ON THE ASSESSMENT OF COMMUNITIES OF WEB SERVICES

by

REEM KATEB

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

Master of Applied Science

Major: Information System Security

Thesis Advisor:

Dr. Jamal Bentahar

Concordia University

Montreal, Quebec, Canada

2013

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Reem Saleh Kateb

Entitled: On The Assessment of Communities of Web Services

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Fung	Chair
Dr. Ben Hamza	Examiner
Dr. Ormandjieva	Examiner
Dr. Bentahar	Supervisor

Approved by ______________________________
Chair of Department or Graduate Program Director

____
Dean of Faculty

Date ______________________________

ABSTRACT

The notion of community of web services has been recently proposed and investigated to gather functionally similar web services in the same virtual space. This allows increasing the visibility of web services and their collaboration, which makes their discovery and composition easier. Using the community infrastructure, users are supposed to direct their requests to the communitys manager (called master), that is in charge of selecting the appropriate web service. Because many communities providing the same functionality are available, selecting the best community to deal with, from the users and providers perspectives, is a key factor that still needs to be investigated. Another particularly challenging issue yet to be addressed is the selection by the master of the appropriate web service to be hosted in the community. Reputation has been proposed as a means to help users, providers, and masters evaluate and rank different candidates. However, reputation is mainly based on users feedback, which is not always accurate. Moreover, other performance parameters should be considered in the selection game.

In this thesis, we propose a new assessment process that focuses on various performance aspects of the community rather than just its reputation. This assessment considers the performance parameters from the users, providers, and masters perspectives. In this approach, the communities performance rate is mainly based on the web services hosted by those communities. Such an assessment approach helps the master of the community differentiate between web services so that only the appropriate ones can be invited or accepted to join based on the communities requirements. It also helps the users and providers select the best available communities.

The proposed method works on three steps. The first step focuses on defining and computing the evaluation metrics used in the assessment process while considering the requirements of all the stakeholders, namely users, providers, and communities. Thus, each community or web service is described by a vector of metrics. The second step includes the clustering of the evaluated communities and web services using the resulted vectors from the first step. During the third step, the resulting clusters are ranked using a function called goodness function. Web services and communities belonging to the best cluster are then selected. The effectiveness of the proposed assessment approach is tested by simulation and comparison to two other approaches in the literature.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Jamal Bentahar for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank Mr. Ehsan for his contributions to this work. I would additionally like to thank all my friends and colleagues who made themselves available by providing support in many ways. Special thanks goes to my parents for their continuous support through my studies far away from home. Finally, I would like to dedicate my ultimate thanks to my beloved husband Youssef and my little princess Aleen for their encouragement and never ending support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

In this chapter, I will introduce the context of my research and discuss the motivations behind it. This chapter also presents the contributions of this thesis. The conclusion of this chapter is an overview of the thesis organization.

1.1 Context of Research

Web services are software components that have emerged as a core technology for sharing resources and merging processes inside companies or organizations. Most organizations relay on web services and use them as an interface to integrate different applications within their boundaries because they provide a concrete implementation of service oriented architecture (SOA). To build and access web services, different standards are used, such as: Web Service Definition Language (WSDL), Universal Description, Discovery and Integration (UDDI), and Simple Object Access Protocol (SOAP). Providers of web services publish their services over the Internet, and consumers request these services. However, to send a request users have to discover the services first to see if they satisfy their needs. As the number of developed web services providing similar functionalities continues to grow, the need for a good discovery method becomes critical.

In general, the objective of the discovery process is to find services that satisfy the users' needs and requirements. However, discovering and selecting a single web service that meets these requirements, from over a hundred available services, is a difficult task. A solution to this problem is to group web services providing similar and complementary functionalities into communities [6, 7, 38].

A community hosts different web services (called slaves) having similar or complementary functionalities but different qualities of service. The community is lead by a master web service, and one of its responsibilities is to invite new web services and eject existing ones if they are not performing as expected. Communities help users direct their requests to the master of the community, which chooses the best service that meets their requirements. As those communities contain a certain number of web services, this will ease the selection of slaves on behalf of users. They also help keep a high standard for available services, as when a web service fails to respond to users' request, it is the master's responsibility to find another web service for replacement.

The main focus of this thesis is to help users choose the best community among many others to seek services, and to assist providers select the best community to join. Furthermore, we will focus on helping the master of the community decide which web service to invite, and which web service to fire.

1.2 Motivations

The first motivation of this thesis is to facilitate and enhance the selection process of web services. With the increased number of web services having similar functionalities, more arguments arise on how to find the best service to fit the requestors' requirements. However, the problem is that some web services do not advertise accurate QoS information, which is directly related to their performance. The QoS is a set of non-functional properties associated with each service. QoS is measured by using different metrics, such as response time, availability, thruput, price, and so on. One of the main issues in SOA, is that when advertising their services, providers could be untrustworthy and may publish high QoS information to receive more requests.

As the communities are designed to handle the users' requests, finding the best service to invite as a member of the community is critical for the master. In fact, users send their requests directly to the master, which puts a huge responsibility on this master when it comes to choosing the best web services to invite. However, if users are satisfied with the provided services, that will lead them to send more requests to the same community. Assessing the web service from the master's point of view is still an open issue in the literature about communities of web services that this thesis aims to address.

The second motivation is to help users and providers select the community that suits them best. The fact that users have to look for appropriate communities that host the requested services and providers have to find suitable communities to host their services requires differentiating between these communities. The community assessment and ranking have been studied in previous proposals based on the reputation parameter [22, 12, 24]. This motivates us to find a way to do this assessment by considering other performance aspects. Although reputation is an important issue reflecting the trust that users, providers and masters have on the web services, other aspects such as the connectivity and wiliness to collaborate are also fundamental and need to be studied.

1.3 Contributions

This thesis presents two main contributions. First, we introduce an assessment method to help the master of a given community select the best services to invite with respect to the community's needs. In this assessment we focus on the non-functional properties (QoS) of each web service to distinguish the best one. We also focus on different community's needs, as different communities may be interested in different properties.

In the second contribution, we introduce an assessment method to differentiate communities by considering the overall performance. This assessment is done from two different perspectives: providers and users.

In our implemented system for simulation purposes, we analyze the effectiveness of our method by comparing it to a reputation based selection and to a random selection. We found that our model is outperforming the other models in terms of satisfying the master, users, and providers (i.e., web services) requirements.

1.4 Thesis Organization

The reminder of this thesis is organized as follows: Chapter 2 presents background information to help understand the rest of the thesis, and an overview of previous related work. First, we discuss the web service definition, architecture, and operation, and present the concept of community of web services, its architecture, and operations. Chapter 3 discusses our proposed methods. We first introduce the communities' assessment process from users' and providers' perspectives. Afterwards, we present the web services assessment. Chapter 4 describes our simulation environment, then we analyze the results to show the effectiveness of our method. Chapter 5 concludes this thesis and provides suggestions for future research.

CHAPTER 2. BACKGROUND AND RELATED WORK

This chapter discusses relevant background information important to understand the rest of the thesis. Section 2.1 presents a brief discussion of web services, web services architecture, and operations. Section 2.2 introduces the concept of communities of web services from the architecture and operations perspectives. Section 2.3 discusses relevant related work that has been done to evaluate and assess communities of web services based on trust and reputation. The assessment of web services based on reputation and QoS will also be discussed in this section.

2.1 Web Services

2.1.1 Definition

In daily life, we can associate the concept of service to many metaphors, which might include utilities. Any service that is available over the internet, such as flight booking, hotel reservation, etc., is called web service if it uses a standardized XML messaging system. Web services are flexible as they are not bonded to any programming language or operating system.

The World Wide Web Consortium (W3C) defines web services as follows:"A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with XML serialization in conjunction with other Web-related standards". When developers declare a new web service, it will be discovered based on its description that fully describes the service. Developers also have to declare a public interface and a readable documentation to help other developers when integrating different services [10].

2.1.2 Web Service Architecture

In this part, we describe the architecture of web services from two perspectives: first, by examining web service components; and second, by examining the web service protocol stack [8].

A. Web Service Components

Web service architecture provides a message exchange mechanism that has three main components [8]:

1. *Service Provider*

   The Service provider is the provider of the web service, whose responsibility is implementing the service and making it available over the internet.

2. *Service Requestor*

   The service requestor is the consumer of the web service who wishes to make use of a *service provider's* service. The requestor opens a network connection with the web service and sends an XML request.

3. *Service Registry*

   The service registry is a centralized directory of services where providers can declare new web services or find an existing one. Figure 2.1 shows the web service components and how they interact with each other.

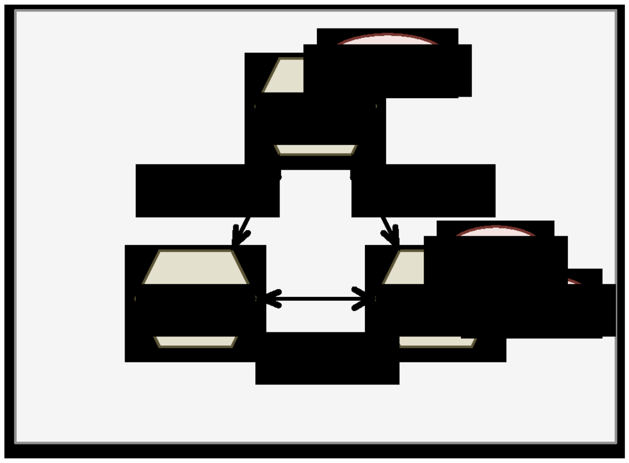


Figure 2.1 Web Service Components and Operations

B. Web Service Protocol Stack

The second way to describe the web service architecture is by examining the emerging web service protocol stack [8]. The stack has four main layers as shown in Figure 2.2.

