarantees on service quality should be monitored and service consumers may be notified of failure to meet these guarantees. The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of service providers and creating agreements based on creational offers, and monitoring agreement compliance at runtime.

To obtain assurance on service quality, the service consumer or an entity acting on its behalf must establish a service agreement with the service provider, or another entity acting on behalf of the service provider. Because the service objectives relate to the definition of the service, the service definition must be part of the terms of the agreement or be established prior to agreement creation. WS-agreement provides a schema for defining overall structure for an agreement document. An agreement includes information on the agreement parties and a set of terms. The terms may comprise one or more service terms and zero or more guarantee terms specifying service level objectives and business values associated with these objectives.

The agreement establishment process typically starts with a pre-defined agreement template specifying customisable aspects of the documents, and rules that must be followed in establishing an agreement, which we call agreement establishment constraints. WS-agreement defines a schema for an agreement template.

The establishment of an agreement can be initiated by the service consumer side or by the service provider side, and the protocol provides hooks enabling such symmetry. The WS-Agreement specification covers various aspects, particularly the relationship of service level objectives with service description, an agreement specifying alternative service description terms and use of logical grouping operators, and agreement establishment constraints in negotiating service level objectives.

2.2 Research problem analysis

It is believed that the SLA is a powerful tool to address the problem of QoS-aware service composition because SLAs can offer the service consumers contractual guarantee that service providers will provide services with promised QoS. Through managing SLAs at different stages of the composite service provision, the issue of QoS-aware service composition can be addressed in a comprehensive and integrated way. The problem of QoS-aware service composition needs to be addressed before, during and after the service provision. Thus, the SLAs must be addressed accordingly at build time, runtime and completion time. Here we give the definitions of build time, runtime and completion time.

- **Build time**: refers to the time between the point when a service consumer starts to negotiate the SLA with a service provider and the point when an SLA is established between the two.

- **Runtime**: refers to the time between the point when the SLA is established and the point when an SLA is fulfilled or terminated.

- **Completion time**: refers to the time after an SLA is fulfilled or terminated.

Accordingly, an SLA between a service consumer and a service provider goes through three stages: establishment, enforcement and completion. The objective of SLA establishment is to reach an agreement on the terms that are going to be included in the SLA between the service consumer and the service provider. The SLA enforcement is the process during which the terms specified in the SLA are enforced, usually along with the provision of the service that the SLA is attached to. When SLA violations occur, measures must be carried out to adapt the service as well as the SLA. Upon the completion of an SLA, the results of the SLA enforcement, are collected and profiled for future analysis of the service provider's reputation and its capability of enforcing SLAs. The results of the analysis can be used for future SLA management, e.g. to identify trustworthy and untrustworthy service providers.

In service composition scenarios, where multiple component services collectively serve the functional and non-functional requirements of the component services, the problem of SLA management is more complicated and challenging. First, a composite service is composed by several component services. The service consumer needs to establish SLAs with each of the service providers that provide the component services. Second, when an SLA violation in component service occurs, the adaptation of the faulty component service might trigger the adaptation of other component services in order to meet the service consumer's requirements for the composite service. The corresponding SLAs must also be adapted accordingly. Finally, although attached with SLAs, services may still fail due to various reasons, e.g. service providers' incapability, which may cause unexpected consequences, such as economic loss of service consumers that the penalty for the service providers specified in the SLAs cannot compensate. In the service composition scenarios, the failure of the component services may lead to exceptions in the composite services. Therefore, selecting trustworthy service providers for the component services can minimise the failure of composite services.

As analysed above, a comprehensive solution to SLA management for service composition should provide mechanisms to manage SLAs at different stages, specifically establishment, enforcement and completion, in order to guarantee the service consumers' requirements for the quality of the composite services. Specific requirements for each of the stages are as follows.

- **SLA establishment at build time.** Service consumers' QoS requirements for services, as well as service providers' capacities of providing QoS, are unnecessarily static. Instead, they usually have acceptable ranges for the QoS. Therefore, service consumers and providers should be allowed to negotiate with each other to reach an agreement on the QoS. As a service usually involves multiple QoS properties, the negotiation approach for SLA establishment should be able to deal with multiple QoS properties. Moreover, in service composition scenarios, a service consumer usually needs to negotiate with multiple groups of service providers, each of the groups competing for a component service. It must be assured that the finally selected service provides can collectively fulfil both the service consumer's functional and non-functional requirements for the composite service. Also, if a service provider can provide more than one of the component services with better quality, it should be given the opportunity to flexibly express its offers.

- **SLA enforcement at runtime.** When SLA adaptation is being carried out to adapt faulty services, not only the benefit but also the cost caused by the adaptation solution must be considered. The reason is that the adaptation is supposed to be performed only when it is worth performing, i.e. brining more benefit than cost. Moreover, if there are more than one adaptation solutions that satisfy this criterion, the one that brings the most profit should be identified and carried out. The SLA adaptation approach should also be able to identify the subset of the composite services that need to be adapted in order to meet the service consumers' QoS requirements for the composite services. Finally, the SLA adaptation should start within a minimised scope and then expand to a larger one if necessary, in order to limit the impact of the SLA adaptation. Therefore, an approach to confine the scope of the SLA adaptation is necessary.

- **SLA completion at completion time.** The service providers' capability of enforcing SLAs can be evaluated based on their performance over past SLA enforcement. Briefly, the service providers that have higher success rates of historic SLA enforcement gain better reputation, and are more likely to enforce SLAs successfully. To allow the service consumers to identify trustworthy service providers, the result must be collected for SLA profiling upon the completion of the SLAs. The evaluation of the service consumers' trust over the service providers based on their reputation must highlight the service providers' reputation in the long term. Moreover, there are some threats existing in the open SOC environment. The SLA profiling approach must be able to well resist these threats in order to protect the service consumers.

As cloud computing is becoming popular with the support from different communities, including industrial, commercial and academic ones, more services are being provided over the Internet in the form of service. Our research can definitely contribute to the research on cloud computing.

2.3 SLA negotiation

2.3.1 Related work

SLA negotiation is the process of negotiating SLAs between service consumers and providers to establish SLAs. During SLA negotiation, service consumers and providers can propose QoS that are acceptable by them until an agreement is made or the negotiation is terminated. In most cases in the open SOC environment, there are a group of service providers that can provide services with the same functionality yet different quality. Therefore, the selection of services (i.e. selection of service providers) usually depends on the results from SLA negotiation – the service provider that can provide the QoS that is most preferable by the service consumers will be selected. In the service composition scenarios, SLA negotiation becomes much more challenging because the service selection involves multiple groups of candidate service providers – each group for one component services.

With QoS constraints, service selection for service composition becomes a much more challenging problem and has been attracting tremendous research attention in recent years. The work in [11] proves that the optimisation of service selection with multiple QoS constraints is NP-complete. The authors built some computational foundation for the problem of QoS-aware service selection by proposing and evaluating some optimal and suboptimal solutions. Suboptimal solutions [15, 47, 106] and optimal solutions [8, 61, 107, 108] have also been proposed.

[15] uses genetic algorithms to address the issue of QoS-aware Web service composition. Their work focuses on domain-specific QoS properties and customised QoS aggregation formulae. WS-Binder Tool is implemented to support both cross-domain and domain-specific QoS properties and to determine suboptimal

solutions for Web service compositions according to given fitness functions and QoS constraint sets. [47] models QoS-aware Web service selection as a 0-1 knapsack problem or resource constraint project scheduling problem and identifies four approaches to find near-optimal solutions, including greedy selection, discarding subsets, bottom-up approximation and pattern-wise selection. [106] also models QoS-aware Web service selection as a 0/1 knapsack problem as well as a multi-constraint optimal path problem. The authors present heuristic algorithms to find near-optimal solutions in polynomial time. For different composition structures, e.g. sequential, parallel, conditional and loops of services, different algorithms are proposed.

Zeng et al. [107, 108] present AgFlow, a middleware platform that enables quality-driven composition of Web services. The selection of component service is performed to meet the service consumers' requirements on the composite service's QoS modelled from multiple dimensions. Integer programming (IP) is used to compute the optimal plan for composite service executions from several execution paths represented by directed acyclic graph (DAG). Authors in [8] design an Web service-based workflow engine named WSQoSX, which aims at optimising QoS-aware service composition in real-time and under heavy load. A heuristic algorithm is proposed. Firstly, linear programming is used to relax the mixed integer programming formulation of the service composition problem constructed in [107, 108]. Then a backtracking algorithm is used to construct a feasible solution based on the result of the relaxed integer problem. Following the work in [107, 108]. In [5], Ardagna and Pernici formulate the QoS-aware service selection problem as mixed integer linear programming (MILP) problem and adopt loops peeling for optimisation. When a feasible solution does not exist, a QoS negotiation algorithm is suggested to enlarge the solution space of the optimisation problem. In [61], the authors use a different philosophy from works described above to address the QoS-aware service selection problem. They use service composition graph (SCG) to represent the composite service. Then multiple criteria decision making (MCDM) and Simple Additive Weighting (SAW) are used to convert the problem of QoS-aware service selection from a multi-constrained problem to a single-constrained problem. Finally, Dijkstra's shortest-path algorithm is employed to find the optimal solution to the service composition problem.

Work in [77] presents a solution for QoS driven Web service composition using combinatorial auctions, named QWESC (QoS-Aware WEb Services Composer). QWESC facilitates combinatorial auctions to capture the Web service providers' willingness to provide a composite service at a lower price than the total price of stand-alone services that form the composite service, as well as other QoS properties. integer programming formulation is provided to solve the problem of QoS-aware Web service composition. We argue that several essential issues in designing a combinatorial auction for Web service composition have not been addressed, e.g. bidding language, winner determination and ask QoS generation. QWESC is far from comprehensive and practical.

2.3.2 Discussion

Current research except [77] assumes that the quality of a component service that a service provider can provide is static. Actually in the open SOC environment the service providers' capacities of providing QoS may dynamically change over time and the QoS they are willing to offer may vary depending on how many component services they can win in the SLA negotiation. In this regard, mechanisms that can motivate service providers to provide satisfactory QoS should be employed in service selection in the SOC environment. In light of the above, combinatorial auction is widely recognised as a promising approach to provide the bidders (i.e. service providers) with flexible bidding options, hence motivating the bidders to compete for the component services involved in the SLA negotiation [27].

None of the current research addresses the following three issues: 1) service providers' capacities for providing QoS may change; 2) the more service providers competing for a service, the more likely the service consumer will be able to obtain favourable QoS; and 3) service providers tend to offer better QoS to win more services.

2.4 SLA adaptation

2.4.1 Related work

Composite services are often executed in volatile environments where the QoS parameters of the participating component services might change during the execution of the composite services. Recently, research has been carried out on adapting composite service in volatile SOC environments. However, current approaches do not consider the internal logic of the composite services and the impact of adaptation for a single component service – the faulty one – on other component services. Other than QoS parameters, effective adaptation requires specific information about the component services in terms of their positions and interactions with other component services.

Harney and Doshi [38] present a mechanism called value of changed information (VOC) which computes the estimated value brought by the changes of the composite service and compares the value to the cost required to make the changes. The update is performed only when it is expected to pay off. In [39], Harney and Doshi utilise service expiration times to reduce the computational overhead of adaptation. The improvement is based on the insight that service providers often keep the quality of their services at a certain level for a period of time. A new approach is proposed, namely VOC with expiration times (VOCε). VOCε manages to reduce the computational burden of adaptation. However, while considering adapting one individual component service, the effect of VOC and VOCε is limited on just one component service without taking into account the composite service. In other words, VOC and VOCε facilitate only local adaptation solutions for composite services which cannot guarantee the satisfaction of the service consumers' requirements for the composite services.

Chafle et al. [17] introduce adaptation on different levels, including the instance level, logic level and physical level. Multiple backup workflows are prepared to substitute the failed components or workflows at any moment. By enabling this, workflow systems can adapt to environmental changes. Verma et al. [98] introduce a suite of stochastic optimisation based methods, including centralised and decentralised ones for adapting business process modelled as Markov decision processes. Exogenous events and inter-service constraints are both taken into account when performing the adaptation. Narendra et al. [79] use the aspect-oriented programming (AOP) technology to dictate modifications in component services in order to meet non-functional requirement changes in the composite service.

2.4.2 Discussion

None of the above approaches considers the cost that may occur in the adaptation of composite services and may generate worthless adaptation solutions which bring cost more than profit. Also, the impact of the SLA adaptation for one component service on other component services has not been considered properly. When SLA adaptation is required, it is sometimes essential to adapt a group of component services and the attached SLAs because otherwise the QoS requirements of the composite services cannot be met. Besides, the adaptation is supposed to be confined within a certain scope, usually the smaller the better, to limit the impact of faulty services on other services and the cost and complexity of the adaptation.

2.5 SLA Profiling

2.5.1 Related work

Service transactions, although attached with SLAs, may still fail due to various reasons – intentionally or accidentally – in the open and volatile SOC environment. In service selection, service consumers often have to estimate the trustworthiness of the service providers without or with limited prior experience and knowledge about them. Moreover, the SOC environment exposes service consumers to various threats, e.g. malicious reputation manipulation and QoS abuse. A trust system which profiles service providers based on their historic performance on SLA enforcement can help service consumers identify trustworthy service providers and protect them from the threats described above[1].

[1] Although profiling may involve information on service providers from various aspects, we

Reputation-based trust research is being carried out in several distinct areas, most notably computer science and economics. An overview of many trust systems for online service provision can be found in [46]. And many key issues in reputation-based trust evaluation mechanisms in e-commerce environments are discussed in [99].

In the domain of distributed computing, several reputation systems have been proposed. Cornelli et al. [26] propose P2PRep, a P2P protocol which complements Gnutella, an existing P2P file-sharing protocol. In P2PRep, peers can keep track of and share information about the reputation of other peers. However, there are no formalised approaches to evaluate the reputation and credibility of the peers and no experimental evaluation is provided. Damiani et al. [28] enhance their previous work in [26] by introducing XRep, a distributed polling protocol that inquires the P2P network for peers' opinions (votes) on targeted resources. Votes are clustered based on IP address to prevent Sybil and collaboration attack. XRep focuses on supporting anonymous and secure services while preserving anonymity to a degree. Kamvar et al. [51] propose EigenTrust, a distributed method for P2P file-sharing networks. Unique global trust values are computed and assigned to each peer in the network. EigenTrust requires pretrusted peers in the network to address the collusion problem. The limitation of their approach is that pretrusted peers may not always be available in all cases. Xiong et al. [104] propose PeerTrust, a feedback based trust management system. PeerTrust incorporates three basic trust parameters (the feedback, the total number of transactions a peer performs and the credibility of the feedback sources) and two adaptive factors (transaction context factor and the community context factor) into computing the trustworthiness of peers. However, the solution adopted to measure feedback credibility, namely Trust-Value based credibility Measure (TVM), assumes that trustworthy nodes be more likely to be honest on the feedback they provide. This assumption is not generally true because peers may send incorrect feedbacks to ruin the reputations of its competitors. Srivatsa et al. [92] propose TrustGuard, a safeguard framework in decentralised

focus on utilising their historic performance in this research.

overlay networks, aiming at countering various vulnerabilities in reputation management. In TrustGuard, a peer rates credibility of feedback from other peers using a personalised similarity measure (PSM). Feedbacks that are similar to the peer's own are considered more credible. This method is limited in the cases where peers with long-term reputation are preferable and credible. For example, if a service provider delivers a poor service transaction to a service consumer by accident, malicious peers can flood negative feedbacks to rapidly ruin the service consumer's trust over the service provider.

Wang et al. [101] presents a model which incorporates transaction amount into trust evaluation. A simple method is proposed to measure the difference between old and new transaction amounts. However, no amount-related malicious behaviour is modelled and no experimental results are presented to validate their approach.

2.5.2 Discussion

In the open SOC environment, service providers may not always successfully enforce the SLAs due to various reasons. SLA violations occur from time to time, intentionally or accidentally. In service selection, the QoS can be negotiated over, but the success rate of a service transaction cannot be provided by the service provider. This problem is especially severe in the service composition scenarios where a composite service is composed by several component services. The failure of an individual component service may result in exceptions of the composite service. Therefore, when selecting service providers, service consumers usually prefer those who are most likely to successfully enforce the SLAs.

In addition, the open and volatile SOC environment exposes service consumers to various threats, including malicious reputation manipulation [58] and QoS abuse [101].

In service selection, approaches should be provided to help the service consumers estimate the trustworthiness of the service providers, as suggested but not specified by [5, 54, 107]. However, it is difficult for a service consumer to determine how much it can trust a service provider due to the lack of sufficient experience and knowledge about the service provider. A direct approach to address this issue is to use a trust system which collects and processes feedbacks about service providers' past behaviour [51, 88, 92, 104]. However, to the best of our knowledge, no trust system has been tailored for service selection in the SOC environment and the threats described earlier have not been sufficiently addressed.

2.6 Requirements analysis

After the literature review of the state-of-the-art and the in-depth analysis of the unsolved problems in the QoS-aware service composition, it can be seen that QoS-aware service composition is challenging yet essential to supporting service applications. While current research on QoS-aware service composition is incomplete by focusing on calculating the quality of the composite services based on static quality of the component services, we advocate that the problem of QoS-aware service composition should be addressed at build time, runtime and completion time. Corresponding approaches for SLA negotiation, adaptation and profiling should be provided.

2.6.1 A motivating example

This section presents a simplified example of trading business process to analyse the research requirements in this thesis.

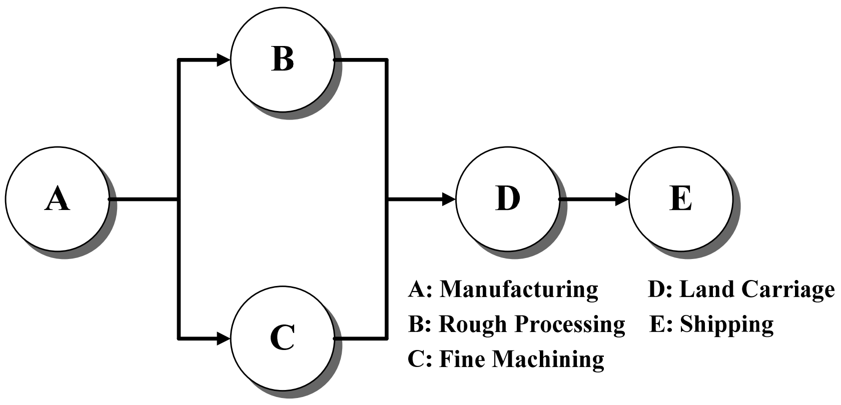


Figure 2.2 A composite service composed of five component services

This trading business process, shown in Figure 2.2, can be modelled as a composite service which consists of five component services – manufacturing, rough processing, fine machining, land carriage and shipping.

The business process starts with a manufacturing service, which produces the goods required by the service consumer. When the goods are produced, half of it is delivered for rough processing and another half for fine machining. When both the rough processing and fine machining services are completed, the goods are delivered to the service consumer by a land carriage service and then a shipping service.

Although the services in the above example are domain specific, the services we are dealing with in this research are generic.

2.6.2 SLA negotiation at build time

Current research on QoS-aware service composition focuses on build time QoS guarantee for service composition – aggregating the QoS of the composite services based on static QoS of the component services given by the service providers before the service provision. However, in the open SOC environment, the QoS that service providers can offer and guarantee during service provision is not static but dynamic – it largely depends on the service providers' resource situations. Therefore, service consumers and providers should be given the opportunity to flexibly exchange their expectations and offers over the terms in the SLAs.

Also, the popularity of e-business and e-commerce pushes the world market closer and closer to the perfectly competitive business environment [25]. In such an environment, competition between the service providers gives the service consumers leverage when negotiating with the service providers. Thus the more service providers there are vying to win the order for a service, the more likely the service consumer will be able to obtain favourable QoS. Apparently, better quality of the component services will result in better quality of the composite service. Hence, an effective approach for SLA negotiation should involve multiple candidate service providers if there are any, create competition between them, and allow them to revise their bids in order to win the order for the service.

Moreover, a service provider's profit increases as the number of obtained orders for services increases. This gives the service providers, which can provide multiple types of services, a very appealing incentive to propose good QoS in order to win more orders for the component services in SLA negotiation for service composition. Those service providers who are capable of providing complementary component services will be able to gain competitive advantages in the competition for the component services. A set of services are complementary when they can be provided with better quality by a single service provider than multiple service providers [27].

Due to the above characteristics, SLA negotiation in service composition scenarios can be very complicated so that an approach that supports effective and efficient service selection for service composition is needed. It has been long proven that auction is effective and efficient in a perfect competition environment [74, 102]. In many e-commerce cases, such as Amazon, eBay and Overstock.com, auction has been proven to be able to well capture buyers and sellers' preferences and to ensure their satisfaction and the auction revenue. We advocate that an auction-based approach is also applicable to SLA negotiation for service composition. In an auction for service composition, multiple candidate service providers bid for the component services and the service consumer selects the final winners for service provision. In the open SOC environment, a dynamic service selection approach for service composition based on auction should address the following key issues:

1. **Bidders biding for multiple component services.** Traditional single-item auctions are unsuitable for service composition scenarios, because they ignore the fact that a service provider (referred to as a bidder in an auction) can and wishes to bid for multiple component services (referred to as items in an auction) at the same time. And a bidder's offer for the quality of the component services, including the quality and the prices[2] of the component

[2] Usually the price is not recognised as the quality of a service, but it is one of service

services, is dependent on the number of items it wins. Therefore, the auction-based service selection should allow the service providers to place bids on combinations of component services flexibly.

2. **Multiple properties of items for auction.** In traditional auctions, items usually have one property, i.e. price. In the SOC environment, besides the price, a service consumer usually has constraints and preferences for other QoS properties of the composite service, such as execution time, throughput, availability, etc. Therefore, the auction-based service selection for service composition should allow bidders to propose offers for multiple properties of the component services. The auction should be aimed at meeting the service consumer's multiple dimensional non-functional requirements for the composite service.

Take the composite service presented in Section 2.6.1 for example, the service consumer demands to obtain the goods as soon as possible at an acceptable price. In order to achieve these goals, the service consumer needs to negotiate with each of the candidate service providers over the execution time and prices of the component services.

As there might be factories that can provide both rough processing and fine machining services, and logistics companies that can provide both land carriage and shipping services, the following four issues may have major impacts on the selection of services.

1. The quality of the composite service is determined by the aggregation of the quality of the component services. For example, the total execution time of the composite service depends on the execution time of the five component services.

---

consumers' most important concerns about the services. Hence, we consider it as a QoS property when evaluating the quality of a service.

2. A bidder may have different preferences and bidding strategies when it bids for different numbers of component services. For example, a factory may charge a lower price for the rough processing service if it also wins the fine machining service.
3. Multiple QoS properties need to be considered when the winning bidders are being determined. For example, different service consumers might have different preferences for price and execution time. As traditional auctions, such as English auction, Dutch auction, First-Price Sealed-Bid Auction and Vickrey auction [73], often involve a single item with a single property only (normally price), they cannot be applied directly to the service composition scenarios described here.

4. Different service consumers might have different criteria for determining when the auctions complete. For example, the service consumer can require the auction to complete when the overall price and execution time of the composite service achieved from the current winning bids meet its pre-specified requirements. A completion criterion can also be set with a preset timer. By setting a timer, the auction can be forced to complete when the time allowed for the auction is up. The auction can also be configured to continue as long as there are new and better bids. When there is only one bid left for each of the component, the auction completes and the standing winning bidders (referred to as final winning bidders) are allocated the orders for the component services with the QoS specified in the remaining bids (referred to as final winning bids).

To summarise, an SLA negotiation approach based on auction is needed to support SLA establishment, which allows candidate service providers to propose offers for combinations of component services with multi-dimensional QoS, and to provide different completion criteria to be set to cater for service consumer's requirements.

2.6.3 SLA adaptation at runtime

Runtime QoS guarantee for composite services is also a very important issue. When exceptions in component services occur during the service provision, e.g. current service providers claim to be incapable of providing promised services or service providers fail to deliver results as specified in the SLA, the faulty component services need to be adapted (updated or replaced), which might lead to modifications to the component services, including their interfaces, implementations and quality. Therefore, an approach is needed to keep service consumers' QoS requirements for the composite services met. Sometimes, adapting the faulty component services only cannot guarantee the required quality of the composite services. In this case, it is necessary to extend the scope of the SLA adaptation – taking more component services into adaptation instead of just the faulty one. For example, it takes a certain amount of time to recover the failed service, which, with the addition of pre-specified time consumption of the remaining unexecuted component services, will lead to violation of the service consumer's requirements for the time consumption of the composite service and thus would require adaptation of more unexecuted component services.

So far, little efforts have been placed on the adaptation for composite services which considers the impact of the adaptation for one service on other services. A comprehensive adaptation approach is needed. Besides, adaptation is supposed to be confined within a certain scope, usually the smaller the better, to limit the impact of failed services on other services and the cost and complexity of adaptation. Hence, defining the scope and then find an optimal adaptation solution in the defined scope is considerably important.

In order to illustrate the requirements for and the complexity of adaptation for composite service, we use the example presented in Section 2.6.1. The SLAs for the manufacturing, the rough processing, the fine machining, the land carriage and the shipping services have been established between the service consumer and the service providers, which collectively fulfil the service consumer's QoS requirements for the composite service. Suppose that the land carriage company suddenly claims a delay and cannot deliver the goods within the timeframe as promised in the SLA, this delays the whole process. Therefore, the composite service must have the adaptation capability to recover from this exceptional situation. In addition, it is possible that the schedule of the shipping company may be interfered. For example, the shipping company may have specified the related term in the SLA such as "the shipping service will be completed with 10 days during May 1 to May 30." With the land carriage service rearranged, this constraint may no longer be held. Therefore, the shipping service needs to be adapted by either updating the SLA between the service consumer and the shipping company or selecting another shipping company that is capable of providing the shipping service as required.

In the adaptation process, cost applies, e.g. the service query cost and the negotiation cost. The service consumer may adapt the land carriage service by either renegotiating with the land carriage company or selecting another land carriage company to replace the original land carriage company. When deciding the adaptation solution for land carriage service, the service consumer needs to estimate both the value and cost that each adaptation solution may bring. Meanwhile, the service consumer needs to compute the probability that the adaptation of land carriage service will lead to interference of the shipping service and the corresponding value and cost from adapting the shipping service. Based on the results, the service consumer can determine the adaptation solution, e.g. how to adapt the land carriage service, whether to adapt the shipping service and if so, how.

In this case, the optimal solution is not necessarily selecting the companies with the best quality but the one with the best tradeoff between the value and cost. This motivating example reveals three important factors for selecting the optimal adaptation solution. First, the service consumer must estimate the tradeoff between the value and cost brought by different solutions. Second, the service consumer must estimate the impact of adapting the faulty service on other services to see if they need to be updated. Third, the value of cost caused by the adaptation of other services must be considered.

To summarise, an SLA adaptation approach is needed to support SLA enforcement, which aims at guaranteeing service consumers' QoS requirements for the composite services by adapting a group of component services within a confined scope and considers both the value and cost of the adaptation solutions.

