University of Windsor
Scholarship at UWindsor

Electronic Theses and Dissertations | Theses, Dissertations, and Major Papers

2007

An adaptive approach for QoS-aware Web service composition

Zhiyang Wang
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Wang, Zhiyang, "An adaptive approach for QoS-aware Web service composition" (2007). *Electronic Theses and Dissertations*. 4620.
https://scholar.uwindsor.ca/etd/4620

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

An Adaptive Approach for QoS-aware Web Service Composition

by

Zhiyang Wang

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2007

© 2007 Zhiyang Wang

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-34938-0
Our file Notre référence
ISBN: 978-0-494-34938-0

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

Web service composition is the process of integrating existing web services. It is a prospective method to build an application system. Current approaches, however, only take service function aspect into consideration. With the rapid growth of web service applications and the abundance of service providers, the consumer is facing the inevitability of selecting the maximum satisfied service providers due to the dynamic nature of web services. This requirement brings us some research challenges including a web service quality model, to design a web service framework able to monitor the service's real time quality. A further challenge is to find an algorithm that can handle extensible service quality parameters and has good performance to solve NP-hard web services global selection problem. In this thesis, we propose a web service framework, using an extensible service quality model. A Cultural Algorithm is adopted to accelerate service global selection. We also provide experimental results comparing between Cultural Algorithm with Genetic Algorithm and Random service selection.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Dedication

To my loving and caring wife who has always patiently supported me.

Thank you for taking care of everything during the program.

This thesis would not be possible without you!

And, of course, to my dearest daughter who brighten my every day.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgement

I would like express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank my supervisor, Dr. Ziad Kobti, for the energy and enthusiasm he invested in this research. His guidance has been essential in the success of this work. I would also like to thank my thesis committee members, Dr. Lu, Dr. Tepe and Dr. Yuan, who have been all generous and patient. Their confidence in my abilities has been unwavering, and has helped to make this thesis a solid work.

I would like to thank my friends Wang Yan and Li Nan's concerns on my daily life and help.

Especially, I am deeply indebted to my wife for her patient love, support and encouragement. I also want to thank my adorable daughter DouDou for her moral support and bringing me happiness.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Content

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1 Introduction

According to the World Wide Web consortium(W3C) "A Web service is a software system identified by a Uniform Resource Identifier (URI), whose public interfaces and bindings are defined and described using extensible mark-up language (XML). Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definitions, using XML based messages conveyed by internet protocols." (W3C 2003).

A web service is conveyed by Simple Object Access Protocol (SOAP) which is based on Hyper Text Transfer Protocol (HTTP). It allows web services to pass through any firewall. Therefore, a web service pulls the web functionality from document oriented to application oriented and provides a new model of distributed computing. Tsalgatidou (2002) states that a web service has the following potential advantages: Reusability, web services can be reused to build value-added service or application; Platform and language independent means any web service can interact with other web services. This is achieved through an XML-based interface definition language and a protocol of collaboration and negotiation; Just-in-time integration, which means web services systems promote significant decoupling and just-in-time integration of new applications and services, as they are based on the notion of building applications by discovering and orchestrating available services. The integration system has self-configuring, adaptive and robust features.

Due to the above benefits, a web service has become an emerging and promising technology to design and build complex enterprise business applications. It has received great attention from industry and academia. Milanovic (2004) predicts that the Internet will become a global common platform where organizations and individuals communicate with each other to carry out various commercial activities and to provide value-added services. In the future, business service developers will simply assemble a set of appropriate web services to implement business tasks. Business applications will be no longer written manually.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Where Web Service Composition (WSC) is concerned, current approaches only take the service function aspect into consideration. The purpose of service discovery is to find the services whose function matches customers' requirements. The major concern of current WSC is the service functional aspect.

With the rapid growth of web service applications and the abundance of service providers, the consumer is facing the inevitability of selecting the "maximum satisfied" service provider given t he dynamic nature of web services. In such a scenario the Quality of Service (QoS) becomes the benchmark to differentiate providers (Kalepu 2004). Yu (2005) presents several major issues which must be considered when integrating distributed quality-aware web services into a business execution process: (1) The set of services capable of providing the same functionality (service community) may be constantly changing; (2) There may be different ways to construct an execution process; (3) An execution process needs to accommodate system exceptions, such as service failure (adaptation ability); (4) The performance of an execution process, which is measured in terms of its global *QoS*, must satisfy the user's requirements.

Due to different concerns, current WSC and QoS-aware WSC have different aims. A current WSC approach is to find a composite service whose function satisfies customer's functional requirements. A QoS-aware WSC approach considers both functional and non-functional customer's requirements.

Before going to QoS-aware WSC, let us take a look at the current framework of service composition which does not take the quality issue into account. This framework only considers the service's functional aspects. It consists of three roles: the service provider, the service registry and the service consumer. Figure 1-1 shows a graphical representation of the current web service working framework.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

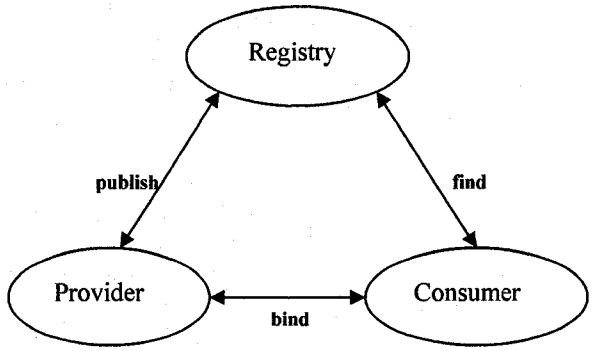


Figure 1- 1 current web service working framework

The service provider publishes the web service by extensible markup language (XML) standard in a central Service Registry. The service registry stores the information about the service provider, such as address and contact of the providing company, and technical details about the service. The Service Consumer retrieves the information from the registry and uses the service description obtained to bind to and invoke the web service. The appropriate methods are depicted in Figure 1 by the keywords 'publish', 'bind' and 'find'. Web services architecture is loosely coupled, and service oriented. The Web Service Description Language (WSDL) uses the XML format to describe the methods provided by a web service. The Universal Description Discovery and Integration standard (UDDI) suggests methods to publish details about a service provider, the services that are stored and the opportunity for service consumers to find service providers and web service details. The Simple Object Access Protocol (SOAP) is used for XML formatted information exchange among the entities involved in the web service model.

Two categories of service composition strategies are mainly applied at present, which are static and dynamic strategies. Milanovic (2004) and Pistore (2004) show us the most often used approach in each strategy. Business Process Execution Language for Web Services (BPEL4WS) is a static composition approach since the composition process takes place during design time. The dynamic service composition solution is a semantic web services strategy. The main idea of a semantic web service strategy is that semantic markup on the web service makes web service automatic discovery.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Since web services are created on the fly, current approach is apparently unsuitable to an unpredictable environment. The current solution cannot guarantee a process executable at run time. Moreover, the current existing framework does not support for service's non-functional quality monitoring, such as performance, scalability, reliability, availability, flexibility, stability, cost, and completeness. For example, there is no mechanism available in the existing web services architecture to determine run-time performance characteristics of a web service, and select a service that best meets the performance requirements of the consumer (Singhera 2006). Therefore, current WSC strategies are hardly employed in real business applications.

According to the concerns of QoS-aware WSC, a QoS-aware WSC framework must support the following functionalities besides those in the current framework: (1) service discovery from both function and non-function aspects; (2) QoS computing; (3) service composition; (4) execution plan adaptation.

Zeng (2004) proposes a middleware supporting service discovery, QoS computing, and service composition. Singhera (2006) devises a so-called extended web service framework to meet non-functional requirements. Yu (2005) uses a broker-based framework for QoS-aware web service composition. Xia (2006) uses an agent implementing probe-based QoS-aware WSC, which support all the functions above. However, they are either lacking in flexibility or only partially implement the functionality mentioned above.

A comprehensive QoS-aware WSC system should have service discovery, QoS computing, service composition, and adaptation functionalities. In current QoS-aware WSC research, the approaches of web service function discovery mostly adopt the current discovery methods like WSDL matching, semantic web service, or based on Database Query. QoS estimation based on historical logs is employed in QoS computing. Linear and Genetic Algorithm integration are the most popular approaches applied in service composition. Re-planning trigger is used in adaptation. More details are found in the literature review section.

QoS-aware service composition involves three major problems: first is the QoS model which means how to choose appropriate quality attributes to exactly describe

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the non-functional feature of a service. Second is WS framework. QoS-aware WSC requires its framework to be able to monitor the quality attributes of each web service in real time. The third problem is the performance of the WSC algorithm. QoS-aware service composition is a late-binding process. From the service composition to invocation of execution process and re-planning must be performed very quickly, long delay may be unacceptable. Performing service composition long before execution may lead to unattended results. For example, some services may be unavailable during execution. QoS-aware WSC solution without time consideration is unacceptable. In current research, many algorithms have been investigated to implement service composition such as integer programming (Zeng 2003), Pinsinger's Algorithm, Reduction Algorithm (Cardoso 2004), and Genetic Algorithm (Canfora 2006) etc. Unfortunately, few of them have considered the performance of the algorithms. For real time, failure to compose and deliver the aggregated service on time can cause big loss in business. Furthermore, in order to keep the consistence of a delivered service, re-planning should be finished and deployed in a timely manner.

In this thesis, we propose a scalable service QoS model and a flexible framework to facilitate dynamic service composition and adaptation of QoS-aware web services with global QoS constraints. We use Cultural Algorithm to accelerate convergence so that the computation duration can be greatly shortened.

1.1 Contributions

We extend the functionality of current web service framework by adding a proxy. It integrates the functionality of modeling web service quality and dynamically composing web service. It overcomes the shortages of previous research that tries to define a complete or formal web service model and uses a linear algorithm or genetic algorithm to compose web services. The proxy looks web service quality model and service composition algorithm as a whole. It also takes the algorithm performance into consideration. The proxy automatically maintains the quality attributes of concrete services by monitoring their run time quality attributes. It adopts Cultural

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Algorithm to enhance the composition speed and support any customized quality parameters. The proxy therefore has excellent scalability and good performance.

1.2 Organization of the thesis

The rest of this thesis is organized as follows. Chapter 2 introduces the background knowledge related to our approach. Chapter 3 gives literature review. Some milestone researches are shown from three aspects service quality model, WS framework and algorithms of web services global selection. Chapter 4 gives the details of our proposed approach. It includes the proposed framework, quality of service model, and the functionality and design idea of each module in the proxy. Chapter 5 explains the implementation process and shows the difference among different strategies used in the proxy. Finally, chapter 6 concludes our contributions, advantages of our approach and shortages for future work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2 Preliminary

2.1 Overview of Web Services

According to the Organization for the Advancement of Structured Information Standards (OASIS), a Web Service is a software component that is described via WSDL and is capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP (OASIS 2001).

Web services presents a standardized way of integrating web-based applications using the XML, SOAP, WSDL and UDDI standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI registry is applied for listing what services are available. Used primarily as means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's systems behind the firewall.

A web service brings a lot of potential advantages (Tsalgatidou 2002): Reusability, web services can be reused to build value-added service or application. Platform and language independence, any web service can interact with other web services. This is achieved through an XML-based interface definition language and a protocol of collaboration and negotiation. By limiting what is absolutely required for interoperability, collaborating Web Services can be truly platform and language independent. Just-in-time integration, web services systems promote significant decoupling and just-in-time integration of new applications and services, as they are based on the notion of building applications by discovering and orchestrating network-available services. This in turn yields systems that are self-configuring, adaptive and robust with fewer single points of failure.