Figure 2.2 Web Service Protocol Stack

1. *Service Transport*

Web services technology uses an interoperable messaging mechanism that makes the transport protocols one of its important foundations. This layer's responsibility is to transport messages between applications. Web services' messages can be transported by using web protocols such as HyperText Transport Protocol (HTTP) or Secure HTTP (HTTPS), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and any communications protocol.

2. *Messaging Services*

   This layer contains the most fundamental web services specifications and technologies such as eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), and WS-Addressing. This layer is in charge of making the encoding messages in a common XML format understandable at both ends. The specifications model the base of interoperable messaging between web services. The following are some important terminologies used in this layer.

   a. XML Messaging

   The world of web services is basically modelled based on the core set of XML specifications. XML is not a language, it is a metalanguage which can be used to define new languages. It has gained its acceptance because it enables various computer systems to share data more effectively, without considering the underlying operating system or programming language. There are many XML tools which are available for mostly every programming language, including C, C++, Java, Perl, and Python, and operating system. XML is always the natural choice for developers when they decide to build a web service messaging system.

   b. SOAP

   It is an XML-based protocol used to provide a simple and lightweight technique to exchange information between services. It is used to minimize the cost and difficulty of merging applications that are build on different platforms. SOAP is an XML document with three elements: an envelope, a header and body. It is defined independently of the inherent messaging transport technique in use, where it makes use of many options of transports for message exchange.

c. WS-Addressing

WS-Addressing defines XML elements to identify the endpoints of web services and to secure end-to-end endpoints in messages. It also enables messaging systems in supporting message transmission through networks.

3. *Service Description*

   This layer's responsibility is to describe the public interface of a given web service using Web Service Description Language (WSDL). WSDL is the most mature metadata used to describe web services. It gives the developers a chance to describe the functional characteristics of the web service. It also offers a standard language-agnostic view of client services. WSDL is an XML file that specifies the public interface of services over the network as a group of endpoints operating on messages with either document-oriented or procedure-oriented information. It allows the description of endpoints and their messages without considering the message formats or network protocols in use. The public interface of web services contains information about all available functions, all data type information about XML messages, information about the transport protocol to be used, and addresses data to locate the service.

4. *Service Discovery*

   This layer is used to centralize services into a common registry and ease the function of finding/publishing web services by storing the important data that describes these services. This data should be in a discoverable and searchable form to consumers [32]. It is handled by using Universal Description, Discovery, and Integration protocol (UDDI). UDDI is a widely recognized specification of a web service registry, which was originally created by Microsoft, IBM, and Ariba. UDDI is a technical specification used to help find and publish web services. This registry enables anyone to search in its existing data. This data is divided into three groups: White pages (for general information), Yellow pages (for general classification data), and Green pages (for technical information). UDDI normally consists of tow parts as follows:

(a) UDDI as a technical specification used to build directory of businesses and web services, where the information is stored using a specific XML format.

(b) UDDI as a business registry, which is an operational implementation of UDDI specification.

UDDI can be rendered in one of the following three ways [11]:

(a) Public UUDI: this way can be used as a resource for Internet-Based web services.

(b) Intra Enterprise UDDI: when an enterprize has a private UDDI registry to provide more control over services description.

(c) Inter Enterprise UDDI: to present the content of the sharable services between specific business partners.

The web service layered architecture as described in [3] considers two additional layers: Quality of Service (QoS) and Component. The QoS layer is in charge of security and reliability. The component layer considers the composition where different protocols can be used such as Business Process Execution Language (BPEL). Figure 2.3 shows the different layers.

2.1.3 Operations

In this section, we describe the operations of web services, which can be classified as follows [3]:

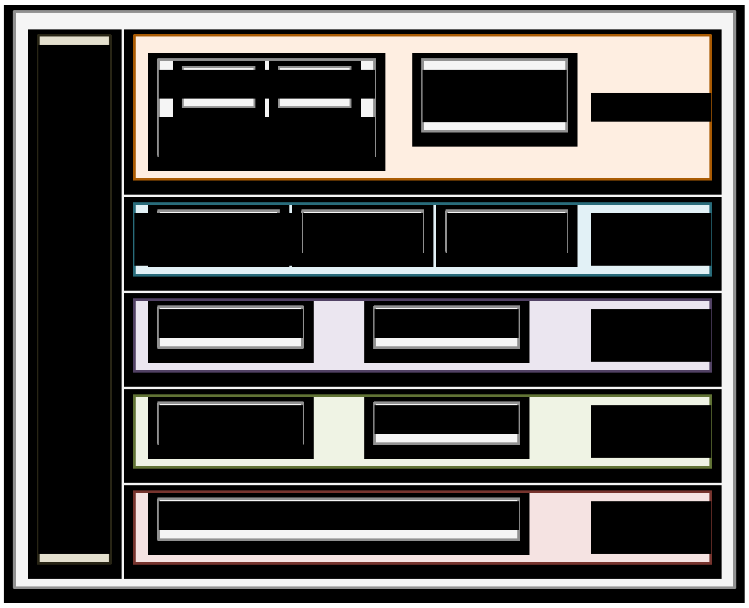


Figure 2.3 Web Service Layered Architecture

1. **Publish**: The service requestor can find the service if its description is published. Such a publication makes the service accessible.

2. **Find**: In this operation, the requestor retrieves the description of the service either directly or from the service registry.

3. **Bind**: A service needs to be invoked. This can happen via the bind operation, where the requestor initiates an interaction with the targeted service at runtime by using the binding details in the service description.

2.2 Communities of Web Services

In this section, we discuss the concept of Communities of Web Services (CWS), and we present its definition, architecture, and operations.

2.2.1 Definitions

Communities have different meanings depending on where we use them. In Oxford Dictionary, community is “a group of people living in the same place or having a particular characteristic in common”. When it comes to web services, Benatallah et al [4] specify community as an aggregation of web services with the same functionality and different non-functional, properties. Medjahed and Boubuettaya [31] use community to provide an ontological organization of web services having the same domain of interest. Medjahed and Atif use community to implement rule-based techniques for comparing context policies of web services [30]. Maamar et al. [24, 23] describe the community as something that can give an explanation of the required functionality without referring to any web service that may respond to this functionality at run-time. The definitions given by Benatallah et al., Medjahed and Bouguettaya, and Medjahed and Atif do not expose the dynamic nature of the community. Communities must have the ability to attract and retain web services, specify the membership rules, and eject web services when they do not accomplish their performance commitments [23, 5, 15, 22].

2.2.2 Architecture

Figure 2.4 describes the architecture of communities of web services and how they connect to web services' providers and UDDI registries. The components of this architecture include: the providers of web services, UDDI registries (or any type of registries like ebXML), and communities of web services. Communities are dynamic by nature. Specific scenarios and protocols are used to establish communities.

When it comes to communities, the traditional way of defining, announcing, and invoking web services is still the same, and the functionalities that UDDI registries normally offer to providers for selection are still the same (see for instance [27, 32]). The selection of web services from communities occur independently from the way these services are grouped into the community. As the master web service leads the community, the other web services are denoted as slaves and have the same functionality that labels the community. Since all web services within the same community implement the same functionality, they compete with each other to participate in ongoing composition scenarios [24, 38, 14].

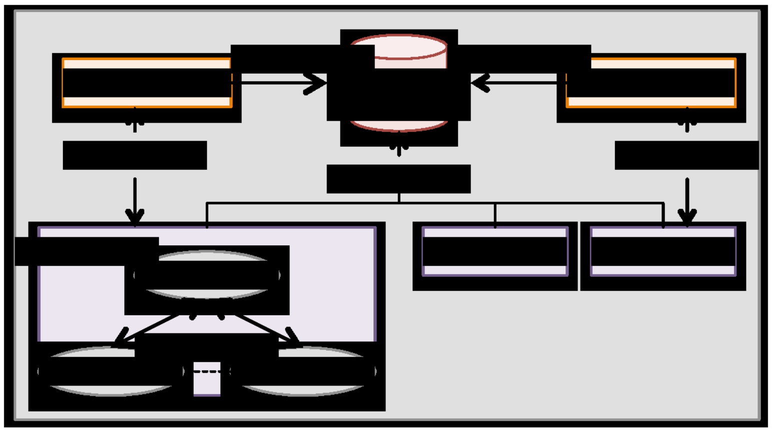


Figure 2.4 Communities of Web Services Architecture

The master component is in charge of different responsibilities. Examples of such responsibilities include attracting new web services to join the community by using rewards such as better exposure to users during the discovery, a high possibility of participation in compositions, and the possibility of collaboration in terms of replacement if a web service cannot answer the request [16, 23]. The master of the community also checks the UDDI registries to keep track of the latest changes and new web services advertisements. Furthermore, new web services can contact the master of the community they wish to join. In both cases, the master studies the credibility of the web service in terms of trust before being invited or accepted to join the community [15, 12, 24]. In service computing, trust is generally evaluated based on the feedback provided by the users on the received quality of service [9, 1, 17, 26].