2.6.4 SLA profiling at completion time

Although associated with SLAs, the service transactions might not be performed exactly as required in the SLAs. SLA violations occur from time to time due to various reasons, e.g. inherent unreliability of the underlying Internet and internal infrastructures of the service providers. Generally speaking, the service providers with lower success rates of past SLA enforcement tend to fail SLA enforcement more frequently. In order to guarantee the quality of the composite services, service providers' capability of successfully enforcing the SLAs must be taken into account. When selecting service providers, those service providers with better reputation, meaning higher success rates of past SLA enforcement, should be more trustworthy. Therefore, upon the completion of SLAs, the results of SLA enforcement should be recorded and profiled in an effective way to support the analysis of service providers' reputation and capability of enforcing future SLAs.

Take the composite service presented in Section 2.6.1 for example, if the land carriage company fails in delivering the goods on time as specified in the SLA, its poor performance must be recorded. Based on a series of such records, the land carriage company can be profiled and its reputation can be evaluated. Next time a service consumer considers selecting this land carriage company, it can evaluate its trust over the land carriage company based on its reputation. On the other hand, if any of the services are provided successfully as specified in the SLAs, the results of the service transactions must also be recorded for future profiling. When selecting service providers during the SLA negotiation and adaptation for any of the component services, the service consumer can incorporate the reputation of the candidate service providers as one of the selection criteria. For example, a land carriage company with a lower reputation – attained from poor performance over time – is usually less likely to be selected by the service consumer as it cannot earn the service consumer's trust.

Besides the abovementioned issues, the following two threats existing in the SOC environment must be addressed through SLA profiling.

1. **Malicious reputation manipulation.** Malicious service providers may manipulate service consumers through such as bribery to provide incorrect ratings in order to boost their reputations or to ruin their competitors' reputations. Malicious service providers can also inject incorrect ratings by faking service consumers.

2. **QoS abuse.** Malicious service providers may strategically alter their behaviour in QoS offering in order to obtain profit. For example, malicious service providers may use successful service transactions with small amounts to build its reputation, obtain service consumers' trust and then defraud them of their money with fraudulent service transactions with large amounts. Genuine service providers may also strategically alter their behaviour under a certain circumstance, e.g. given an order of a service transaction with an unusually large amount, a genuine service provider might make the transition into being a malicious service provider and then provide a fraudulent service transaction.

Take the composite service presented in Section 2.6.1 for instance, in order to boost its reputation, a malicious land carriage company can manipulate or fake several clients to get good ratings over poor or fake service transactions. These manipulative ratings will jeopardise the accuracy of service consumers' evaluation of their trust over the land carriage company. If such service providers are selected to provide the component services, the actual success rate of the composite service will be lower than the service consumers estimate it would be. In addition, a malicious land carriage company can perform some service transactions with low transaction amounts to boost its reputation. By doing so, it can take advantage of its high reputation to attract potential big clients and defraud them of their money with fraudulent services with large amount. This may cause violations of the service consumer's requirements for the composite service because 1) the land carriage service has to be rearranged; and 2) the shipping service may have to be rearranged too.

To resist the threat of malicious reputation manipulation, service consumers' trust over service providers should be evaluated based on service providers' long-term reputations which are evaluated based on service providers' long-term performance. Long-term reputations can smooth out short-term fluctuations and highlight long-term trend of service providers' reputation. Another benefit of basing service consumers' trust over service providers' long-term reputation is that it encourages service providers' trustworthy and consistent behaviour at present.

To resist the threat of QoS abuse, when evaluating service consumers' trust over individual service transactions, the QoS of the past successful service transactions that the service providers have performed must be taken into account. By comparing service providers' current offers for the QoS against their historic QoS, potential fraudulent service transactions can be identified and avoided.

To summarise, a service trust system based on SLA profiling upon SLA completion is needed to facilitate reputation-oriented service selection. By employing the service trust system, service consumers should be able to identify trustworthy service providers, and the threats described above, including malicious reputation manipulation and QoS abuse, should be well resisted.

2.7 Summary

In this chapter, the analysis of the research problems has been presented. The major problem of current research on service composition is the lack of a comprehensive solution that addresses the issue of service composition at different stages of the composite service provision. The literature in relation to the issues that arise at different stages of the composite service provision has been intensively reviewed. Based on the problems analysis and literature review, the research requirements have been analysed in detail.

Chapter 3

Service level agreement lifetime

In this chapter, we introduce the SLA lifetime, the major SLA operations at each stage of the lifetime and the interactions among the operations.

This chapter is organised as follows. Section 3.1 gives a general introduction about the SLA lifetime and the operations at each step. Sections 3.2, 3.3 and 3.4 introduce SLA negotiation, SLA adaptation and SLA profiling respectively. Section 3.5 discusses the interactions among the three SLA operations. Finally, Section 3.6 summarises this chapter.

3.1 Introduction

SLA management is a complicated process involving operations at different stages of the service provision. Briefly, before the service provision, an SLA needs to be established between the service consumer and the service provider. The key objective of SLA establishment is to achieve an agreement between the service consumer and the service provider on QoS guarantees and constraints, which represent the service consumer's QoS requirements and the service provider's QoS promises. During the service provision, the established SLA needs to be enforced. The key operations of SLA enforcement include monitoring of actual QoS values, evaluation of the SLA compliance by comparing actual QoS values against agreed values in the SLA, detection of SLA violation and SLA adaptation. SLA violations can be remedied by SLA adaptation. Upon the completion of an SLA, the enforced

SLA needs to be profiled to provide valuable information for future SLA management. The key operation of SLA profiling is to manage service trust. Clearly, operations at these three SLA management stages are inherently interrelated and should be provided in an integrated way. SLA management supported by provision of incomplete SLA operations or operations in an isolated manner would be either ineffective or inefficient. Therefore, we developed a model of SLA management which logically connects all the SLA operations, as presented in Figure 3.1.

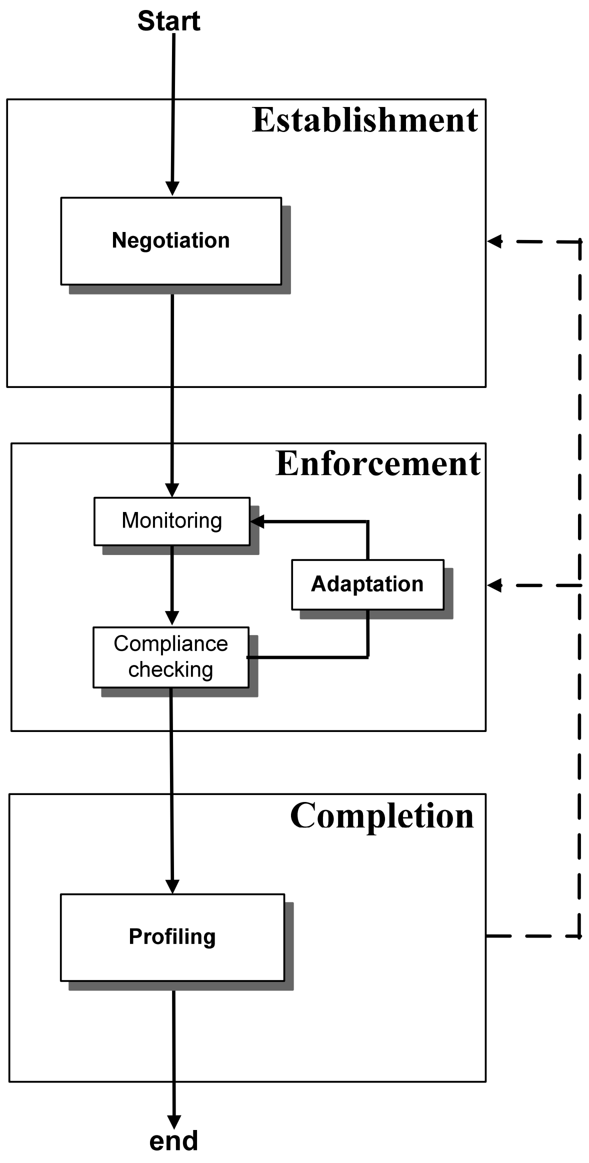


Figure 3.1 Lifetime of an SLA

As depicted in Figure 3.1, the lifetime of an SLA includes three major steps: establishment, enforcement and completion. An SLA is established between the service consumer and the service provider before the service provision. In this step, we develop a negotiation approach to support SLA establishment. During the service provision, the attached SLA is enforced. In this step, SLA violations are detected by whoever it is in charge of SLA monitoring and compliance checking. We develop an SLA adaptation technique to support SLA enforcement. Upon the completion of an SLA, the results of SLA enforcement are collected for SLA profiling in order to enhance future SLA management, specifically SLA negotiation and adaptation.

We have developed the SLA negotiation, adaptation and profiling approaches to support SLA establishment, enforcement and completion respectively. Chapters 4, 5 and 6 will present the details. SLA monitoring and compliance checking are not included in our research, because they are usually delegated to third authentic parties for credibility and security purposes.

3.2 SLA negotiation

The major objective of SLA establishment is to select, among (possibly many) functionally equivalent service providers, a service provider with essential QoS that is "right" for the usage context. In the case of service composition, a set of service providers will be selected for component services in the composition that are collectively "right" for the usage context [28]. Whether service providers can offer essential quality for the specific service provision context depends on their resource situations at the requested time of service. Therefore, QoS that depends on actual resource usage cannot simply be advertised as an invariant property of the service and then bound to by a service consumer. Instead, the service consumer must obtain state-dependent guarantees from the service provider, represented as an agreement on the service and the associated QoS guarantees.

We develop a negotiation technique to facilitate SLA establishment. The use of negotiation as a means of establishing service contracts has been a topic of considerable interest for quite some time [18]. In this research, negotiation refers to the process by which the service consumer and the service provider arrive at a desired outcome for QoS. In the context of service composition, because the service consumer must negotiate with multiple service providers for multiple services, a coordinated negotiation approach is needed, enabling the outcomes of individual negotiation to be achieved collectively.

When an agreement on all the terms in the SLA attached to the service is reached, the SLA can be established between the service consumer and the service provider. At this point, the SLA establishment is completed.

3.3 SLA adaptation

Because of the dynamic features of the services, the actual QoS may vary over time. Therefore, it is essential to monitor the service provision closely, check the compliance of the SLA and detect any QoS violation, in order to finally adapt service and SLA to these changes at runtime. This will enable the agreed SLA to be enforced despite the changing environment, thereby ensuring that the service remain "right" throughout the service provision.

The first issue to be addressed in SLA enforcement is SLA monitoring. Due to authority and credibility issues, the task of SLA monitoring is usually commissioned to supporting third parties. Therefore, the implementation of SLA monitoring is not included in our research. Due to the same reason, the second issue to be addressed in SLA enforcement, SLA compliance checking, is also excluded from our research.

The second issue to address in SLA enforcement is SLA adaptation. The key to effective SLA adaptation is the accurate analysis of the impact of any SLA violation on the service. Adaptation in this research is generally achieved by updating or replacing the service and the attached SLA to best restore the faulty service.

When the SLA is fulfilled successfully or terminated, the SLA enforcement is completed.

3.4 SLA profiling

Results collected upon the completion of SLA enforcement can provide useful information about the service providers' historical performance in various contexts, for future SLA negotiation and adaptation. SLA profiling can deal with the issues of maintaining up-to-date service profiles to support the evaluation of service consumers' trust over service providers based on their reputation. Well-managed service profiles can largely support decision making in tasks such as SLA negotiation and SLA adaptation. Therefore, an SLA profile is much appreciated in the comprehensive and integrated solution to SLA management for service composition.

The key issue in SLA profiling is quantifying and comparing the trustworthiness of service providers, based on their historic performance over SLA enforcement, via the management of service trust.

3.5 Discussion

SLA negotiation, adaptation and profiling are the three major operations in the lifetime of SLA. Working together, they provide a comprehensive and integrated solution to lifetime SLA management for service composition.

SLA negotiation usually happens at build time, i.e. before the service provision. However, during the execution of the composite services, SLA violations will trigger SLA adaptation, in which component services might need to be updated or replaced. During SLA adaptation, the QoS properties of the faulty component services must be renegotiated between the service consumer and the original service provider, or negotiated between the service consumer and other service providers, or the combination of the two.

The aim of SLA negotiation and adaptation is to select appropriate service providers for the services in order to meet the service consumers' requirements. Service providers' reputation, as one of the service consumers' most important concerns, needs to be taken into account when selecting service providers. An effective way to identify trustworthy service providers can help enhance SLA negotiation and adaptation. It can also protect the service consumers from the jeopardy caused by disreputable service providers. An effective way to identify trustworthy service providers is through their historic performance over past SLA enforcement. Upon the completion of a service transaction, the outcome of the service transaction can be compared against the attached SLA to evaluate the SLA enforcement. With analysis of service providers' performance over past SLA enforcement in long term, their reputation can be quantified and used for service selection.

3.6 Summary

In this chapter, we have introduced the lifetime of an SLA, and the major SLA operations that constitute the lifetime SLA management. We have also discussed the interactions among the three major SLA operations: namely SLA negotiation, adaptation and profiling.

Chapter 4

Flexible negotiation in SLA establishment

As discussed in Section 2.3 and Section 3.2, SLA establishment, as the first stage of the lifetime of an SLA, needs the support of SLA negotiation before the service provision to reach an agreement between the service consumer and the service provider. SLA negotiation in service composition scenarios is a more complicated issue as it often involves multiple QoS properties of the composite service and the component services, and multiple service providers for each of the component services. In this chapter we introduce Combinatorial Auction for Service Selection (CASS), an innovative approach that supports effective and efficient SLA negotiation for service composition based on combinatorial auction.

This chapter is organised as follows. Section 4.1 introduces some preliminaries about combinatorial auctions. Then Section 4.2 presents the procedure of CASS, followed by supporting mechanisms discussed in Section 4.3. Experimental evaluation is given in Section 4.4. Discussion about how CASS can be used for SLA adaptation and how CASS can be enhanced by incorporating reputation consideration based on SLA profiling are presented in Section 4.5. Finally, Section 4.6 summarises this chapter.

4.1 Preliminaries

4.1.1 Combinatorial auction

Combinatorial auctions are auctions where bidders can bid for combinations of items [27]. It has been proven that combinatorial auctions can lead to more effective allocation of multiple items than traditional auctions [90]. In combinatorial auctions for service composition, let $K = \{1, 2, \ldots, m\}$ be the set of items for auction and $I = \{1, 2, \ldots, n\}$ be the set of the participating bidders. An arbitrary bid B_i proposed by a bidder can be represented by a tuple (S_i, V_i), where $S_i \subseteq K\ (S_i \neq \varnothing)$ represents the set of items the bid is placed on, and $V_i = (v_{i1}, v_{i2}, \ldots, v_{ip})$ denotes the QoS property vector (e.g. price, execution time, etc.) for S_i. The key problem of a combinatorial auction for service composition is to find the set of bids from all proposed bids that can maximise the auction revenue which is evaluated according to the service consumer's preferences for the QoS properties of the composite service. Optimised quality of the composite service can be achieved from the winning bids that maximise the auction revenue. The issue of getting the optimised QoS is essentially equivalent to weighted set packing problem [91] and hence the winner determination problem for combinatorial auctions is NP-complete [89].

4.1.2 Bidding language

The bidding language for a combinatorial auction specifies the structure of bids and the operators for combining the bids. A bidding language should allow bidders to bid in a straightforward manner. It should not be overly expressive because that would unnecessarily increase the complexity of the auction. For example, if a bidding language requires a bidder to attach a value to each possible combination, then given m items, a bidder has to submit a bid of size $C_m^1 + \ldots + C_m^m = \sum_{n=1}^{m} C_m^n$, which is only practically feasible for very small m. There are two general bidding languages for combinatorial auctions, exclusive-or (XOR) and additive-or (OR) [12]. The former allows bidders to submit bids in the form of *(S_1, p_1) xor (S_2, p_2) xor ... xor (S_n, p_n)* stating the semantics "I will buy at most one of these combinations." while the latter allows bids in the form of *(S_1, p_1) or (S_2, p_2) or ... or (S_n, p_n)* stating the semantics “I’ll buy one or more of these combinations”. In this research, the XOR bidding language is adopted because it is expressive and straightforward in expressing the complementarity of items and has emerged as the definitive choice in recent combinatorial auction designs [27].

4.2 CASS procedure

In CASS, the set of component services that constitute a composite service are open for auction. Bidders (service providers or representatives of service providers) can place bids on individual or bundled items (component services) round by round and the final winning bidders are selected to provide the component services. CASS is a simultaneous multi-round auction, which allows bidders to update their bids round by round. This section presents the procedure of CASS.

The five steps of CASS are depicted in Figure 4.1, as explained below.

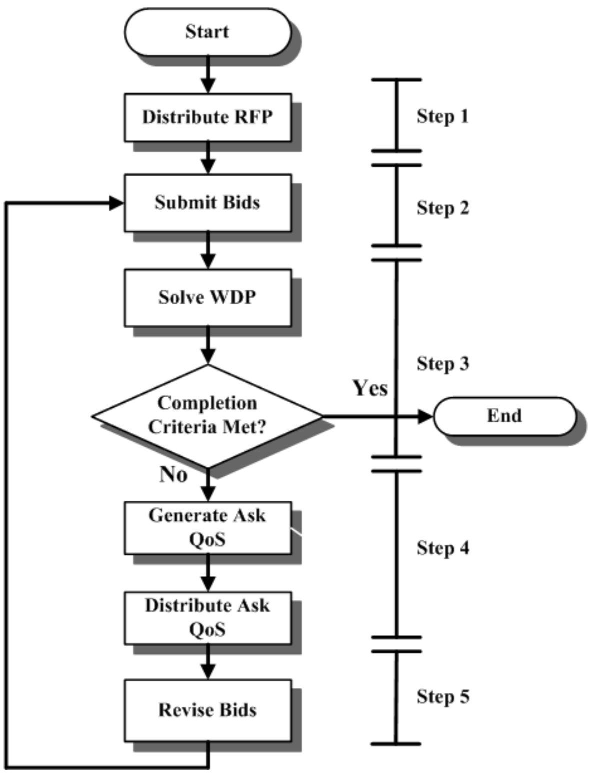


Figure 4.1 CASS procedure

Step 1: At the start of the auction, a predetermined set of component services are established and included in the Request for Proposal (RFP) which is then distributed to potential service providers. The main content in the RFP specifies the composite service consisting of a set of component services. According to the RFP, potential service providers can decide if they are interested and which component services they will bid for.

Step 2: The service providers interested in participating the auction submit their bids and become candidate service providers.

Step 3: The auctioneer, i.e. the service consumer or an agent of the service consumer, receives the submitted bids and solves the Winner Determination Problem (WDP). If the completion criterion is met, the auction completes.

Step 4: If the auction needs to continue, the auctioneer generates and then distributes the ask QoS, which contain guiding information such as the current winning bids and the required bids for the bidders to enter the next round, to the bidders.

Step 5: According to the ask QoS, the bidders can choose to accept it or propose boosted bids, i.e. bids that specify better QoS than required in the ask QoS, to enter the next round, or simply quit the combinatorial auction.

Steps 2 to 5 are repeated until the completion criteria are met. Sometimes a final deal-sealing round will be triggered to determine the final winning bidder for an item or a combination of items. The details will be provided in Section 4.3.1.3.

4.3 CASS mechanisms

In this section, we present the mechanisms that support CASS, including bidding constraints, QoS model, bid evaluation, winner determination, ask QoS generation and completion criteria.

4.3.1 Bidding constraints

In CASS, three bidding constraints are imposed: XOR bidding language, dynamic minimum increment and final deal-sealing round.

4.3.1.1 XOR bidding language

Given the reason described in Section 4.1.2, CASS adopts the exclusive-or (XOR) bidding language with which a bidder can bid for multiple items and combinations of items, but can win at most one of them at the end of the auction. A bid is submitted in the form of *$((S_1, V_1)$ xor (S_2, V_2)* xor ... *xor $(S_n, V_n))$*, where $S_i \subseteq K$ denotes the items or combinations of items that the bidder is bidding for and V_i is the vector of QoS properties for S_i. The component units that assemble the bids, in the form of (S_i, V_i), is referred to as *component bids*.

In the open SOC environment, services usually have multiple QoS properties and service consumers and providers have diversified preferences for those QoS properties. CASS allows bidders to express their component bids associated with a vector V_i to specify the QoS properties of corresponding items or combinations of items S_i. Note that the additive QoS properties in component bids, e.g. price, execution time, etc, can be non-linear. For example, a bidder can propose a bid which involves three component bids: (S_1, V_1) *xor* (S_2, V_2) *xor* (S_3, V_3), where $S_3 = S_1 \bigcup S_2, S_1 \bigcap S_2 = \varnothing$ and $V_3 \neq V_1 + V_2$ [3]. This allows bidders to apply discounts to negative properties of or premiums to positive properties of the component services according to the complementarities of the items they are bidding for. The definitions of negative and positive properties of services will be given in Section 4.3.2.

4.3.1.2 Dynamic minimum increment

In each round, the auctioneer distributes ask QoS to the bidders. The bidders can choose to accept the ask QoS or revise their bids. And those that cannot afford the ask QoS will have to quit the auction. Ask QoS is used to coordinate the auction process. When generating the ask QoS, minimum increments (or decrements in the

[3] Here "+" means the aggregation of QoS properties.

case of negative properties), denoted by σ, can be imposed on the properties of the items. For example, "at least an increase of price by 5% than the previous bid" requires a bidder to bid a monetary price at least 5% higher than its previous bid in the last round.

In the open SOC environment, some functionally-equivalent service providers may have huge different reserved capacities for the QoS while some others may have similar ones. In order for a combinatorial auction for service composition to be efficient, bidders with low reserved capacities should be filtered fast while bidders with high reserved capacities should be given more chances to propose better bids.

In addition, the most notable approaches, which adopt Integer Programming (IP) and Mixed Integer Linear Programming (MILP) to solve the problem of QoS-aware service composition [5, 107, 108], are subject to computational uncertainty and scaling-up concerns. Specifically, there is no guarantee that the solution can be found in a reasonable amount of time when the number of bidders and items becomes larger.

To address the above issues, we design a novel mechanism, named dynamic minimum increment (DMI). In general, at the early stage, higher minimum increment is adopted to generate ask QoS in order to efficiently filter the bidders with low capacities. By doing this, DMI efficiently decreases the number of bidders and bids at the early stage of the auction and thus reduces the complexity of the winner determination problem. And at the late stage, low minimum increment is adopted to distinguish the small difference among the remaining bidders' reserved capacities, which guarantees the final outcome of the auctions (i.e. the quality of the composite services). Consequently, DMI can improve the efficiency of the combinatorial auction without sacrificing the effectiveness. Figure 4.2 illustrates three example models for the generation of DMI based on Chi-Square distribution, F-Distribution and Linear functions respectively.

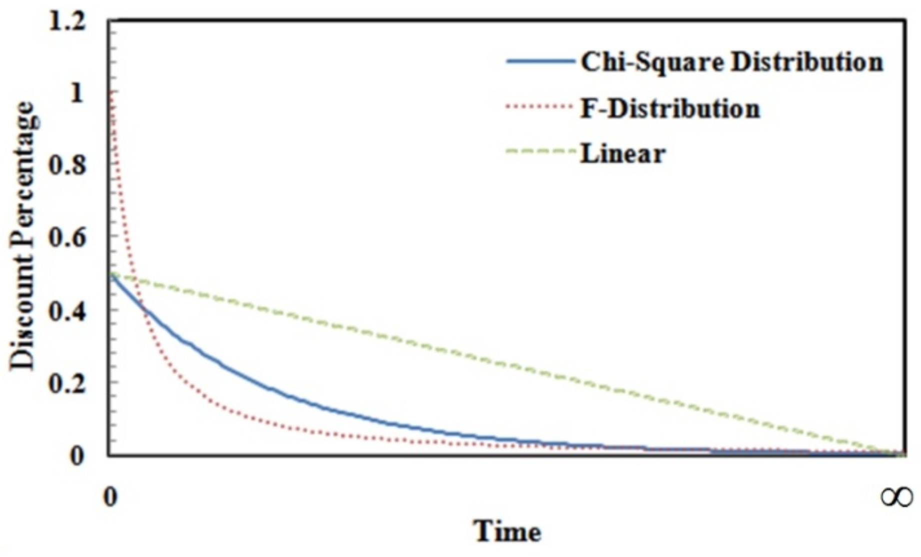


Figure 4.2 Example models for dynamic minimum increment

4.3.1.3 Final deal-sealing round

If a timer has been set for auction, when the time is up, a final deal-sealing round will be held in which the remaining bidders have a last chance to propose their best offers. And the result will determine the final winning bidders. If the bids proposed in the final deal-sealing round are estimated equivalent, the final winning bidder will be selected randomly.

4.3.2 QoS model

There are various QoS properties in the open SOC environment. They are often treated in accordance with domain-specific semantics. Nevertheless, as long as the QoS properties are negotiable between service consumers and providers and specifiable in SLAs, they can be accommodated by CASS. In addition, in service composition scenarios, the component services collectively fulfil the QoS requirements for the composite services. In order to evaluate the quality of a composite service, the quality of the component services need to be aggregated.

Component services may have various QoS properties, some of which can be aggregated[4] while others cannot. The former properties are defined as *compatible properties* (e.g. price and execution time) and the latter are defined as *incompatible properties* (e.g. colours and shapes of goods). The formal definitions are given as follows:

Definition 1: (Compatible property). A property is compatible to two services when:

- both services have the property; AND
- the corresponding property of the combination of the two services *can* be attained by aggregating the properties of the two services.

Definition 2: (Incompatible property). A property is incompatible to two services when:

- one service has the property while the other does not; OR
- both services have the property but the corresponding property of the combination of two services *cannot* be attained by aggregating the two services.

We aggregate the compatible QoS of the component services to obtain the QoS of the composite service based the workflow patters [95]. For the demonstration purpose, the definitions and aggregation functions for price and execution time, which are used in the example presented in this thesis, are as follows.

1. **Price**. The monetary amount that the service consumer has to pay to the service provider for the service transaction. Price can be measured in any

[4] The operation of aggregating a QoS property of two services is to obtain the QoS property of the service composed by the two.

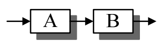
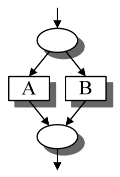
currencies as agreed by the service consumer and provider. The aggregation function for the price of a combination of services, noted as S_{A+B}, composed of two services, S_A and S_B, is:

$$S_{A+B}.v_{pr} = S_A.v_{pr} + S_B.v_{pr} \tag{4.1}$$

where $S.v_{pr}$ denotes the price of service S.

2. **Execution time**. The delay between the moment when the request is sent to the service and the moment when expected result is returned. Execution time is usually measured in seconds. Different from price, the execution time of a combination of services cannot be calculated by simply summing the execution time of individual component services. It depends on the local structure of composite service where the component services are located. Table 4.1 provides the aggregation functions for execution time [95].