2.2 Web Service Composition

From Srivastava (2003), Milanovic (2004) and Dustdar (2005), when the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

implementation of a web service's application involves the invocation of other web services, it is necessary to combine the functionality of several web services. This is called a composite service. The process of developing a composite service is called service composition. Service composition can be either performed by composing elementary or composite services. Composite services in turn are recursively defined as an aggregation of elementary and composite services.

In web service composition, services are executed in different platforms distributed by firewalls and other trust barriers. Srivastava (2003), Milanovic (2004) and Granell (2005) state that a QoS-aware composition mechanism of web services must satisfy several requirements: connectivity, non-functional quality-of-service properties, correctness, and scalability. Connectivity means each composition approach must guarantee connectivity because only with reliable connectivity, services can be composed and reason about the input and output messages. Secondly, web services are based on message passing, service providers must also address nonfunctional QoS properties which are the foundation of web service selection. Next, composition correctness requires verification of the composed service's properties, such as function, security or dependability. Finally, because complex business processes are likely to involve multiple services in an invocation workflow, composition frameworks must scale with the number of composed services.

Many approaches within different categories are applied to compose web services. In (Milanovic 2004), it simply grouped the approaches of web service composition into two types. On type, it is based on a number of XML-based standards to formalize the specification of web services. This approach is primarily syntactical: Web service interfaces are like remote procedure calls and the interaction protocols are manually written. The other type is based on the Semantic Web reasoning about web resources by explicitly declaring their preconditions and effects in ontology to dynamically compose web services. Dustdar (2005) presents two categories which are static and dynamic strategies. The difference between the two strategies is the time concern when web services are composed. They are equivalent to design-time and run-time composition.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3 Quality of Service (QoS)

The international quality standard ISO 8402 within ISO 9000 (ISO9000 2002) describes quality as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs." Quality of web service is a set of non-functional attributes which reflects the quality offered by a web service.

In a web service, QoS requirement mainly refers to the non-functional quality of the web service. Quality of a web service is not only associated with its own services but also relates to the network environment they are located. In fact, at each layer of a web service stack there are different corresponding quality attributes. Figure 2-1 shows the relation in a QoS stack.

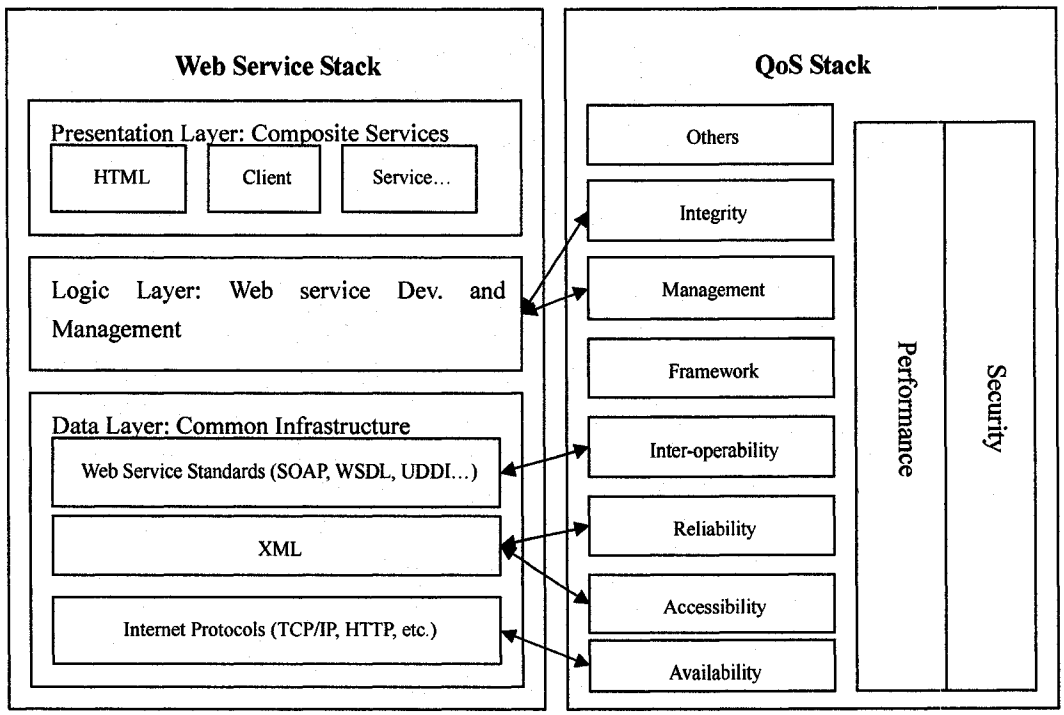


Figure 2- 1 QoS stack

QoS is a combination of quality properties of a service (Menasce 2002). Araban (2004) expresses that the main classification of QoS attributes are internal attributes, which are independent of the service environment, and external attributes which are dependent on the service environment. These attributes mainly include performance, reliability, integrity, accessibility, availability, interoperability, and security.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Performance is measured in terms of throughput, latency, execution time, and transaction time. Reliability is the overall measure of a Web Service to maintain its service quality. The number of failures per day, week, month, or year represents an overall measure of reliability for a web service. Reliability also refers to the assured and ordered delivery for messages being sent and received by service requestors and service providers. Integrity is the degree to which a system or component prevents unauthorized access to, or modification of, computer programs or data. Accessibility defines whether the Web Service is capable of serving the client's request. Availability is the percentage of time that a service is available. The fundamental goal of interoperability in Web Services is to cross the lines between the developing environments used to implement services so that developers using those services do not have to think about which programming language or operating system the services are hosted on. Security properties include the existence and type of authentication mechanisms the service offers.

The QoS measure is observed by web service users. These users are not human beings but programs that send requests for services to web service providers. QoS issues in web services can be evaluated from the perspective of the providers of web services and from the perspective of the users of these services.

2.4 QoS model

Quality of service is a big umbrella which covers a variety of quality attributes. QoS attributes are in fact defined by a subjective notion. The meaning of each attribute is different due to end-user factors, contextual circumstances as well as the perspective of interest. QoS model has no standard feature which means we do not have a formal or complete quality model to specify web service quality. Therefore, each provider has to define its web service quality model before delivering its quality aware services.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Araban (2004) states the two domain classifications of QoS attributes are internal attributes, which are independent of the service environment, and external attributes which are dependent on the service environment.

Another popular model is expressed in (Ran 2003). They classify the QoS attributes as: Runtime Related QoS, Transaction Support Related QoS, Configuration Management and Cost Related QoS and Security Related QoS. The author further explains Runtime Related QoS includes Scalability, Capacity, Performance (response time, latency, and throughput), Reliability, Availability, Robustness/Flexibility, Exception Handling, and Accuracy. Transaction QoS is related to transaction Integrity. Configuration Management and Cost Related QoS include regulatory, support standard, stability/change cycle, cost, and completeness. Security Related QoS is composed of Authentication, Authorization, Confidentiality, Accountability, Traceability and Auditability, Data encryption, and Non-Repudiation.

According to Cardoso (2004), quality of service can be characterized by various dimensions. Web service composition of a business system is targeted two distinct areas, operations management of organization and quality of service of software system. On the organizational side, the result indicates that the success of a company is related to the capability to compete with other organizations. It is based upon three essential pillars: time, cost, and quality. On the software system side, it presents a set of practical dimensions for distributed object systems' reliability and performance, which include TTR (time to repair), TTF (time to failure), and availability.

In the thesis, in order to associate our approach with real business system, we have constructed a four-dimension QoS model which consists of response time, service price, reliability and availability.

2.5 Genetic Algorithm

From GA (2002) and Goldberg (1989), GA originated with an idea, born over 30 years ago, of applying the biological principle of evolution to artificial systems. Genetic Algorithm (GA) is the search algorithm that works via the process of natural

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

selection. It begins with a sample set of potential solutions which then evolves toward a set of more optimal solutions. Within the sample set, solutions that are poor tend to die out while better solutions mate and propagate their advantageous traits, thus introducing more solutions into the set that boast greater potential (the total set size remains constant; for each new solution added, an old one is removed). A little random mutation helps guarantee that a set won't stagnate and simply fill up with numerous copies of the same solution.

In general, genetic algorithm tends to work better than traditional optimization algorithms because it is less likely to be led astray by local optima. This is because it does not make use of single-point transition rules to move from one single instance in the solution space to another. Instead, GA takes advantage of an entire set of solutions spread throughout the solution space, all of which are experimenting upon many potential optima.

The basic operations of the genetic algorithm are simple and straight-forward. They have three steps. Reproduction: The act of making a copy of a potential solution. Crossover: The act of swapping gene values between two potential solutions, simulating the "mating" of the two solutions. Mutation: The act of randomly altering the value of a gene in a potential solution.

Roughly speaking, a GA is an iterative procedure that searches for the best solution of a given problem among a constant-size population, represented by a finite string of symbols, named the *genome*. The search is made starting from an initial population of individuals, often randomly generated. At each evolutionary step, individuals are evaluated using a *fitness function*. High–fitness individuals will have the highest probability to reproduce.

The evolution (i.e., the generation of a new population) is made by means of two operators: the *crossover operator* and the *mutation operator*. The crossover operator takes two individuals (the *parents*) of the old generation and exchanges parts of their genomes, producing one or more new individuals (the *offspring*). The mutation operator has been introduced to prevent convergence to local optima, in that it randomly modifies an individual's genome (e.g., by flipping some of its bits, if the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

genome is represented by a bit string). Further details on GA can be found, for example, in (Goldberg 1989).

2.6 Cultural Algorithm

From (Reynolds 2004), Cultural Algorithm has two basic components: Population Space and Belief Space. First, individuals in the Population Space are evaluated with a performance function *obj()*. An acceptance function *accept()* will then determine which individuals are to impact the Belief Space. Experiences of those chosen elites will be used to update the knowledge / beliefs of the Belief Space via function *update()*, which represents the evolution of beliefs. Next, the beliefs are used to influence the evolution of the population. New individuals are *generated* under the *influence* of the beliefs, and from then, together with old individuals, individuals are *selected* and form a new generation of population. The two feedback paths of information, one through the *accept()* and *influence()* functions, and the other through individual experience and the *obj()* function create a system of dual inheritance of both population and belief. The population component and the belief space interact with and support each other, in a manner analogous to the evolution of human culture. Figure 3 describes such a framework.

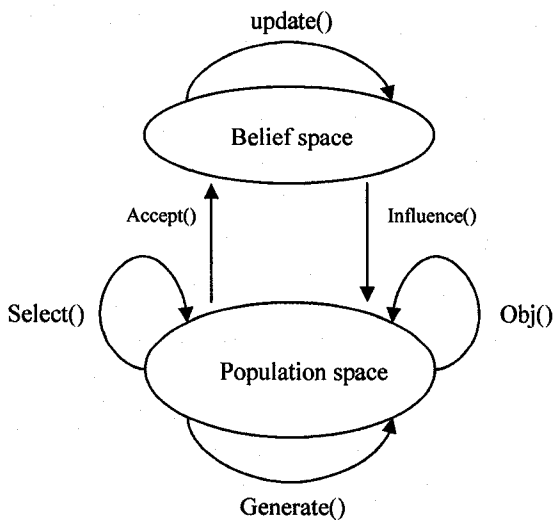


Figure 2- 2 cultural algorithm (Reynolds 2004)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3 Literature Review

In order to implement a business function by integrating existing web services, we should carefully consider the management of services relationship, service quality model, WS framework and a strategy to select services in global. The following papers from above aspects introduce their solutions.