Another master's responsibility is to run the Contract-Net Protocol (CN Protocol) [35] to nominate the slave web service, which will participate in an ongoing composition of web services. The master runs the CN protocol by sending a call for bids to all slave web services inside the community (CN Step 1). This call usually considers the non-functional criteria (QoS) that the user required for selecting web services. The slave web services check their capacities to meet these requirements and their ongoing commitments in other compositions before getting back to the master (CN Step 2). The slave who is interested in biding replies back to the master web service; then the master will check all bids before choosing the winner (CN Step 3). The winner is notified, and then it gets ready to execute the composition task (CNStep 3), and other slaves are notified as well(CN Step4). The winner is decided based on the requested and offered QoS and other criteria such as trust and reputation [23, 7].

In a community of web services, the master is designed in two different ways. First, by having a dedicated web service to play the master role during the entire time of being in the community. This master web service is independently developed and never participates in any composition. Second, by identifying a web service out of web services already inside the community [23].

2.2.3 Operations

The operations of the community include the community development, the attraction and retention of web services, and web services selection [6, 7, 24, 23].

a. Community Development

The main purpose of designing a community is to gather web services with similar functionality to ease the discovery process. This is a designer-driven activity which occurs in two steps. First, by defining the functionality of the community by attaching it to a specific ontology. This binding is important because providers of web services use different terminologies to explain their functionality. Using a special ontology is important when mapping the description of web service's functionality onto the description of the community's functionality. Second, by deploying the master web service to lead the community and take over its responsibilities. One of the master's responsibilities is to attract web services to join its community and to check the credentials of web services such as QoS before admitting it to the community. This checking has two advantages: it improves the security level among web services in the community, and it enhances the trustworthiness level of the master web service. Disassembling the Community of web services is also a designer-driven activity requested by the master. This oversees all the community activities such as arrival of new web service, departure of some web services, nominating slaves to be part of composition, etc.

When the master notices that the number of web services inside the community is less than a certain threshold, and when the amount of participation in composite web services over a specific period of time is less than another threshold, the community would be dismantled since it is not performing well. The designer sets both thresholds. When the web service is ejected from a community, it can join another community as long as it provides the same functionality as the new community and meets the community's requirements and policies.

b. Web Service Attraction and Retention

Attracting and retaining web services is one of the master's responsibilities. We discussed how the community of web services could disappear if the number of membered web services drops below a certain threshold. As attracting new web services drives the master of the community to regularly check the UDDI registries searching for new web services, the master can use rewards and incentives to attract the providers, such as high visibility, free security infrastructure, and high rate of collaboration [16, 19, 21]. Any changes in a web service's description could raise challenges as this service may not suit the community's objective. Hereafter, the master starts interacting with the provider of this service, and asking the provider to register its service within the community. Tow arguments are used during this interaction. First, the high rate of participation of the existing web services in composition scenarios. Second, the short response-time when handling users' requests, and the effectiveness of security techniques against malicious web services [6, 7]

The retention of web services for a long time inside a community is a good index for the following elements:

1. As web services in a community are in competition, they also show an attitude of cooperation [24, 14]. In the community of web services, peers are not subject to attack each other.

2. Providers of web services set a satisfaction rate in composite web services, and web services are, to a certain extent, satisfied with this rate [19, 21].

3. Web services are aware of peers in the community, where they can substitute them in case of failure with the minimum impact on the ongoing composite web services [16].

Web services' attraction and retention scenarios bring the attention to a third scenario, where the master of the community asks a web service to leave the community. Such request could be issued upon the assessment of the following properties:

1. When a web service has some changes in its functionality, which do not match the community's functionality.

2. The web service inside the community is unreliable when it fails to participate in compositions due to some operational problems.

3. When a web service loses its reputation, so it cannot get opportunities to participate in composite business scenarios.

c. Web Services Selection

In a community of web services, the master runs the Contract-Net Protocol (CN Protocol) to select web services to participate in compositions. CN Protocol contracting and subcontracting are between two types of agents: initiator (the master of the community) and participants (slaves inside the community). An agent can be initiator, participant, or both at any time. Mapping CN Protocol onto a community's operations happens when the user selects a community based on its functionality; the user contacts the master of this community in order to identify the appropriate slave web service to implement this functionality. Then the master runs the CN Protocol to select the appropriate web service as discussed in Section 2.2.2.

2.3 Related Work

In this section, we discuss relevant related work about evaluating web services and communities of web services. While there are significant proposals on web services' evaluation, little work exists on classifying communities of web services.

First, we focus on web services assessment. A lot of work has been done to evaluate web services from the user's perspective by mainly considering the reputation of web services [26, 15, 42, 28, 34] and its QoS [19, 22, 21, 36]. However, evaluating web services from the master's perspective, considering the overall performance and not only the reputation, has not been addressed yet.

2.3.1 Evaluating Web Services

In [37], Y. Wang and J. Vassileva noted that in addition to helping in the selection process, the reputation model aims to distinguish the good and bad web services, where in a dynamic environment it is unusual to have a web service associated to a fixed reputation level all the time. Web services can appear and disappear, come in new shape, be upgraded without previous notice, and resume operation after interruption. They may accommodate their good performance level that they provided in the past. Their proposed mechanism uses a centralized or decentralized, person or resource, global or personalized criteria. This model is focused on reputation and is mainly based on users' feedback which cannot be considered trustworthy all the time.

In [41], Xu, Martin, Powley and Zulkernine used web services' non-functional properties to assess their overall reputation. They suggested a reputation model combining an augmented UDDI registry and a manager. The service reputation is assessed by assigning scores based on the users' feedback. The reputation model has been experienced to find suitable services and the results showed high performance as long as all users feedback are honest. However, malicious users are not taken into account in this work although users' rating affect the reputation score, which may end up not to be accurate based on the malicious rating.

In [34], Ali, Majitha, Rana, and Walker noted that the discovery approaches using UDDI could not help locate web services based on their behaviors and capabilities. The proposed approach consists of the matchmaker service, composer service, discovery manager service, and reputation manager service. The matchmaker compares requested service reputation with the advertised services, then the discovery manager service requests the composer if the matchmaker is not able to do the comparison. The composer puts together the services with the same functionality, then matches the requested reputation with the advertised ones. The reputation manager computes the reputation in three different phases including: 1) reputation interrogation phase; 2) reputation rating phase; and 3) reputation computation phase. Then the reputation score is associated to the service along with the computation time. Similar to the previous work, malicious feedback are not distinguished from the honest one, which can jeopardize the overall system performance.

In [13], Jurca and Falting noted that web services should be contracted through service level agreements to specify certain QoS. Monitoring these agreements at run-time occur via a reputation mechanism. This mechanism tries to identify selfish providers who do not put the required effort into maintain the announced QoS but it still cannot prevent them from cheating.

Maximilien and Singh argued in [28] that an efficient reputation model for web services should be based on independent and trusted parties. Those parties are known as agencies. The role of an agency is to gather the right information a bout the quality of web services and present it in an appropriate format to potential users at selection time.

In [32], Ran proposed a discovery model that considers the functional and non-functional properties equally important in the discovery process. This model can co-work with the existing UDDI registries to help users find the appropriate service with the required quality. A new role, named *certifier*, is introduced to verify the advertised QoS to the user. The QoS is organized into different categories, each category needs to be associated with different metrics. However, in the proposed model the author did not provide equations to compute those metrics so that the user can obtain an accurate QoS.

In [29], Maximilien and Singh proposed a dynamic service selection using a QoS ontology integrated with an agent framework. They used agents and agencies theories to solve the dynamic selection of web services by taking into consideration the non-functional requirements when computing the QoS. They used a QoS ontology so that involved agents can match the advertised QoS with the requested one. The QoS ontology has been defined but equations to calculate them to help evaluate the quality of the web service are not provided.

2.3.2 Evaluating Communities

In [12], Elnaffar et al. proposed a reputation-based community architecture where users and providers look forward to communicate with reputable communities. Providers aim to increase their visibility to reap lucrative outcomes, and users look forward to receive high quality of service. The proposed reputation model considers the perspectives of both users and providers. The authors proposed an extended UDDI registry, which supposes to host the entries for communities in addition to the conventional entries of web services. They focused on how to structure and update the reputation system, how to maintain the reputation up-to-date, and how to make it accessible.

In [17], Khosravifar et al. proposed a reputation model based on four metrics, namely responsiveness, in-demand, satisfaction, and time recency. They analyzed the feedback logging mechanism used to provide users' feedback and provided a reliable mechanism able to manage malicious acts of agents. The controller agent which they introduced is making sure that any violation is correctly identified and involved agents are penalized. They conducted simulations showing how their model provides a system that is able to adjust the level of reputation.

Compering the performance of web service communities with single agent-based web services from trust and reputation perspectives has been investigated by Khosravifar et al. in [15]. A reputation model is used to rank communities and web services where different reputation parameters are considered. The authors discussed what would encourage a single web service join a community of web services, even if this joining could impact other parameters in a negative way. In addition, they measure the benefits that a single web service gains when it joins a community. In measuring the general service reputation, they considered two parameters, namely satisfaction and in-demand, to reflect the basic reputation assessment of a web service. Finally, they analyzed the efficiency of community of web services in comparison twith a single web service in different aspects.

CHAPTER 3. COMMUNITIES AND WEB SERVICES ASSESSMENT

3.1 Introduction

In recent years, we have seen a significant increase of interest in web services. As the number of available web services continued to grow, there was a need to make these web services cooperate and coordinate their actions within virtual structures such as communities of web services. The concept of community of web services arose to ease and improve the discovery and selection processes of web services in an open environment such as the Internet, and to successfully answer user requests as discussed in Chapter 2. Communities of web services facilitate the discovery process and help find the best service available. Communities also help find substitutes in case the web service cannot answer the request after being committed to it.