Table 4.1 Aggregation functions for execution time

Sequence			$S_{A+B}.v_{exe} = S_A.v_{exe} + S_B.v_{exe}$
Parallel		Parallel + Synchronisation	$S_{A+B}.v_{exe} = max(S_A.v_{exe}, S_B.v_{exe})$ ($S.v_{exe}$ denotes the execution time of service S.)
		Parallel + Discriminator	$S_{A+B}.v_{exe} = min(S_A.v_{exe}, S_B.v_{exe})$
		Multi-Choice + Synchronising Merge	$S_{A+B}.v_{exe} = min(isSelected_{MC+SM}(S_A) \cdot S_A.v_{exe}, isSelected_{MC+SM}(S_B) \cdot S_B.v_{exe})$ (Function $isSelected_{\mathrm{MC+SM}}(S)$ returns 1 if S is selected for execution, 0 otherwise)
		Parallel + Synchronisation	$S_{A+B}.v_{exe} = min(isSelected_{MC+D}(S_A) \cdot S_A.v_{exe}, isSelected_{MC+D}(S_B) \cdot S_B.v_{exe})$ (Function $isSelected_{\mathrm{MC+SM}}(S)$ returns 1 if S is selected for execution, 0 otherwise)

Although the QoS properties discussed in this section is limited (for the purpose of demonstration), our model can be similarly applied to other QoS properties (either generic or domain specific) without fundamental alterations.

Moreover, we divide the numerical QoS properties into two categories: positive and negative properties, defined as follows.

Definition 3: (Positive property). A positive property is a property whose evaluation will increase as its value increases. Given the utility function *u(S.v)* to evaluate a property *v* of a service *S*, *v* is a positive property when:

$$\forall v_1, v_2,\ \ v_1 \le v_2 \Rightarrow u(S.v_1) \le u(S.v_2) \tag{4.2}$$

Definition 4: (Negative property). Similarly, a negative property is a property whose evaluation will decrease as its value increases. Given the utility function *u(S.v)*, a property *v* of a service *S* is a negative property when:

$$\forall v_1, v_2,\ \ v_1 \le v_2 \Rightarrow u(S.v_1) \ge u(S.v_2) \tag{4.3}$$

The above two concepts usually apply to numerical properties. Properties like "colour" and "shape" cannot be termed positive or negative. Although there are types of multi-property utility functions [87], we assume that the utility functions used by the auctioneer and the bidders in the combinatorial auction have the same monotonicity on the same properties.

4.3.3 Bid evaluation

To facilitate the heuristic backtracking algorithm to solve the winner determination problem in the combinatorial auction, a bid evaluation function, denoted as *E(b),* is needed to estimate how much a component bid is likely to maximise the auction revenue – optimise the quality of the composite service. The reason for adopting the heuristic backtracking approach will be given later in Section 4.3.4.

Given a service consumer' explicit utility function, it will be simple to implement *E(b)* which calculates the total utility of a component bid by summing up the utilities of individual QoS properties [84]. When the service consumer's utility function is unknown, CASS provides a novel scheme named ARPE (Additive Ranking Position Evaluation) to evaluate the component bids.

The philosophy of ARPE is to compare each of the QoS properties shared by the set of component bids proposed for the same component service. The procedure is as follows:

Step 1: Organise all the component bids into groups according to the items involved in the component bids. A component bid can fall into multiple groups if it involves several items. For example, the bid *({A, B, C}, {$3,000})* goes into groups *A*, *B* and *C*.

Step 2: For each group, compare the bids based on each of the QoS properties (basically numerical QoS properties) of the corresponding component services.

a) Average the QoS properties of the component service of all bids in the group. The average value of QoS properties *j* of item *i* is denoted as $v_{ave}^{i,j}$;

b) Evaluate individual QoS properties of the bids. For QoS property *j* of component service *i* of bid *n*, denoted as $v_n^{i,j}$, the evaluation value is computed as:

$$e_n^{i,j} = \begin{cases} \dfrac{v_n^{i,j} - v_{ave}^{i,j}}{v_{ave}^{i,j}} & \textit{for positive attributes} \\ \dfrac{v_{ave}^{i,j} - v_n^{i,j}}{v_{ave}^{i,j}} & \textit{for negative attributes} \end{cases} \tag{4.4}$$

c) Sum up the evaluation values of all QoS properties of the corresponding component service in the bids. For component service *i* with *m* QoS properties, the evaluation of bid *n* is:

$$E_n^i = \sum_{j=1}^{m} \lambda_j e_n^{i,j} \quad \sum_{j=1}^{m} \lambda_j = 1 \tag{4.5}$$

where λ_j is the weight assigned to capture the service consumer's preferences for different QoS properties. If the service consumer's preferences are not given, λ_j will be assigned with 1/m.

Step 3: For bids that involve a combination of component services, sum up the respective E_n^i of the involved component services to attain the total evaluation value, and then divide it by the number of the involved component services. The evaluation value of a bid that involves *p* component services is calculated as:

$$E(b) = \frac{1}{p} \sum_{i=1}^{p} E_n^i \tag{4.6}$$

The higher $E(b_i)$ is, the more likely the QoS proposed in b_i is to maximise the quality of the composite service. This is the basis of the heuristic backtracking approach to the winner determination problem. In order to facilitate ARPE, bidders will be required to specify the QoS of individual component services when they bid for combinations of component services.

4.3.4 Winner determination

The objective of winner determination is, from all submitted bids, to search for a set of component bids, denoted as $B = (b_1(S_1, V_1), b_2(S_2, V_2) \ldots b_n(S_n, V_n))$ where $S_i \subseteq K$ $(i = 1, 2, \ldots, n)$ $(S_i \neq \varnothing)$, which fulfils the following two constraints:

$$S_1 \cup S_2 \cup \ldots \cup S_n = K \tag{4.7}$$

$$\forall S_i, S_j,\ S_i \cap S_j = \varnothing \tag{4.8}$$

and meanwhile, maximises the quality of the composite service:

$$E(B)=\sum_{i=1}^{n} E(b_i) \tag{4.9}$$

Constraint (4.7) ensures that all component services are covered by $S_1 \cup S_2 \cup ... \cup S_n$ because all the component service must to be allocated. Constraint (4.8) ensures that each component service is included in only one selected component bid because a component service can be allocated to only one service provider. Finding the optimal solution is equivalent to the weighted set packing problem, and hence is NP-complete [89]. Accordingly, we propose a heuristic method based on backtracking algorithm [13] to solve the winner determination problem. The heuristics selects the most satisfactory component bid first. In determining an appropriate order of the component bids, function *E(b)* is used to evaluate how much a component bid is likely to maximise the auction revenue.

Prior to being ordered, component bids need to be extracted out of the bids which are in the form of $B_i((S_1, V_1)\ xor\ (S_2, V_2)\ xor\ ...\ xor\ (S_j, V_j))$, so that they can be considered individually in the heuristic winner determination. Here a concept is defined that will be used in the heuristic algorithm.

Definition 5: (Compatible bids). Two component bids (S_m, V_m) and (S_n, V_n) are compatible when $S_m \cap S_n = \varnothing$.

Definition 6: (Incompatible bids). Two component bids (S_m, V_m) and (S_n, V_n) are incompatible when $S_m \cap S_n \neq \varnothing$.

The heuristic algorithm adopted to solve the winner determination problem in CASS is as follows.

Step 1: Order the component bids by their evaluation values in a non-increasing order.

```
Input: B[]  // Component bids
       B_S[]  // Candidate set initiated as Ø
1.  for i=1 to B.Length
2.      B[i].Value = E(B[i])
3.      B[i].Active = True
4.  end for
5.  B.OrderByValueDesc()
6.  while B_S.ServiceSet!=S_T
7.      for i=1 to B.Length
8.          if B[i].isActive==True and B_S.isCompatible(B[i])
9.              B_S.Push(B[i])
10.             if B_S.ServiceSet==S_T
11.                 return B_S
12.             end if
13.             B.deactiveSiblingComponentBids(B[i])
14.         end if
15.     end for
16.     S_temp=B_S.Pop()
17.     B.Deactivate(S_Temp)
18.     B.activateSiblingComponentBids(S_Temp)
19. end while
end algorithm
```

Figure 4.3 Winner determination – heuristic backtracking algorithm

Step 2: Set a candidate bid set B_s, starting as $\varnothing$.

Step 3: From the set of ordered component bids, select the first one that is compatible to B_s, denoted as b_t, add it to B_s, i.e. $B_s \leftarrow B_s \cup b_t$. If there are multiple such component bids with the same evaluation value, select the one with the largest cardinality. The cardinality of a component bid b_t is defined as $|b_t|$, meaning the number of component services involved in the component bid. Then temporarily deactivate other component bids that were submitted along with b_t by the same bidder.

Step 4: Check if constraint (4.7) is satisfied, if so, the heuristic process completes. Otherwise, go to Step 5.

Step 5: If no more component bids compatible to B_s can be found and constraint (4.7) is not yet satisfied, backtrack to the component bid that was last selected, remove it from B_s, activate the component bids that were deactivated when the removed component bid was previously selected. Return to Step 3.

The corresponding pseudo code for the algorithm is presented in Figure 4.3.

4.3.5 Ask QoS generation

Ask QoS, as an extension of ask price, specifies the QoS asked by the auctioneer from a bidder. In each round, CASS provides bidders with non-linear and non-anonymous ask QoS to guide their bidding. Non-linear and non-anonymous ask QoS are defined as follows.

Definition 7: (Non-linear). Given a compatible QoS property v_i about services S_m and S_n, proposed in an ask QoS, where $S = S_m \bigcup S_n$ and $S_m \bigcap S_n = \varnothing$, non-linear ask QoS allows:

$$S.v_i \neq S_m.v_i + S_n.v_i \,. \tag{4.10}$$

For example, the auctioneer can propose to a bidder an ask price of \$200.00 for component service *M*, \$300.00 for item *N* and \$470.00 for the combination of *M* and *N*.

Definition 8: (Non-anonymous). Given a QoS property v_i about a service *S* in two ask QoS proposed to two different bidders, non-anonymous ask QoS allows:

$$S_m.v_i \neq S_n.v_i \,. \tag{4.11}$$

For example, the auctioneer can propose to a bidder an ask price of \$200.00 for a component service while proposes to another bidder an ask price of \$250.00 for the same component service.

Non-linear ask QoS allows the auctioneer to take into account the complementarities of the component services. Non-anonymous ask QoS allows the auctioneer to elicit bidders' discriminatory capacities for providing the QoS. For instance, a bidder with potential capacity of providing low price might be asked for lower price while another bidder being able to provide short execution time might be asked for shorter execution time.

We compute the ask QoS by analysing bidders' historical performance retrieved from an SLA profiling system [41], e.g. ServiceTrust presented in Chapter 6. In general, an SLA profiling system is an infrastructure deployed to collect service providers' historical performance based on which the service providers can be evaluated and ranked. A novel concept called Surplus Bidding Space is defined below, which models the bidders' potential capacity for boosting their offers for the QoS.

Definition 9: (Surplus Bidding Space). Given the QoS property *j* of component service *i* proposed by bidder *k* in the previous round, denoted as $v_k^{i,j}$, the surplus bidding space of bidder *k* for QoS property *j* of component service *i* is defined as:

$$\varepsilon_k^{i,j} = \begin{cases} \dfrac{v_{max}^{k,i,j} - v_k^{i,j}}{v_{max}^{k,i,j} - v_{min}^{k,i,j}} + \dfrac{v_{ave}^{k,i,j} - \frac{1}{2}\left(v_{max}^{k,i,j} + v_{min}^{k,i,j}\right)}{v_{max}^{k,i,j} - v_{min}^{k,i,j}} & \textit{for positive attributes} \\[2ex] \dfrac{v_k^{i,j} - v_{min}^{k,i,j}}{v_{max}^{k,i,j} - v_{min}^{k,i,j}} + \dfrac{\frac{1}{2}\left(v_{max}^{k,i,j} + v_{min}^{k,i,j}\right) - v_{ave}^{k,i,j}}{v_{max}^{k,i,j} - v_{min}^{k,i,j}} & \textit{for negative attributes} \end{cases}$$

$$\varepsilon_k^{k,i,j} \in (0,1) \text{ when } v_{min}^{k,i,j} < v_k^{i,j} < v_{max}^{k,i,j}. \tag{4.12}$$

where $v_{max}^{k,i,j}$, $v_{min}^{k,i,j}$ and $v_{ave}^{k,i,j}$ denote the maximum, minimum and average value of QoS property *j* of component service *i* that bidder *k* has ever provided. When an unknown bidder participates in the combinatorial auction, whose historical data cannot be provided by the SLA profiling system, equation (4.12) cannot be adopted to compute the bidder's surplus bidding space. In this case, the average value of the surplus bidding space of the other bidders competing for the same component service will be used by default to generate the ask QoS for the unknown bidder.

A higher $\varepsilon_k^{i,j}$ means a greater potential capacity a bidder has for providing better QoS, thereby the bidder can be pushed harder by the auctioneer with a higher minimum increment. Besides, relatively weak bidders, i.e. bidders with smaller surplus bidding space, will be excluded from the auction soon by the increasingly demanding ask QoS. This reduces the complexity of the winner determination problem and hence improves the efficiency of the auction.

As described in Section 4.3.1.2, high minimum increments can speed up the auction process at the early stage of the auction by efficiently filtering out weak bidders while low minimum increments can maximise the outcome of the auction at the late stage by effectively capture the difference among the remaining bidders. Ask QoS generation in CASS is based on dynamic minimum increment, aiming at guaranteeing both the efficiency and the effectiveness of the auction. In general, there are two factors that affect the calculation of the dynamic minimum increments: surplus bidding space of the bidders and the adopted model for generating dynamic minimum increment. As illustrated in Figure 4.4, the generation of ask QoS for bidder *n* which specifies asked QoS property *j* of component service *i*, proceeds as follows:

Step 1: Calculate $\varepsilon_k^{i,j}$, the surplus bidding space of the bidder *k* for property *j* of component service *i*, using equation (4.12);

Step 2: Draw a straight line from the origin with $\varepsilon_k^{i,j}$ as the slope;

Step 3: Locate the point at which the straight line drawn at Step 2 intersects with the curve used to generate the dynamic minimum increment.

Step 4: Obtain the *y* value of the intersection point, which is the dynamic minimum increment of QoS property *j* of component service *i* for bidder *k*.

We point out that $\varepsilon_k^{i,j} \leq 0$ when $v_i^j \geq v_{max}^j$ (for positive properties) or $v_i^j \leq v_{min}^j$ (for negative properties). In such cases, $\varepsilon_k^{i,j}$ will no longer work the way it is supposed to be. This situation happens when the QoS property of the current bid leaves the range of the bidder's historic performance. In such cases, using the bidder's historic performance to estimate its surplus bidding space becomes infeasible. According to Figure 4.4, the straight line with $\varepsilon_k^{i,j}$ as the slope does not intersect with the curves of dynamic minimum increment. To tackle this issue, we will use half of the dynamic minimum increment in the previous round as default to generate the ask QoS for the bidder.

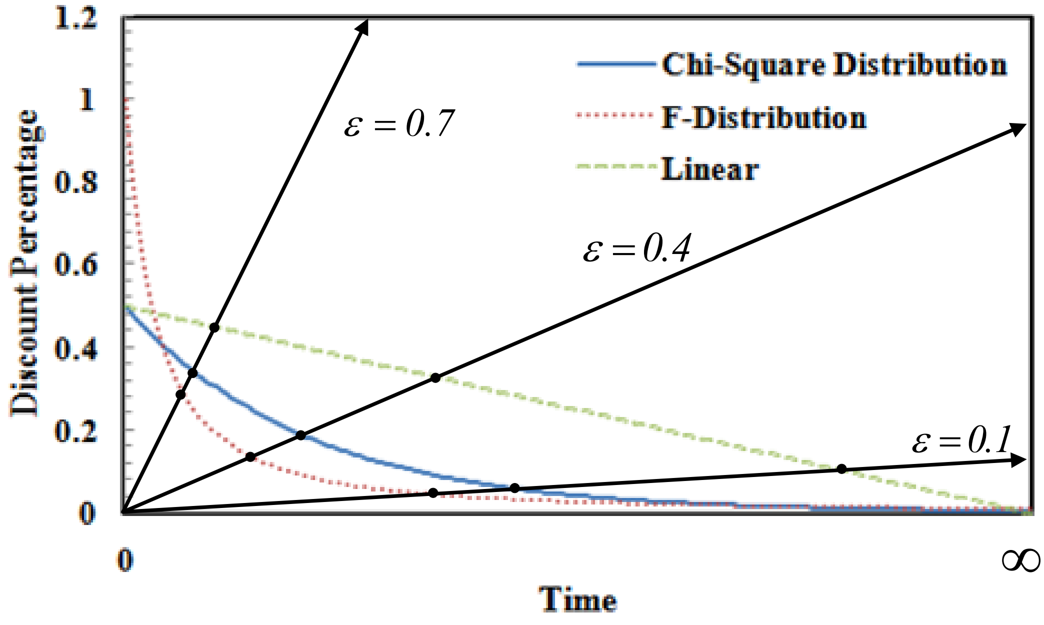


Figure 4.4 Generation of dynamic minimum increment

4.3.6 Completion criteria

In the combinatorial auction, when the winning bids have been determined in a certain round, the auctioneer needs to determine whether the auction completes by checking if the completion criteria are met. CASS provides flexible completion criteria. Three primitive completion criteria can be used, i.e. preset reserved QoS, pre-set time constraint and final-best-bids.

The first completion criterion is to meet reserved QoS pre-specified by the service consumer. When the current winning bids fulfil or exceed the reserved QoS, the auction completes and the winning bids constitute the final solution. This completion criterion is frequently used in service composition scenarios because service consumers often have certain QoS requirements. The reserved QoS cannot be changed throughout the auction process. Reserved QoS can be pre-specified for the quality of the composite service, e.g. “total price < $300,000”, “total execution time < 30 days”, etc. Service consumers can also pre-specify reserved quality for component services, such as required quality for the goods and particular size of the containers during shipping. Furthermore, those constraints can be combined to form complicated ones using logic operators, e.g. “(total price < $300,000) AND (total execution time < 30 days)”, or conditional operators, e.g. “IF total price ≥ $300,000 THEN total execution time ≤ 28 days”.

The second primitive completion criterion is related to a preset time constraint for the combinatorial auction. Similar to eBay, a timer can be set to determine when the combinatorial auction ends. The timer begins to count down as the auction starts. When time is up, the bidders are no longer allowed to propose bids. The component services are allocated to the current winning bidders. Or, final deal-sealing rounds can be performed, which is fair by giving the bidders a last chance to propose bids, in order to obtain better quality of the composite service.

The third primitive completion criterion, i.e. final-best-bids, is to let the combinatorial auction continue until only one bidder left for each component service. As the auction proceeds, bidders that cannot afford to accept the ask QoS would have to quit the auction while the capable ones enter the next round. Finally, only one bidder remains for each component service. The component services are allocated to the remaining bidders.

The three primitive completion criteria can be combined to form more complicated completion criteria. Usually, reserved QoS might be used together with a time constraint on the auction as some service consumers have specific QoS requirements of the composite services and they want the solutions within a certain amount of time. Also, time constraints are often used in combination with the final-best-bids completion criterion. For example, an obvious issue of adopting final-best-bids completion criterion is that sometimes it takes a long time to sort out the best bids when the bidders competing for the same component service have very similar reserved capacities for providing QoS. The reason is that a very small dynamic minimum increment, which will only appear in late rounds, is needed to differentiate the remaining bidders. This prolongs the combinatorial auction and thus reduces the efficiency. Therefore, setting a proper timer for the combinatorial auction is a direct method to address this issue.

4.4 Experimental evaluation

We conducted experiments on CASS to evaluate its effectiveness and efficiency. The mechanisms supporting CASS were implemented in the agent-based framework presented in [41] in which SLA negotiation can be automated through agents' interactions.

4.4.1 Experiment configuration

We utilised the example presented in Section 2.6.1 for evaluation. Bidders' reserved capacities for price and execution time for individual component services were picked randomly from the fixed intervals as presented in Table 4.2. And bidders were willing to give discounts when they bided for combinations of component services. In the experiments, some bidders would bid for the combination of services *B* and *C*, and some for the combination of services *D* and *E* (see Figure 2.2). Considering that the varying degrees of complementarity of the component services lead to different maximum discounts, we modelled the maximum discounts that the bidders were willing to apply to combinations of component services with small, medium and large degrees of complementarity using *beta* distribution functions presented in Figure 4.5. The Chi-Square distribution model presented in Figure 4.4 was used to compute the dynamic minimum increments and the ask QoS.

Table 4.2 Intervals for bidders' reserved capacities

Service	Price (*10^3)	Execution Time (Days)
Manufacturing	[100, 120]	[15, 20]
Rough Processing	[100, 120]	[30, 40]
Fine Machining	[200, 240]	[35, 45]
Land Carriage	[5, 7]	[3, 6]
Shipping	[3, 4]	[6, 10]

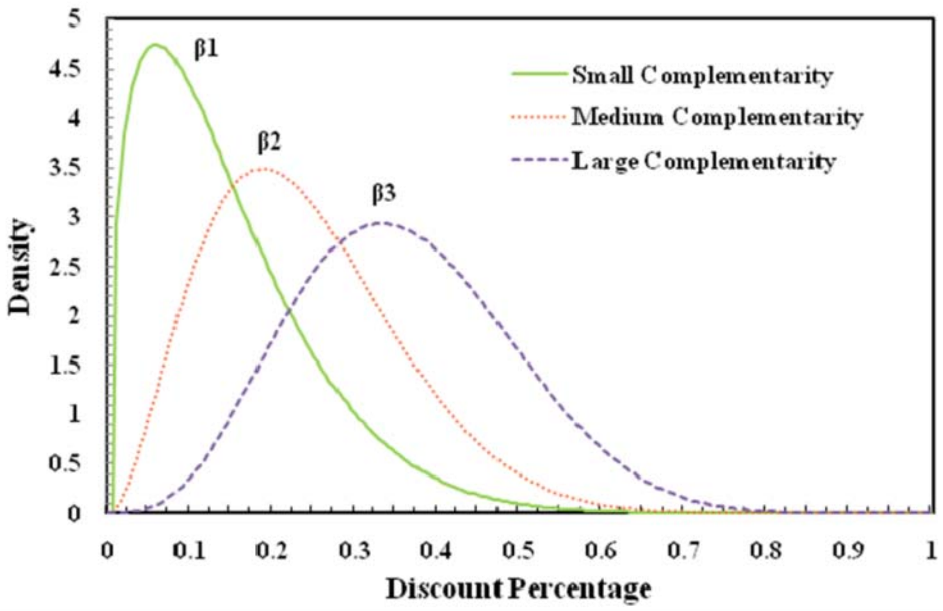
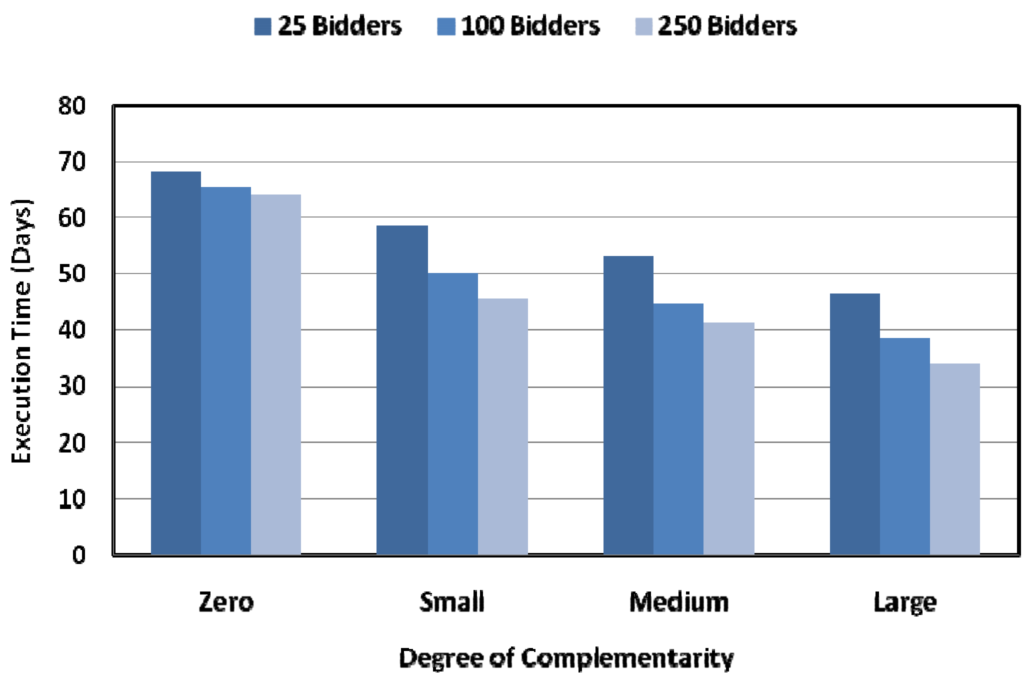


Figure 4.5 Maximum discounts for component services with small, medium and large degrees of complementarity

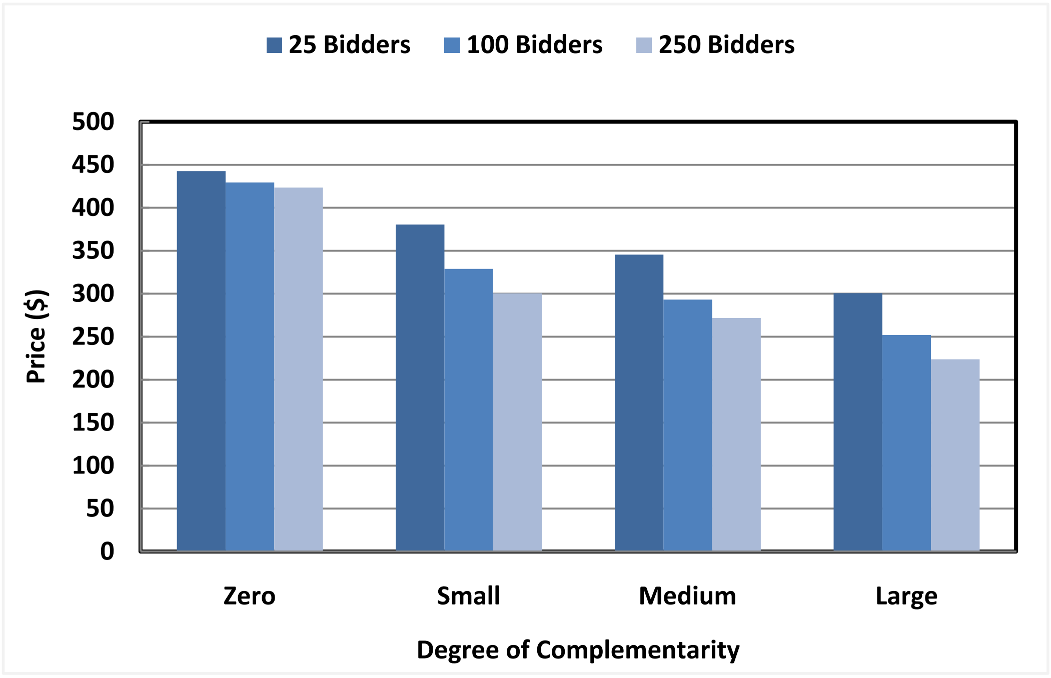
We conducted four sets of experiments with zero, small, medium and large degrees of complementarity assigned to the combinations of services B and C, and services D and E, respectively. With zero complementarity assigned to the combinations of component services the combinatorial auction is actually equivalent to traditional single-item auctions because no discounts or premium would be applied by the bidders. Random values from the intervals presented in Table 4.2 were picked to specify the reserved capacities of the bidders. The QoS proposed in a bidder's first bid (the bid proposed in the first round as response to the request for proposal) is selected from a reasonable range conforming to the normal distribution. For example, if a bidder's reserved capacity for the price of the rough processing service is \$100,000, then the price proposed in its first bid will be selected between [\$150,000, \$200,000]. This configuration captures the nature of the bidders with various concession spaces. To access the performance of CASS in environments on different scales, we conducted each set of the experiments with 25, 100 and 250 bidders. By doing so, an average number of 5, 20 and 50 bidders competed for each service, creating different competitive environments. In each experiment, a random proportion (between 30% to 50%) of the bidders was given the ability to bid for combinations of component services, either services *B+C* or *D+E.* Those bidders would be able to apply discounts to the bids for combinations of component services. The maximum discount percentages depended on the adopted *beta* distribution functions. $\beta 1$ presented in Figure 4.5 was adopted to generate the maximum discount percentages for combinations of component services with small degrees of complementarity, and $\beta 2$ and $\beta 3$ for medium and large degrees of complementarities respectively. For each set of experiments, 300 instances were executed, 100 with 25 bidders, 100 with 100 bidders and 100 with 250 bidders. And the outcomes were averaged.