3.1 Workflow in WSC

From Gouscos (2003) and Singhera (2006), future business systems require a seamless integration of many business processes, applications, intelligence, and web services over the Internet. Delivering services to meet user needs is a significant and critical challenge because of the dynamic and unpredictable nature of business applications and Internet traffic. Business applications with very different characteristics and requirements compete for resources used to provide web services. Without a careful management of service quality, critical business applications may suffer detrimental performance degradation, and result in functional failures and/or financial losses.

The area of Quality of Service (QoS) management covers a wide range of issues to match the needs of service requesters with those of the service providers. QoS has been a major area of interest in communication networks, real-time computing, and multimedia systems. For web service, QoS guarantee and enhancement have started to receive great attention.

Workflow technology has been used since a decade and has been proved successful in automating many complex business processes. In addition, a significant number of works have been done to deal with different aspects of workflow technology. Current research on web service workflow-based composition has obvious advantages including scalability, heterogeneity, reuse and maintenance of services (Patel 2003). The major issues in service composition workflow are service discovery, service binding, service QoS contracts and service composition. Service

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

workflow aids such as Web Service workFlow Language (WSFL) overcome the limitations of traditional workflow tools which manually specify the composition of programs to perform some tasks. Other industrial initiatives such as BPEL4WS, XLANG concentrate on service representation issues to tackle the problem of service contracts, compositions and agreements.

The process-based composition of web services is ,in particular, gaining a considerable momentum as an approach for the effective integration of distributed, heterogeneous, and autonomous applications in (Zeng 2003), (Cardoso 2004) and (Canfora 2006). In process-based approach, applications are encapsulated as web services and the logic of their interactions is expressed as a process model. In this thesis, the BEPL-alike workflow is employed similar to (Zeng 2003) and (Canfora 2004).

3.2 QoS Model and QoS Monitoring

QoS model is a set of non-functional attributes which reflect the quality offered by a web service. The main problems of QoS model have quality parameter selection, quality parameter specification, and run time quality value monitoring. Namely, the major problems in QoS model are what quality parameters can be appropriately used to describe a service quality, how to define each quality attribute and how to monitor the quality values in real time. The following papers present their solutions on these issues. Since modeling web service quality is complicated, Gouscos (2003) recommends "a simple way to go, when talking about quality of web services, is to identify some facets whose meaning is intuitive and whose importance is also recognized in the literature. This line of thought, although not resulting in a formal and complete model of quality, at least offers an appropriate basis for modeling work that can be later on extended and enriched with additional features."

Gouscos (2003) presents a simple approach to model and represent QoS of web service, such as, availability, accessibility, integrity, performance, reliability and provision price, in lower-level WSDL specification. In its conceptual model, it uses

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

deterministic or fuzzy two different ways to present quality attributes of a web service quality attributes. At last, it introduces a trusted third party offered "web service management information broker" to refresh run-time quality attributes. The problem of their approaches is the QoS definition is tied to the individual operation (local binding), rather than the service as a whole. Furthermore, runtime QoS issue is not addressed.

Kalepu (2004) states most QoS descriptions are subjective notion. The author recommends mapping the objective system quality to the users' subjective perception of quality. The author, therefore, introduces a new web service quality attribute termed objective verity and proposes a framework to quantify it.

Lau (2001) explains the importance of monitoring and refreshing QoS information at run-time. He proposes the notion of a "QoS broker service", which would be responsible for publishing and disseminating QoS data based on historical as well as real-time QoS measurements of web services.

Jin (2002) presents an approach to manage web service quality in the context of Service Level Agreements (SLAs). The approach directly uses UDDI registry as the storage for both service description and service quality attributes. In order to observe the real-time quality values, the author proposes that the registered web services periodically refresh their values at UDDI. The major problem of this approach is that client applications have always access to up-to-date quality values. It brings a big security problem to UDDI registry.

Tian (2003) uses XML schema defining QoS that both service consumers and providers apply to specify the agreed QoS parameters. The approach allows for the web service dynamic selections based on various QoS requirements. On the negative side, the life-cycle of agreements is not taken into account, and it is not possible to define expiration for a negotiation.

Cardoso (2004) explicates why and how to choose quality attributes to mostly describe a web service. The author recommends using time, cost, reliability and availability to define a service quality. The author also introduces quality monitor and trigger to fresh certain service quality values and re-planning a business process.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Since quality description of web service does not have a formal or complete model, the scalability of QoS model is significant important. Unfortunately, few of above paper mentioned it. None of them mentioned how to handle a scalable QoS model in their WSC process.

3.3 Framework of Web Service

Singhera (2006) proposes an extended web services framework that enables to deal with a collection of functional and non-functional service characteristics at run time, and uses the collected data for service discovery, binding, and execution. It extends the current web service framework by a monitor agent which is responsible for collecting services non-functional attributes. The author presents an approach of using a collection of historical non-functional data to implement local service selection. It enhances the current approach using static non-function attribute to choose services.

Penta (2006) describes WS Binder, a framework for services dynamic binding. The framework is implemented on top of a BPEL technology and supports pre-execution binding (able to satisfy global composition constraints), run-time binding of single service invocations and runtime re-binding of the whole workflow.

However, the authors of above papers do not mention if their framework can handle different no-standard QoS model.

3.4 Web Service Global Selection Algorithm

Zeng (2003) proposes a global linear planning approach to select component services during running time rather than design time. It uses state-charts to model the process of composing web services. It transfers the WSC plan to a path plan problem. In web service quality model, it clearly explains the meaning of each quality attribute. However, it does not give an approach to compute the value of each quality attribute.

Canfora (2006) presents the problems of linear algorithm to implement web service dynamic composition. It proposes an approach which is more scalable and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

more suitable to handle generic QoS attributes, Genetic Algorithm. It also gives the comparison between linear algorithm and genetic algorithm.

None of the above paper, however, mentions the performance of their algorithm or the method to enhance their algorithm performance.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

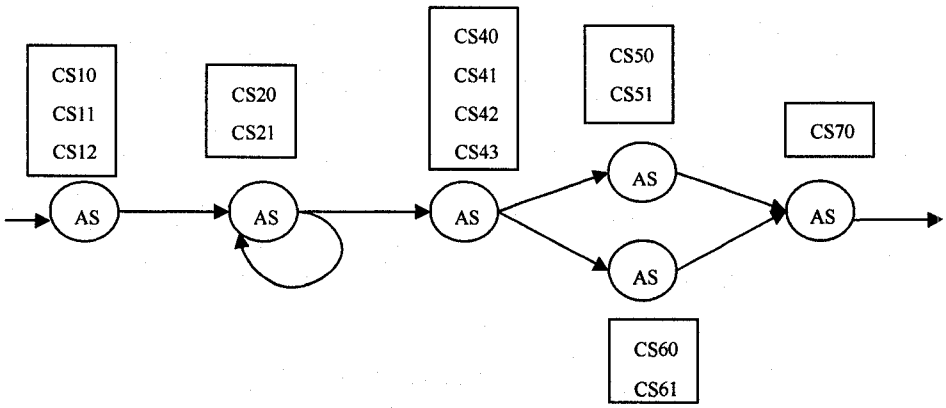
4 Proposed Method

In this section, the approaches from the client requirement description to the combination of concrete services are given. The terms and techniques involved are explained. The framework and its components are introduced. We also present functionality and implementation details of each module in proxy entity at the final.

4.1 Service Composition Workflow

In a service composition workflow, function modules (e.g. application) are encapsulated as abstract web services. The logic relation is represented as a process model. In the other words, the application requirement is described as a workflow containing several abstract services. The abstract services are combined as sequence, branch or loop. Here, an abstract service represents as functional module and is associated with a web service community which contains several concrete web services with the same functionality. The process of selecting a concrete service from a web service community for an abstract service by QoS attributes is called local selection. Obviously, a task presented by the service composition can be solved by a significant numbers of combinations. The process of selection from the numerous combinations according to the global QoS constraints is called global selection. The global selection problem is a NP-hard problem (Canfora 2006). Figure 4-1 shows an example of service composition workflow.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



CS: concrete service AS: Abstract Service
Local binding: selecting a concrete service for a abstract service
Global binding: combination of local binding

Figure 4- 1 Example of web service composition workflow (Canfora 2006)

In service composition workflow, there are four different models that individual services use to integrate and build a process. The four basic models are sequential, parallel, conditional, and loop, as shown in figure 4-2.

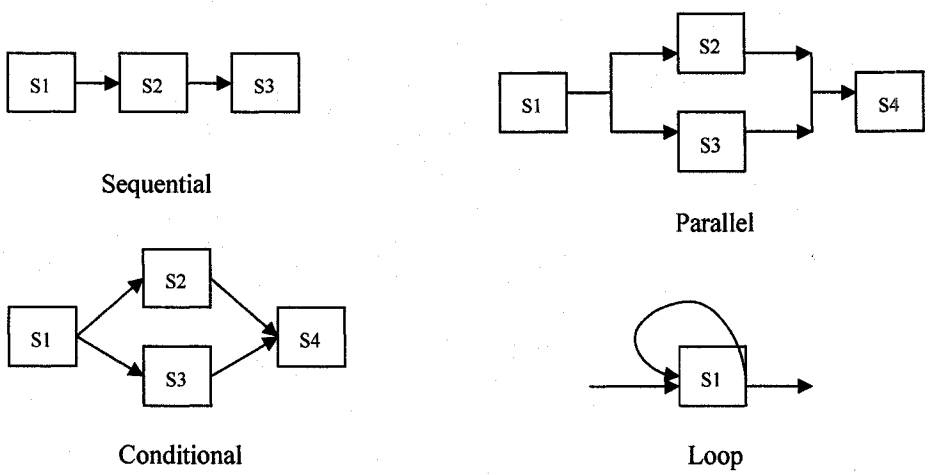


Figure 4- 2 service composition model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2 Proposed QoS Model of Component Services

Since the meaning of QoS attributes is different by a variety of end-user factors, contextual circumstances as well as the perspective of interest, each provider must unambiguously define its QoS model before delivering its QoS aware service.

In the thesis, we use four typical quality attributes execution cost, response time, reliability and availability to model the extendable quality of web services (the reason has been mentioned in preliminary section). The brief explanation of each attribute is as follows.

Execution cost: The execution cost of an operation of a service is the amount of money that a service requester has to pay for executing the operation. Web service providers either directly advertise the execution cost of their operations, or they provide means to enquire about it.

Execution duration: The execution duration measures the maximum delay in seconds between the moment when a request is sent and the moment when the results are received by a client. The execution duration wraps the complex transmission factor into account. It includes service response time and network transmission latency. Service response time is the maximum seconds that elapses from the moment that a web service receives a legitimate SOAP request until it produces the corresponding legitimate SOAP reply. The execution duration expression is $q_{duration} = t_{rep}(s,op) + t_{trans}(s,op)$, meaning that the execution duration is the sum of the response time $t_{rep}(s,op)$ and the transmission time $t_{trans}(s,op)$. Services advertise their response time or provide methods to enquire about it. The transmission time is estimated based on past executions of the service operations, e.g. $t_{trans}(s,op) = \sum_{i=1}^{n} t_i(s,op)/n$, where $t_i(s,op)$ the past observation of the transmission time, and n is the number of execution times observed in the past. The average Execution Cost is also used to evaluate the execution cost quality attribute of a service.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reliability: the reliability of a service is the probability that a request is correctly responded within a maximum expected time frame which is published in the web service description. In the thesis, reliability measures the degree of compliance between providers claimed value with the actual value. Reliability is a technical measure related to hardware and/or software configuration of web services and network connections between the service consumers and providers. The value of reliability is computed from historical data about past invocations using the expression $q_{rel}(s) = N_c(s)/k$, where $N_c(s)$ is the number of times that the service s has been successfully delivered within the maximum expected time frame, and k is the total number of invocations.