In general, single web services may fail to accept all requests from users, which may cause their overall reputation to decrease within the environment and may also lead to lose some users. In a community of web services, the community collects a set of web services having the same functionality. Many communities come online and some of them provide the same functionality, which lead to a competition between them. Users will be more interested in binding to the best community that hosts the appropriate web services, which will successfully respond to their requests. Moreover, providers are interested in cooperating with the best communities that can combine a high participation rate with a good way to advertise their services.

In communities of web services which offer the same functionality, the reputation assessment is considered as a distinguisher factor that provides the masters with good incentives to act truthfully. However, a good assessment should go beyond the traditional trust and reputation so that other performance aspects are taken into consideration.

The remainder of this chapter is organized as follows. In Section 3.2, we define the procedure of assessing web services communities from the perspectives of providers and users. In Section 3.3, we discuss the assessment of web services from the master of the community's perspective. Section 3.4 concludes the chapter.

3.2 Assessing Communities of Web Services

In this section, we define the assessment process of Community of Web Services (CWS) from two different perspectives: providers and users. This assessment follows the following three steps as shown in Figure 3.1:

1. Defining the evaluation metrics, which forms the evaluation process.
2. Clustering using the *k-means* algorithm.
3. Calculating the goodness function of each cluster's centroid (mean).

3.2.1 The Evaluation Process

This process is performed from two different perspectives. First, the users' perspective; second, the providers' perspective.

Henceforth, C_i denotes the candidate community i, which is under consideration by user U_i and provider WS_i. $|C_i|$ denotes the number of web services hosted by the community C_i.

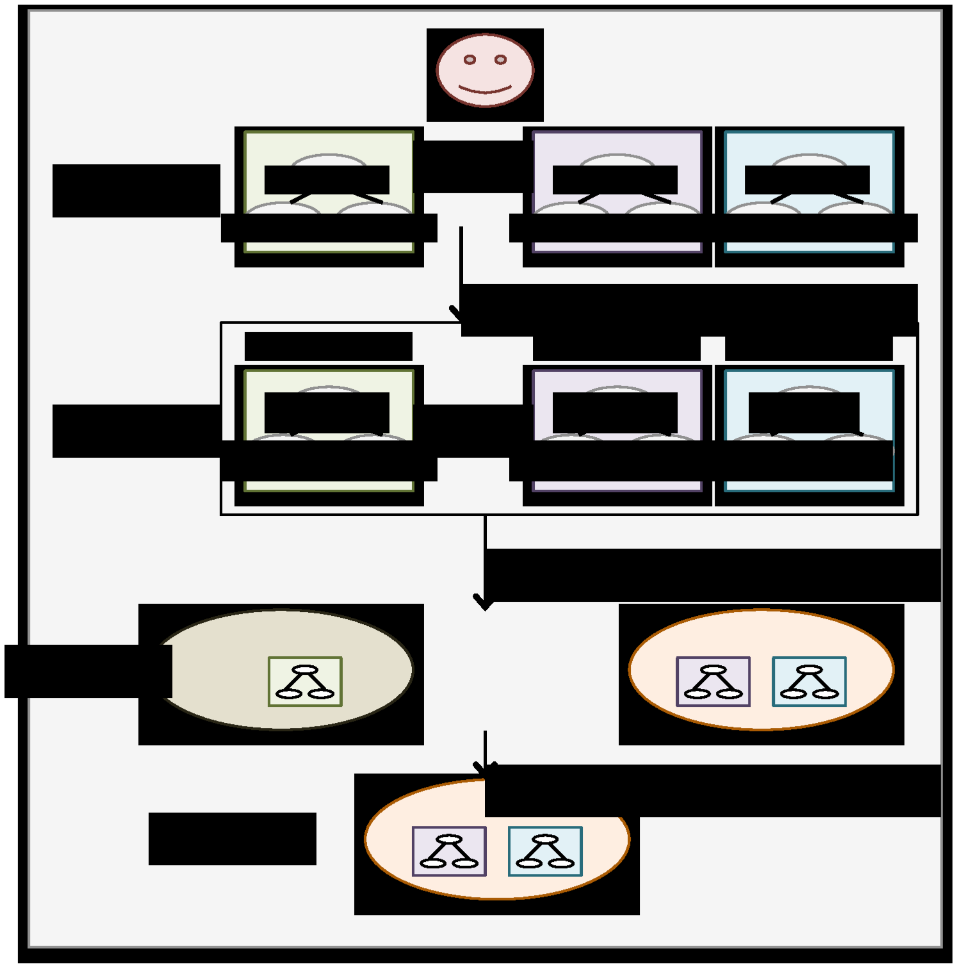


Figure 3.1 Assessing Communities of Web Services

1. *Provider's Perspective*

   As the number of communities increases, single web services would prefer to join a community to increase their visibility and benefit from the community infrastructure. From a large number of communities, providers are interested in joining the community that can help them achieve their goals in terms of having access to a large number of users. For this reason, providers need to differentiate between communities under consideration to decide which community to join. This process gives the provider a sense of how each community is performing by considering different factors defined based on the IEEE std. 1061 to evaluate the quality of each community [45].

   For each factor we proposed different metrics that are calculated for a given period of time $[t1, t2]$ using different parameters. We consider large window $[t1, t2]$ so that the considered parameters are not null. However, if the denominator of an equation is null, the metric is also considered null. In the rest of this chapter, we only consider the case where the denominators are different from 0.

   (a) *Connectivity:*

   i. Substitutability: Substitutability measures the connectivity between web services in C_i in term of substitution. The substitutability of a community C_i at a specific period of time, denoted by Sub^{C_i}, is measured by calculating the total number of failed requests and the number of substituted requests among the failed ones.

$$Sub^{C_i} = \frac{R^{C_i}_{substituted}}{R^{C_i}_{failed}}$$

   where $R^{C_i}_{substituted}$ is the total number of substituted fail requests at time

$[t1, t2]$, and $R^{C_i}_{failed}$ refers to the total number of failed requests at the same period of time.

ii. Internal Connection: To measure the internal connection we gather web services that can substitute each other into groups, and the lowest number of groups reflects the highest community connectivity. The concept of group means a collection of at least two web services, which are strongly connected to one another so that each web service can substitute the other. In fact, web services are gathered in one group based on the similarity of their non-functional proprieties where each one can substitute the other in case of failure or compose with each other to respond to complex requests. A group is therefore a strongly connected graph where nodes are web services and edges mean substitutability or composition links. The internal connection of a community is computed as follows:

$$IntCon^{C_i} = \frac{|G|^{C_i}}{|C_i|}$$

where $|G|^{C_i}$ is the number of disjoint groups inside the community, and $|C_i|$ is the number of web services hosted by the community, which is assumed to be non empty during the time interval $[t1, t2]$.
Let us assume we have three communities C_a, C_b, and C_i with 10 web services in each as shown in Figure 3.2. The community C_i has the highest connectivity between its web services in term of substitution/composition. The community C_a has the lowest connectivity because it has the most number of groups. In fact, the community C_i shows a strongly homogeneous community compared to C_a and C_b.

iii. External Connection (for collaboration): External connection measures the ability of community C_i to connect with external web services (outside the community) in terms of collaboration and composition. It is

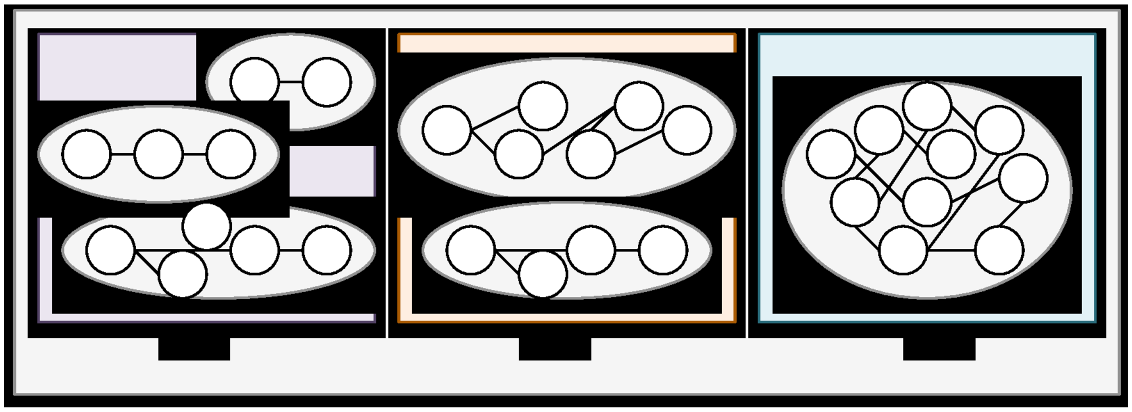


Figure 3.2 Community's Internal Connection Using The Concept of Group

calculated by considering the total number of complex requests that the community C_i received during a specific period of time, and the total number of those complex requests that are successfully responded during the same period of time.

$$ExtCon^{C_i} = \frac{R^{C_i}_{Crespond}}{R^{C_i}_{Crecived}}$$

where $R^{C_i}_{Crespond}$ is the total number of responses to complex requests and $R^{C_i}_{Crecived}$ is the total number of complex requests sent to the community C_i during $[t1, t2]$.