4.4.2 Effectiveness evaluation

Figure 4.6 compares the quality of the composite service from traditional auctions (with zero complementarity of the component services CASS is equivalent to traditional single-item auctions) and CASS with different degrees of complementarity assigned to the combinations of component services. The results demonstrate that the traditional auctions yielded the worst quality of the composite service. And the quality of the composite service gets better as the degree of complementarity increases. Specific increments are presented in Table 4.3.



(a) Execution time



(b) Price

Figure 4.6 Comparison of the quality of the composite service achieved from traditional auction and CASS[5]

[5] Since price and execution time are both negative QoS properties, decreased prices and execution times imply increments of auction revenues.

Table 4.3 Increment of quality of the composite service

Number of Bidders	Degree of Complementarity		
	Small	Medium	Large
25	18.6%	29.1%	42.5%
100	31.0%	42.0%	54.7%
250	38.5%	47.4%	62.5%

We observe that as the degree of complementarity increases, the increment in the quality of the composite service increases. The reason is that the maximum discounts the bidders were capable of offering increase as the degree of complementarity increases. We also observe that an auction involving a larger number of bidders yields better quality of the composite service. This demonstrates that CASS facilitates effective SLA negotiation through capturing the basic economic theory: the more sellers there are, the more likely it is that the buyer will be able to obtain a favourable price.

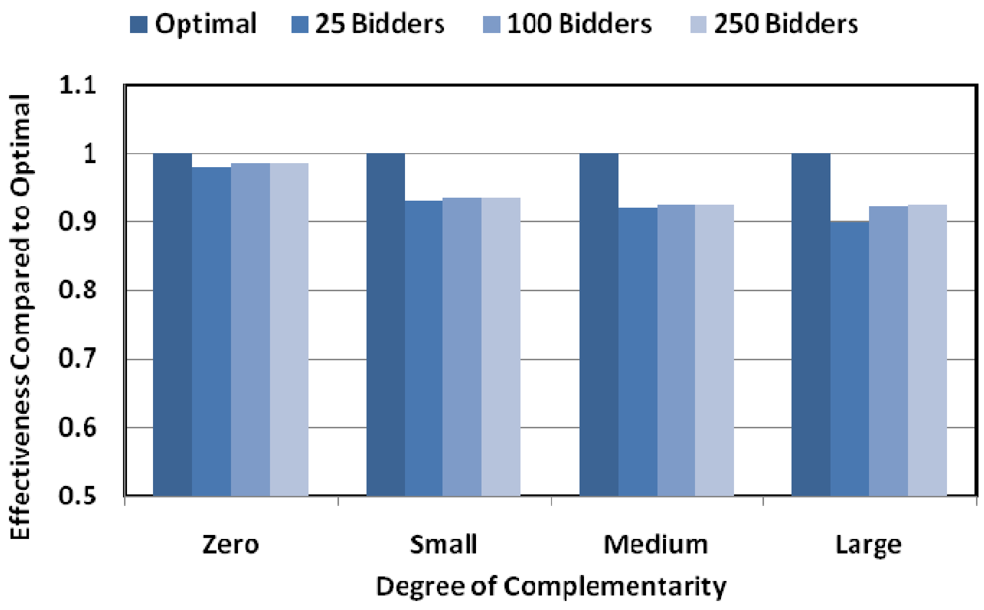


Figure 4.7 Comparison of the quality of the composite service achieved from CASS against the optimal results

We also compare the results from CASS against the optimal results, as demonstrated in Figure 4.7. The specifics are presented in Table 4.4. The optimal resultes are manually calculated using the best bidders' reserved capacitites. We observe that the results from CASS are very close to the optimal results.

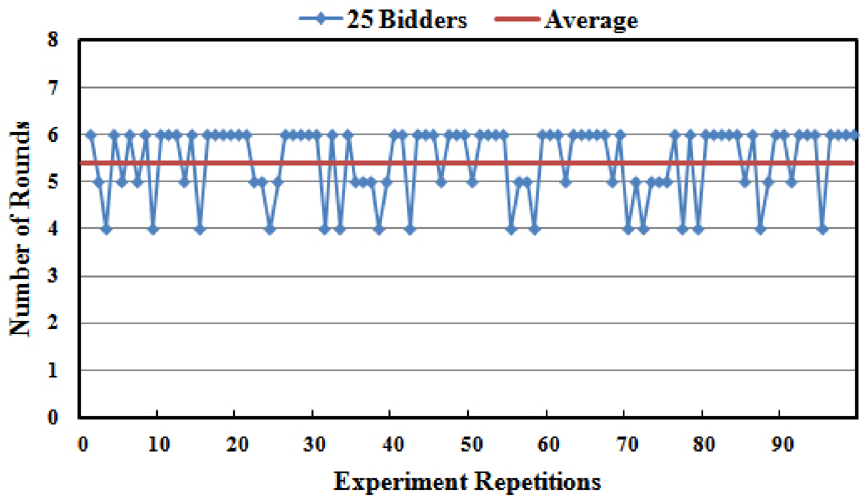
Table 4.4 Quality of the composite service compared to the optimal results

Number of Bidders	Degree of Complementarity			
	Zero	Small	Medium	Large
25	97.8%	93.0%	92.0%	89.8%
100	98.7%	93.6%	92.4%	92.3%
250	98.7%	93.6%	92.5%	92.4%

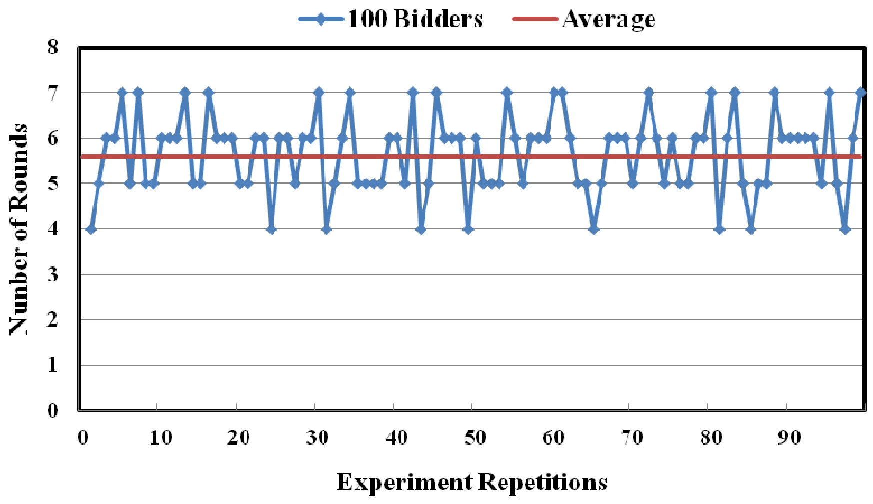
4.4.3 Efficiency evaluation

The auction overhead, measured in auction duration, is also a relevant concern of CASS because long auction duration usually implies low efficiency. Also, increased auction duration may reduce service consumer's profit from the auction because usually a certain amount of administrative cost for maintaining the auction applies in each round of the auction. When the auction proceeds to later stages, the marginal benefit for the service consumers from the bidders' competition may decrease dramatically which, in the worst case scenario, is not even worth the additional administrative cost. However, with the final-best-bids completion criterion applied, a longer auction has potentials to improve the quality of the composite service because it gives the auctioneer more time to elicit the high-value bidders' reserved capacities. Apparently, there is a tradeoff between the efficiency and the effectiveness of the auction. The bottom line is that the auction duration must be generally tolerable to the service consumers.

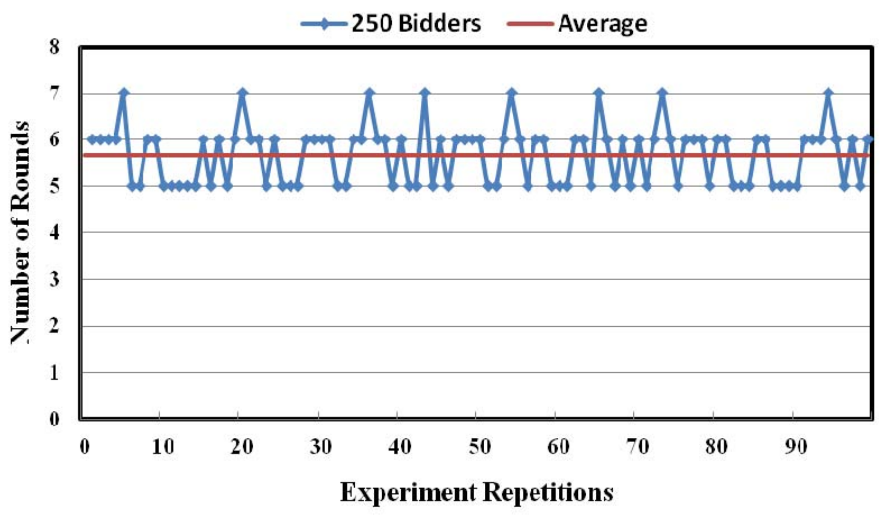
In the experiments, we assessed the auction duration measured by rounds. Recall that there are three primitive completion criteria. In the experiments, we adopted the final-best-bids completion criterion because it leads to the most time-consuming auctions. To evaluate the performance of CASS in environments on different scales, again, we ran the experiments with different numbers of bidders, i.e. 25, 100 and 250. Figure 4.8 plots the rounds taken for the auction to complete as the number of bidders increases. Specifically, an average of 5.42, 5.58 and 5.67 rounds were taken respectively for the auctions with 25, 100 and 250 bidders to complete. We observe that the fluctuations of the number of rounds varied in environments on different scales and no auctions took unusually high numbers of rounds. All the auctions with 25 bidders completed within 4 to 6 rounds, 100 bidders 4 to 7 rounds and 250 bidders 5 to 7 rounds. This demonstrates that a larger number of bidders would result in more fierce competition and hence increase the auction duration. However, 4 to 7 rounds is a reasonable range that we believe is tolerable in most scenarios especially compared to the improvement gained on the quality of the composite service. From the experimental results we conclude that CASS can achieve high efficiency in various environments on different scales.



(a) 25 bidders



(b) 25 bidders



(c) 250 bidders

Figure 4.6 Rounds for completion of CASS

4.5 Discussion

The QoS can be negotiated between the service consumers and providers before service provision. However, SLA violations may still happen due to faulty services, leading to failed SLA enforcement. Although penalty in the case of faulty services can be specified in the SLA beforehand, the service consumers, in most cases, would still expect the SLAs to be enforced successfully. Generally speaking, the service providers with good performance over past SLA enforcement, which implies good reputation, tend to be reputable and thus trustworthy to the service consumers. Therefore, during the process of service selection based on combinatorial auction, service consumers' trust over the service providers can be taken into account as a selection criterion. In this way, not only what the service providers advertise but also what essentially they can do is taken into account for the service selection.

The trust information can be utilised in two stages of CASS, before the auction and during the auction. Before the auction, candidate service providers with unsatisfactory reputation can be filtered out. By doing this, the complexity and duration of CASS can be reduced because the number of bidders involved in the auction decreases. During the auction, the service providers' reputation can be seen as one of the QoS properties. Although it must be noted that the reputation is a nature of the service provider based on its performance over the past SLA enforcement and thus which cannot be negotiated – essentially different from other QoS properties.

As the reputation of a service provider is calculated based on its performance over past SLA enforcement, upon the completion of each of its SLA enforcement, no matter successfully or unsuccessful, the result must be collected to generate the basic reputation metric which can later be used to calculate more useful reputation metric. Therefore, SLA profiling is an indispensable part to facilitate the enhancement of SLA negotiation. We have proposed ServiceTrust, a novel service trust system for reputation-oriented service selection based on SLA profiling which is detailed in Chapter 6.

4.6 Summary

In this chapter, we introduced CASS, an innovative effective and efficient SLA negotiation approach based on iterative multi-property combinatorial auction to support SLA establishment for service composition. CASS can significantly improve the effectiveness of the SLA negotiation with reasonable overhead via an effective information exchange of service consumers' preferences and service providers' offers. Thus, the quality of the composite service, which is achieved from the final winning bids, can be improved in an efficient way. The key features of CASS are:

1. CASS allows candidate service providers to bid for combinations of component services. This feature captures the dynamics, e.g. preferences and capacities of the candidate service providers. This feature also benefits the service consumers because it improves the quality of the composite services by allocating the component services to the service providers who are capable of providing the best QoS.

2. CASS allows SLA negotiation over multi-dimensional QoS. This feature captures the service consumers' preferences for different properties of the composite services. A novel approach for the generation of ask QoS (an extension of ask price) that involves multiple QoS properties, is proposed to coordinate the auction process by guiding the bidders in each round.

3. CASS allows different completion criteria to be set to cater for service consumer's requirements. The completion criteria are optional and can be combined to create complicated completion criteria.

Chapter 5

Runtime adaptation in SLA enforcement

As discussed in Section 2.4 and Section 3.3, SLA enforcement, as the second stage of the lifetime of an SLA, needs the support of SLA adaptation during the service provision to adapt the services and attached SLAs when SLA violations occur at runtime. However, in service composition scenarios, adapting the failed service only may not be able to adapt the composite service as a whole. Hence, the adaptation approach needs to consider updating a certain subset of the composite service while adapting the faulty component service. To determine the adaptation solution, the SLA adaptation approach needs to identify the adaptation scope and analyse the profit from different adaptation solutions [40]. In this chapter, we discuss how the value of changed information (VOC) [18, 38] is extended and applied to SLA adaptation based on workflow patterns. Specifically, we will analyse and present how to compute the VOC for sequence and parallel patterns. If the adaptation solution is expected to pay off, it is performed within a certain scope defined by workflow patterns. By doing so, the SLA adaptation can deliver satisfactory results while being kept within a reasonable scope. The experimental results show that our approach can significantly improve the satisfaction rates of service consumers' requirements for the composite services in different situations.

This chapter is organised as follows. Section 5.1 presents the approach for SLA adaptation for composite service based on workflow patterns. Section 5.2 presents the adaptation method that implements the mechanisms presented in Section 5.1.

Section 5.3 presents the experimental results from the evaluation of the proposed approach. Section 5.4 discusses the support of SLA negotiation for SLA adaptation. Section 5.5 summarises this chapter.

5.1 Adaptation for composite service based on workflow patterns

Composite services can be modelled as executable business processes using service composition languages like BPEL and WSFL which incorporates the concepts and mechanisms in the workflow community. Therefore, the internal logic of a composite service can be captured using workflow patterns. In this section, we introduce the workflow patterns presented in [95], and discuss and analyse how the VOC mechanism can be extended based on these workflow patterns to facilitate SLA adaptation for composite service.

5.1.1 Sequence pattern (Seq)

- **Pattern 1 Sequence** A *Sequence* pattern describes the structure where a component service starts after the completion of another component service in the same process.

When a service violates the SLA, there are two ways to adapt it. The service consumer can either renegotiate with the current service provider – the one that violated the SLA – to see if it can still provide the service with satisfactory quality to meet the service consumers' requirements for the composite service, or find a new service provider to replace the current service provider. Before performing the adaptation process, the VOC of the adaptation solution needs to be computed.

Since the actual values of the adapted QoS are not known until after negotiating with the service providers, we average all the possible combinations of the values of adapted QoS using current belief distributions which can be obtained from the pre-defined SLAs, previous interactions with the service providers or ServiceTrust – a service trust system based on SLA profiling as detailed in Chapter 6.

Suppose there are two component services, *A* and *B*, that are compliant with the Sequence pattern, when an SLA violation occurs in service *A*, the SLA adaptation mechanisms must be triggered. If there are several candidate service providers for service *A* – very common in the open SOC environment, for each candidate service provider (including the current service provider), an estimated value is computed. Formally,

$$V(A|A') = \int_{\Omega_{A'}} u(e,p) \cdot Pr(E_{A'} = e, P_{A'} = p) d\Omega_{A'} \text{ [6]} \tag{5.1}$$

where $V(A|A')$ is the estimated value from adapting service *A* with service provider A', $u(e,p)$ is a utility function which computes the utility of a service based on the service consumer's preferences for execution time, *e*, and price, *p*, $Pr(E = e, P = p)$ denotes the belief distribution of the combination *(e, p)* and $\Omega_{A'} = <(e_1, p_1), (e_2, p_2), \ldots, (e_n, p_n)>_{A'}$ represents the possible combinations of the values of execution time and price from the candidate service providers for service *A*. Here we take the current service provider as a candidate service provider if it is still potentially capable of providing the service with satisfactory SLA.

Then the service consumer computes the probability that service *B* needs to be rearranged due to selecting adapting service *A* with service provider A'. The probability, denoted as $P(A|A' \to B)$, is affected by three main aspects:

1) the extra resource consumption caused by the adaptation of service *A*. For example, delay and increased price from adapting service *A* might trigger the need of adapting service *B*;

2) the margins of the related terms in the SLA attached to services *A* and *B*. If the service consumer had succeeded in contracting SLAs for service *A* or *B* or both with extra marginal violation tolerance, there would be a relatively slim chance that service *B* needs to be adapted; and

[6] In this chapter we use the execution time and price of the services for the purpose of demonstration.

3) the estimated capability of candidate service providers for service *A*. If it is expected to adapt service *A* with better QoS, the need of updating service *B* will decrease.

In the sequence pattern, the formal computation of $P_{Seq}(A|A' \to B)$ is:

$$P_{Seq}(A|A' \to B)$$
$$= \frac{1}{|\Omega_{A'}|} \cdot \int_{\Omega_{A'}} isRequirementViolated(e,p) \cdot Pr(E_{A'} = e, P_{A'} = p) d\Omega_{A'} \qquad (5.2)$$

where *isRequirementViolated(e, p)* is a function which, given the execution time, *e*, and price, *p*, returns *1* if the service consumer's requirements for the composite service will be violated and *0* otherwise. This function involves QoS aggregation [45] and multiple criteria decision making (MCDM) [49] and is implemented application-specifically.

Then we formulate the VOC due to the service adaptation as:

$$VOC_{A|A'+B|B'} = V(A|A') + P(A|A' \to B) \cdot V(B|B')$$
$$= \int_{\Omega_{A'}} u(e,p) \cdot Pr(E_{A'} = e, P_{A'} = p) d\Omega_{A'} + P_{Seq}(A|A' \to B) \cdot \int_{\Omega_{B'}} u(e,p) \cdot Pr(E_{B'} = e, P_{B'} = p) d\Omega_{B'}$$
(5.3)

where $VOC_{A|A'+B|B'}$ is the value brought from adapting service *A* with service provider A' and service *B* with service provider B'.

Since adapting services *A* and *B* may be expensive, the adaptation is performed only when it is expected to pay off. The adaptation cost, including querying information, renegotiating and negotiating with service providers, and switching service providers, must be lower than the corresponding VOC. Formally, the adaptation is performed when:

$$VOC_{A|A'+B|B'} > COST(A\,|\,A'+B\,|\,B') \tag{5.4}$$

where $COST(A\,|\,A'+B\,|\,B')$ is the cost of adapting service *A* with service provider A' and service *B* with service provider B'. When there are more than one qualified combination of candidate service providers for services *A* and *B*, the one with the greatest profit, i.e. $VOC_{A|A'+B|B'} - COST(A\,|\,A'+B\,|\,B')$, is selected.

From formulas (5.1) – (5.4), we can observe that it is the combination of candidate service providers for services *A* and *B* that collectively determines the profit of the adaptation solution. The service consumer needs to compute the VOC of each of the possible combinations of candidates for services *A* and *B* – possible adaptation solutions – which is computationally intensive if the number of candidate service providers is large. Under this circumstance, the service consumer can select several candidates according to the ranking based on their trust over the candidature service providers calculated using ServiceTrust which is presented in Chapter 6.

5.1.2 Parallel patterns

Besides the sequence pattern, parallel is another major pattern in business processes as well as composite services. The authors in [95] consider the parallel patterns in terms of (1) how the branches are picked, (2) how they are executed and (3) how they converge. In this section, we analyse the SLA adaptation mechanisms for different types of parallel patterns considering the three aspects above.

There are three typical split patterns that describe the logic of processes being split and enacted:

- **Pattern 2 Parallel Split** A *Parallel Split* describes the structure where a single thread splits into multiple threads which can be executed in parallel. In this pattern, component services *A* and *B* will both be executed and can be executed simultaneously in any order.

- **Pattern 3 Exclusive Choice** An *Exclusive Choice* pattern descries the structure where, based on a decision or process control data, only one selected branch is activated and executed.

- **Pattern 4 Multi-Choice** A *Multi-Choice* pattern describes the structure where, based on a decision or process control data, a number of branches are chosen.

We must also consider how the branches will converge (if they will). There are five typical patterns that model the logic of the branches converging:

- **Pattern 5 Synchronisation** A *Synchronisation* pattern describes the structure where multiple parallel branches converge into one single thread synchronised.

- **Pattern 6 Simple Merge** A *Simple Merge* pattern describes the structure where more than one branches converge without synchronisation and only one of them has ever been executed.

- **Pattern 7 Synchronising Merge** A *Synchronising Merge* pattern describes the structure where synchronising happens only when more than one branches are active (i.e. they are being executed).

- **Pattern 8 Multi-Merge** A *Multi-Merge* pattern describes the structure where the branches converge without synchronisation and the service succeeding the mergence will be activated by the completion of every incoming branch.

- **Pattern 9 Discriminator** A *Discriminator* pattern describes the structure where the subsequent service will be activated by the first and only the first completed branch. The remaining branches will be ignored.

The combination of the split patterns and converge patterns determines the solution to SLA adaptation. Starting with the simplest combination, i.e. *Parallel Split + Synchronisation*, the corresponding SLA adaptation mechanisms are discussed as follows.

Parallel Split + Synchronisation (PSS**)**

In this pattern combination, multiple branches split at a certain point and then converge with synchronisation after the completion of all the branches. Here we suppose there is only one service on each parallel branch. This assumption is realistic because if there are more than one service on any branch, they can be considered as a composite service. When the service on one of the parallel branches needs to be adapted, the service consumer must compute the VOC and select an appropriate adaptation solution. Moreover, to guarantee the satisfaction of its requirements for the composite service, the service consumer must also consider if it is necessary to adapt the services on other branches.

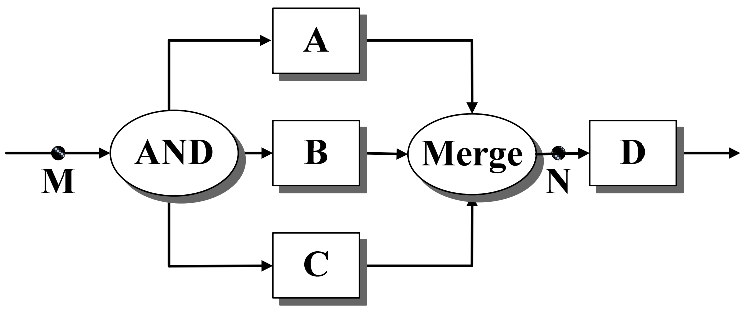


Figure 5.1 Parallel Split + Synchronisation pattern

We consider the adaptation under two circumstances:

1. All the branches have not been activated and executed when the SLA adaptation is triggered. In this case, the SLA adaptation is performed before point M in Figure 5.1; and

2. All the branches have been activated when the SLA adaptation is triggered. In this case, the SLA adaptation is performed between point M and N in Figure 5.1.

In case 1), the service consumer needs to estimate if the adaptation of the faulty service, i.e. service *A* in Figure 5.1, will cause the delay in the activation of service *D* due to the newly contracted SLA and the time consumption of the adaptation process. If the answer is yes, the service consumer can consider giving services *B* and *C* more time to complete because given more flexible time constraint, the service consumer may be able to renegotiate with the providers of services *B* and *C* to obtain lower prices or better quality for services *B* and *C*.

For the PSS pattern combination, the probabilities of additional execution time for services *B* and *C* caused by the adaptation of service *A*, assuming service *A* is adapted with service A', are computed as:

$$P_{PSS}(A \mid A' \to B) = \frac{1}{|E_{A'}|} \cdot \int_{E_{A'}} \cdot Pr(E_{A'} = e) \cdot isBigger(e, e_B) dE_{A'} \quad (5.5)$$

$$P_{PSS}(A \mid A' \to C) = \frac{1}{|E_{A'}|} \cdot \int_{E_{A'}} \cdot Pr(E_{A'} = e) \cdot isBigger(e, e_C) dE_{A'} \quad (5.6)$$

where e_B and e_C are the execution times for services *B* and *C*, $|E_{A'}|$ is the modulo of $E_{A'}$, and *isBigger(x, y)* is a function returns *1* when *x* is bigger than *y* or *0* otherwise. Then the VOC of different adaptation solutoins, involving services *A*, *B* and *C*, can be computed as:

$$VOC_{PSS}^{A|A'+B|B'+C|C'} = V(A \mid A') + P(A \mid A' \to B) \cdot V(B \mid B') + P(A \mid A' \to C) \cdot V(C \mid C') \quad (5.7)$$

In case 2), when the SLA violation of service *A* is detected, services on other branches, i.e. services *B* and *C*, have already been activated. When the adaptation of service *A* is being performed, services *B* and *C* are already under execution.