Availability: A vailability is the quality aspect of whether the web service is present or ready for immediate use. The availability $q_{av}(s)$ of a service is the probability that the service is accessible. In the thesis, the value of the availability of a service is computed using the following expression $q_{av}(s) = t_a(s)/k$, where $t_a(s)$ is the amount of time (in seconds) in which service is available during given k seconds.

In conclusion, the quality vector $Q(s) = (q_{cost}(s), q_{duration}(s), q_{rel}(s), q_{av}(s))$ is used to represent the quality of a service in the thesis. This proposed QoS model also supports extended custom QoS attribute as long as customers give its unambiguous definition and computing approach.

4.3 Proposed QoS Model of Composite service

The above quality attributes are also applied to evaluate the quality of composite services. In order to simplify the computation, a composite service, first of all, will be unfolded and only composed of a set of sequential component services, no loops and conditions. Namely, we can consider a composite web service containing several concretized abstract services with sequential structure. For example, a composite service can be defined as S = *{AS1, AS2, ASi ...}* and each component *ASi* must be concretized. The next section explains the approach of unfolding a complex

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

composition workflow.

4.3.1 Unfolding Service Workflow

A complex composite service may be composed of conditional branches and loops. In order to simplify the computation of the QoS of a complex composite service, an unfolding approach is introduced as follows based on (Canfora 2004) and (Zeng 2003).

For a *Switch* construct in the service composition, each case statement is annotated with the probability to be chosen. A conditional branch can be serialized as sequential workflow in which the QoS of each component service becomes its invocation probability times its original QoS value. For example, for a workflow containing a *Switch* structure composed of two Cases, with costs *C1* and *C2* respectively and probabilities p and 1-p, the overall cost is computed as follows: $p*C1 + (1-P)*C2$. The probabilities are initialized by the workflow designer. Therefore, a conditional branch can be unfolded as Figure 4-3.

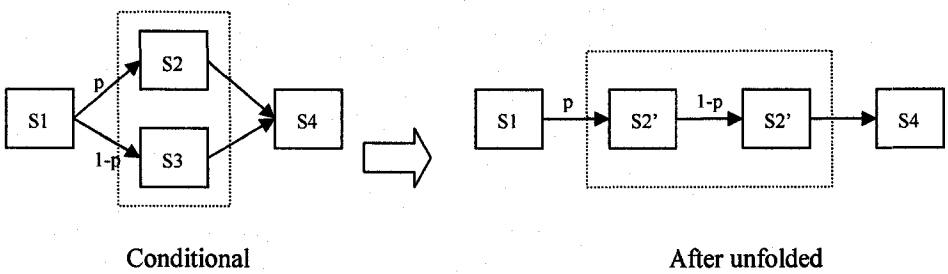


Figure 4- 3 Conditional workflow unfolding approach

Loop construct is unfolded with an estimated number of iterations *n*. Here, the QoS of the *Loop* is computed taking into account the factor *n*. for example, if the *Loop* compound has a cost *C1*, the estimated cost of the *Loop* will be $n*C1$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

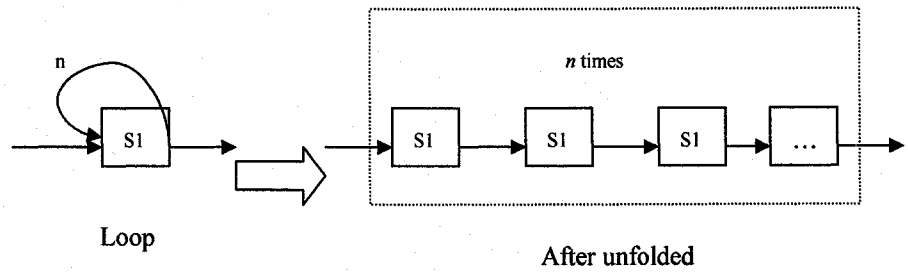


Figure 4- 4 Loop workflow unfolding approach

4.3.2 Computation of the QoS of Composite Service

Table 4-1 provides the aggregation functions of computing the QoS of a composite service. A brief explanation is given as follows.

Execution cost: the execution cost of an unfolded composite service is the sum of every component service. For instance, for a workflow containing a conditional branch, with cost *C1* and *C2* and probabilities *p* and *1-p*, the overall cost is *p*C1 + (1-p)*C2*. For a workflow containing a Loop, with cost *C1* and n times iteration, the overall cost is *n*C1*.

Execution duration: the execution duration of a sequential composite service is the sum of every component service.

Reliability: the reliability of a sequential composite service is the multiplication of the reliability of each component service.

Availability: the availability of a sequential composite service is the multiplication of the availability of each component service.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 4- 1 QoS aggregation function of different composite structure

QoS Attributes	Sequential	Conditional	Parallel	Loop
Execution Cost (*C*)	$\sum_{i=1}^{n} C(t_i)$	$\sum_{i=1}^{n} p_i * C(t_i)$	$\sum_{i=1}^{n} C(t_i)$	$k * C(t)$
Execution Duration (*D*)	$\sum_{i=1}^{n} D(t_i)$	$\sum_{i=1}^{n} p_i * D(t_i)$	$Max\{T(t_i)_{i\in\{1..n\}}\}$	$k * D(t)$
Reliability (*R*)	$\prod_{i=1}^{n} R(t_i)$	$\sum_{i=1}^{n} p_i * R(t_i)$	$\prod_{i=1}^{n} R(t_i)$	$R(t)^k$
Availability (*A*)	$\prod_{i=1}^{n} A(t_i)$	$\sum_{i=1}^{n} p_i * A(t_i)$	$\prod_{i=1}^{n} A(t_i)$	$A(t)^k$
Custom Attributes (*F*)	$f_S(F(t_i)_{i\in\{1..m\}})$	$f_C(p_i, F(t_i)_{i\in\{1..m\}})$	$f_P(F(t_i)_{i\in\{1..m\}})$	$f_L(k, F(t_i)_{i\in\{1..m\}})$

4.4 Proposed Framework

In order to overcome the deficiency of current web service framework (details in introduction section) and facilitate dynamic QoS-aware service composition and adaptation with global QoS constraints, we devise a flexible framework which is completely compatible with current web service model and supports the following functionalities:

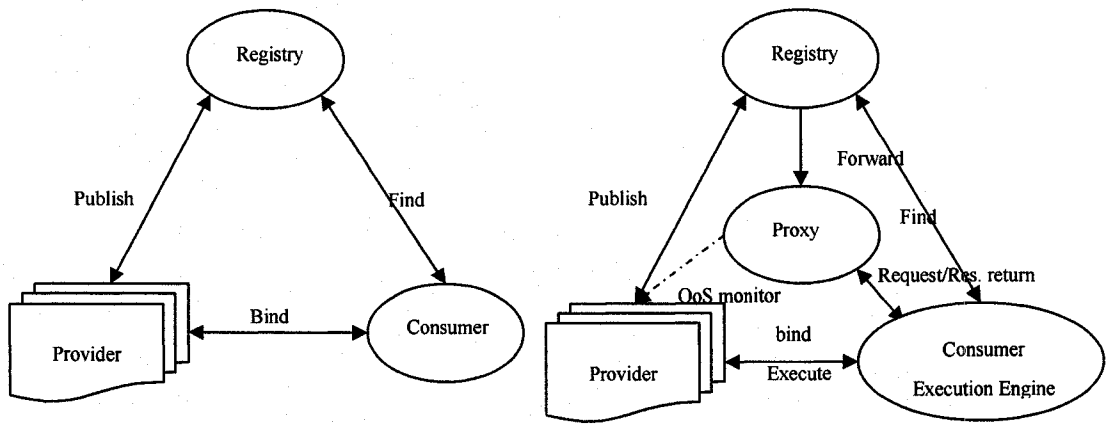
1. convert the user's requirement into a workflow process that can be implemented by existing services;
2. identify the most capable and efficient service for each component service in the workflow to meet users' functional and non-functional QoS requirements;
3. compose services that meet with user's global QoS requirement;
4. monitor the workflow invocation process, perform adaptation during service failures or unacceptable;

A suggested proxy is added in the proposed framework besides the existing registry, providers and consumers roles. The principle of the proposed framework design is to make the change as few as possible under current web service framework.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The proxy is the central controller which works as service and manages the consumers' requests, selects appropriate services and sends results back to consumers.

In the proposed framework, all current publishing, finding and binding operations are supported. The typical difference between the proposed from the current framework is that the proxy takes over the consumers' requests instead of manually marshalling the WSC workflow. The proxy identifies a component service composite service in its service repository satisfied with customers' requests. After global optimization, an execution plan with services' location is then sent back to the user. The user side execution engine binds with concrete providers, invokes the concrete services in the execution plan and returns the QoS attributes of each concrete service to the proxy. The appropriate methods are depicted in Figure 4-5.



a. Current Framework b. Proposed Framework

Figure 4- 5 Current and Proposed WS framework

4.5 Architecture of Proxy

Proxy plays an important role in the proposed WS framework during the WSC lifecycle. It produces a set of optimal global service composition plan. It also monitors the plan execution process to dynamically re-plan once the previous plan violates the client's requirement.

The proxy has a proxy central control module, request interpreter, workflow

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

analyzer, QoS evaluator, service composer, QoS adapter, QoS monitor modules and a service repository (shown in figure 4-6). Each module works independently and is managed by the central control module.

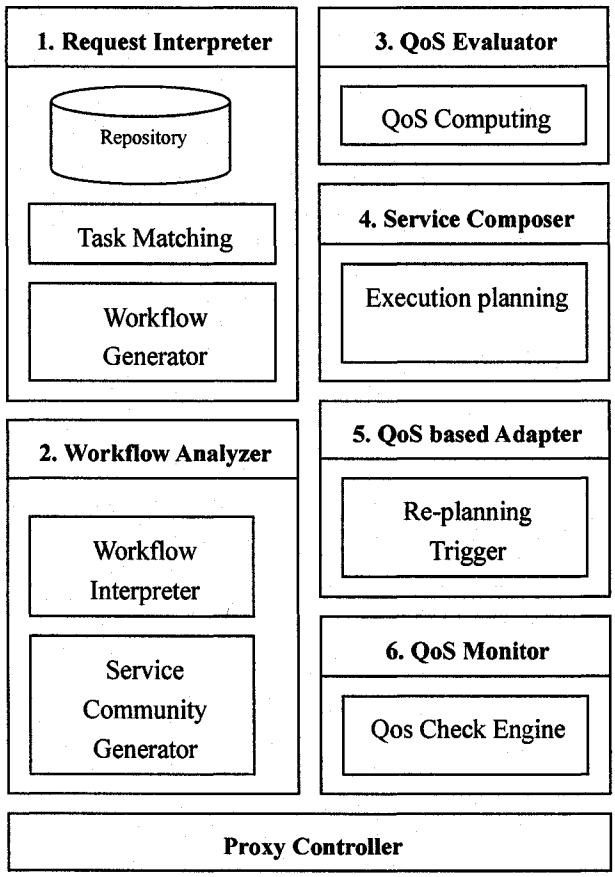


Figure 4- 6 Proxy Modules

The motivation of the proxy is to automatically generate a service combination execution plan which is the most satisfied solution for the user's request. In the thesis, we focus on user's non-functional requirements. We assume all services in the community using the same ontology or registered at the same TModel contract in UDDI registry, so that the proxy can generate the service community easily.