This metric is useful in measuring the performance of the community and comparing different communities, which could be very important to the providers of web services to help them choose the most connective community to join.

(b) *Productivity:* Productivity is the factor that determines how productive the community is in comparison to other communities. It is measured by computing the total number of requests the community receives and the number of web services in this community at a specific period of time as follows:

$$Pro^{Ci} = \frac{Req^{C_i}}{|C_i|}$$

where Req^{C_i} is the total number of requests C_i received during $[t1, t2]$, and $|C_i|$ denotes the number of web services in C_i during the same period of time.

(c) *Efficiency:*

i. *Responsiveness:* Responsiveness is the metric that shows how fast the master is in nominating the best slave web service from the community, which can handle the user's request with respect to required quality of service, how fast the slave web service is in responding to the requested service, and how fast the substitution is in case of failure. Let C_i be the community under evaluation by user U_j. The Responsiveness metric is measured by computing the total number of respond time (the time a master takes to complete the nomination and the time a slave takes to respond) to the requests R_k of the user U_j, and the extra time in case of substitution during the considered period of time.

$$Res_{U_j}^{C_i} = \frac{1}{n}\sum_{k=1}^{n}(Res_{U_j}^{C_i,R_k} + Sub_{U_j}^{C_i,R_k})$$

where $Res_{U_j}^{C_i}$ is the response time community C_i takes to answer all user U_j requests during the interval $[t1, t2]$, n is the number of requests received from Uj to this community during this period of time, and $Sub_{C_i}^{U_j,R_k}$ is the time needed to substitute the slave responsible for the request R_k by another one. If no substitution is happening, then $Sub_{U_j}^{C_i,R_k} = 0$.

(d) *Satisfaction:* This factor measures the users' opinion of a community with which they recently interfaced, and considers the provided service during a specific period of time. This metric is computed by observing the positive feedback from user U_j, $PosFed_{C_i}^{U_j}$ and the total feedbacks received from the same user during the time window $[t1, t2]$ $TotFed_{C_i}^{U_j}$.

$$Sat_{U_j}^{C_i} = \frac{PosFed_{U_j}^{C_i}}{TotFed_{U_j}^{C_i}}$$

The overall users satisfaction is given by the following equation where u is the number of users interacted with the community between $t1$ and $t2$.

$$Sat^{C_i} = \frac{\sum_{k=1}^{u} Sat_{U_k}^{C_i}}{u}$$

(e) *Availability:* This metric measures the availability rate of the community so that it can receive requests from the users. It gives the provider an indication of how many requests the community can receive during a given period of time in percentage. Each web service ws_k within the community comes with its own availability Avl^{C_i,ws_k}, which can be monitored during the window $[t1, t2]$. The community availability is then computed as the average availability of the members as follows:

$$Avl^{C_i} = \frac{\sum_{k=1}^{|C_i|} Avl_{ws_k}^{C_i}}{|C_i|}$$

(f) *Popularity:* To measure the popularity of the community we consider two factors: 1) the in-demand, which reflects the popularity from the number of requests directed to the community; and 2) the number of web services hosted in the community. The popularity of the community is a significant property to attract new web services to sign-up with the community. Joining communities with high popularity means having access to a high number of requests and benefiting from high rate of substitutability from peers. We compute popularity as follows:

$$Pop^{C_i} = \mu_1 InDe^{C_i} + \mu_2 \frac{|C_i|}{\sum_{k=1}^{M} |C_k|}$$

where M is the number of considered communities, $\mu_1 + \mu_2 = 1$, and μ_1 and μ_2 are fixed by the evaluator depending on his preferences and

$$InDe^{C_i} = \frac{R^{C_i}_{received}}{\sum_{k=1}^{M} R^{C_k}_{received}}$$

where $R^{C_i}_{received}$ is the number of requests received by C_i during $[t1, t2]$, and M is the number of communities under consideration.

(g) *Objectiveness*: This factor measures the objectiveness of the master of the community when selecting which web service to accept and which web service to select for an ongoing composition.

i. *Selectivity:* This metric measures the selectivity rate of the community in terms of acceptance rate with regard to membership requests sent by web services. Selective communities is an indication of the quality standard used by those communities when it comes to evaluating slave web services. Selectivity is computed considering the number of total membership requests TMR^{C_i} and the number of accepted requests AMR^{C_i}:

$$Sel^{C_i} = \frac{AMR^{C_i}}{TMR^{C_i}}$$

ii. *Expected Requests:* This metric is used by the provider of a web service to measure the web service's probability of being selected by the master and having some requests. For a typical community C_i, the service requests is a discreet event that can be modelled as random variables that follow a *Poisson* distribution. Poisson distribution is a discrete probability distribution that expresses the probability of a number of events occurring in a period of time. The expected number of requests is referred to as the Poisson rate λ. However, this distribution assumes that the events occur independently of the time since the last event. This assumption means that requests are independent of any other factor such as the web service reputation and QoS, which changes with time. To better model the dynamics of requests that depend on other factors, we use the *non-homogeneous Poisson process* [33], which is a Poisson process with dynamic rate $\lambda^{C_i}(x)$ denoting the mean number of requests received by community C_i at time moment x. The rate $\lambda^{C_i}(x)$ is a function of time and the arrival of requests to the community C_i during the time window $[t1, t2]$ ($m^{C_i}([t1, t2])$) can be formulated as a non-homogeneous Poisson process as follows:

$$m^{C_i}([t1, t2]) = \int_{t1}^{t2} \lambda^{C_i}(x)\,\mathrm{d}x$$

Thus, the probability of having exactly n requests is given as follows:

$$p(m^{C_i}([t1, t2]) = n) = \frac{m^{C_i}([t1, t2])^n}{e^{m^{C_i}([t1,t2])} \times n!}$$

The number of requests a web service ws_k can expect, for a given n, is given by:

$$E_{ws_k}^{C_i[t1,t2]}(n) = \frac{p(m^{C_i}([t1, t2]) = n) \times n}{|C_i|}$$

iii. *fairness:* This metric is used to reflect the fairness rate of the master in selecting slave web services to answer users' requests. It is computed using the standard deviation of the number of requests directed to each web service within the community during the considered interval of time.

$$Fair^{C_i} = \begin{cases} \frac{1}{\sqrt{\frac{1}{|C_i|}\sum_{l=1}^{|C_i|}(x_l-\mu)^2}} & \text{if } \sqrt{\frac{1}{|C_i|}\sum_{l=1}^{|C_i|}(x_l-\mu)^2} \neq 0 \\ FMAX & \text{otherwise} \end{cases}$$

where x_l is the number of requests given by the master to the community member ws_l during the interval $[t1, t2]$; $\mu = \frac{1}{|C_i|}\sum_{l=1}^{|C_i|} x_l$ is the population average during the same period; and $FMAX$ if the fixed maximum number that the fairness can take.

2. *User's Perspective*

As perceived by users, a community assessment would help them select communities hosting web services having the capability to meet their quality expectations. The performance metrics that can play a role in the evaluation process as seen by users are in general similar to the ones considered by the providers as both of them are seeking good quality communities. Users generally focus on the "health" of the community and its overall reputation. This can be reflected by the following parameters defined in the previous subsection: *connectivity, responsiveness, in-demand, satisfaction, productivity, popularity, and availability.* Users are less interested in some internal issues such as fairness. However, knowing the expected number of requests is a significant indicator of the reputation evolution. On the one hand, if the number of expected requests for the future is high, this reflects a good management, but if the availability is low, the user will probably select another community providing high availability. On the other hand, if the expected number is low, this means high availability, but less popularity. Users should then consider a tradeoff between these two metrics. Assuming the availability is fixed, we formalize this problem as follows:

$$C^* = \underset{C \in \mathcal{C}}{\text{argmax}}(\chi(\sum_{n=1}^{MAXR} p(m^C([t1, t2]) = n) \times n) + (1 - \chi)Avl^C)$$

where $\mathcal{C}$ is the set of all communities under consideration, $MAXR$ is the maximum number of requests a community can receive during the interval $[t1, t2]$, and χ is a coefficient stated by the evaluator.

3.2.2 Clustering with K-*Means*

After the evaluation process, we analyze the communities by clustering all the evaluated communities into groups in order to further facilitate the selection process. Clustering is the task of classifying a set of data into groups where each cluster contains objects similar in some sense to each other. The members of a cluster have some common characteristics compared to members of other clusters. The objective is to find a structure in a collection of data. Clustering algorithms are used extensively not only to organize and categorize data, but also for data compression and model construction. Clustering can be done using different algorithms, which differ significantly in their notation of what constitutes a cluster and how to efficiently find them. We choose for this work a centroid model where the cluster is represented by a single mean vector (k-*means*). However, since the ordering of the input data set does not matter and our problem is expectation-maximization oriented, k-*means* clustering is the most suitable [25].

The k-*means* algorithm is a simple iterative method to divide a given data set into a number of clusters specified by users. It has been proposed first by Lioyd in 1957 and published later in [20]. This algorithm operates on a set of d-dimensional vectors, then chooses k points as the initial k cluster representatives or ”centroids”. The way of selecting these initial points is random from the dataset. Then the iteration between two steps takes place. First, data assignment occurs where each data point is assigned to its closest centroid. Second, relocation of ”means” where each centroid is relocated to the center ”mean ” of all data in this cluster. The Algorithm ends when assignment no longer changes.