Therefore, the estimated time consumption from adapting service A, must be included in the computation of $P(A|A' \to B)$ and $P(A|A' \to C)$:

$$P_{PSS}(A|A' \to B)$$

$$= \frac{1}{|E_{A'}|} \cdot \int_{E_{A'}} \cdot Pr(E_{A'} = e, E_{adaptation} = e_{adaptation}) \cdot isBigger(e + e_{adaptation}, e_B) dE_{A'} \tag{5.8}$$

$$P_{PSS}(A|A' \to C)$$

$$= \frac{1}{|E_{A'}|} \cdot \int_{E_{A'}} \cdot Pr(E_{A'} = e, E_{adaptation} = e_{adaptation}) \cdot isBigger(e + e_{adaptation}, e_C) dE_{A'} \tag{5.9}$$

where $e_{adaptation}$ is the estimated time to adapt service A.

Then the VOC of different adaptation solutions can be computed using formula (5.7) with $V(A|A')$, $V(B|B')$ and $V(C|C')$ computed using formula (5.1).

Parallel Split + Multi-Merge (PSMM)

In a *Multi-Merge* pattern, the service succeeding the mergence will be executed every time an incoming branch completes. Most of the workflow products, e.g. Eastman, Verve Workflow and Forte Conductor, implement the Multi-Merge pattern by replicating the service(s) succeeding the mergence (see Figure 5.2 for a simple example) [95]. And the replicated services will be made sequential to the services on each of the original branches, generating several independent sequence structures. Actually, at runtime, no *Multi-Merge* structures will be found. Therefore, VOC computation for *Sequence* pattern will be applied to the created *Sequence* structures in adaptation.

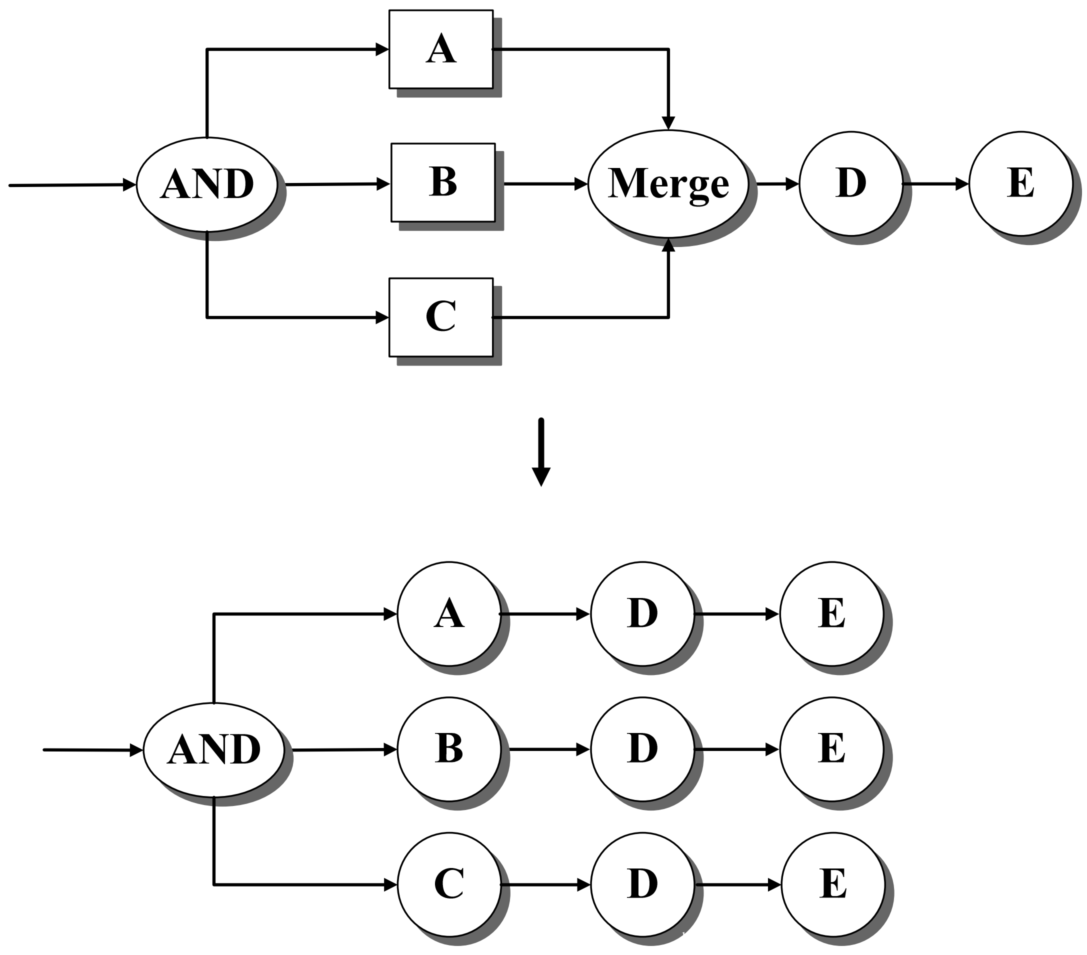


Figure 5.2 Implementation of Multi-Merge pattern

Parallel Split + Discriminator (PSD)

In a *Discriminator* pattern, the service succeeding the mergence waits for the first completed incoming branch and ignores the rest. In other words, the succeeding service will be activated once one of the incoming branches is firstly completed. We now discuss how to adapt the parallel services when SLA violations occur.

In this pattern combination the fastest branch (the one with the shortest execution time) determines the start time of the service succeeding the mergence. Once the succeeding service is activated, the uncompleted branches will be ignored. In fact, the *Discriminator* pattern is not found often in business processes where

SLA is enabled because generally the execution time of the services will be specified in the SLAs and hence it can be estimated that which branch is likely to complete first and which branches will be ignored. However, there is one exception – the service consumer wants to hedge the risk of delay caused by SLA violations. In this case, by employing the *Discriminator* pattern, when a branch is broken, other branches can still deliver expected results in a relatively tolerant period of time. Intuitively, the most effective way to hedge the risk is to allocate low execution time for individual branches while obtaining a high successful global execution rate of the branches. It is also the major objective of the adaptation of the services when SLA violations occur. Since the services on different branches are functionally equivalent, they share a group of candidate service providers and a utility function. For the demonstration purpose, we use the example in Figure 5.3.

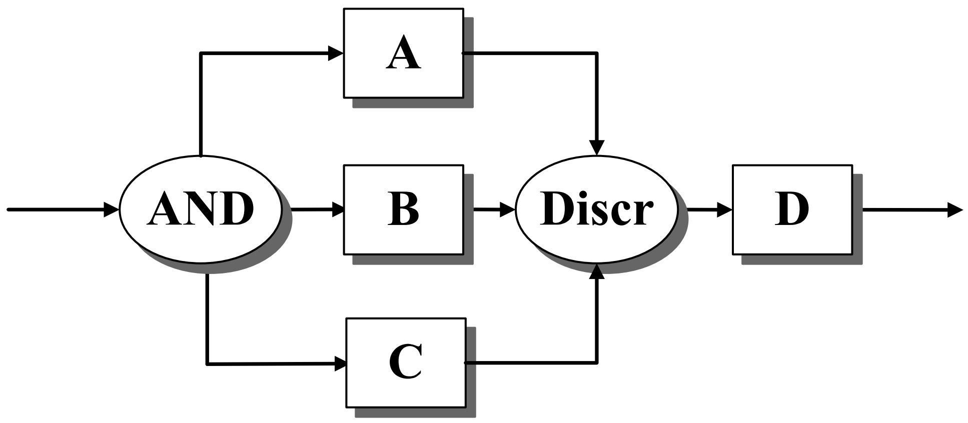


Figure 5.3 Parallel Split + Discriminator pattern

First, the normalised successful execution rate weights, w_1, w_2 and w_3, are assigned to the candidate service providers. The success rate weights range from *0* to *1*, representing the service consumer's trust over the service providers, based on their historical performance which can be provided by ServiceTrust detailed in Chapter 6. Service providers with better reputation, obtained from higher successful rates of past SLA enforcement, will be assigned with higher weights. The historical performance of service providers can also be provided by the service providers through pre-defined SLAs or be learnt from the service consumer's previous interactions with the service providers.

Second, the VOC can be computed as:

$$VOC_{PSD}^{A|A'+B|B'+C|C'} = \int_{\Omega_{(A',B',C')}} w_1 \cdot u(e_{A'}, p_{A'}) \cdot Pr(E_{A'} = e_{A'}, P_{A'} = p_{A'})$$

$$+w_2 \cdot u(e_{B'}, p_{B'}) \cdot Pr(E_{B'} = e_{B'}, P_{B'} = p_{B'}) + w_3 \cdot u(e_{C'}, p_{C'}) \cdot Pr(E_{C'} = e_{C'}, P_{C'} = p_{C'}) d\Omega_{(A',B',C')} \quad (5.10)$$

Based on formula (5.10), the combination of candidate service providers that maximises $VOC_{PSD}^{A|A'+B|B'+C|C'} - Cost(A|A') - Cost(B|B') - Cost(C|C')$ will be selected.

Sometimes which branches will be executed is dependent on the runtime decision making. Pattern 3 *Exclusive Choice* and pattern 4 *Multi-Choice* describe the two different situations in this category. In pattern 3 *Exclusive Choice*, only one branch will be chosen and executed in a running process instance and it leads to a *Simple Merge*. This makes the branches uninfluential in one another. Therefore, when the service on one branch needs to be adapted, the services on other branches do not need to be considered. The VOC mechanism can still be applied here because the broken branch, together with the service succeeding the mergence, can be seen to constitute a *Sequence* pattern. In pattern 4 *Multi-Choice*, multiple branches will be chosen for execution. Therefore, the service consumer needs to consider the services on other branches when trying to adapting the faulty service on one branch. Next we will discuss the pattern combinations involving the *Multi-Choice* pattern.

Multi-Choice + Synchronising Merge (MCSM)

In this pattern combination, multiple branches will be chosen for execution and they will be synchronised when they merge. The way the branches merge is similar to pattern 5 *Synchronisation*. The difference is that not all the incoming branches will be activated for every running instance. If one of the branches is broken and needs to be adapted, the service consumer needs to estimate the probabilities of whether other branches need to be activated, and whether they can benefit from the adaptation.

Generally speaking, there is no way to ascertain which branches will be activated for a specific running instance until the dynamic decision is made. However, the service consumer can estimate the probability that a branch will be activated based on the historical performance of the branches. For example, if a branch was executed 80 times out of the last 100 composite service instances, we consider the branch will be selected with a probability of 80%. The probability of being selected for each branch can be normalised as weights, s_1, s_2, … s_n, ranging between *0* and *1* representing the importance of the branches. Take the composite service in Figure 5.1 as an example, with *Parallel Split* replaced with *Multi-Choice*. Based on this assumption the computation of the VOC for the adaptation solution can be formalised as:

$$VOC_{MCSM}^{A|A'+B|B'+C|C'} = s_1 \cdot V(A|A') + s_2 \cdot P(A|A' \rightarrow B) \cdot V(B|B') + s_3 \cdot P(A|A' \rightarrow C) \cdot V(C|C') \quad (5.11)$$

where $V(A|A')$, $V(B|B')$ and $V(C|C')$ are computed using formula (5.1), $P_{MCSM}(A|A' \rightarrow B)$ and $P_{MCSM}(A|A' \rightarrow C)$ are computed using formulas (5.5) and (5.6).

Multi-Choice + Multi-Merge (MCMM)

Similar to the *PSMM* combination, in a *MCMM* pattern combination the branches will be transformed into several independent sequence structures before being executed. Thus, VOC computation for the *Sequence* pattern can be applied when necessary.

Multi-Choice + Discriminator (MCD)

In this pattern combination, a number of branches are selected and executed in parallel based on a decision dynamically made. The first branch that completes will trigger the service succeeding the mergence and after that other branches will be ignored. As discussed before, the aim of employing the *Discriminator* pattern is to achieve relatively tolerant execution time when SLA violations occur. Therefore, in adaptation solution determination, the branches that have higher successful rates and higher probabilities of being selected should be given higher preferences. Here we adopt the weights used before: *w*, *to* represent the successful execution rate, and *s*, to represent the probability of branches being selected. Again, for three parallel branches, the formalised VOC computation is:

$$VOC_{MCD}^{A|A'+B|B'+C|C'} = \int_{\Omega_{(A',B',C')}} s_1 \cdot w_1 \cdot u(e_{A'}, p_{A'}) \cdot Pr(E_{A'} = e_{A'}, P_{A'} = p_{A'}) + s_2 \cdot w_2 \cdot u(e_{B'}, p_{B'}) \cdot Pr(E_{B'} = e_{B'}, P_{B'} = p_{B'}) + s_3 \cdot w_3 \cdot u(e_{C'}, p_{C'}) \cdot Pr(E_{C'} = e_{C'}, P_{C'} = p_{C'}) d\Omega_{(A',B',C')} \quad (5.12)$$

5.1.3 Other patterns

In addition to the nine patterns addressed so far, there are other eleven patterns introduced in [95]. Some of them are used to describe the global properties and special activities of the business processes, including *Arbitrary Cycles*, *Implicit Termination*, *Multiple Instance*, *Cancel Activity* and *Cancel Case*. Patterns of *Deferred Choice* and *Milestone* are used to specify the triggering condition of services based on decision dynamically made. The last pattern, *Interleaved Parallel Routing*, describes a set of services that are executed one by one in an arbitrary order decided at runtime. However, our work, VOC calculation based on workflow patterns, is dedicated to analysing adaptation solution for composite services based on specific influence between services in a confined adaptation scope described using workflow patterns. The above eleven patterns do not serve the goal and thus are excluded from this thesis.

5.2 Adaptation method

Figure 5.4 shows the pseudo code for adapting the composite services using the mechanisms presented in Section 5.1. The algorithm takes one input – the component service that needs to be adapted, denoted as *S_0*. The algorithm starts with identifying the pattern that *S_0* belongs to (line 3). After the pattern identification, the VOC of different adaptation solutions is computed (line 7 for Sequence pattern and line 18 for Parallel pattern). The adaptation solution that is estimated to bring the most profit will be performed (lines 9 and 20). If the adaptation solution within the current scope is not satisfactory, the component services in the current scope will be considered as a composite service (lines 12-13 and lines 23-24) and the adaptation will expand to a larger scope. The algorithm returns *true* if the composite service is successfully adapted or *false* if all the unexecuted component services have been taken into account and still no satisfactory adaptation solution can be found.

```
Algorithm for Adapting Composite Service
Input: $S_0$  // service requiring adaptation
1.  $V_{S0}[] \leftarrow$ Candidates($S_0$)  //fetch candidates for $S_0$
2.  while end of composite service not reached
3.      identify the pattern that $S_0$ belongs to
4.      if it is sequence
5.          $S_1 \leftarrow S_0$.Next
6.          $V_{S1}[] \leftarrow$ Candidate($S_1$)  //fetch candidates for $S_0$ and $S_1$
            //Iterate through the combinations of $V_{S0}[]$ and $V_{S1}[]$
7.          if Max(VOC($S_0|S_0'+S_1|S_1'$) - Cost($S_0|S_0'+S_1|S_1'$)) > 0
8.          and Service consumer's requirements met
9.              Adapt $S_0$ with $S_0'$ and $S_1$ with $S_1'$
10.             return true
11.         else
12.             $S_0 \leftarrow S_0 + S_1$
13.             $V_{S0}[] \leftarrow$ Combine($V_{S0}[]$, $V_{S1}[]$)  //combine candidates of S0 and $S_1$
14.         end if
15.     end if
16.     if it is parallel
            //fetch candidates for $V_{S1}[], \cdots, V_{Sn}[]$
17.         $V_{S1}[], \cdots, V_{Sn}[] \leftarrow$ Candidate($S_1$), $\cdots$, Candidate($S_n$)
            //Iterate through the combinations of candidates for $S_0, S_1, \cdots S_n$:
18.         if Max(VOC($S_0|S_0'+S_1|S_1'+\cdots S_n|S_n'$) - Cost($S_0|S_0'+S_1|S_1'+\cdots S_n|S_n'$)) > 0
19.         and Service consumer's requirements met
20.             Adapt $S_0$ with $S_0'$, $S_1$ with $S_1'$,$\cdots$, $S_n$ with $S_n'$
21.             return true
22.         else
23.             $S_0 \leftarrow S_0 + S_1$
24.             $V_{S0}[] \leftarrow$ Combine($V_{S0}[]$, $V_{S1}[]$)  //combine candidates of S0 and $S_1$
25.         end if
26.     end if
27. end while
28. return false
end algorithm
```

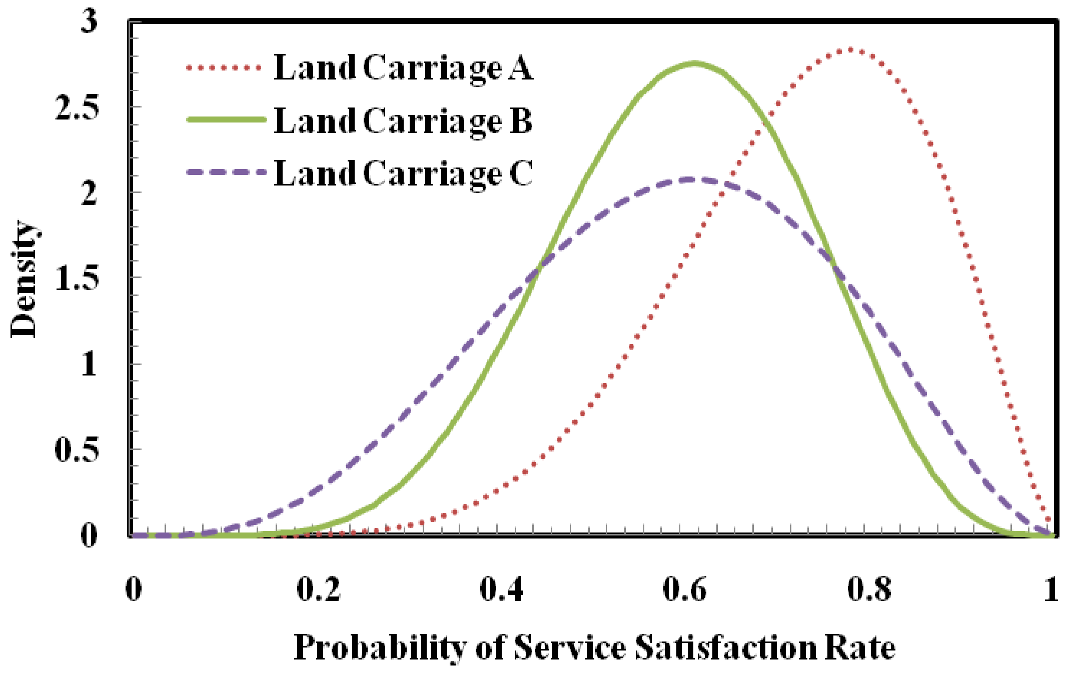
Figure 5.4 Pseudo code for adapting a composite service

5.3 Experimental evaluation

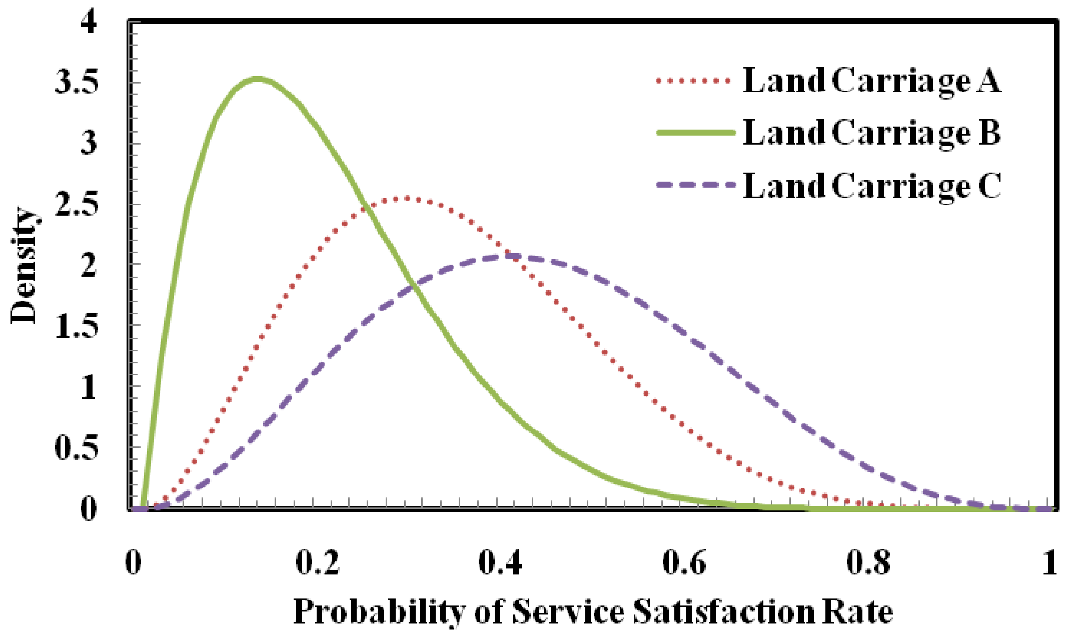
To evaluate the performance of the proposed SLA adaptation approach for composite services, we conducted experiments in a simulated volatile environment. The experimental evaluation is aimed to show the effectiveness of our approach in guaranteeing the satisfaction of the service consumers' requirements for the composite services.

We utilised the example composite service presented in Section 2.6.1. Component service *D* (land carriage) and *E* (shipping) are compliant with the *Sequence* pattern. We evaluated the satisfaction rate of the service consumer's requirements for the composite service with our SLA adaptation approach enabled that considers both adapting the land carriage service and the shipping service while the land carriage needs to be adapted. Considering that in different situations the difficulty levels of adapting and updating services might vary significantly, we model the land carriage companies' distribution and the shipping companies' distribution over their service satisfaction rates at two difficulty levels, i.e. easy and difficult, using *beta* distribution functions presented in Figure 5.5. Intuitively, service providers expose relatively high and low service satisfaction rates in easy and difficult situations respectively.

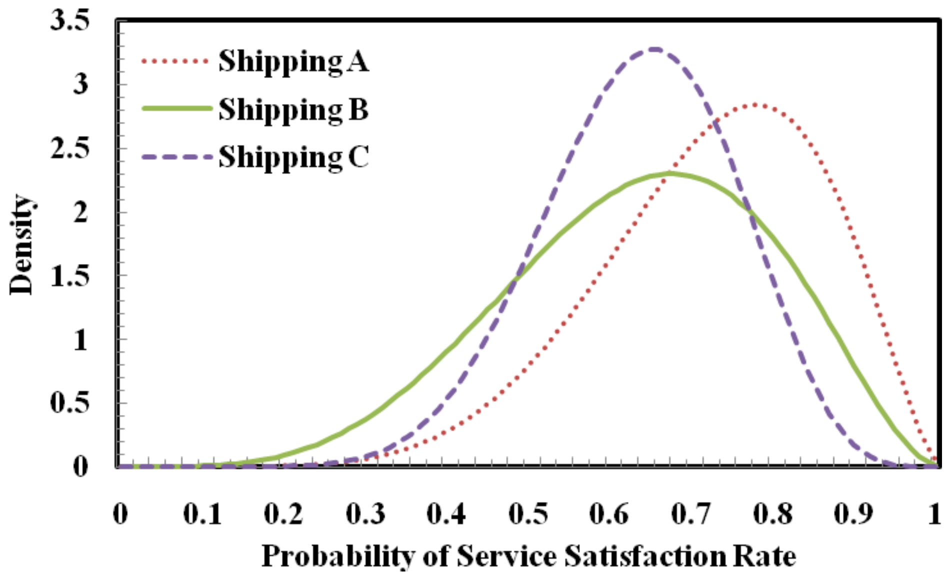
We ran 1,000 independent composite service instances for each experiment within a simulated volatile environment. Since the difficulty of adapting two sequential services might vary in different situations, we conducted comprehensive experiments with all the four combinations of difficulty levels. We measured the satisfaction rate of the service consumer's requirements by calculating the number of successful cases, i.e. cases where service consumer's requirements can be met after the adaptation, out of the overall cases.



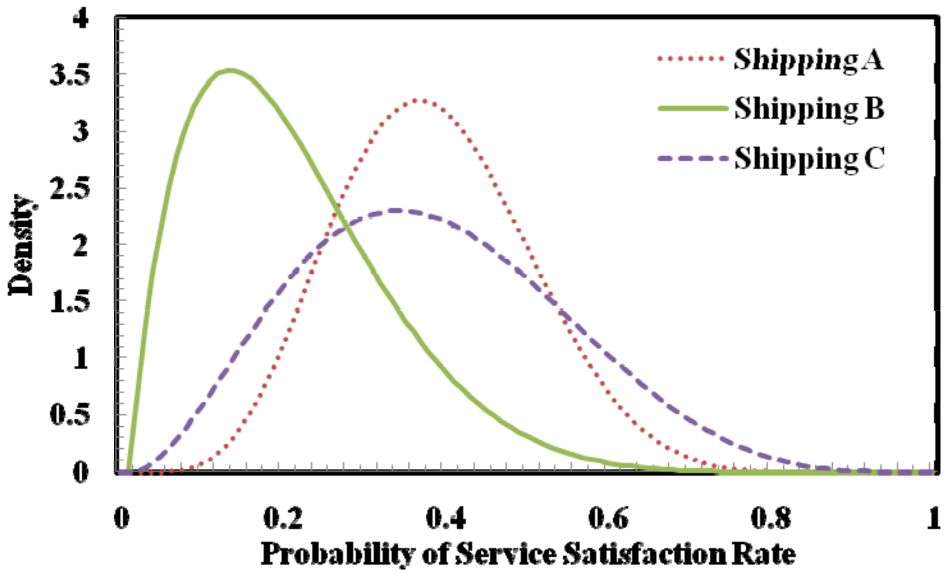
(a) Easy service satisfaction rates of land carriage companies



(b) Difficult service satisfaction rates of land carriage companies



(c) Easy service satisfaction rates of shipping companies



(d) Difficult service satisfaction rates of shipping companies

Figure 5.5 Probability density functions

Figure 5.6 compares the satisfaction rates of service consumer's requirements for the composite service in different situations with and without our approach enabled. The results demonstrate that in all situations our adaptation approach provides a more effective solution to the satisfaction of service consumer's requirements for the composite service. In situations with different combinations of difficulty levels, including difficult-difficult, easy-difficult, difficult-easy and easy-easy, our approach provides an increment of 22%, 12%, 46% and 25% respectively in the satisfaction rates of service consumer's requirements for the composite service.