The default process of user requests in proxy is stated as follow and illustrated in figure 4-7.

1. User request is sent or forwarded from registry to proxy service.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. *Request interpreter* matches the user's request with an abstract workflow from proxy service repository.
3. *Workflow analyzer* takes the abstract workflow from *request interpreter* to unfold it and generate service communities for each abstract service.
4. *Service composer* gets the abstract workflow, QoS statistics and service communities from *QoS analyzer and service repository* and returns an execution plan to client-side execution engine.
5. *Execution engine* returns QoS values to *QoS evaluator* after each component service executed.
6. *QoS evaluator* records service runtime QoS values into *service repository* and also passes to *QoS adapter.*
7. *QoS adapter* asks *execution planning engine* to re-plan the execution plan if the QoS downgrades to the pre-defined threshold.

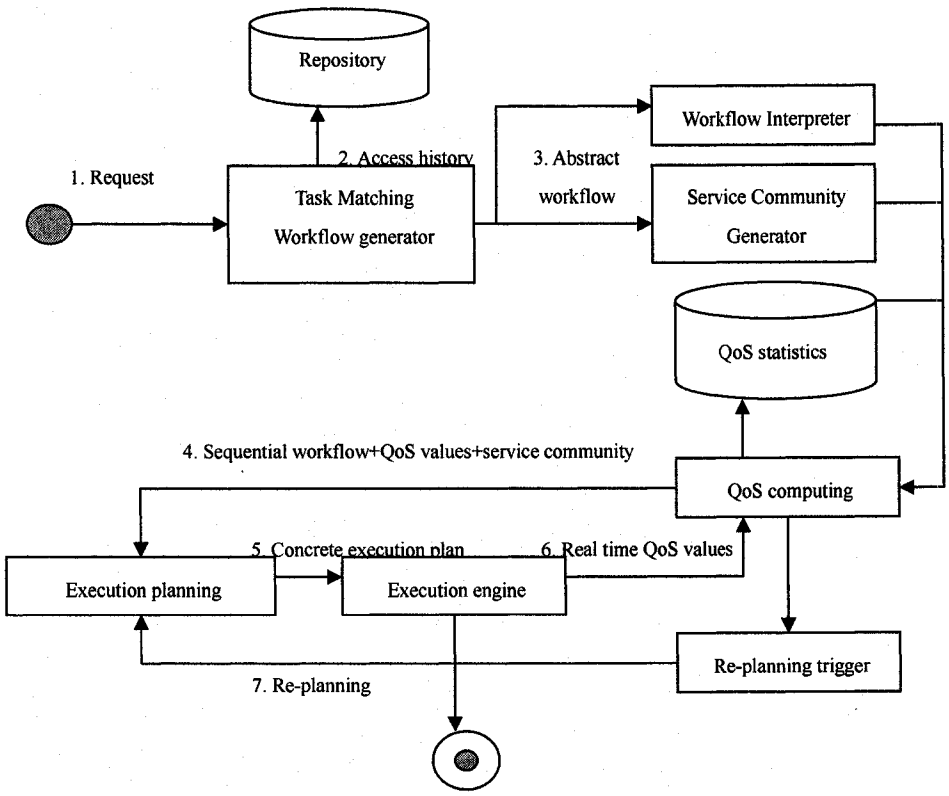


Figure 4- 7 Default enacting planning in proxy

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.1 Service Repository

The Service repository stores the information of web services candidates. It includes the service functional description and non-functional quality properties. There are two key tables: Service Table and QoS Statistics Table.

• ***Service Table***

Each web service in the Service Table includes the following fields (see Table 4-2):

1. Primary ID: a unique representation of the record;
2. Service Name: the name of the web service;
3. URL: the location of the Web service which points to the WSDL file of the service;
4. Namespace URI: a unique namespace for each web service to distinguish it from other services on the internet;
5. Service community: it shows the different functionalities of services. The services in the same service community have the same functionality with possibly different nonfunctional parameters (QoS);
6. Description: description of the operation;
7. Worst Execution Duration (TDmax): the maximum time needed for service which is claimed by providers;
8. Average Execution Duration (TDavg): the average time needed for service;
9. Execution Cost: the service cost;
10. Availability;
11. Reliability.

Columns 7 to 11 are non-functional QoS parameters. The table is initialized by the service provider. The attributes are refined by QoS evaluator module according to the service's runtime values received from client side execution. A web service may have many records in a table depending on the number of operations the service provides.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 4- 2 Service Table

PID	Service name	URL	URI	SC	Des	EDmax	EDavg	Exe Cost	Rel.	Ava.
1	CS1	http: //	http: //	AS1	Des1	70	45	20	.7	.8
2	CS2	http: //	http: //	AS1	Des2	30	40	10	.9	.9
3	...	...	...	...	...	...	...	...	...	...

• ***QoS Statistics Table***

QoS Statistics Table (Table 4-3) records the statistical QoS data for services. The information includes number of times a service having been invoked, last invoke time, average response time based on previous invocations, standard deviation of the response time to indicate the stability of service, reliability, and availability. Based on the statistical QoS data, we can update the Service Table with more accurate values of TDavg, reliability and availability.

Table 4- 3 QoS Statistics Tables

Service name	Invoke times (sec.)	Last invoked time (sec.)	TD (second)	Rel. (percentage)	Av. (percentage)
CS1	10	timestamp	45	.77	.90
CS2	20	Timestamp	40	.80	.90
...	...	...	...	...	...

4.2.2 Proxy Controller

The proxy control module acts as the central controller, which coordinates the invocations of all modules in proxy. It reads the initial file and creates the instances of other modules. It delivers the customer's request to the first module, manages the return results and forwards the results to the next module, and so on. The control module makes the proposed proxy more flexible and scalable since the configuration of the proxy is set by an external file.

Example of proxy configuration:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
<proxy>
    <source>
        <driver></driver>
        <user></user>
        <password></password>
    </source>
    <module name="RequstInterpreter">
        <config>requestInterpreter.xml</config>
    </module>
    < module name="WorkflowAnalyzer">
    </ module >
</proxy>
```

4.2.3 Request Interpreter

The function of request interpreter is to translate the user function request into the workflow composed of abstract services. The user request can be a task description, a process workflow or any other style as long as the request interpreter recognizes. The translation process can be implemented either based on history knowledge or service ontology.

To knowledge based request interpreter, the interpreter module simply translates the user request based on request history knowledge.

To ontology based solution, it assumes the services have ontology meaning. The first step is to bring the user request to the ontology level. The second step is semantic mapping during which the interpreter module matches the client request to services by ontology.

Since the interpreter module is not the major concern in the thesis, we assume the customer's requests are workflow-based. Therefore, we do not need to care about how to create history knowledge or service ontology. Table 4-4 shows the abstract process.

Table 4- 4 Process Table

Request	Abstract process
Task1	Task1.xml
Task2	Task2.xml

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Example of Task1.xml:

```
<workflow task="Task1">
    <sequence>
        <invoke>AS1</invoke> //calls an abstract service in a synchronous fashion
        <switch>
            <case>
                <invoke>AS5</invoke>
            </case>
            <case>
                <invoke>AS9</invoke>
            </case>
        </switch>
        <loop times="N">
            <invoke>AS2</invoke>
            <invoke>AS3</invoke>
            <invoke>AS4</invoke>
        </loop>
        <invoke>AS2</invoke>
        <parallel>
            <invoke>AS7</invoke>
            <invoke>AS8</invoke>
        </parallel>     //calls in asynchronous fashion (in parallel)
    </sequence>
</workflow>
```

4.2.4 Workflow Analyzer

The function of Workflow Analyzer is to simplify a complex abstract process to a sequential workflow and collect concrete services for each abstract service in the abstract sequential process. The two sub-modules are called workflow interpreter and service community generator.

The function of workflow interpreter module is to unfold a composite XML-based workflow to a sequential process and save it in Sequential Process table (SPtable) or a XML file for future requests. Table 4-5 shows the sequential workflow results. A composite service is composed of several services in a bracket. N*(AS2, AS3, AS4) means the composite service executes N times. ((AS7), (AS8)) means the components execute in parallel. Service community is generated based on history

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

knowledge or ontology matching. Table 4-5 shows the service community.

Table 4- 5 Sequential Process Table

Request	Abstract process
Task1	AS1,AS5,AS9,N*(AS2,AS3,AS4),Max((AS7),(AS8))
Task2	AS1,AS3,AS8,AS2

Table 4- 6 Service Community Table

Abstract service	Community members
AS1	S3,S4,S1,S9
AS5	S4,S5

4.2.5 QoS Evaluator

The function of QoS evaluator is to calculate the quality values for component and composite services in the execution plan. The other function is to update the each quality attribute based on the real-time values from client-side execution engine and store in QoS statistics table (Table 4-3 shown). QoS evaluator works as a QoS data resource center. It provides the QoS values for execution plan optimization. It also supplies the QoS resources for re-planning trigger (Figure 4-7 presents the function of QoS evaluator).

4.2.6 Service Composer

Service composer is the most important module in the proxy. Other modules serve as data collector or data analyzer of it. The data will be used in the module to find the final solution for client.

The idea of service composer is to determine the best global integration of concrete services. The goal of the optimization algorithm is to maximize a fitness function of the available QoS attributes and meet the client's request. In the thesis, we implement both Genetic Algorithm and Cultural Algorithm to optimize service composition. The advantage and operation process of GA and CA have been given in preliminary section. In order to shorten the computation duration of service combination, we take Culture Algorithm in this model.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.7 QoS Adapter

QoS adapter decides when and how to re-plan a failed execution plan. It includes two respective components: re-planning trigger and re-planning strategies.

The design of re-planning trigger is tightly bounded with QoS computing module. The QoS attributes and customer's QoS requirement should be used to determine the QoS thresholds.

The strategy of re-planning is to keep the re-planning scope as small as possible. Therefore, if a component service fails, the best solution is that a suitable candidate service can be found to substitute the failure service. In worst solution, there is no suitable candidate found, the service composer has to re-plan the whole execution workflow to find an alternative. The whole process has to start over from scratch. Re-planning strategies need to be carefully designed to improve the efficiency of this procedure.

4.6 Execution Engine

The function of execution engine is to initiate and invoke the optimal concretized business process, monitor the execution process at each component service and send run time QoS values to QoS evaluator module. The QoS evaluator calculates the quality values and stores them to the QoS statistics table (Table 4-3).

If a service problem occurs during process execution or one of its quality values downgrades to the threshold, the QoS evaluator fires the re-planning trigger in QoS adapter to re-plan the execution process from the failure point. The execution engine will adopt the alternative path and repeat above operations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5 Implementation and Experiment

The experiment aims to prove the reason we use CA algorithm in service composer module. We assume the client request is to maximize *availability* and *reliability* and also minimize *execution Cost* and *execution Duration.* In the experiment, the performance of Cultural Algorithm, Genetic Algorithm and Random selection approaches for QoS-aware WSC are compared.

5.1 Experiment Environment

This experiment is conducted under Windows XP and Java5 SDK which is running on Dell DIMENSION 440, Pentium® 4 CPU 2.80GHz, 2.79GHz 0.99GB of RAM.

5.2 Simulation Process

5.2.1 Initialization

The initial parameters of the experiment include the number of available web services, the size of abstract services, the quality properties of web services, population size and the number of generations.

A large population size of 500 is used so that we can rapidly reach to the convergent point. The fixed number of generations is 40. The initial service quality values are compliant to better execution duration and availability offers corresponding higher execution cost. In the experiment, we consider a simple workflow including 5 distinct abstract services which include 3 to 15 available services respectively. Table 5-1 and 5-2 show the samples of service QoS and abstract service (i.e. service community) description in the experiment (complete table are attached in Appendix). We assume a complex service workflow has been unfolded and stored as Table 5-3.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 5- 1 Web Service description

Service						
serviceName	**rspMax**	**rspAvg**	**price**	**Reliability**	**availability**	**comName**
S1	100	63	100	0.627	0.544	AS1
S2	500	451	30	0.598	0.532	AS1
S3	400	318	30	0.653	0.584	AS1
S4	30	13	300	0.955	0.956	AS2
S5	80	54	160	0.592	0.527	AS1
S6	140	68	70	0.937	0.929	AS2
S7	130	92	65	0.595	0.529	AS3

Table 5- 2 Abstract Service description

Community	
communityName	**Members**
AS1	S1,S3,S2,S5,S15,S36,S37,S38,S39,S40
AS2	S10,S6,S4,S16,S33,S34,S35
AS3	S7,S9,S17,S18,S28,S29,S30,S31,S32
AS4	S8,S11,S19
AS5	S14,S13,S12,S20,S21,S22,S23,S24,S25,S26,S27

Table 5- 3 Unfolded service workflow

Task	
taskName	**process**
TASK1	AS1,AS5,AS2,AS3,AS4
TASK2	AS3,AS4,AS6,AS7

According to the assumption of client request (mentioned at the beginning of this chapter), the solutions must satisfy the quality requirements, maximum *availability* and *reliability*, minimum *Execution Cost* and *execution Duration*. Therefore, our problem can now be modeled by means of a fitness function which maximizes *reliability* and *availability*, while minimizing *Execution Cost* and *execution Duration*. The fitness function is defined for a genome *g* as follows:

$$F(g) = \frac{w_1 Availability + w_2 \,\mathrm{Re}liability}{w_3 ExecutonCost + w_4 ExecutionDuration}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Where $w1, \ldots, w4$ are real, positive weighting factors. In general, calibrating weights is guided by observing the landscape of fitness function, as well as from the analysis of the evolution of the different factors. In our experiment, in order to simplify the calculation, we define the fitness function as follows:

$$F'(g) = \frac{Availability + \mathrm{Re}liability}{ExecutionCost + ExecutionDuration} * 1000$$

In this fitness function, the four quality parameters are considered having the same priority so that they have the same weight factor. The calibrating coefficient 1000 is to make the fitness values easily analyze.

5.2.2 Complete Process

5.2.2.1 Random Service Selection

The main idea of random service selection is randomly select an available concrete service from each service community following a workflow. The combination of these concrete services is the solution of client's requirement.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Pseudo code:

```
Initialization
    load web services and service communities
    For (curGen=1; curGen<=MaxGenerations;curGen++){
        For (pop=1; pop<=NumOfPopulation;pop++)}
            while (workflow is not over){
                Randomly select a service from a service community;
            }
            Create newGenome(curGen);
            Evaluate QoS of the genome;
        }
        evaluate QoS of the generation;
    }
```

Figure 5- 1 Proposed WSC by Random Selection Approach

5.2.2.2 Genetic Algorithm

To use GA searching for WSC solutions, we first need to have a suitable genome to present the problem. In our case, the genome is represented by a java ArrayList with a number of items that equals to the number of distinct *abstract services* composing our service. Each item contains a HashMap of the *concrete services*. Figure 5-2 illustrates the idea of how the genome is made.

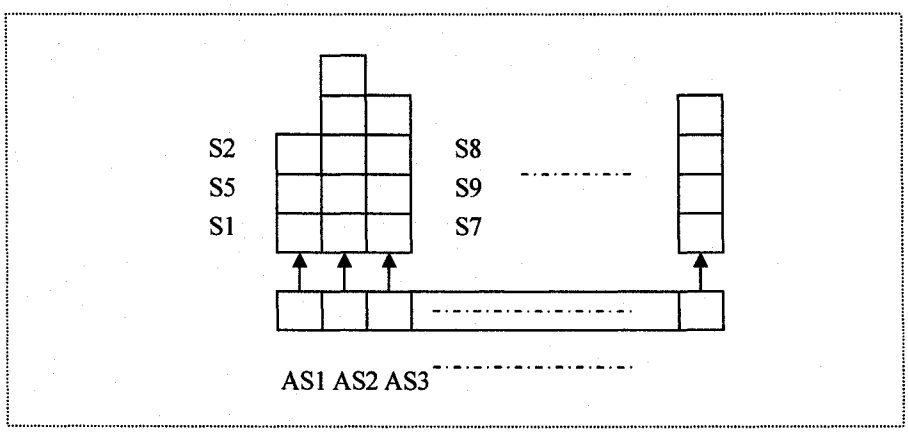


Figure 5- 2 genome model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In our experiment,

- Crossover operator is the standard two-point crossover. The crossover point is randomly selected according to the length of the genome.
- Mutation operator randomly selects an *abstract service* (a position in the genome) and randomly replaces a *concrete service* by another available concrete service in the community.
- Service selection operator picks up the individuals which have best fitness values in current generation as the parents of next generations.
- Stop criteria is a certain number of generation reached.

Pseudo code:

```
Initialization
    load web services and service communities;
    randomly create the population number of solutions;
    evaluate QoS for each individuals;
    save the individuals into Population Space as sample data set;
    For (curGen=1; curGen<=MaxGenerations;curGen++){
        pick up top 20% individuals according to finessVal as parents;
        For (pop=1; pop<=NumOfPopulation;pop++)}
            Randomly select two individuals from sample data set;
            Crossover;
            Mutation;
            make a new genome(curGen);
            Evaluate QoS of the genome;
        }
        evaluate QoS of the generation;
    }
```

Figure 5- 3 Proposed WSC by GA

5.2.2.3 Cultural Algorithm

We apply the same genome model as Genetic Algorithm used to present our problem. The same methods in GA are used to evaluate the individuals in Population Space. Belief Space is maintained by two knowledge sources, Situational Knowledge and History Knowledge. Situational Knowledge chooses the best individuals into

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Belief Space. History Knowledge takes the most recently used individuals into Belief Space. The individuals in Belief Space also influence the production of the next generation in population space.

In the experiment:

- In Population Space, individuals come from Belief Space to produce the individuals of next generation.
- The individuals in Population Space have two paths to be accepted into Belief Space:
    - Situation knowledge: The fitness is better than the individuals in Belief Space.
    - History knowledge: The most recent selected individuals.
- Belief Space is updated by selecting the certain number of best individuals.

Pseudo Code:

```
Initialization
    load web services and service communities;
    randomly create the population number of solutions;
    evaluate QoS for each individuals;
    initialize Population Space;
    initialize Belief Space by Situational and History Knowledge;
    For (curGen=1; curGen<=MaxGenerations;curGen++){
        Update Population Space by roulette selection;
        Update Belief Space by Situational and History Knowledge;
        For (pop=1; pop<=NumOfPopulation;pop++)}
            Randomly select two individuals from Belief and Population
            Space;
            Crossover;
            Mutation;
            make a new genome(curGen);
            Evaluate QoS of the genome;
        }
        evaluate QoS of the generation;
    }
```

Figure 5- 4 Proposed CA for WSC

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3 Test with Different Algorithm

5.3.1 Randomly Compose Service

Figures 5-5 illustrates the evolution of average quality parameters and fitness among 40 generations. Apparently, the convergent point of the average fitness is very low by random service selection approach. In other words, random selection approach is not an ideal approach to search for "maximum satisfied" WSC solution.

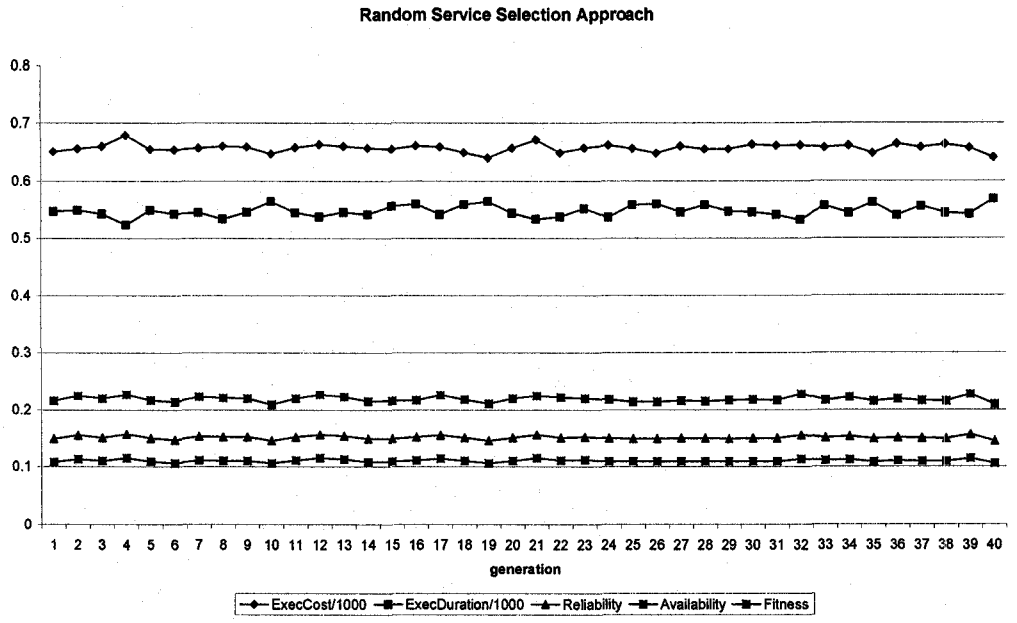


Figure 5- 5 evolution of quality and fitness parameters by Random service selection approach

Table 5- 4 Convergent point of quality parameters and fitness using Random service selection approach

Parameters	Convergent Value
Fitness	0.2 ± 0.02
Reliability	0.15 ± 0.001
Availability	0.11 ± 0.005
Execution Cost	0.657 ± 0.01
Execution Duration	0.547 ± 0.011

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3.2 Service Composition by using Genetic Algorithm

Figure 5-6 plots the *Execution Cost, Execution Duration, Reliability, Availability* and *Fitness* evolution track across GA generations. From this figure, the average fitness is converged at the central value 0.85 ± 0.01 from the 10^{th} generation. Namely, it takes 10 generations to reach to the peak point 0.85 ± 0.01 in GA WSC approach. The quality parameter and fitness convergent points are listed in table 5-5.

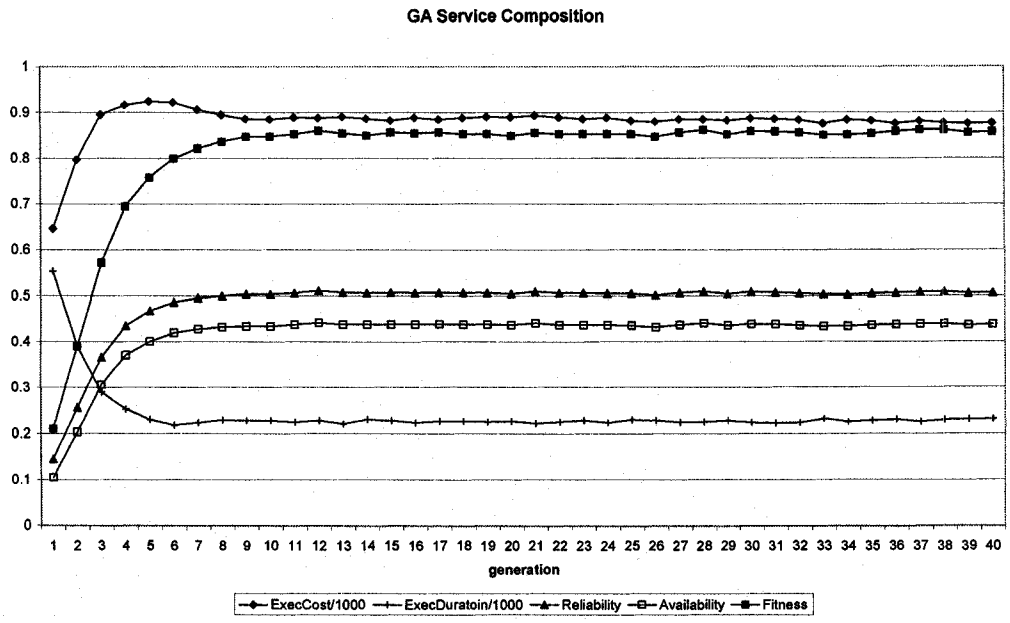


Figure 5- 6 evolution of quality parameters and Fitness by GA service composition approach

Table 5- 5 Convergent point of quality parameters and fitness using GA WSC approach

Parameters	Convergent Value
Fitness	0.85 ± 0.01
Reliability	0.5 ± 0.01
Availability	0.43 ± 0.01
Execution Cost	0.885 ± 0.005
Execution Duration	0.229 ± 0.006

5.3.3 Service Composition by using Culture Algorithm

Figure 5-7 presents the *Execution Cost, Execution Duration, Reliability,*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Availability and *Fitness* evolution track across CA generations. From this figure, the average fitness is converged at the central value 0.885 ± 0.007 from the 3rd generation. Namely, it takes three generations to reach to the peak point 0.885 ± 0.007 in CA. The quality parameter and fitness convergent points are listed in table 5-6.

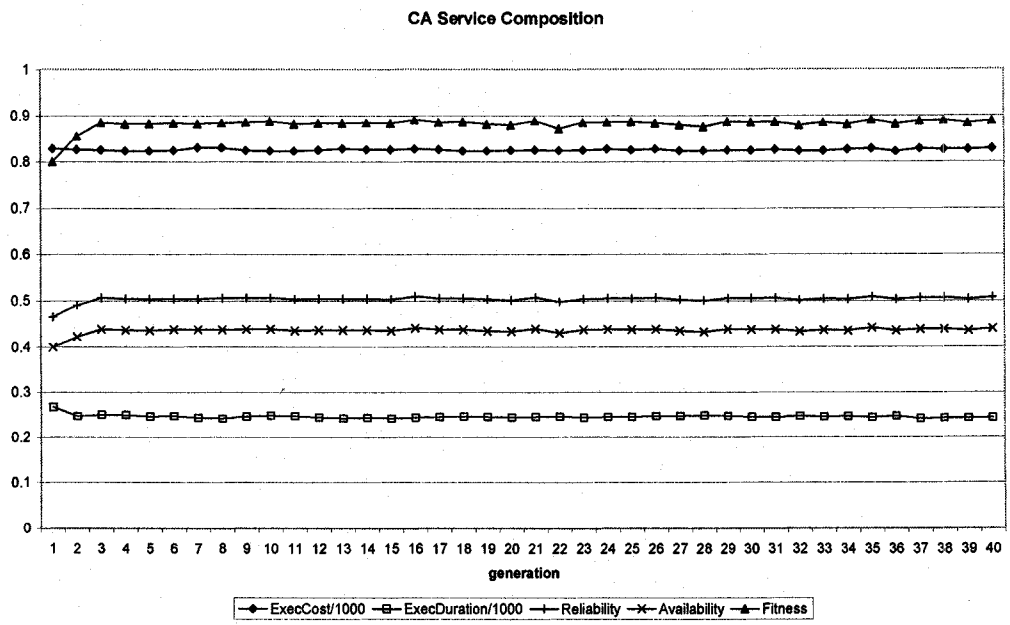


Figure 5- 7 evolution of quality parameters and Fitness by CA service composition approach

Table 5- 6 Convergent point of quality parameters and fitness using CA WSC approach

Parameters	Convergent Value
Fitness	0.885 ± 0.007
Reliability	0.506 ± 0.004
Availability	0.438 ± 0.004
Execution Cost	0.827 ± 0.004
Execution Duration	0.245 ± 0.004

5.4 Comparison

Figure 5-8 illustrates the *Fitness* evolution track across Random service selection, GA and CA generations. The average fitness values of Random Service Selection, GA and CA are respectively converged at 0.2 ± 0.02, 0.85 ± 0.01 and 0.885 ± 0.007. From the curve of Random Service Selection, we could not find at what generation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the convergent point reached, but we can clearly find GA reached its convergent point at the 10^{th} generation, CA only took 3^{rd} generation reaching its convergent point. Further, CA has the best convergent point which is 0.885 ± 0.007. In table 5-6, it shows the fitness improvement compared to random service selection approach.

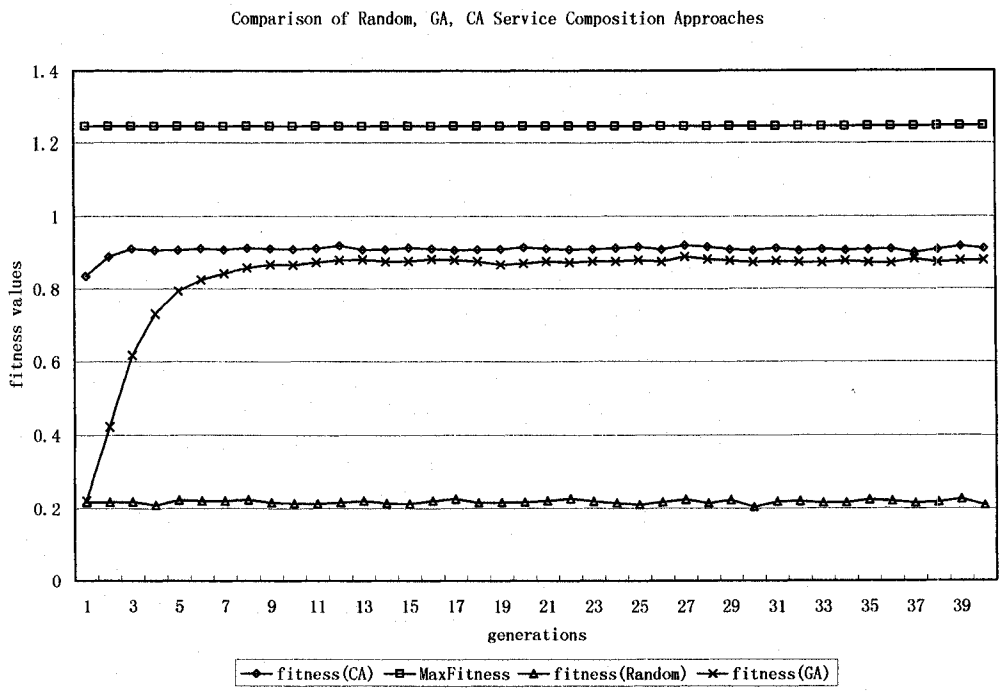


Figure 5- 8 comparison of fitness evolution of Random,GA and CA

Table 5- 7 Comparison of fitness among Random service selection, GA and CA WSC approaches

Approaches	# of Runs	# of Generations	Convergent Point	Fitness Increase
Random	5	/	0.2 ± 0.02	/
GA	5	11	0.85 ± 0.01	76.5%
CA	5	3	0.885 ± 0.007	77.4%

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6 Conclusion and Future Work

This thesis proposes a framework using proxy agents to monitor QoS in web service composition. According to the results reporting on a limited set of quality parameters a near optimal solution was reached in a reasonable amount of generations using a Cultural Algorithm, slightly surpassing the performance of a GA.

The proxy works well with any customized quality attributes by unfolding approach and cultural algorithm. It solves the irresolvable web service quality model problem. Therefore, the proxy has excellent scalability and good performance.

Future work will focus on additional knowledge structures in the Belief Space of the CA and investigate further quality parameters in the evaluation process. We will also try other algorithms to implement QoS-aware web service dynamic composition, such as Greedy Algorithm and compare them with GA and CA algorithm.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix A:

The experiment files are arranged at 4 packages: thesis.appinit, thesis.basicClass, thesis.simulation, and thesis.algorithm. The files in each package are listed as follow:

Package: thesis.basicClass: (data structure)

SampleSolutions.java, ServCommunity.java, Services.java, Solution.java, Task.java

Package: thesis.appinit

AppInit.java

Package: thesis.algorithm

AlgorithmImp.java

Package: thesis.simulation

WSCSimulation.java

Design of the simulation system:

The purpose of the system design is to make the system has good performance and good scalability. In order to improve the performance of the simulation system, we used data source connection pool, concrete service pool, service community pool and solution pool. In order to improve the scalability, we reduced the hard code as much as possible. Client can setup the initial values in a table or an initial file. The main control class is thesis.simulation.WSCSimulation. First of all, it initializes the system by creating a connection pool and loading all initial values from external data sources into the data structure which likes the genome model we mentioned in chapter 5. Secondly, according to the client requirement, it chooses corresponding algorithm to compose web services. Thirdly, it pours out the solutions from solution pool into database table through connection pool.

The complete initial parameters of the experiment are list in following 3 tables:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

service						
serviceName	**rspMax**	**rspAvg**	**price**	**reliability**	**availability**	**comName**
S1	100	63	100	0.627	0.544	AS1
S2	500	451	30	0.598	0.532	AS1
S3	400	318	30	0.653	0.584	AS1
S4	30	13	300	0.955	0.956	AS2
S5	80	54	160	0.592	0.527	AS1
S6	140	68	70	0.937	0.929	AS2
S7	130	92	65	0.595	0.529	AS3
S8	220	167	25	0.494	0.425	AS4
S9	100	51	190	0.858	0.861	AS3
S10	45	16	200	0.983	0.978	AS2
S11	30	11	250	0.969	0.953	AS4
S12	75	50	150	0.499	0.425	AS5
S13	55	35	210	0.569	0.521	AS5
S14	50	24	200	0.869	0.863	AS5
S15	90	54	180	0.722	0.661	AS1
S16	100	69	100	0.513	0.452	AS2
S17	35	18	250	0.803	0.749	AS3
S18	100	66	100	0.543	0.483	AS3
S19	300	241	20	0.487	0.422	AS4
S20	400	356	30	0.465	0.4	AS5
S21	100	90	98	0.627	0.6	AS5
S22	500	451	30	0.598	0.6	AS5
S23	400	318	30	0.653	0.7	AS5
S24	30	13	300	0.955	0.956	AS5
S25	80	54	160	0.592	0.6	AS5
S26	140	68	70	0.937	0.929	AS5
S27	130	92	65	0.595	0.529	AS5
S28	220	167	25	0.494	0.425	AS3
S29	100	51	190	0.858	0.861	AS3
S30	45	16	200	0.983	0.978	AS3
S31	30	11	250	0.969	0.953	AS3
S32	75	50	150	0.499	0.425	AS3
S33	55	35	210	0.569	0.521	AS2
S34	50	24	200	0.869	0.863	AS2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

service						
serviceName	**rspMax**	**rspAvg**	**price**	**reliability**	**availability**	**comName**
S35	90	54	180	0.722	0.661	AS2
S36	100	69	100	0.513	0.452	AS1
S37	35	18	250	0.803	0.749	AS1
S38	100	66	100	0.543	0.483	AS1
S39	300	241	20	0.487	0.422	AS1
S40	400	356	30	0.465	0.4	AS1

Web Service description

Community	
communityName	**Members**
AS1	S1,S3,S2,S5,S15,S36,S37,S38,S39,S40
AS2	S10,S6,S4,S16,S33,S34,S35
AS3	S7,S9,S17,S18,S28,S29,S30,S31,S32
AS4	S8,S11,S19
AS5	S14,S13,S12,S20,S21,S22,S23,S24,S25,S26,S27

Abstract Service description

Task	
taskName	**process**
TASK1	AS1,AS5,AS2,AS3,AS4
TASK2	AS3,AS4,AS6,AS7

Unfolded service workflow

The updating formula of Execution Duration, Reliability, and Availability as follows:

ExecDuration:

$$ExecDuration_{avg} = (CurExecDuration_{avg} * InvokeTimes + CurExecDur)/(InvokeTimes + 1)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reliability:

$$if\,(CurExecDuration <= ExecDuration_{worst})then$$

$$\mathrm{Re}liability = (Cur\,\mathrm{Re}liability * InvokeTimes + 1)/(InvokeTimes + 1)$$

else

$$\mathrm{Re}liability = (Cur\,\mathrm{Re}liability * InvokeTimes)/(InvokeTimes + 1)$$

Availability:

$$if\,(CurExecDuration <= ExecDuration_{worst})then$$

$$Availability = (CurAvailability * ExecDur_{avg} * (InvokeTimes + 1) + CurExecDur)/$$
$$(ExecDur_{avg} * (InvokeTimes + 1) + CurExecDur)$$

else

$$Availability = CurAvailability * ExecDur_{avg} * (InvokeTimes + 1)/$$
$$(ExecDur_{avg} * (InvokeTimes + 1) + CurExecDur)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reference

(Fan 2005): Fan, J.C.; Subbarao, K.; 2005; A Snapshot of Public Web Services; *Research articles and surveys;* Page: 24-32; ACM Press;

(Jorge 2005): Jorge, C.; Miller, J.; Su, J.W.; Pollock, J.; 2005; Academic and industrial research: Do their approaches differ in adding semantics to web services? ; Page: 14-21; Springer Press;

(Milanovic 2004): Milanovic, N.; Malek, M.; 2004; Current solutions for web service composition; *Internet Computing, IEEE Volume: 8, Issue: 6* Page: 51- 59; IEEE Press;

(Tsalgatidou 2002): Tsalgatidou, A.; Pilioura, T.; 2002; an Overview of Standards and Related Technology in Web Services; *a Journal of Distributed and Parallel Databases; Springer Press;*

(Lau 2001): Lau, T.C.; 2001; "QoS for B2B Commerce in the New Web Services Economy", *ISEC 2001 Workshop on Performance and QoS for E-Commerce Applications, Hong-Kong, China, April 2001*

(Jin 2002): Jin, L.; 2002; "Analysis on Service Level Agreement of Web Services"; *Software Technology Laboratory, HP Laboratories Palo Alto, Technical Paper HPL-2002-180, June 2002*

(Maximilien 2002): Maximilien, E.M.; Singh M.P.; 2002; Conceptual model of web service reputation; *ACM SIGMOD Record, Special section on semantic web and data management,* Pages: 36-41, ACM Press

(Ran 2003): Ran, S.; 2003; a model for web service discovery with QoS; *ACM SIGecom Exchanges*, Pages: 1-10, ACM Press

(Zeng 2003): Zeng, L.Z.; Benatallah, B.; Dumas, M.; Kalagnanam, J.; Seng, Q.Z.; 2003; Quality driven web services composition; *International World Wide Web Conference, Proceedings of 12th*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

international conference on World Wide Web Session: Web engineering; Pages: 411-421, ACM Press

(Zeng 2004): Zeng, L.Z.; Benatallah, B.; Ngu, A.H.H.; Dumas, M.; Kalagnanam, J.; Chang, H.; 2004; QoS-aware middleware for web services composition; *Software Engineering, IEEE Transactions;* Pages: 311-327; IEEE Press

(Gouscos 2003): Gouscos, D.; Kalikakis, M.; Georgiadis, P.; 2004; An Approach to Modeling Web Service QoS and Provision Price; *the 4th* international conference on web information systems engineering workshops, Pages: 121-130; IEEE Press

(Kalepu 2004): Kalepu, S.; Krishnaswamy, S.; Loke, S.W.; 2004; Verity: a QoS metric for selecting web service and providers; *Web information systems engineering workshops, Proceedings 4th international conference;*Pages: 131-139;

(Yu 2005): Yu, T.; Lin, K.J.; 2005; service selection algorithms for web services with end-to-end QoS constraints; Pages: 103-126; Springer Press;

(Sherchan 2006): READ PLS Sherchan, W.; Loke, S.W.; Krishnaswamy, S.; 2006; A fuzzy model for reasoning about reputation in web services; *Symposium on Applied Computing, Proceedings of the 2006 ACM symposium on Applied computing;* Pages: 1886-1892; ACM Press;

(Bouch 2000): Bouch, A.; Kuchinsky, A.; Bhatti, N.; 2000; Quality is the eye of the beholder; *meeting user's requirements for internet quality of service, Proceedings of the SIGCHI conference on human factors in computing systems*

(Aggarwal 2004): Aggarwal, R.; Verma, R.; Miller, J.; Milnor, W.; 2004; Constraint driven web service composition in METEOR-S. *Proceedings of the 2004 IEEE International Conference on Services Computing*; pages 23-30; IEEE Press

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(Canfora 2004): Canfora, G.; Penta, M.D.; Esposito, R.; Vilanni, M.L.; 2004; A lightweight approach for QoS-aware service composition; *Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 36-47

(Cardoso 2004): Cardoso, J; Sheth, A.; Miller, J.; Arnold, J.; Kochut, K.; 2004; Quality of service for workflows and web service processes; *Journal of Web Semantics*

(Tian 2003): Tian, M.; Gramm, A.; Naumowicz, T.; Ritter, H.; Schiller. J.; A concept for QoS integration in web services; *Proceedings of the First Web Services Quality Workshop at WISE*, Rome, Italy, December

(Singhera 2006): Singhera, Z.U.; Shah, A.; 2006; Extended services framework to meet non-functional requirements; *ACM International Conference Proceeding Series, Workshop proceedings of 6th international conference on web engineering;* ACM Press;

(Milanovic 2005): Milanovic, N.; Malek, M.; 2005; Architectural Support for Automatic Service Composition; *Proceedings of the 2005 IEEE International Conference on Services Computing (SCC'05)*

(Penta 2006): Penta, M.D.; Esposito, R.; Villani, M.L.; Codato, R.; Colombo, M.; Nitto, E.D.; 2006; WS Binder: a framework to enable dynamic binding of composite web services; *International conference on software engineering, Proceedings of the 2006 international workshop on service-oriented software engineering;* Pages: 74-80; ACM Press;

(Xia 2006): Xia, J.; 2006; QoS-based Service Composition; *Computer software and applications conference, 2006, COMPSAC'06 30th;* Page: 359-361; IEEE Press;

(Canfora 2006): Canfora, G.; Penta, M.D.; Esposito, R.; Vilanni, M.L.; 2006; An Approach for QoS-aware Service Composition based on Genetic Algorithms; *Genetic And Evolutionary Computation*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Conference, Proceedings of the 2005 conference on Genetic and evolutionary computation; Pages: 1069-1075; ACM Press;

(Zhou 2007): Zhou, J.H.; Niemela, E.; Savolainen, P.; 2007; An Integrated QoS-Aware Service Development and Management Framework; *Software Architecture, IEEE/IFIP Conference;* Pages: 13-13; IEEE Press

(Li 2005): Li, B.; Tang, X.Y.; Jian, L.; 2005; The Research and Implementation of Services Discovery Agent in Web Services Composition Framework; *Machine Learning and Cybernetics, Proceedings of 2005 International Conference;* Pages: 78-84; IEEE Press

(Menasce 2002): Menasce, D.A.; 2002; QoS issues in web services; *IEEE internet computing;* Pages: 72-75; IEEE Press

(Araban 2004): Araban, S.; Sterling, L.; 2004; Measuring Quality of Service; *1st Australian Workshop on Engineering Service-Oriented Systems (AWESOS 2004).* Melbourne, Australia, 2004.

(Dustdar 2005): Dustdar, S.; Schreiner, W.; 2005; A survey on web services composition; *International Journal of Web and Grid Services,* Pages:1-30

(Patel 2003): Patel, C.; Supekar, K.; Lee, Y.; 2003; A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows; *Book: Database and Expert Systems Applications;* Springer Press

(W3C 2003): http://www.w3.org/TR/2003/WD-ws-arch-20030808/#whatis

(OASIS 2001): Glossary for the OASIS Web Service Interactive Applications http://www.oasis-open.org/committees/wsia/glossary/wsia-draft-glossary-03.htm

(ISO9000 2002): http://www.iso.ch/iso/en/ISOOnline.frontpage

(GA 2002): http://jgap.sourceforge.net/doc/gaintro.html

(Reynolds 2004): Reynolds, R.G.; Peng, B.; 2004; Cultural Algorithms: Modeling

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of how cultures learn to solve problems; *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence;* IEEE Press;

(Goldberg 1989): Goldberg, D.E.; 1989; Genetic Algorithms in Search, Optimization and Machine Learning; Addison-Wesley Pub Co.;

(Paolucci 2002): Paolucci, M.; Kawamura, T.; Payne, T.R.; Sycara, K.; 2002; Semantic Matching of Web Service Capabilities; *The Semantic Web – ISWC 2002; Proceeding of First International Semantic Web Conference, Sardinia, Italy;* Springer Press;

(Rose 2004): Rose, D.E.; Levinson, D.; 2004; Understanding user goals in web search; *Proceedings of the 13th international conference on World Wide Web, New York, USA;* IEEE Press; Pages: 13-19;

(Hu 2005): Hu, J.; Guo, C.; Wang, H.; Zuo, P.; 2005; Web Service Peer-to-Peer Discovery Service for Automated Web Service Composition; Springer Press; *Journal World Wide Web;* Pages: 211-229;

(Aversano, 2004): Aversano, L.; Canfora, G.; Ciampi, A.; 2004; An algorithm for Web service discovery through their composition; *Web Service, 2004. Proceedings of IEEE International Conference;* IEEE Press; Pages: 332-339;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Vita Auctoris

Zhiyang Wang was born in Changde, Hunan, china. He received his bachelor degree of communication engineering at Huazhong University of Science and Technology. Currently, he is completing his masters' degree in computer science from the University of Windsor and expects to graduate in October 2007.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.