3.2.3 Goodness Function

Once the clusters are identified, we use a goodness function to distinguish the best cluster. We calculate the goodness function of each cluster centroid (mean) to decide from which cluster the community will be chosen. To calculate the goodness function, we weight each parameter according to the evaluator's preferences. For example, if the satisfaction is important for the user, he will assign a high weight to this metric. The goodness function is as follows:

$$GF = \sum_{i=1}^{10} w_i * par_i$$

where $\sum_{i=1}^{10} w_i = 1$ and each par_i is a metric previously defined.

3.3 Assessing Web Services

This section presents the assessment of web services from the master perspective. Such an assessment takes place when the master of the community decides to invite a new web service to join its community. First, the master usually checks the available registries such as UDDI to get information about new web services. Then, the master starts the assessment process so only the appropriate web services can be invited. This assessment is done using the same steps as discussed in Section 3.2, namely 1) defining evaluation metrics, which forms the evaluation process; 2) clustering web services; and 3) calculating the goodness function. Steps 2 and 3 are very similar to the ones defined in Section 3.2. Here we only focus on step 1.

3.3.1 The Evaluation Process

In this section, we present some metrics, which can be used by the master of the community to decide which web service to invite. This evaluation is divided into two different categories. The first category evaluates new web services. The second category is meant to evaluate existing web services having participated to previous transactions.

1. Evaluating new web services

   Before inviting a new web service, the master checks the available registries for web services providing the same functionality as the community it belongs to. This is done by parsing the web service description language (WSDL) file. When it comes to the evaluation of a new web service that has not participated in any composition or transaction, one of two scenarios might occur. The first scenario is when the new web service is provided by a specific provider, which is already known to the master from other services it provides. In this case, the master gives the new web service weight based on the existing services inside the community belonging to this provider. The second scenario is when the master evaluates a completely new web service. In this case, we follow a similar strategy to what has been proposed to solve the bootstrapping problem. Bootstrapping is the problem of assigning an initial value to a new entity [42]. There are few solutions proposed in the literature to solve this problem, and any solution can be adapted to web services.

One particular strategy used to solve the bootstrapping problem is the adaptive strategy. In the adaptive strategy the new entity is assigned a trust value, which depends on the rate of maliciousness in the system, which is dynamic in the sense it changes over time. The new entity is assigned a high initial value when the maliciousness rate is low and, on the other hand, if that rate is high the initial value is low. Another strategy is the default value strategy, where the new entity is assigned a default value. This value is generally considered as a threshold and under this value the entity is considered as malicious. The disadvantage of this strategy is that depending on this value it can either give advantage existing entities or new entities. If the initial value is high the existing nodes are negatively affected as the new entities have a higher value which will encourage malicious entities to leave and join again to increase their values.

Beyond trust, the following factors can be used to evaluate web services, especially new arrivals:

(a) *Cost:* This metric is used to evaluate the web service from an economic point of view, which can be helpful for the community when deciding to invite a new web service. We define this metric from the following parameters.

i. Execution Price: Is the amount that should be paid to the provider from the user in order to use the service.

ii. Penalty Price: Is the price that the user should pay in case of any cancelation.

iii. Compensation Price: Is the rate that should be paid to the user in case of any delay or failure from the provider.

(b) *Security:* The purpose of this parameter is to take into account the different security aspects for web services, as the service provider might apply different levels and techniques of security according to the service requestor.

As the number of web services increases, the concern about the security of services transferred over the public internet increases as well. Some of the security properties the master should check are presented as follows.

i. Authentication: Is the process of determining if the user or provider is truthful about his identity. It is implemented using different methods such as username and password, certificates, or, in an advance way, by using a more sophisticated system such as Kerberos, a computer network authentication protocol that allows nodes communicating over a non-secure network to prove their identity to one another in a secure manner requiring a securely encrypted message to be transferred. In password-based authentication, for example, the provider should use powerful and strong passwords. This technique cannot be sufficient alone and it should be combined with other authentication and authorization processes such as certificates, Public Key Infrastructure(PKI), Kerberos, Remote Authentication Dial-in User Service (RADIUS), and Lightweight Directory Access

Protocol (LDAP).

ii. Authorization: Is the ability of web services to control access to resources where each service can be accessed only by authorized users or providers. Web services should set policies to determine the access rights and when those rights are given.

iii. Encryption: Is the process that applies to a text message (plaintext) to make it unreadable (cipher) to others except those who have the secret key and know how to decrypt it. Encryption is used to protect important data such as files and insure their confidentiality It is also used to protect information while transferring it through a network, public internet, mobile telephones and other devices. In web services, important data that should be encrypted to protect the secrecy and privacy. The master should evaluate the type and strength of technique this web service uses to encrypt and decrypt data before inviting it to see whether or not it satisfies the security levels required in the community.

iv. Non-Reputation: Is the ability to ensure that the message has been sent and received by the users or providers that claim to send or receive the message. With non-reputation, the users cannot later deny having sent or received the message, as it can be obtained by using digital signatures, confirmation services, and timestamps.

v. Confidentiality: Confidentiality of a web service measures the ability of the web service to protect its data from unauthorized access.

2. Evaluating existing web services

To evaluate an existing web service which has previous transactions, we follow the same steps as discussed in section 3.2. First, the evaluation process; second, clustering using *k-means*; and third, calculating the goodness function of each cluster's centroid (mean). The master of the community will use the following metrics to help in the evaluation process to determine which web service has performed well, which web service come with a good yield to the community, and which web service is there whenever it is needed to participate in a composition or to replace a failed web service.

1. *Performance:* In order to measure the performance, we compute different metrics such as response time, throughput, and latency. A good performance means that web services can execute and complete the requested function quickly and it can minimize any delay in responding to users' requests with increasing loads.

   (a) Response Time: Is the mean elapse time from the moment of receiving the request until providing the corresponding response.

   $$Res = \frac{\sum_{i=1}^{n} T_{respond}(i) - T_{request}(i)}{n}$$

   Where $T_{respond}(i)$ is the time when the user receives the respond for the request i, $T_{request}(i)$ is the time the web service receives the request i, and n is the total number of requests received by the web service. We calculate the average of the response time for more than one request because the response time of one request does not represent the typical response time for the web service. We can also calculate the standard deviation in the usual way.

   (b) Throughput: Is the rate of successfully completed requests for the given period of time $[t1, t2]$.

   $$Thro = \frac{R_{responded}}{n}$$

   Where $R_{responded}$ is the number of successfully responded requests and n the total number of requests.

   (c) Latency: Is the time delay that might happen when sending a request and receiving the respond. As the latency metric increases, the chance of satisfying more users' requests decreases, which negatively affects the web service overall evaluation. It is measured by considering the difference between the experienced response time Res and the promised one $PRes$.

$$Lat = Res - PRes$$

(d) *Availability:* Availability of a web service is the probability that it can respond to a user's request, which means that the request successfully reaches the service. This metric can be used by the master of the community to know the availability rate of a particular web service when it comes to consider this web service in the nomination process for ongoing compositions, and when to call this web service to replace another service. The availability during the interval of time $[t1, t2]$ can be computed by the following formula:

$$Avl = \frac{uptime}{uptime + downtime}$$

Where *downtime* refers to the period of time when the web service has failed to respond to users requests, and *uptime* is the total time the service has been ready.

2. *Maintainability:*

(a) *Stability:* Is the ability of the web service to remain stable without major changes that can affect its performance under different circumstances, or, in other way, it is the ability of the web service to cope with any changes that may occur without affecting its performance. It is computed as follows:

$$Sta = \frac{NC}{t2 - t1}$$

Where NC is the number of major changes the web service undertook during the window time $[t1, t2]$. After noticing the changes, the master has to re-evaluate the web service to see if this changes affect its performance. This can help the master know which web service performs optimally under different circumstances. Those changes could affect web service's resources, the way that the web service handles the requests, or the quality of service provided by this web service.

(b) *Scalability:* Is the ability of web service to provide the same quality of service under rising request demands, or it is the ability to handle the growing amount of requests without a diminishment in the quality of service. It can be calculated by computing the throughput of the web service at a round time trip.

(c) *Robustness:* it is the ability of the web service to perform correctly in the presence of errors or any slight disturbance.

(d) *Replaceability :* is the ability of the web service to replace others in case of failure after the call from the master of the community. It can be measured by considering the number of successful replacements this web service made in the elapsed time, and the number of received calls from the master to replace other web services.

$$Rep = \frac{R_{replaced}}{R_{replacedcall}}$$

Where $R_{replacedcall}$ is the total number of calls this web service has received to replace another service, and $R_{replaced}$ is the number of successful replacements.

3. *Reliability:* Is the ability of the web service to perform its task correctly and repeatedly for a specific period of time under some stated conditions. The master of the community is interested to keep more reliable web services within the community to satisfy users and community requirements. Mathematically, reliability is the probability of the web service operating for a certain amount of time $[t1, t2]$ without failure. It is evaluated as usual as follows:

$$R(t1, t2) = 1 - \int_{t1}^{t2} f(s)ds$$

Where f is the distribution that models the failure behaviour of a web service, and it is generally assumed to be Exponential:

$$f(t) = \lambda e^{-\lambda t}$$

4. *Accessibility:* It measures the ability of web services to answer as many users' requests as possible. The web service might be available but not accessible due to high demands. This metric can give the master information about the yield of the web service when it is available, which is a good way to distinguish the active web services inside the community. It is computed as follows:

$$Acs = \frac{R_{responded}}{avl}$$

   Where $R_{responded}$ is the number of successfully completed requests.