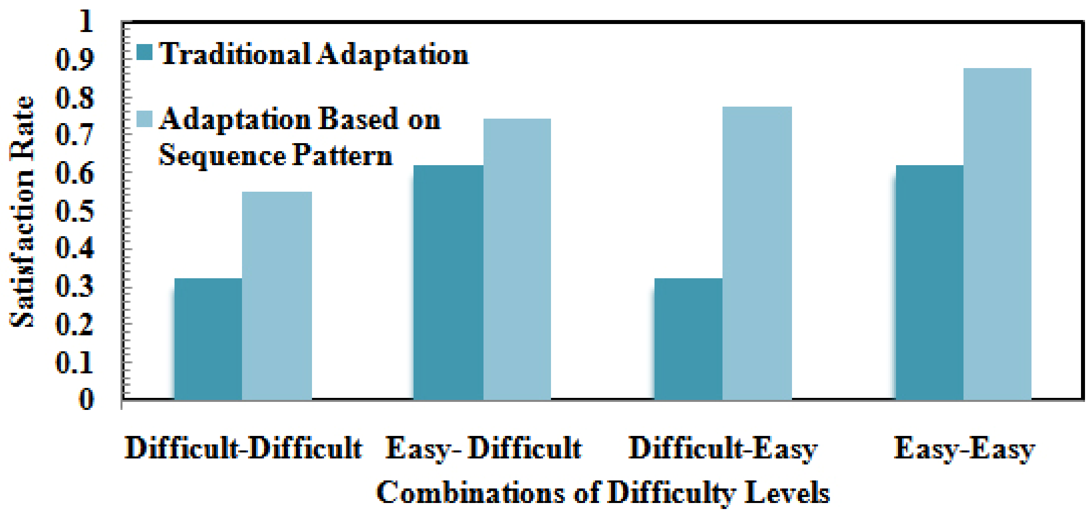


Figure 5.6 Comparison between traditional adaptation and our adaptation for Sequence pattern

The results from experiments demonstrate that our SLA adaptation approach a promising for the sequence pattern. Since the parallel patterns are also addressed using the same method (VOC), the results would be similar.

5.4 Discussion

SLA negotiation may also happen during the execution of the composite services due to the possibility of SLA violations. When a component service fails, it must be adapted (either updated or replaced). And the adaptation of a faulty service may trigger the adaptation of other component services. Accordingly, the SLAs attached to these component services must be adapted (either renegotiated or replaced). In SLA adaptation, if multiple component services need to be adapted, CASS introduced in Chapter 4 can be held in which the component services that need to be adapted are open for auction. In the cases where only one component service needs to be adapted, simpler SLA negotiation techniques can be adopted, e.g. one-on-one SLA negotiation between the service consumer and the failed service provider, or between the service consumer and new service providers, or the combination of the two.

Sometimes the SLA adaptation is required to be rapid, for example in time-sensitive business processes such as stock trading. In those cases, CASS needs to be configured to complete with accepted results within the time limit required. The analysis of the tradeoff between effectiveness and efficiency of CASS under these circumstances has yet to be undertaken.

5.5 Summary

In this Chapter, we have discussed how the value of changed information (VOC) is extended and applied to a new SLA adaptation to support SLA enforcement for service composition based on workflow patterns. Specifically, we have analysed and presented how to compute VOC for sequence and different parallel pattern scenarios. The SLA adaptation is performed within a certain scope defined by workflow patterns only when it is expected to pay off. By doing so, SLA adaptation for service composition can deliver satisfactory results while being kept within a reasonable scope. The experimental results show that our approach can significantly improve the satisfaction rates of service consumers' QoS requirements for the composite services in different situations.

Chapter 6

Effective profiling in SLA completion

As discussed in Section 2.5 and Section 3.4, SLA completion, as the third and also the last stage of the lifetime of an SLA, needs the execution of SLA profiling after the service provision to collect the results of SLA enforcement for future analysis of the service provider's reputation and its capability of enforcing SLAs. The basic principle is that the service providers with higher success rates of past SLA enforcements are more trusted by the service consumers in having the capability of enforcing SLAs successfully.

A service trust system based on service providers' performance over their past SLA enforcement can facilitate reputation-oriented service selection, which can enhance SLA negotiation and SLA adaptation, and protect the service consumers from various threats in the open and volatile SOC environment.

This chapter introduces ServiceTrust, a novel reputation-based service trust system which supports reputation-oriented service selection by estimating service consumers' trust over service providers based on their historic performance on SLA enforcement.

This chapter is organised as follows. Section 6.1 addresses the design consideration of ServiceTrust. Section 6.2 introduces the mechanisms that implement ServiceTrust. Section 6.3 describes how ServiceTrust evaluates the reputation of new services. Section 6.4 presents the experimental results regarding the effectiveness of ServiceTrust, and the resistibility of ServiceTrust against the two threats, i.e. malicious reputation manipulation and QoS abuse. Section 6.5 discusses the support of ServiceTrust for SLA negotiation and adaptation. Finally, Section 6.6 summarises this chapter.

6.1 Design consideration

One design goal of ServiceTrust is to help the service consumers evaluate the trustworthiness of the service providers. We believe that the evaluation of service providers' reputation should be based on their long-term reputation. The reasons are twofold. First, long-term reputation can highlight service providers' performance in the long term and smooth out fluctuations in the short term. Second, service providers' expectation of long-term reputation creates an incentive for their good performance at present. Another design goal of ServiceTrust is to help protect the service consumers from threats in the open SOC environment. A widely recognised threat that the service consumers are exposed to in the open SOC environment is that malicious service providers manipulate service consumers to report incorrect feedbacks in order to boost their reputations or to ruin their competitors' reputations [58]. It can also be done by malicious service providers to fake service consumers. Another major threat is QoS abuse, where service providers strategically alter their behaviour in QoS offering and then provide fraudulent service transactions in order to earn profit [101].

Due to the above issues, in service selection, approaches should be provided to help the service consumers evaluate the trustworthiness of the service providers, as suggested but not specified in [5, 54, 107]. However, sometimes it is difficult for a service consumer to determine how much it can trust a service provider due to the lack of sufficient experience and knowledge about the service provider. A direct approach to address this issue is to use a trust system which collects and processes feedbacks about service providers' past behaviour [51, 88, 92, 104]. However, to the best of our knowledge, no trust system has been tailored for service selection in the open SOC environment and the threats described earlier have not been properly addressed. Furthermore, different from the peers in the peer-to-peer (P2P) environment, the service providers in the SOC environment usually have unique identifications in order for the service consumers to identify their services. Therefore, existing trust systems in the P2P environment, which usually put a lot of effort in maintaining peers' anonymity property, are generally unsuitable to be directly applied in the SOC environment.

To summarise, ServiceTrust should provide two functions:

1. Evaluate service consumers' trust over service providers based on their performance over past SLA enforcement in the long term; and

2. Protect the service consumers from two threats: malicious reputation manipulation and QoS abuse

6.2 ServiceTrust mechanisms

In this section, we introduce a trust structure which consists of *local transactional rating*, *local trust*, *global trust* and *transactional trust*, and the supporting mechanisms.

6.2.1 Generating local transactional ratings

A local transactional rating describes a service consumer's experience of an individual service transaction with a service provider. Some early work [28, 51, 104], which use binary rating systems for calculating a given peer's reputation, demonstrate that binary-value ratings work quite well for file-sharing systems, in which a file is either a complete version or an incomplete version. An SLA in the SOC environment can be seen as an equivalent of a file in a file-sharing system because an SLA also only has two finalised status: fulfilled or unfulfilled, representing a successful service transaction or a failed one. Using binary values to rate service transactions is simple and does not require service consumers' physical participation. Another advantage of adopting binary-value ratings is that the ratings are explicit – a service transaction is either successful or unsuccessful in fulfilling the attached SLA. However, some recent work [100, 105] adopt numeric rating systems, in which the ratings are in a certain interval, e.g. [*0, 1*]. Compared to binary-value ratings, numeric-value ratings can model more accurately a service consumer's experience in a service transaction. However, it requires service consumers' direct participation in the rating process, which sometimes becomes an obstacle to the extensive use of the application. Moreover, service consumers' lack of incentive and knowledge to report authentic and accurate ratings over service transactions may result in undesired, inaccurate or even incorrect ratings.

To give application developers flexible choices, ServiceTrust supports both binary-value and numeric ratings. For binary-value ratings, *0* and *1* are used to represent a failed service transaction and a successful one respectively. The definition of service consumer *i*'s local transactional rating over the *n*th service transaction with service provider *j*, denoted as $r_{i,j}^{(n)}$, is defined as follows:

$$r_{i,j}^{(n)} = \begin{cases} 0 & \textit{service transaction failed} \\ 1 & \textit{service transaction succeeded} \end{cases} \tag{6.1}$$

Service consumers are also allowed to rate service transactions using a value in the interval of [*0, 1*], with *0* and *1* representing complete dissatisfaction and complete satisfaction respectively. Considering that service consumers might lack the knowledge of QoS satisfaction, it is advisable for application developers to provide the service consumers with necessary assistance during the rating process.

The mechanisms described in the rest of this chapter can accommodate both binary-value and numeric ratings.

6.2.2 Aggregating local transactional ratings

To obtain a service consumer's local trust over a service provider, local transactional ratings generated from the service consumer's past service transactions with the service provider need to be aggregated. In the aggregation, we consider the temporal dimension when evaluating the credibility of the local transactional ratings. It is not only their values that matter, but also at what time they are recorded – we assume that the local transactional ratings are recorded upon the completion of the service transactions, which is also when the attached SLA is completed. The credibility of a local transactional rating diminishes as time elapses. The ratings over a service consumer's recent service transactions with a service provider are more credible than the old ones. Also, when combining a service consumer's and other service consumers' personal local trust (as detailed in Section 6.2.3), the recent ratings from one service consumer are more credible than the old ones from another service consumer.

We use exponential moving average (EMA) scheme [21] to aggregate a service consumer's local transactional ratings over a service provider. Weights are computed to represent the credibility of the ratings according to how old they are. The weight of each older rating decreases exponentially, giving more credibility to recent ratings whilst not entirely discarding older ones. By doing so, short-term fluctuation of ratings can be smoothed out and long-term trend can be highlighted. Since the threshold between short-term and long-term is application specific, ServiceTrust uses parameter θ, as a time window, to specify valid ratings when aggregating the local transactional ratings. Ratings lying outside of θ are considered obsolete and thus discarded in the aggregation. θ can be set accordingly by the application developers to meet the requirements of applications.

The elapsed time since a service transaction has been performed is used to express how old the corresponding rating is. In order to compute the elapsed time of the ratings, ServiceTrust requires the rating time, noted by $t_{i,j}^{(n)}$, i.e. the time when the service transaction, to be recorded along with the rating in the form of 2-tuple: $(r_{i,j}^{(n)}, t_{i,j}^{(n)})$.

The process of calculating service consumer *i*'s local trust over service provider *j* by aggregating the series of local transactional ratings over the past service transactions between them, i.e. $[(r_{i,j}^{(1)},t_{i,j}^{(1)}),(r_{i,j}^{(2)},t_{i,j}^{(2)}),...,(r_{i,j}^{(n)},t_{i,j}^{(n)})]$, consists of the following five steps.

1. Compute the elapsed time, denoted as $et_{i,j}^{(n)}$, since each transaction was rated. The series of local transactional ratings becomes: $[(r_{i,j}^{(1)},et_{i,j}^{(1)}),(r_{i,j}^{(2)},et_{i,j}^{(2)}),...,(r_{i,j}^{(n)},et_{i,j}^{(n)})]$;

2. Determine the value of the time window θ;

3. Divide the time frame confined by θ into s time slots;

4. Compute the arithmetic average value of the local transactional ratings in each time slot, denoted as $ar_{i,j}^{(1)},ar_{i,j}^{(2)},...,ar_{i,j}^{(s)}$;

5. Aggregate $ar_{i,j}^{(1)},ar_{i,j}^{(2)},...,ar_{i,j}^{(s)}$ to obtain service consumer *i*'s local rating over service provider *j*, denoted as $R_{i,j}$, using exponential averaging scheme as follows:

$$R_{i,j} = \sum_{k=1}^{s} \alpha(1-\alpha)^k ar_{i,j}^{(k)} \tag{6.2}$$

where $0 < \alpha < 1$ controls how fast the credibility of the ratings decreases over time.

Besides θ, two other parameters, s and α, are manoeuvrable. They can be set by application developers to control the weight decrease in order to meet application specific requirements. The bigger s and α are, the faster the weight decreases, meaning the faster the old ratings in θ become incredible. Figure 6.1 gives an example of θ with s=7 and Figure 6.2 depicts an example of the weight decrease.

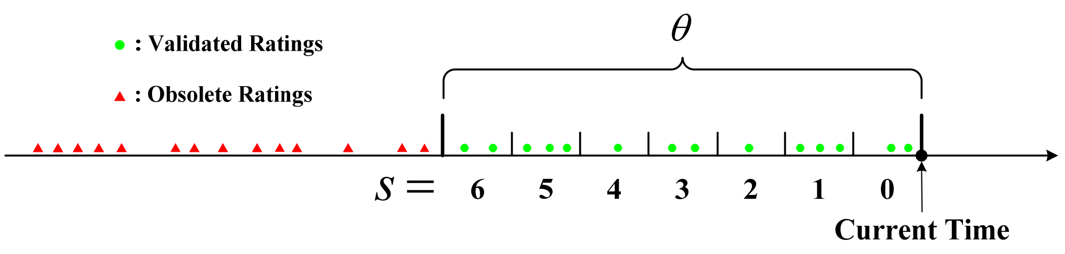


Figure 6.1 A sample time window (*s*=*7*)

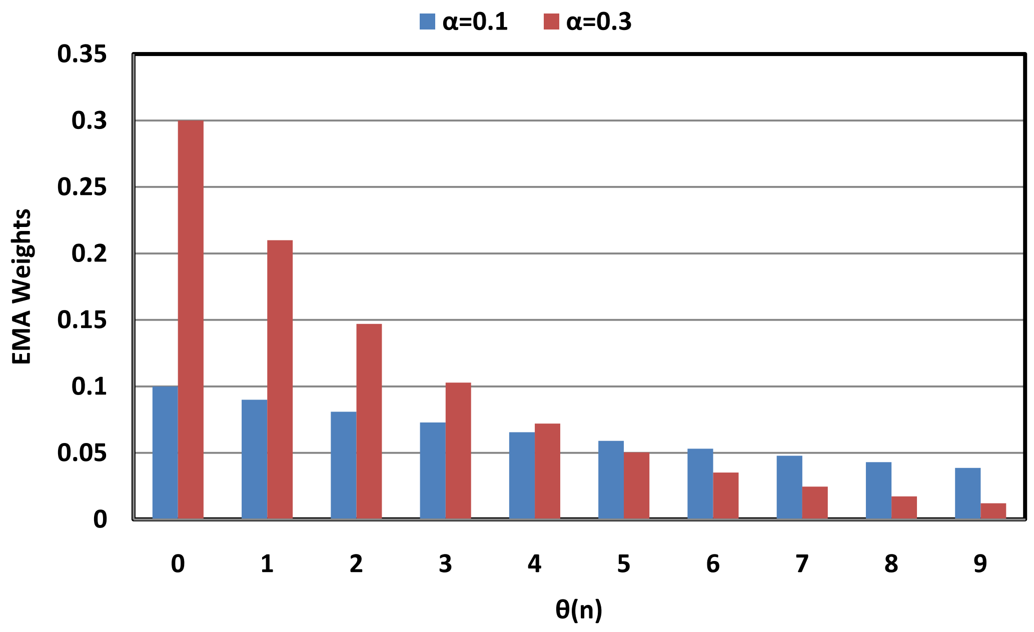


Figure 6.2 An example of EMA weight decrease (*α=0.1* and *0.3*, and *s =10*)

6.2.3 Combining personal trust

The local trust introduced in Section 6.2.2 reflects a service consumer's personal opinion of a service provider. To comprehensively analyse a service provider's capability to enforce SLAs based on its historic performance, a service consumer's local trust should be combined with other service consumer's local trust when evaluating the service consumer's global trust over the service provider. By doing so, the service consumer can obtain a global and comprehensive view of the service provider. A simple approach to the combination is to simply average all the local trust. An advanced approach is to compute a weighted average of all the local trust, where the weights represent the credibility of the local trust.

The credibility of a service consumer's local trust over a service provider depends not only on how old the local transactional ratings are (see Section 6.2.1), but also on how long the service consumer has had interactions with the service provider. Experience with the service provider in the longer-term gives the service consumer more information and knowledge about the service provider, thus enabling the service consumer to predict the service provider's ability and behaviour better [29, 93]. It also provides a firmer basis for calculating the credibility of the service consumer's local trust over the service provider. Therefore, when incorporating other service consumers' local trust into evaluating a service consumer's global trust over a service provider, we consider the relationship duration between the service consumers and the service provider, measured by the number of past service transactions between them. The longer relationship duration a service consumer has had with the service provider, the more credible its local trust over the service provider is.

We adopt Rayleigh cumulative distribution functions [30] to calculate the weights according to the number of a service consumer's past service transactions with the service provider. The credibility of service consumer *i*'s local trust over service provider *j*, denoted as $\beta_{i,j}$, is calculated as follows:

$$\beta_{i,j} = 1 - exp(\frac{-x^2}{2\sigma^2}) \quad (\sigma > 0) \tag{6.3}$$

where σ is a parameter that inversely controls how fast $\beta_{i,j}$ increases as the number of interactions, denoted as *x*, increases. σ can be set by the application developers, from *0* to theoretically ∞, to capture the characteristics of different application scenarios. Figure 6.3 presents an example of the calculation of $\beta_{i,j}$ using different σ.

Compared to other service consumers' local trust, a service consumer can choose to trust its own local trust more or less when evaluating its global trust over the service provider. To reflect this nature, the weight assigned to the service consumer's own local trust over the service provider, denoted as $\beta'_{i,j}$, is computed as follows:

$$\beta'_{i,j} = 1 - exp(\frac{-x^2}{2(\sigma+\varepsilon)^2}) \quad (\sigma+\varepsilon) > 0 \tag{6.4}$$

where x is the number of service transactions that service consumer i has had with service provider j and ε specifies *how much more* (using a negative number) or *how much less* (using a positive number) the service consumer trusts its own local trust over service provider j than other service consumers'.

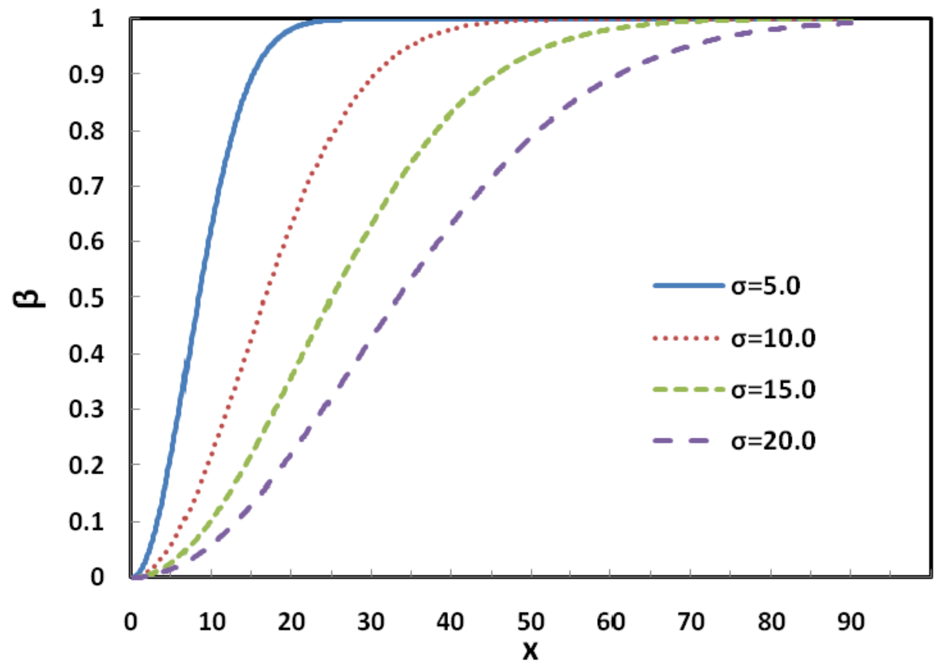


Figure 6.3 An example of the calculation of $\beta_{i,j}$

Then service consumer i's global trust over service provider j, denoted as $\tilde{R}_{i,j}$, can be calculated as follows:

$$\tilde{R}_{i,j} = \beta'_{i,j} \cdot R'_{i,j} + \sum_k \beta_{k,j} \cdot R_{k,j} \tag{6.5}$$

where $R'_{i,j}$ is service consumer i's own local trust over service provider j and $R_{k,j}$ is the k^{th} other service consumer's local trust over service provider j.

6.2.4 Evaluating transactional trust

The scheme presented in this section can be applied to resist various types of QoS abuse, e.g. execution time, availability and throughput, etc. Since the transaction amount is usually one of a service consumer's most important concerns about the service in the SOC environment, we present a solution to resist transaction amount abuse for demonstration.

To protect service consumers from transaction amount abuse, we incorporate the transaction amount into estimating service consumers' transactional trust for individual service transactions. We define transactional trust as the probability at which a service consumer believes the service provider will perform an individual service transaction and deliver expected outcomes specified in the attached SLA.

Transaction amount abuse usually consists of two steps. First, the malicious service provider fulfils service transactions with relatively small amounts to obtain a service consumer's trust. Second, the malicious service provider entices the service consumer to give it an order for a service transaction with a large amount, and then defraud the customer with fraudulent service transactions or inferior goods afterwards. Under other circumstances, a fraudulent service transaction might also be performed, e.g. a genuine service provider may make the transition into being malicious when it gets an order for a service transaction with an unusually large amount which reaches or crosses its threshold for being genuine.

We address this issue by evaluating the transactional trust in consideration of the similarity between the quote on the forthcoming service transaction and the average transaction amount of the successful service transactions the service provider has performed. The base for this approach is the spirit of situational trust [69]: experience from situations of a similar nature will give a means of determining risk accurately. When evaluating the transactional trust, we consider two factors:

1. *The average amount of successful service transactions that the service provider has performed.* In general, the larger the quote on a service

transaction is than the average amount of its past successful service transactions, it is more likely that the service provider will provide a fraudulent service transaction.

2. *The extent of amounts of successful service transactions that the service provider has performed.* If a service provider has a large extent of amounts of successful service transactions, the chance that it will provide a fraudulent service transaction is slim.

Combining the considerations on the above two factors, we evaluate service consumer *i*'s transactional trust for a forthcoming service transaction provided by service provider *j*, denoted as $\bar{R}_{i,j}$, using formula (6.6).

$$\bar{R}_{i,j} = \gamma \cdot \tilde{R}_{i,j} \tag{6.6}$$

$$\gamma = \left(\frac{1}{\Delta^2}\right)^k \tag{6.7}$$

$$\Delta = \frac{q_{new}}{a_j^{ave}} \cdot \frac{1}{cv_j} \tag{6.8}$$

$$cv_j = \frac{\sqrt{\sum_{m=1}^{M} (a_j^m - a_j^{ave})^2}}{a_j^{ave}} \tag{6.9}$$

where γ is the *transactional amount impact factor*, *k* is the parameter that controls how fast the transactional trust decreases as Δ increases, q_{new} is the quote on the forthcoming service transaction, a_j^{ave} is the average amount of the successful service transactions provider j has performed, a_j^m is the amount of the m^{th} successful service transaction provider j has performed, and cv_j is the coefficient of variation of $a_j^1, a_j^2, \ldots, a_j^m, \ldots, a_j^M$. Parameter *k* can be set by application developers according to the requirements of the applications. For example, in the scenario where the fluctuation of prices is relatively violent, such as the global crude oil market, a small *k* is advisable.

Usually the smaller the transaction amount is, the more satisfactory it is for a service consumer because the transaction amount is a negative property (see Definition 4 in Section 4.3.2). However in relation to some positive QoS (see Definition 3 in Section 4.3.2) such as availability and throughput, the higher the better it is for the service consumers. In those cases, formula (6.10) can be used to replace formula (6.8):

$$\Delta = \frac{a_j^{ave}}{q_{new}} \cdot \frac{1}{cv_j} \qquad (6.10)$$

6.3 Initial trust for new services

In the discussion so far, we assume that a service provider provides one type of service. However, in the SOC environment, the service providers might be able to provide multiple types of services with respective service identifications. Accordingly, in ServiceTrust, a service consumer's trust over a service provider is service specific, and is estimated based on the service provider's historic performance over individual types of services. One issue that may occur is that when a service provider starts offering a new service, there is no historic performance information about the new service for service consumers to refer to. In this case, a service consumer's trust for this new service cannot be evaluated.

The development of a service consumer's initial trust for a new service usually goes through two stages: an exploratory stage and a commitment stage, which reflects the general belief in the trust literature [10]. At the exploratory stage, the service provider's reputation will influence the service consumer's intention to trust the service provider. At the commitment stage, experience-based knowledge will readily replace the tentative trust built at the exploratory stage [71]. Another factor that influences a service provider's tentative trust over a service provider is its familiarity with the service provider [34, 67]. Familiarity is referred to as understanding of the context which the service transaction is involved, and hence is considered the precondition for tentative trust [67].

From the perspectives of both the reputation and the context, we assume that a service provider with good reputation obtained from its existing services tends to provide the new service at a high success rate. This assumption is acceptable at least at the early stage of the new service's appearance because the service provider has to cater for the service consumers in order to quickly develop its reputation for the new service and to attract more potential service consumers [76]. Therefore, a service consumer's initial trust for a new service can be estimated through looking into the service provider's reputations for its existing services.

We evaluate a service provider's *global reputation* by aggregating its reputations for individual services. The estimation of a service consumer's initial trust for the new service is based on the service provider's global reputation. After interacting with the service provider, the service consumer can gradually incorporate its direct experience and knowledge of the service into developing its trust for the service following the procedure presented in Section 6.2. In ServiceTrust, service consumer *i*'s global trust over service provider *j*, denoted as $\hat{R}_{i,j}$, based on its trust for service provider *j*'s *N* individual existing services is calculated as:

$$\hat{R}_{i,j} = \frac{1}{N}\sum_{n=1}^{N}\tilde{R}_{i,j}^{(n)} \tag{6.11}$$

where $\tilde{R}_{i,j}^{(n)}$ is service consumer *i*'s trust for the n^{th} individual existing service provided by service provider *j*.

6.4 Experimental evaluation

In this section, we first assess the effectiveness of ServiceTrust as compared to a random service selection with no trust system enabled. Then we demonstrate ServiceTrust's resistibility against the threats of malicious reputation manipulation and QoS abuse. All the experiments are based on the composite service presented in Section 2.6.1. The issue of initial trust for new services is not directly related to either the effectiveness on service selection or the resistibility against threats, and hence is not included in the experiments.

6.4.1 Simulation configuration

Network model. We set up a SOC environment based on our previous work [42] in which the service consumers look up the service providers in an efficient decentralised way. The simulation environment consists of 400 service consumers and 200 service providers. Service consumers can issue requests for services and service providers respond to these requests. The service consumers and providers communicate in a P2P manner. The service consumers can access all the information about the service providers' historic performance.

Node model. Five types of services are provided by the 200 service providers, 40 for each. Each service provider has an inherent success rate randomly picked from a certain interval for its past and forthcoming service transactions. Different intervals for inherent success rates, including [*0.9, 1*], [*0.8, 1*] [*0.7, 1*], [*0.6, 1*], [*0.5, 1*] and [*0.4, 1*], are used to simulate different volatile environments, [*0.9, 1*] being the best and [*0.4, 1*] being the worst. In all experiments, service providers perform service transactions at their inherent success rates except under threat model #5 in experiment #6. In the experiments with ServiceTrust enabled, genuine service consumers select the available service provider they have the highest trust over (global trust in experiments #1 to #5 and transactional trust in experiment #6), and rate service transactions honestly. Malicious service consumers select service providers and rate service transactions under corresponding threat models. The threat models will be detailed in Section 6.4.3. In experiments where ServiceTrust is disabled, service consumers randomly select service providers.

Simulation execution. The simulation proceeds in simulation cycles. Each simulation cycle is subdivided into an evaluation cycle, a transaction cycle and a rating cycle. The details about each cycle are as follows:

- In an evaluation cycle, each service consumer initiates an instance of the composite service. Then the service consumers evaluate their global trust or transactional trust over the service providers for the five component services – manufacturing, rough processing, fine machining, land carriage and shipping – that compose the composite service.

- In a transaction cycle, each service consumer requests five services – one for each component service – based on the results from the trust evaluation in the evaluation cycle. Service providers correspond and complete service transactions. In each transaction cycle, each service provider can accommodate up to a maximum of 40 service requests. If a service provider is fully loaded, the service consumer will turn to the functionally equivalent service provider it has the next highest trust over.

- In a rating cycle, service consumers rate the service transactions honestly or under corresponding threat models. Binary rating values, described in Section 6.2.1, are used[7].

Upon the completion of each simulation cycle, the success rates of overall composite service instances are collected. Each experiment is run 20 times and the results of all runs are averaged. We analyse the statistics to assess ServiceTrust by measuring the average success rates of overall composite service instances.

ServiceTrust parameters. Table 6.1 summarises the parameters carefully chosen for the simulation in order to calculate service consumers' trust over service

[7] We choose not to use numeric ratings to avoid unnecessary issue of modeling service consumer's satisfaction from QoS.

providers based on their historic performance in the long term.

Table 6.1 ServiceTrust parameters used in simulation

α	θ	s	σ	ε	k
0.1	10 simulation cycles	10	15	-5	1/7

The base configurations used in each experiment are summarised in Table 6.2.

Table 6.2 Experimental settings

	Experiments #1 - #6					
	#1	**#2**	**#3**	**#4**	**#5**	**#6**
Number of Service Types	5	5	5	5	5	5
Number of Service Providers	200	200	200	200	200	200
Number of Service Consumers	400	400	400	400	400	400
Percentage of Malicious Service Consumers and Providers	0%	10% - 70%	10% - 70%	10% - 70%	10% - 70%	10% - 70%
Interval for Inherent Success Rates of Service Providers	[0.9, 1] to [0.4, 1] in steps of 0.1	[0.4 1]	[0.4 1]	[0.4 1]	[0.4 1]	NA
Number of Simulation Cycles	20	20	20	20	20	20

6.4.2 Effectiveness of ServiceTrust

In experiment #1, we compare the average success rates of overall composite service instances with ServiceTrust enabled against disabled in volatile environments without malicious attacks.

Experiment #1.

Setup. In experiments where ServiceTrust is enabled, service consumers evaluate their global trust over service providers in the evaluation cycle. In the transaction cycle, each service consumer issues five service requests to the available service providers which it has the highest global trust over. The service providers correspond by performing the service transaction at their own inherent success rates. In the rating cycle, service consumers rate the service transactions honestly. In experiments where ServiceTrust is disabled, the only difference is that service consumers do not evaluate their trust over the service providers. Instead, they select service providers randomly. Six volatile environments are simulated using different intervals for the service providers' inherent success rates.

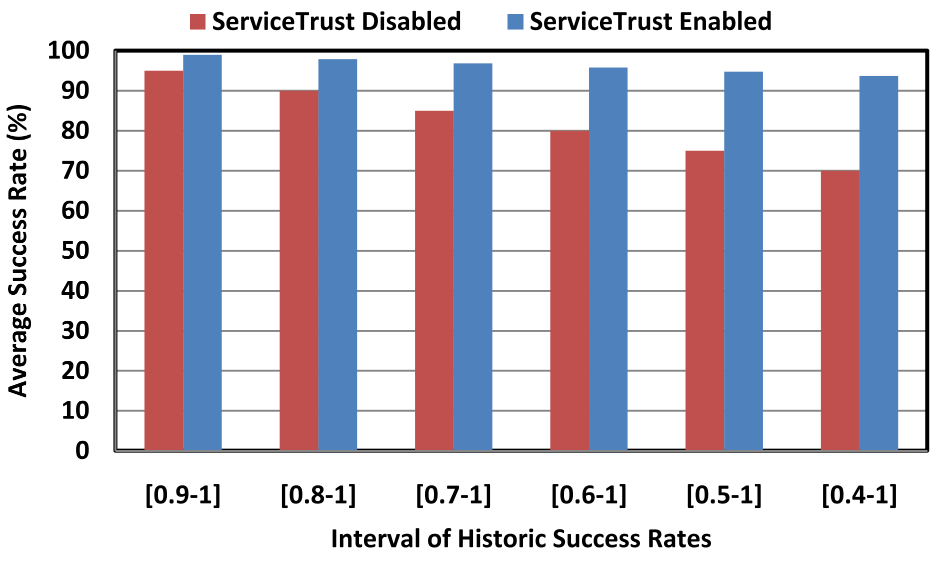


Figure 6.4 Increase of average success rates of overall composite service instances with ServiceTrust enabled

Discussion. Figure 6.4 depicts and compares the results from experiment #1. The results show that ServiceTrust can significantly increase the average success rates of overall composite service instances in different volatile experiments. As the environment becomes more volatile, the average success rate decreases drastically in the absence of ServiceTrust. But with ServiceTrust enabled, even when different service providers' success rates vary in the largest interval, i.e. [*0.4, 1*], the average success rate of overall composite service instances still remains at 93% in contrast to around 70% in the case of no ServiceTrust enabled.

6.4.3 Resistibility against threats

Table 6.3 Service consumers and providers' behaviour

Threat Model	**Malicious Service Providers**	**Malicious Service Consumers**	
		Service Selection	**Rating**
Threat Model #1	NA	randomly select service providers	rate 1 over all service transactions with malicious service providers
Threat Model #2	NA	select only malicious service providers	rate 1 over all service transactions with malicious service providers
Threat Model #3	NA	randomly select service providers	rate 0 over all service transactions with genuine service providers
Threat Model #4	NA	select only genuine service providers	rate 0 over all service transactions with genuine service providers
Threat Model #5	provide fraudulent services at the probability of 1-λ	NA	NA

We now evaluate ServiceTrust's resistibility against the threats of malicious reputation manipulation and QoS abuse. Malicious reputation manipulation includes patterns described by four threat models: individual and collective malicious reputation boost, individual and collective malicious reputation ruin. The four threat models describe different strategic malicious behaviour of the service consumers who are manipulated or faked by malicious service providers to interact with targets intentionally and to provide incorrect ratings. The threat model for QoS abuse describes the service providers' strategic change of behaviour in QoS offering. Table 6.3 lists the service consumers and providers' behaviour under each threat model. For QoS abuse, we evaluate the ServiceTrust's resistibility against transaction amount abuse for demonstration.

Experiment #2.

Threat model #1. *Individual malicious reputation boost***.** Under this model, malicious service consumers and providers do not recognise each other until they interact. Malicious service consumers rate *1* over all the service transactions with malicious service providers.

Setup. In each experiments, the service providers' success rates of service transactions are picked from the interval of [*0.4, 1*]. 10%, 20%, 30%, 40%, 50%, 60% and 70% service consumers and providers are set malicious to simulate different volatile environments. Service consumers are randomly picked to be malicious while service providers with lower inherent success rates are more likely to be set malicious. A malicious service consumer randomly selects available service providers to request for services. Malicious service consumers rate *1* over all the service transactions with malicious service providers.

Discussion. Figure 6.5 depicts the average success rates of overall composite service instances in different volatile environments with ServiceTrust enabled and disabled as the simulation proceeds through 20 simulation cycles. The results show that as the percentage of malicious service consumers and providers increases the average success rate throughout the simulation decreases, but still remains higher than random service selection for most of the time (more than 99%). We observe that the malicious attack can slightly decrease the average success rates from time to time by boosting the malicious service providers' reputations. It makes the genuine service consumers trust and select the malicious service providers more. Those malicious service providers have relatively low inherent success rates and thus bring down the average success rates of overall composite service instances. We also observe that the average success rates always recover to a normal level soon after decreasing due to malicious attack (usually no more than 2 simulation cycles). The reason is that the malicious service providers lose the genuine service consumers' trust with their poor performance due to relatively low inherent success rates. And the service consumers' trust over the genuine service providers can be recovered by the correct ratings from successful service transactions with the genuine consumer providers. The conclusion is that ServiceTrust can well protect the service trust system from being undermined by individual malicious reputation boost.

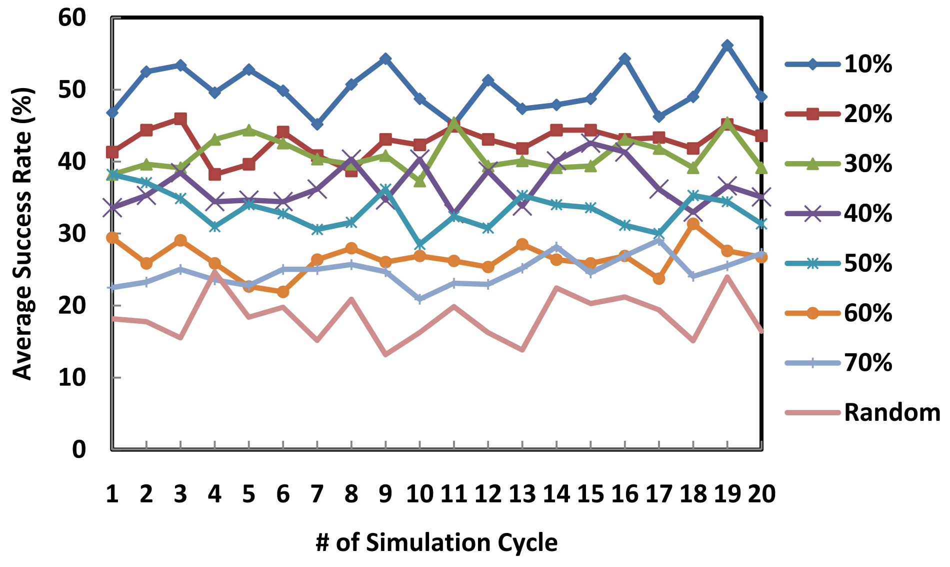


Figure 6.5 Average success rates in different volatile environments under threat model #1

Experiment #3.

Threat model #2. *Collaborative malicious reputation boost.* Under this threat model, malicious service consumers know about malicious service providers in the first place. They form a malicious collective by performing service transactions among themselves and rate *1* over all such transactions.

Setup. The simulation is run with the similar configuration described in experiment #2 albeit with the malicious service consumers behaving under threat model #2.

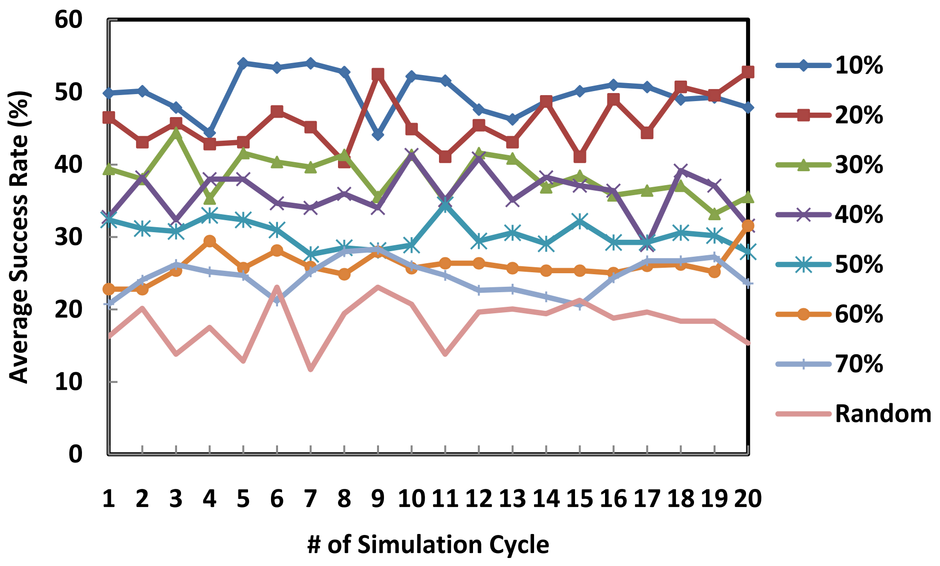


Figure 6.6 Averagge success rates in different volatile environments under threat model #2

Discussion. Figure 6.6 depicts the results from experiment #3. The results are similar to those from experiment #2. Service selection with ServiceTrust enabled performs better than random service selection for most of the time. The average success rate drops down from time to time but always recovers soon. The attack from malicious service consumers can slightly decrease the average success rate in the short term but cannot last long. The service trust system is safely guarded by ServiceTrust from collective malicious reputation ruin.

Experiment #4.

Threat model #3. *Individual malicious reputation ruin*. Under this model, malicious service consumers and providers do not recognise each other until they interact.

Malicious service consumers rate *0* over all the service transactions with genuine service providers.

Setup. The experiment configuration is similar to that in experiment #2, albeit with malicious service consumers behaving under threat model #3.

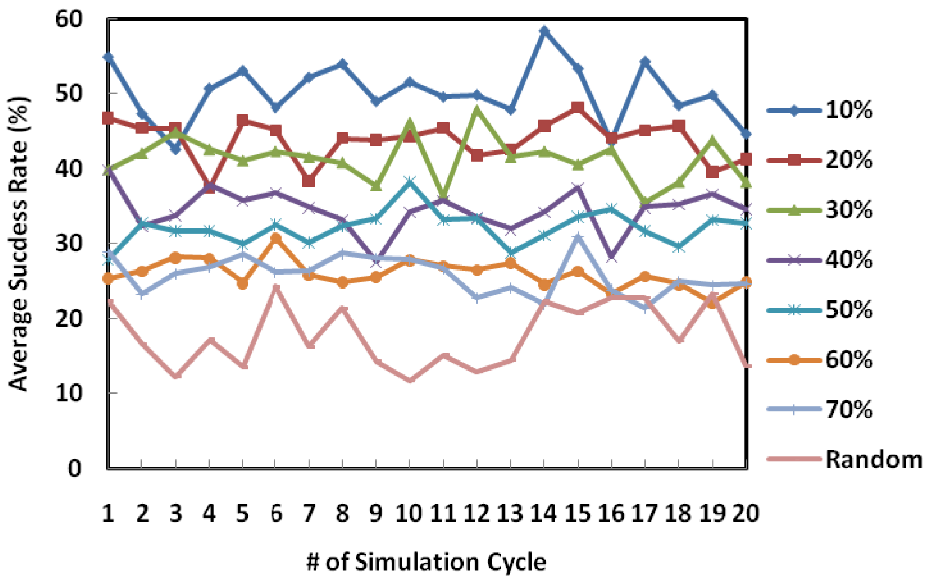


Figure 6.7 Averagge success rates in different volatile environments under threat model #3

Discussion. Figure 6.7 depicts the results from experiment #4. The average success rates of composite service instances when ServiceTrust is enabled are better than random service selection for most of the time. The short-term decrease of the average success rate caused by the malicious attack is insignificant. Service consumers' trust over the genuine service providers always recover soon after being ruined by the malicious service consumers. This experiment demonstrates that ServiceTrust can well resist the threat of individual malicious reputation ruin.

Experiment #5.

Threat model #4. *Collective malicious reputation ruin*. Under this model, the malicious service consumers know about the malicious service providers beforehand.

They form a malicious collective. Malicious service consumers perform service transactions only with service providers that do not belong to the malicious collective and rate *0* over all such transactions.

Setup. The experiment configuration is similar to experiment #3. The only difference is that malicious service consumers only request for services from genuine service providers.

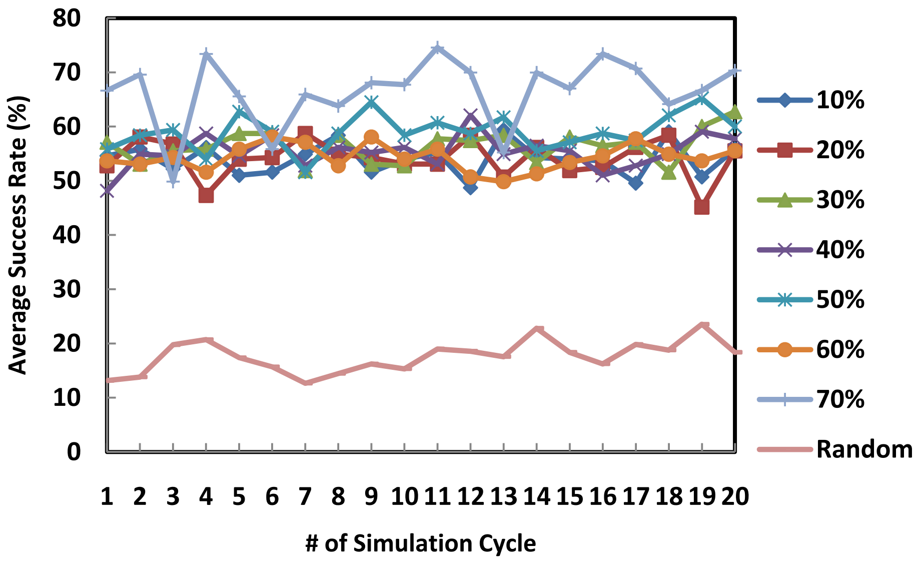


Figure 6.8 Averagge success rates in different volatile environments under threat model #4

Discussion. Figure 6.8 depicts the results from experiment #5. The average success rates of overall composite service instances rarely drop down to lower than 50% under the malicious attack. The waves in Figure 6.8 demonstrate that the attack still takes effect only in the short term. Both the genuine and malicious service consumers select the genuine service providers, albeit with different aims. The malicious service providers – the ones with relatively low inherent success rates – are not selected by any service consumers at all. As a result, there are no ratings over them in any simulation cycles. As presented in Section 6.2.2, those malicious service providers will lose the service consumers' trust and thus will not be selected any more. Meanwhile, the genuine service consumers' correct ratings can protect the genuine service providers from the malicious attack in the long term. Therefore, the average success rate of overall composite service instances always remains at a very high level. This experiment demonstrates that ServiceTrust has very good resistibility against collective malicious reputation ruin.

Experiment #6.

Threat model #5. *Transaction amount abuse.* Under this model, the malicious service providers provide fraudulent services at the probability of $1-\gamma$ (see formula (6.7) in Section 6.2.4).

Setup. Service providers' history of transaction amounts are generated using randomly created normal distribution functions, ranging from 50 to 250 monetary units. An example of three normal distribution functions used can be found in Figure 6.9. The quotes of requested service transactions are randomly picked from the interval of [*50, 250*]. In this set of experiments, a malicious service provider is a service provider whose average transaction amount is smaller than the quote of the requested service transaction. A malicious service provider performs a fraudulent service transaction at the probability of $1-\gamma$. We control the generation of the quotes for requested service transactions and the history of transaction amounts of the service providers to create different volatile environments with malicious service consumers and providers consisting of a fraction of 10% to 70% in steps of 10%. To identify ServiceTrust's resistibility against transaction amount abuse, we only look at the numbers of fraudulent services caused by unusually high transaction amounts. The failed service transactions caused by service providers' inherent failure rates are not taken into statistics.

Discussion. Figure 6.10 depicts the results from experiment #6. The results show that ServiceTrust can almost perfectly protect the service consumers from being deceived by transaction amount abuse. In the most volatile environment with 70% malicious service consumers and providers, the average success rate is still above 99%. The reason is that when the transaction amount is unusually higher than the normal amounts that a service provider used to deal with, the service consumer's transactional trust over that service provider drops immediately and drastically. The chance is very slim that a malicious service provider will be selected by a service consumer.

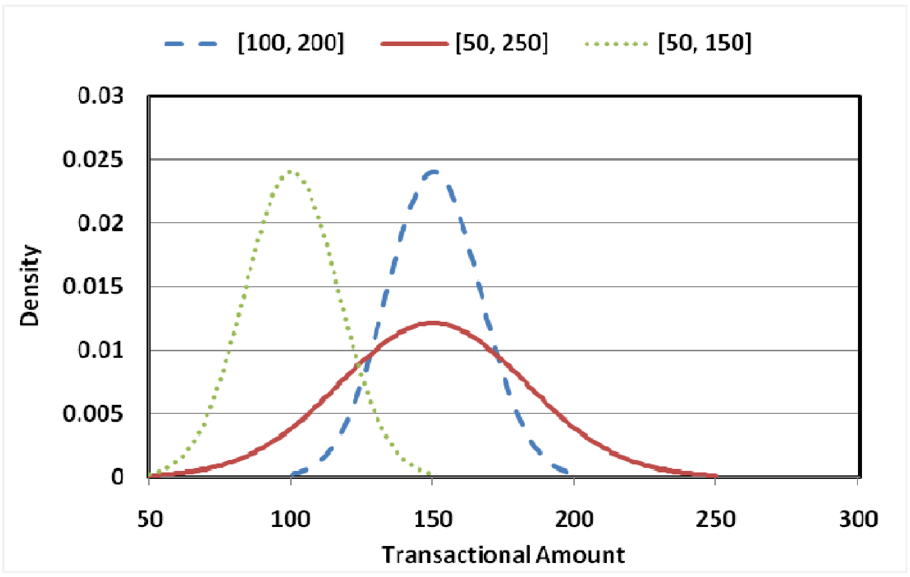


Figure 6.9 An example of normal distribution functions used to generate service providers' historic transaction amounts

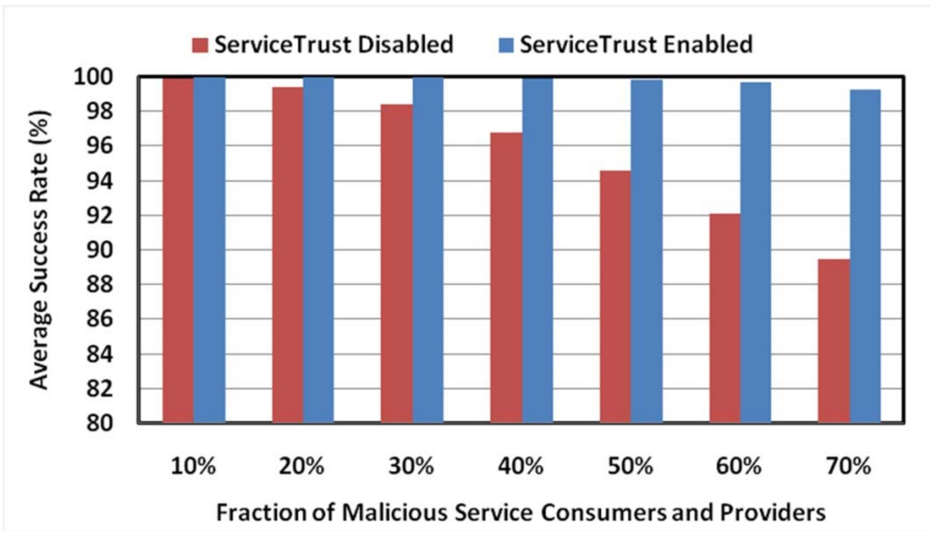


Figure 6.10 Average success rates in different volatile environments under threat model #5

6.5 Discussion

ServiceTrust can help service consumers identify trustworthy service providers and resist the threats of malicious reputation manipulation and QoS abuse. The service consumers' trust over the service providers can be taken into account as one of the criteria to compare the service providers. Both SLA negotiation and SLA adaptation can benefit from the trust information provided by ServiceTrust as they both involve the process of service selection, i.e. service provider selection. Obviously, selecting trustworthy service providers can minimise service failures caused by service providers' incapability at runtime.

ServiceTrust can provide more than the abovementioned functionality through collecting and analysing more comprehensive information other than the results of SLA enforcement, for example, QoS that service providers have ever proposed in the process of SLA negotiation. As discussed in Chapter 5, the possible QoS that service providers can provide can be used to facilitate SLA adaptation. Recall that ServiceTrust estimates transaction trust based on service providers' historic QoS, i.e. QoS service providers have ever provided in past SLA enforcement. The information of service providers' historic QoS, collected by ServiceTrust, can be used to estimate the possible QoS that the service providers can provide. By doing so, the accuracy of the results from formulas (5.1)-(5.3), (5.5)-(5.6), (5.8)-(5.9), (5.10) and (5.12) can be improved, which will improve the effectiveness of the SLA adaptation approach presented in Chapter 5.

6.6 Summary

In this chapter, we presented ServiceTrust – a novel service trust system which supports SLA enforcement based on SLA profiling. ServiceTrust can improve the success rate of composite services by helping service consumers identify trustworthy service providers through analysing service providers' long-term reputation as demonstrated by the experimental results. In addition, experimental results show that ServiceTrust can well resist the following malicious threats: 1) individual malicious reputation boost; 2) collective malicious reputation boost; 3) individual malicious reputation ruin; 4) collective malicious reputation ruin; and 5) QoS abuse.

Chapter 7

Conclusions and future work

In this chapter, we summarise this thesis and the contribution of this thesis and discuss some future research directions for SLA management for service composition.

7.1 Summary of this thesis

The primary objective of this research is to investigate a comprehensive solution to SLA management for service composition. The summary of this thesis is as follows:

- Chapter 1 introduced the basic concepts of SOC and service composition. It also introduced the key issues addressed in this thesis and the structure of this thesis.

- Chapter 2 introduced the related concepts involved in this thesis, including service, service composition, SLA, as well as the major work related to SLA negotiation, SLA adaptation and SLA profiling. It also presented the research requirements analysis. Based on the requirements analysis, it was argued that a comprehensive solution to SLA management for service composition is needed. The issue of QoS-aware service composition must be addressed at build time, runtime and completion time. Also, effective and efficient SLA management for service composition must be provided in an integrated way.

- Chapter 3 introduced the lifetime of SLA, including three major stages: establishment, enforcement and completion. Each of these stages involves one or more supporting SLA operations. In the model of SLA lifetime proposed in this thesis, SLA negotiation is adopted to support SLA establishment, SLA monitoring, compliance checking and adaptation constitute SLA enforcement, and SLA profiling is performed upon SLA completion. The SLA operations not only provide the required functionalities at respective stage of the SLA lifetime, but also interact with each other to provide a comprehensive and integrated solution to SLA management for service composition.

- Chapter 4 presented Combinatorial Auction for Service Selection (CASS) – an innovative SLA negotiation approach to support SLA establishment for service composition based on combinatorial auction. In CASS, the component services that compose a composite service are open for auction. Candidate service providers are allowed to bid for multiple component services that have multiple QoS properties. Round by round, the candidate service providers can revise their bids. The final winning service providers are selected to provide corresponding component services. According to the experimental results, CASS can significantly improve the effectiveness of the SLA negotiation for service composition in an efficient way.

- Chapter 5 introduced a new SLA adaptation approach to support SLA enforcement for service composition. The proposed approach considers both the benefit and the cost of the adaptation solution. The adaptation solution that brings the most profit will be selected. In addition, the impact of the adaptation of the faulty component service on other component services is taken into account. Workflow patterns are used to specify the scope of adaptation, aiming to fulfil the service consumers' QoS requirements for the composite services. As demonstrated by the experimental results, the proposed SLA adaptation approach can achieve the goal of satisfying service consumers' QoS requirements for the composite services much better.