5. *Efficiency:*

   (a) *Capacity:* Is the maximum number of users' requests this web service can handle. When the web service works beyond its capacity, some of its quality attributes will be affected, such as availability and reliability. This metric is declared by the web service in its WSDL file and helps the master of the community when nominating web services to participate in compositions, or when calling web service to replace failed ones.

6. *Out Degree:* Is used to measure the popularity of web service among the other web services.

(a) Direct Out Degree: This metric represent the direct composition or substitution of the web service with regard to other peers. This relation normally happens within the same community. We consider the wight of the edge between two web services (W_{ws_i,ws_j}), which reflects the successful number of compositions and substitutions over the interval $[t1,t2]$, and the time recency of this relation ($Tr_{ws_i}^{ws_j}$). This metric computes in fact the degree of cooperativeness of the web service.

$$Doutdegree_{ws_i} = \sum_{ws_j \in \mathcal{C}} W_{ws_i,ws_j} * Tr_{ws_i}^{ws_j}$$

(b) Indirect Out Degree: It is used for evaluating the popularity of a web service outside its community, when the relation between web services is established in more than one step.

$$Ioutdegree_{ws_i}^{ws_j} = \prod_{k=i}^{j-1} W_{ws_k,ws_{k+1}} * Tr_{ws_k}^{ws_{k+1}}$$

Let $\mathcal{WS}$ be the set of all considered web services. Then we have:

$$Ioutdegree_{ws_i} = \sum_{ws_j \in \mathcal{WS}} Ioutdegree_{ws_i}^{ws_j}$$

7. *Effectiveness:*

(a) *Accuracy:* It refers to the ability of the web service to perform its functions in a correct way without errors or mistakes. We simply compute it by counting the number of correct responses $R_{cresponded}$ over the total number of provided responses $R_{responded}$.

$$Acu = \frac{R_{cresponded}}{R_{responded}}$$

(b) *Contributability:* Is to compute the contribution of the web service to the community in terms of throughput. It measured by calculating the throughput of the web service and the total throughput of the community as follows:

$$Con_{ws_i} = \frac{thro_{ws_i}}{\sum_{ws_j \in \mathcal{C}} thro_{ws_j}}$$

3.4 Conclusion

In this chapter, we discussed different steps to assess both the communities of web services and the web services themselves. First, we presented the evaluation process where we used different factors for each assessment. Then, we used the *k-means* algorithm to cluster the evaluated communities and web services. Finally, we defined the goodness function to distinguish the best cluster. Implementation results of the clustering algorithm and goodness function are reported in the next chapter.

CHAPTER 4. IMPLEMENTATION

In this chapter, we discuss the settings of our simulation experiments that aim to show the effectiveness of our assessment approach. Then, we analyze the obtained results.

4.1 Simulation Environment

Our simulated system consists of two different parts. The first part is about the assessment of individual web services to help the master decide which web service to invite to the community. To accomplish this, the master checks the UDDI registries to get the descriptions of potential new members, then the assessment process begins to evaluate the nominating services. The second part is about the assessment of communities, which begins when the provider of the web service intends to join a community or when the user starts looking for a community.

This simulation is written with Java in the Eclipse environment. Most of web services (QoS) values are taken from a dataset of 2507 real web services with 9 proprieties for each web service [2]. We have simulated 100 web services and 30 communities with different number of clusters. The experiment's parameters are shown in Table 4.1. The "Change Value" column represents the values that might change during the experiments based on the evaluator's preferences.

To analyze the effectiveness of our proposed method, we compare our three-step assessment and selection process with two different approaches. The idea of the first approach, called *random selection*, is to select web services and communities randomly

Table 4.1 Experiments Parameters

Parameters	**Default value**	**Change value**
Number of web services	100	No
Number of communities	30	No
Number of clusters	3 to 20	Yes
Parameter's wight	Equally	Yes

without any pre-processing. The key element of the second approach, called *reputation-based selection*, is to use web services and communities reputation during the selection process. This approach has been discussed in [13] and [15]. Unlike our approach, reputation-based approach does not consider all the relevant web services and communities parameters, but only focuses on in-demand, satisfaction, execution and response times, and contributability. For a fair comparison, only those parameters are evaluated once the different approaches identify the selected community or web service. Moreover, we considered different k during the implementation to observe the impact of the number of clusters on the performance of the selected communities and web services.

As mentioned in Chapter 3, we start our assessment by computing the metrics using the equations we have proposed. For each evaluated web service and community, the outcome is a vector of n dimension, where n is the number of computed metrics. The obtained vectors are then used as input to the *k-means* algorithm, which we implement to cluster the evaluated communities and web services. Finally, the goodness function is computed to determine the best cluster that mostly suits the requestor's needs. To make the comparison meaningful, we assign to each chosen component (web service or community), based on each method, the same number of tasks, and we observe the performance of each one in the same period of time using the aforementioned parameters.

4.2 Results and Analysis

The goal of this analysis is to demonstrate that by using our assessment process, web services and communities that best meet the requirements of communities, users, and providers have more chance to be selected. To this end, and in order to compare the three investigated methods against a benchmark, we defined three different classes: high performing class, medium performing class, and low performing class. For each class we set a range for each parameter as illustrated in Tables 4.2 and 4.3. In these tables, except response time, latency, and responsiveness, which are given in s, stability, which is given in terms of the number of changes during the considered period of time, and direct and indirect out degrees, which are given in terms of connection weights and time recency, all the values are given as rates. For instance, 0.7 for availability means the web service is available 70% of the time. Thus, once the web service or community is being identified using one of the three methods, we identify the class that fits it more based on the majority of the metrics. The component (web service or community) is then considered belonging to the class to which most of its parameters belong. By doing so, we can measure the quality of each selection process.

Table 4.2 Web Services' Classes: Benchmark

Feature	**High Performing**	**Medium Performing**	**Low Performing**
Availability	0.7-1.0	0.4-0.6	0.1-0.3
Response Time	0.1-0.3	0.4-0.6	0.7-1.0
Throughput	0.7-1.0	0.4-0.6	0.1-0.3
Latency	0.1-0.3	0.4-0.6	0.7-1.0
Stability	1-3	4-6	7-10
Reliability	0.7-1.0	0.4-0.6	0.1-0.3
Accessibility	0.7-1.0	0.4-0.6	0.1-0.3
Direct Out Degree	0.7-1.0	0.4-0.6	0.1-0.3
Indirect Out Degree	0.7-1.0	0.6-0.4	0.3-0.1
Accuracy	0.7-1.0	0.4-0.6	0.1-0.3
Reblaceability	0.7-1.0	0.4-0.6	0.1-0.3
Contributability	0.7-1.0	0.4-0.6	0.1-0.3

Table 4.3 Communities' Classes: Benchmark

Feature	High Performing	Medium Performing	Low Performing
Substitutability	0.7-1.0	0.4-0.6	0.1-0.3
Internal Connection	0.7-1.0	0.4-0.6	0.1-0.3
External Connection	0.7-1.0	0.4-0.6	0.1-0.3
Productivity	0.7-1.0	0.4-0.6	0.1-0.3
Responsivness	0.1-0.3	0.4-0.6	0.7-1.0
InDemand	0.7-1.0	0.4-0.6	0.1-0.3
Satisfaction	0.7-1.0	0.4-0.6	0.1-0.3
Availability	0.7-1.0	0.4-0.6	0.1-0.3
Popularity	0.7-1.0	0.4-0.6	0.1-0.3
Selectivity	0.7-1.0	0.4-0.6	0.1-0.3

4.2.1 Communities Assessment

We analyze the communities' assessment by considering the providers' and users' perspectives. The goal is to demonstrate that using the proposed method gives a high probability of selecting a community that best meets the provider's or user's requirements.

1. Provider's Perspective

   Providers are looking for the best community to join to increase the number of requests. Thus, we first compare the number of received requests by the communities (which is used to compute the in-demand) after using the three selection methods: our evaluation and clustering-based assessment process, random selection, and reputation-based selection. Figure 4.1 compares the received requests using these three methods with different number of clusters. Our method shows better performance than the two other methods over time specially when we chose a small k ($k \in [3, 10]$) The simulation also shows that the community selected using our approach belongs to the high performing class, while the ones selected by the reputation-based approach and the random selection approach belong respectively to medium and low performing classes.