- Chapter 6 introduced ServiceTrust – a novel service trust system to support reputation-oriented service selection based on SLA profiling which is performed upon the completion of SLAs. The trust architecture consists of local transactional rating, local trust, global trust and transactional trust. Those trust values are calculated based on service providers' historic performance over past SLA enforcement. As illustrated by the experimental results, ServiceTrust can significantly improve the success rates of the composite services and well protect service consumers from the threats of malicious reputation manipulation and QoS abuse. The trust information provided by ServiceTrust can also be used to enhance SLA negotiation and SLA adaptation in an integrated manner.

7.2 Contribution of this thesis

The significance of this thesis is that it addresses the problem of QoS-aware service composition at build time, runtime and completion time. This research investigates a set of technologies that supports different SLA operations at different stages of SLA lifetime. This research contributes significantly a lot to the challenging research area of service composition. The major outcome of this research is a comprehensive and integrated solution to SLA management for service composition. The effectiveness of the proposed technologies that constitute the solution has been evaluated by experiments. Therefore, this research illustrates cutting-edge technologies for SLA management for service composition. The key contribution of the work presented in this thesis is fourfold:

- The lifetime of SLA is proposed, which consists of three stages: establishment, enforcement and completion. The SLA operations that support each of the stages are also identified: SLA negotiation for enforcement, SLA monitoring, compliance checking and adaptation for enforcement, and SLA profiling for SLA completion. Moreover, the interactions among the SLA operations that facilitate a comprehensive and integrated solution to SLA management for service composition are also presented.

- An innovative SLA negotiation approach for service composition, namely Combinatorial Auction for Service Selection (CASS), is designed to support SLA establishment. CASS captures the dynamics of the service providers by allowing bidders to bid for combinations of component services and to flexibly express their offers and preferences for the quality of the component services. CASS also addresses the issue of multi-dimensional QoS by adopting a flexible QoS model and a multi-dimensional QoS-based bid evaluation scheme named Additive Ranking Position Evaluation (ARPE). A heuristic solution is provided to solve the winner determination problem. The ask QoS generation is designed for coordinating the auction process. The experimental evaluation shows that, by allowing the service providers to bid for combinations of component services and the giving them the incentives to bid aggressively, the effectiveness of the SLA negotiation can be significantly improved – the quality of the composite service achieved from the SLA negotiation can be improved. In addition, CASS is also demonstrated to be efficient in environments on different scales.

- A new SLA adaptation approach to support SLA enforcement for service composition based on workflow patterns is designed. Value of changed information (VOC) is extended and replied to SLA adaptation for service composition. VOC is used to compute the tradeoff between the expected value and the cost from adapting the service. The SLA adaptation is performed only when it will pay off. Workflow patterns are used to define the scope of the adaptation when multiple component services need to be adapted. By doing so, the adaptation can be confined within a certain scope, limiting the impact of faulty component services on other component services.

- A novel service trust system called ServiceTrust is designed to support reputation-oriented service selection based on SLA profiling upon SLA completion. ServiceTrust can quantify and compare the trustworthiness of service providers based on their historic performance over SLA enforcement. ServiceTrust combines a service consumers' and other service consumers' personal trust to estimate the service consumer's global trust over a service provider. One unique feature of ServiceTrust is to calculate the credibility of a service consumer's trust over a service provider by taking into account the temporal factor and the relationship duration between them. Another feature of ServiceTrust is to evaluate service consumers' transactional trust over service providers by considering the QoS of their past service transactions.

7.3 Future work

In the future, further investigation into SLA management for service composition can be carried out in several directions:

- During the process of SLA establishment, before the SLA negotiation, the service consumers' requirements must be mapped to machine-readable terms according to standard SLA specifications. This process is actually dependent on who will automatically perform the SLA negotiation on behalf of the service consumers. For example, if agent technologies are adopted to automate the SLA negotiation, before the negotiation, the service consumers' QoS requirements must be translated and specified in an agent-readable way.

- The impact from different bidder behaviours on the effectiveness and efficiency of CASS can be taken into account when further elaborating

CASS. Furthermore, some add-on mechanisms can be designed to automate or semi-automate the process of CASS by allowing service consumers and providers to preset the thresholds of their offers to be proposed in CASS.

- The accuracy of VOC computation adopted in the SLA adaptation approach can be improved by using ServiceTrust. The efficiency of the proposed SLA adaptation approach can be evaluated through corresponding experiments.

- The impact of adapting one faulty component service on a region, i.e. a group of component services, instead of just one service can be investigated.

- A complementary scheme can be designed to offer incentive to service consumers to participate in ServiceTrust and provide correct ratings over SLA enforcement. Furthermore, the resistibility of ServiceTrust against more threats that exist in the open SOC environment can be further investigated.

- Experiments can be conducted to assess the performance of the proposed approaches in the cloud computing environment.

Bibliography

[1] Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N. M., Kumar, A., Mittal, S., Srivastava, B.: "Synthy: A System for End to End Composition of Web Services", *Journal of Web Semantics*, vol. 3, no. 4, pp. 311-339, 2005.

[2] Agarwal, V., Dasgupta, K., Karnik, N. M., Kumar, A., Kundu, A., Mittal, S., Srivastava, B. "A Service Creation Environment Based on End to End Composition of Web Services", *Proc of 14th International Conference on World Wide Web (WWW2005)*, Chiba, Japan, 2005, pp. 128-137.

[3] Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C. K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A., "Web Services Business Process Execution Language Version 2.0", OASIS, 2007, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[4] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Toshiyuki, N., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M., "Web Services Agreement Specification (WS-Agreement)", World-Wide-Web Consortium (W3C), 2007, http://www.ogf.org/documents/GFD.107.pdf.

[5] Ardagna, D., Pernici, B.: "Adaptive Service Composition in Flexible Processes", *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369-384, 2007.

[6] Baligand, F., Rivierre, N., Ledoux, T. "A Declarative Approach for QoS-Aware Web Service Compositions", *Proc of 5th International Conference on Service-Oriented Computing (ICSOC2007)*, Vienna, Austria, 2007, Springer, pp. 422-428.

[7] Banerjee, D., Dürst, M. J., McKenna, M., Phillips, A., Suzuki, T., Texin, T., Trumble, M., Vine, A., Noji, K., "Web Services Internationalization Usage Scenarios", W3C, 2004, www.w3.org/TR/2004/WD-ws-i18n-scenarios-20040512/.

[8] Berbner, R., Spahn, M., Repp, N., Heckmann, O., Steinmetz, R. "Heuristics for QoS-aware Web Service Composition", *Proc of IEEE International Conference on Web Services (ICWS2006)*, Chicago, Illinois, USA, 2006, IEEE Computer Society, pp. 72-82.

[9] Biron, P. V., Permanente, K., Malhotra, A., "XML Schema Part 2: Datatypes Second Edition", W3C, 2004, http://www.w3.org/TR/xmlschema-2/.

[10] Blau, P.: "Exchange and Power in Social Life", John Wiley & Sons. New York, 1964.

[11] Bonatti, P. A., Festa, P. "On Optimal Service Selection", *Proc of 14th International Conference on World Wide Web (WWW2005)*, Chiba, Japan, 2005, pp. 530-538.

[12] Boutilier, C., Hoos, H. H. "Bidding Languages for Combinatorial Auctions", *Proc of 17th International Joint Conference on Artificial Intelligence (IJCAI2001)*, Seattle, Washington, USA, 2001, pp. 1211-1216.

[13] Brassard, G., Bratley, P.: "Fundamentals of Algorithmics", Prentice-Hall, 1995.

[14] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., Cowan, J., "Extensible Markup Language (XML) 1.1 (Second Edition)", W3C, 2006, http://www.w3.org/TR/xml11.

[15] Canfora, G., Penta, M. D., Esposito, R., Perfetto, F., Villani, M. L. "Service Composition (re)Binding Driven by Application-Specific QoS", *Proc of 4th International Conference on Service-Oriented Computing (ICSOC2006)*, Chicago, IL, USA, 2006, Springer, pp. 141-152.

[16] Cardellini, V., Casalicchio, E., Grassi, V., Presti, F. L. "Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes", *Proc of IEEE International Conference on Web Services (ICWS2007)*, Salt Lake City, Utah, USA, 2007, IEEE Computer Society, pp. 743-750.

[17] Chafle, G., Dasgupta, K., Kumar, A., Mittal, S., Srivastava, B. "Adaptation in Web Service Composition and Execution", *Proc of IEEE International Conference on Web Services*, Chicago, Illinois, USA, 2006, IEEE Computer Society, pp. 549-557.

[18] Chafle, G., Doshi, P., Harney, J., Mittal, S., Srivastava, B. "Improved Adaptation of Web Service Compositions Using Value of Changed Information", *Proc of IEEE International Conference on Web Services (ICWS 2007)*, Salt Lake City, Utah, USA, 2007, IEEE Computer Society, pp. 784-791.

[19] Charfi, A., Khalaf, R., Mukhi, N. "QoS-Aware Web Service Compositions Using Non-intrusive Policy Attachment to BPEL", *Proc of 5th International Conference on Service-Oriented Computing (ICSOC2007)*, Vienna, Austria, 2007, Springer, pp. 582-593.

[20] Chinnici, R., Moreau, J.-J., Ryman, A., Weerawarana, S., "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", W3C, 2007, http://www.w3.org/TR/wsdl20/.

[21] Chou, Y.-l.: "Statistical Analysis: With Business and Economic Applications", Holt, Rinehart and Winston, 1969.

[22] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C, 2001, http://www.w3.org/TR/wsdl.

[23] Clement, L., Hately, A., von Riegen, C., Rogers, T., "UDDI Version 3.0.2", OASIS, 2004, http://www.uddi.org/pubs/uddi_v3.htm.

[24] Colan, M., "Service-Oriented Architecture Expands the Vision of Web Services", 2004, http://www.ibm.com/developerworks/webservices/library/ws-soaintro.html?S_TACT=105AGX04&S_CMP=LP.

[25] Combe, C.: "Introduction to e-Business: Management and Strategy", Butterworth-Heinemann, 2006.

[26] Cornelli, F., Damiani, E., di Vimercati, S. D. C., Paraboschi, S., Samarati, P. "Choosing Reputable Servents in a P2P Network", *Proc of 11th International Conference on World Wide Web (WWW2002)*, Honolulu, Hawaii, USA, 2002, ACM Press, pp. 376-386.

[27] Cramton, P. S., Y., Steinberg, R.: "Combinatorial Auctions", MIT Press. Cambridge, Mass., 2006.

[28] Damiani, E., Vimercati, D. C. D., Paraboschi, S., Samarati, P., Violante, F. "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks", *Proc of 9th ACM Conference on Computer and Communications Security (CCS2002)*, Washington, DC, USA, 2002, ACM Press, pp. 207-216.

[29] Doyle, S. X., Roth, G. T.: "Selling and Sales Management in Action: The Use of Insight Coaching to Improve Relationship Selling", *Journal of Personal Selling & Sales Management*, vol. 12, no. 1, pp. 59-64, 1992.

[30] Evans, M., Hastings, N., Peacock, B.: "Statistical Distributions", Wiley-Interscience, 2000.

[31] Fallside, D. C., Walmsley, P., "XML Schema Part 0: Primer Second Edition", W3C, 2004, http://www.w3.org/TR/xmlschema-0/.

[32] Farrell, J., Lausen, H., "Semantic Annotations for WSDL and XML Schema (SAWSDL)", World Wide Web Consortium (W3C), 2007, http://www.w3.org/TR/sawsdl/.

[33] Fielding, R., Gettys, J., Mogul, J. C., Frystyk, H., Berners-Lee, T. "Hypertext Transfer Protocol -- HTTP/1.1," *RFC 2616*, 1997.

[34] Gefen, D.: "E-Commerce: the Role of Familiarity and Trust", *Omega*, vol. 28, no. 6, pp. 725-737, 2000.

[35] Girish B. Chafle, Sunil Chandra, Vijay Mann, Mangala Gowri Nanda. "Decentralized Orchestration of Composite Web Services", *Proc of 13th International Conference on World Wide Web (WWW2004)*, New York, NY, USA, 2004, ACM, pp. 134-143.

[36] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A., Lafon, Y., "SOAP Version 1.2", W3C, 2007, http://www.w3.org/TR/soap12-part1/.

[37] Haas, H., Brown, A., "Web Services Glossary", W3C Working Group, 2004, http://www.w3.org/TR/ws-gloss/.

[38] Harney, J., Doshi, P. "Adaptive Web Processes Using Value of Changed Information", *Proc of 4th International Conference on Service-Oriented Computing*, Chicago, IL, USA, 2006, Springer, pp. 179-190.

[39] Harney, J., Doshi, P. "Speeding Up Adaptation of Web Service Compositions Using Expiration Times", *Proc of 16th International Conference on World Wide Web*, Banff, Alberta, Canada, 2007, ACM, pp. 1023-1032.

[40] He, Q., Yan, J., Jin, H., Yang, Y. "Adaptation of Web Service Composition Based on Workflow Patterns", *Proc of 6th International Conference on Service-Oriented Computing (ICSOC08)*, Sydney, Australia, 2008, Lecture Notes in Computer Science, pp. 22-37.

[41] He, Q., Yan, J., Kowalczyk, R., Jin, H., Yang, Y. "Lifetime Service Level Agreement Management with Autonomous Agents for Services Provision," *Information Sciences*, 2009, http://dx.doi.org/10.1016/j.ins.2009.01.037.

[42] He, Q., Yan, J., Yang, Y., Kowalczyk, R., Jin, H. "Chord4S: A P2P-based Decentralised Service Discovery Approach", *Proc of IEEE International Conference on Services Computing (SCC2008)*, Honolulu, Hawaii, USA, 2008, IEEE Computer Society, pp. 221-228.

[43] Hoffmann, J., Bertoli, P., Pistore, M. "Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty", *Proc of 22nd AAAI Conference on Artificial Intelligence (AAAI2005)*, Vancouver, British Columbia, Canada, 2007, pp. 1013-1018.

[44] Huhns, M. N., Singh, M. P.: "Service-Oriented Computing: Key Concepts and Principles", *IEEE Internet Computing*, vol. 9, no. 1, pp. 75-81, 2005.

[45] Hwang, S.-Y., Wang, H., Tang, J., Srivastava, J.: "A Probabilistic Approach to Modeling and Estimating the QoS of Web-Services-Based Workflows", *Information Sciences*, vol. 177, no. 23, pp. 5484-5503, 2007.

[46] Jøsang, A., Ismail, R., Boyd, C.: "A Survey of Trust and Reputation Systems for Online Service Provision", *Decision Support Systems*, vol. 43, no. 2, pp. 618-644, 2007.

[47] Jaeger, M. C., Mühl, G., Golze, S. "QoS-Aware Composition of Web Services: A Look at Selection Algorithms", *Proc of IEEE International Conference on Web Services (ICWS2005)*, Orlando, FL, USA, 2005, IEEE Computer Society, pp. 807-808.

[48] Jin, L.-j., Machiraju, V., Sahai, A.: Analysis on Service Level Agreement of Web Services. Technical Report, HP Laboratories, http://www.hpl.hp.co.uk/techreports/2002/HPL-2002-180.pdf, 2002.

[49] Köksalan, M., Zionts, S.: "Multiple Criteria Decision Making in the New Millennium", Springer, 2001.

[50] Küster, U., König-Ries, B., Stern, M., Klein, M. "DIANE: An Integrated Approach to Automated Service Discovery, Matchmaking and Composition", *Proc of 16th International Conference on World Wide Web (WWW2007)*, Banff, Alberta, Canada, 2007, ACM, pp. 1033-1042.

[51] Kamvar, S. D., Schlosser, M. T., Garcia-Molina, H. "The EigenTrust Algorithm for Reputation Management in P2P Networks", *Proc of 12th International World Wide Web Conference (WWW2003)*, Budapest, Hungary, 2003, ACM Press, pp. 640-651.

[52] Keller, A., Ludwig, H.: "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services", *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57-81, March 2003.

[53] Khalaf, R., Mukhi, N., Weerawarana, S. "Service-Oriented Composition in BPEL4WS", *Proc of 12th International Conference on World Wide Web (WWW2003)*, Budapest, Hungary, 2003.

[54] Ko, J. M., Kim, C. O., Kwon, I.-H.: "Quality-of-Service Oriented Web Service Composition Algorithm and Planning Architecture", *Journal of Systems and Software*, vol. 81, no. 11, pp. 2079-2090, 2008.

[55] Kona, S., Bansal, A., Gupta, G. "Automatic Composition of Semantic Web Services", *Proc of International Conference on Web Services (ICWS2007)*, Salt Lake City, Utah, USA, 2007, IEEE Computer Society, pp. 150-158.

[56] Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: "SAWSDL: Semantic Annotations for WSDL and XML Schema", *IEEE Internet Computing*, vol. 11, no. 6, pp. 60-67, 2007.

[57] Kreger, H.: Web Services Conceptual Architecture (WSCA 1.0). IBM, 2001.

[58] Lam, S. K., Riedl, J. "Shilling Recommender Systems for Fun and Profit", *Proc of 13th International Conference on World Wide Web (WWW2004)* New York, NY, USA, 2004, ACM Press, pp. 393-402.

[59] Leff, A., Rayfield, J. T., Dias, D. M.: "Service-Level Agreements and Commercial Grids", *IEEE Internet Computing*, vol. 7, no. 4, pp. 44-50, 2003.

[60] Leymann, F., "Web Services Flow Language (WSFL 1.0)", IBM, 2001, http://xml.coverpages.org/WSFL-Guide-200110.pdf.

[61] Li, Y., Huai, J., Deng, T., Sun, H., Guo, H., Du, Z. "QoS-aware Service Composition in Service Overlay Networks", *Proc of IEEE International Conference on Web Services (ICWS2007)*, Salt Lake City, Utah, USA, 2007, IEEE Computer Society, pp. 703-710.

[62] Litke, A., Konstanteli, K., Andronikou, V., Chatzis, S., Varvarigou, T.: "Managing Service Level Agreement Contracts in OGSA-based Grids", *Future Generation Computer Systems*, vol. 24, no. 4, pp. 245-258 2008.

[63] Liu, Y., Tham, C.-K., Jiang, Y.: "Conformance Analysis in Networks with Service Level Agreements", *Computer Networks*, vol. 47, no. 6, pp. 885-906, 2005.

[64] Liu, Z., Squillante, M. S., Wolf, J. L. "On Maximizing Service-Level-Agreement Profits", *Proc of 3rd ACM Conference on Electronic Commerce (EC2001)*, Tampa, Florida, USA, 2001, ACM, pp. 213-223.

[65] Ludwig, H., Dan, A., Kearney, R. "Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements", *Proc of 2nd International Conference on Service Oriented Computing (ICSOC2004)*, New York, USA, 2004, pp. 65-74.

[66] Ludwig, H., Keller, A., Dan, A., King, R. P., Franck, R., "Web Service Level Agreement (WSLA) language specification, Version 1.0", IBM, 2003, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[67] Luhmann, N.: "Trust and Power", Wiley, 1979.

[68] Mani, A., Nagarajan, A., "Understanding Quality of Service for Web services", IBM, 2002, http://www.ibm.com/developerworks/library/ws-quality.html.

[69] Marsh, S. P.: "Formalising Trust as a Computational Concept", PhD Thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[70] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D. M., Sheila, Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K., "OWL-S: Semantic Markup for Web Services", World Wide Web Consortium (W3C), 2004, http://www.w3.org/Submission/OWL-S/.

[71] McKnight, D. H., Choudhury, V., Kacmar, C. "Trust in E-Commerce Vendors: A Two-Stage Model", *Proc of 21st International Conference on Information Systems (ICIS2000)*, Brisbane, Queensland, Australia 2000, ACM Press, pp. 532-536.

[72] Milanovic, N., Malek, M.: "Current Solutions for Web Service Composition", *IEEE Internet Computing*, vol. 8, no. 6, pp. 51-59, 2004.

[73] Milgrom, P.: "Putting Auction Theory to Work", Cambridge University Press, 2004.

[74] Milgrom, P.: "The Structure of Information in Competitive Bidding", PhD Thesis, School of Humanities and Sciences, Stanford University, 1979.

[75] Milner, R.: "A Calculus of Communicating Systems", Springer, 1982.

[76] Mitchell, W.: "Dual Clocks: Entry Order Influences on Incumbent and Newcomer Market Share and Survival When Specialized Assets Retain Their Value", *Strategic Management Journal*, vol. 12, no. 2, pp. 85-100, 1991.

[77] Mohabey, M., Narahari, Y., Mallick, S., Suresh, P., Subrahmanya, S. V. "A Combinatorial Procurement Auction for QoS-Aware Web Services Composition", *Proc of 3rd IEEE Conference on Automation Science and Engineering (CASE2007)*, Scottsdale, AZ, USA, 2007, IEEE Computer Society, pp. 716-721.

[78] Nanda, M. G., Chandra, S., Sarkar, V. "Decentralizing Execution of Composite Web Services", *Proc of 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA2004)*, Vancouver, BC, Canada, 2004, ACM pp. 170-187.

[79] Narendra, N. C., Ponnalagu, K., Krishnamurthy, J., Ramkumar, R. "Run-Time Adaptation of Non-functional Properties of Composite Web Services Using Aspect-Oriented Programming", *Proc of 5th International Conference on Service-Oriented Computing (ICSOC2007)*, Vienna, Austria, 2007, Springer, pp. 546-557.

[80] Nielsen, H., Leach, P., Lawrence, S. "An HTTP Extension Framework," *RFC 2774*, 2000.

[81] OASIS, "OASIS: Advancing Open Standards for the Information Society", http://www.oasis-open.org/home/index.php.

[82] Papazoglou, M. P. "Service-Oriented Computing: Concepts, Characteristics and Directions", *Proc of 4th International Conference on Web Information Systems Engineering (WISE2003)*, Rome, Italy, 2003, IEEE Computer Society, pp. 3-12.

[83] Papazoglou, M. P., Georgakopoulos, D.: "Service-Oriented Computing", *Communications of the ACM*, vol. 46, no. 10, pp. 25-28, 2003.

[84] Parkes, D. C., Kalagnanam, J.: "Models for Iterative Multiattribute Procurement Auctions", *Management Science*, vol. 51, no. 3, pp. 435-451, 2005.

[85] Pistore, M., Marconi, A., Bertoli, P., Traverso, P. "Automated Composition of Web Services by Planning at the Knowledge Level", *Proc of 19th International Joint Conference on Artificial Intelligence (IJCAI2005)*, Edinburgh, Scotland, UK, 2005, pp. 1252-1259.

[86] Ragone, A., Noia, T. D., Sciascio, E. D., Donini, F. M., Colucci, S., Colasuonno, F.: "Fully Automated Web Services Discovery and Composition through Concept Covering and Concept Abduction", *International Journal of Web Services Research*, vol. 4, no. 3, pp. 85-112, 2007.

[87] Ralph L. Keeney, H. R.: "Decisions with Multiple Objectives: Preferences and Value Trade-Offs", Cambridge University Press, 1993.

[88] Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: "Reputation Systems", *Communications of the ACM*, vol. 43, no. 12, pp. 45-48, 2000.

[89] Rothkopf, M. H., Pekec, A., Harstad, R. M.: "Computationally Manageable Combinatorial Auctions", *Management Science*, vol. 44, no. 8, pp. 1131-1147, 1998.

[90] Sandholm, T.: "Algorithm for Optimal Winner Determination in Combinatorial Auctions", *Artificial Intelligence*, vol. 135, no. 1-2, pp. 1-54, February 2002.

[91] Skiena, S. S.: "The Algorithm Design Manual", Springer, 1997.

[92] Srivatsa, M., Xiong, L., Liu, L. "TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks", *Proc of 14th International Conference on World Wide Web (WWW2005)* Chiba, Japan, 2005, ACM Press, pp. 422-431.

[93] Swan, J. E., Nolan, J. J.: "Gaining Customer Trust: A Conceptual Guide for the Salesperson", *Journal of Personal Selling & Sales Management*, vol. 5, no. 2, pp. 39-48, 1985.

[94] Thompson, H. S., Beech, D., Maloney, M., Mendelsohn, N., "XML Schema Part 1: Structures Second Edition", W3C, 2004, http://www.w3.org/TR/xmlschema-1/.

[95] van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., P., B. A.: "Workflow Patterns", *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5-51, 2003.

[96] Vedamuthu, A. S., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boube, T., Ü., Y.: Web Services Policy framework (WS-Policy), Version 1.5. World Wide Web Consortium, http://www.w3.org/TR/ws-policy/, 2007.

[97] Vedamuthu, A. S., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T., Yalçinalp, Ü., "Web Services Policy 1.5 - Attachment", W3C, 2007, http://www.w3.org/TR/ws-policy-attach/.

[98] Verma, K., Doshi, P., Gomadam, K., Miller, J. A., Sheth, A. P. "Optimal Adaptation in Web Processes with Coordination Constraints", *Proc of IEEE International Conference on Web Services (ICWS2006)*, Chicago, Illinois, USA, 2006, IEEE Computer Society, pp. 257-264.

[99] Wang, Y., Lin, K.-J.: "Reputation-Oriented Trustworthy Computing in E-Commerce Environments", *IEEE Internet Computing*, vol. 12, no. 4, pp. 55-59, 2008.

[100] Wang, Y., Varadharajan, V. "A Time-Based Peer Trust Evaluation in P2P E-commerce Environments", *Proc of International Conference on Web Information Systems Engineering (WISE2004)*, Brisbane, Australia, 2004, pp. 730-735.

[101] Wang, Y., Wong, D. S., Lin, K.-J., Varadharajan, V.: "Evaluating Transaction Trust and Risk Levels in Peer-to-Peer E-Commerce Environments ", *Information Systems and E-Business Management*, vol. 6, no. 1, pp. 25-48, 2008.

[102] Wilson, R.: "A Bidding Model of Perfect Competition", *Review of Economic Studies*, vol. 44, no. 3, pp. 511-518, 1977.

[103] Wolter, R., "XML Web Services Basics", Microsoft, 2001, http://msdn.microsoft.com/en-us/library/ms996507.aspx.

[104] Xiong, L., Liu, L.: "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities", *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843--857, 2004.

[105] Yu, B., Singh, M. P., Sycara, K. "Developing Trust in Large-Scale Peer-to-Peer Systems", *Proc of 1st IEEE Symposium on Multi-Agent Security and Survivability (MAS&S2004)*, Philadelphia, PA, USA, 2004, IEEE CS Press, pp. 1-10.

[106] Yu, T. Z., Yue, Lin, K.-J.: "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints", *ACM Transactions on the Web*, vol. 1, no. 1, pp. 2007.

[107] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q. Z. "Quality Driven Web Services Composition", *Proc of 12th International Conference on World Wide Web (WWW2003)*, Budapest, Hungary, 2003, pp. 411-421.

[108] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., Chang, H.: "QoS-Aware Middleware for Web Services Composition", *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311-327, May 2004.