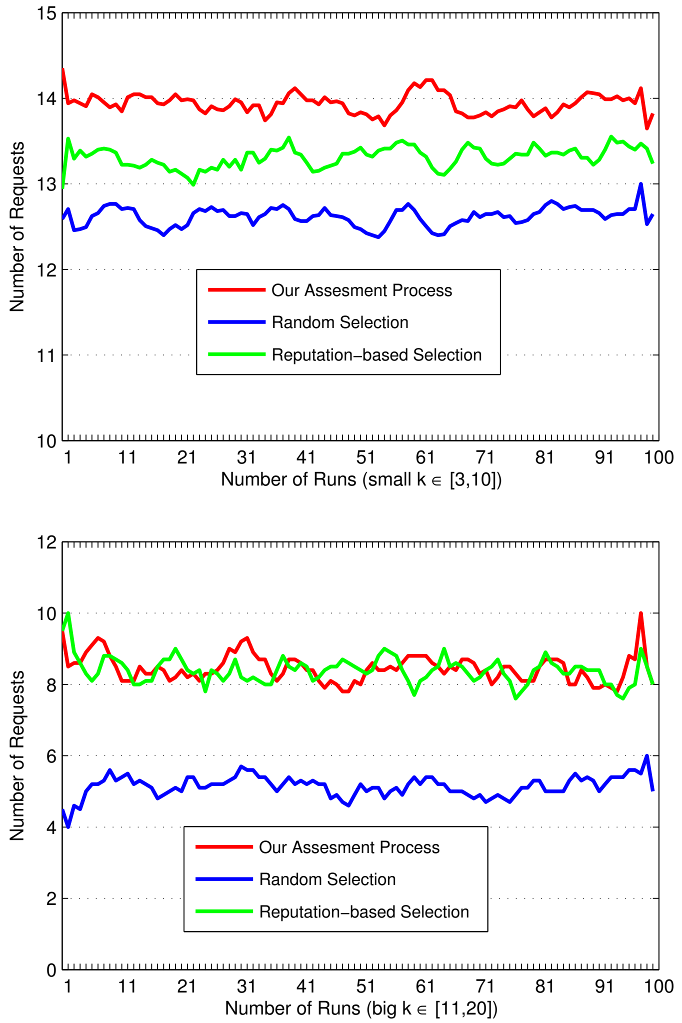


Figure 4.1 Comparison of Communities Received Requests

2. User's Perspective

   Users are asked to interact with the masters of communities to request services. Looking for the appropriate community leads the user to evaluate different communities to choose the best one. Users are more interested in requesting services from communities that can satisfy their QoS, with minimum execution time.

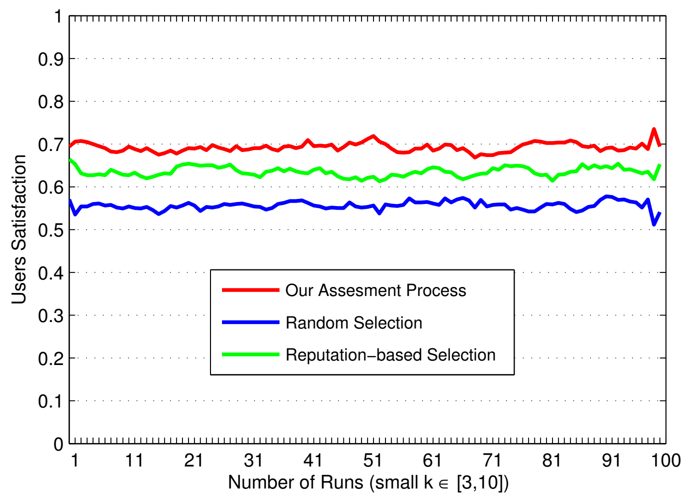
   An interesting observation from the users' perspective is made when we compare the users' satisfaction of each community chosen based on the three different methods. We plot this comparison in Figure 4.2. We notice that users are more satisfied with the community selected using our method (with small k) than with the two other communities.

   We also compare the execution time each community takes to answer the user's request (nominating the best slave web service, the response time, and the substitute time in case of failure) in Figure 4.3. We notice form this figure that the community chosen based on the proposed method is taking less time to execute users' requests, while the community chosen based on the random selection is showing a significantly high execution time.

4.2.2 Web Services Assessment

In this section, we present the analysis of the web service assessment from the master's point of view. When the master of the community decides to invite new web services, the evaluation of all potential services should be done before the selection. The master of the communities is supposed to invite web services that will have a positive contribution to the community and satisfy the users' QoS in a short execution time.

From this perspective, we conducted a comparison of the response time of the chosen web services using the three considered approaches. We plot the resulting graph in Figure 4.4, where we observe that the response time of web services chosen based on our method



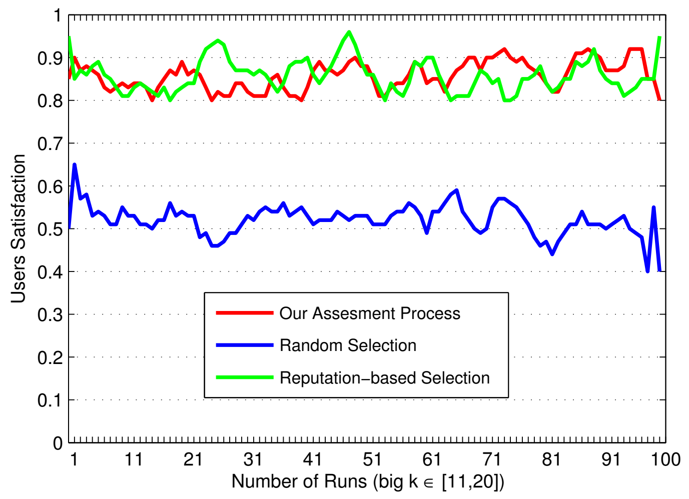
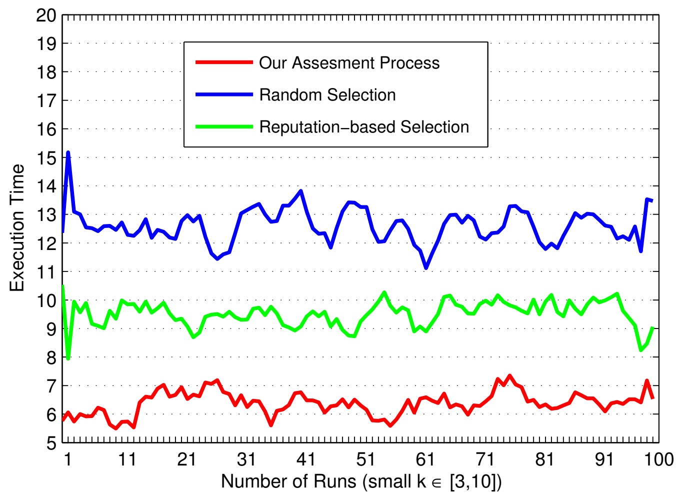


Figure 4.2 Comparison of Users' Satisfaction about Communities



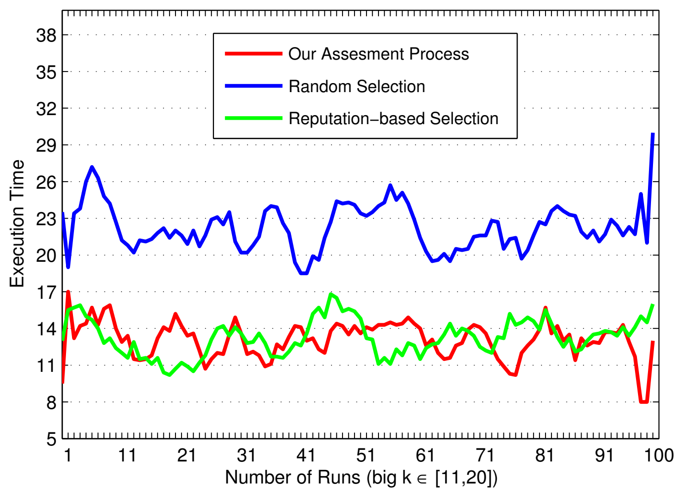
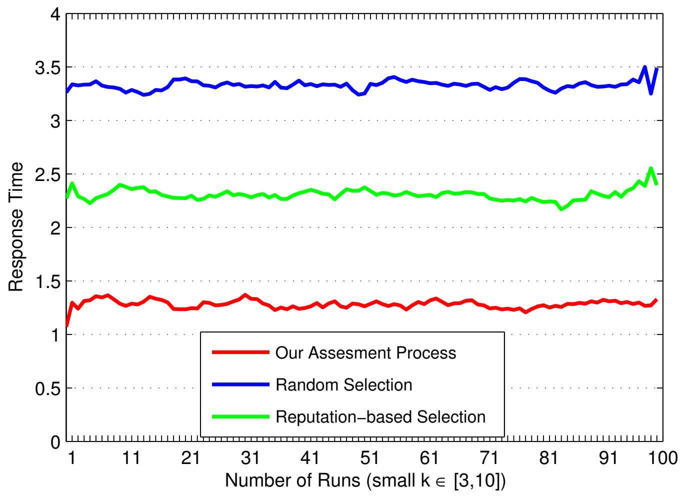


Figure 4.3 Comparison of Communities Execution Time

is significantly lower than the other web services.



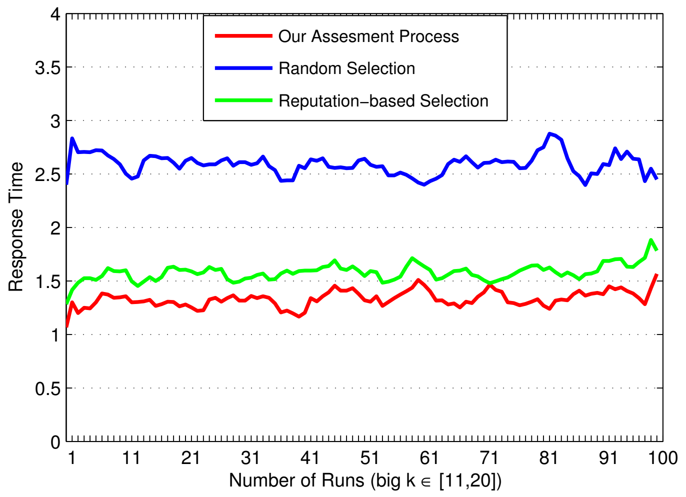
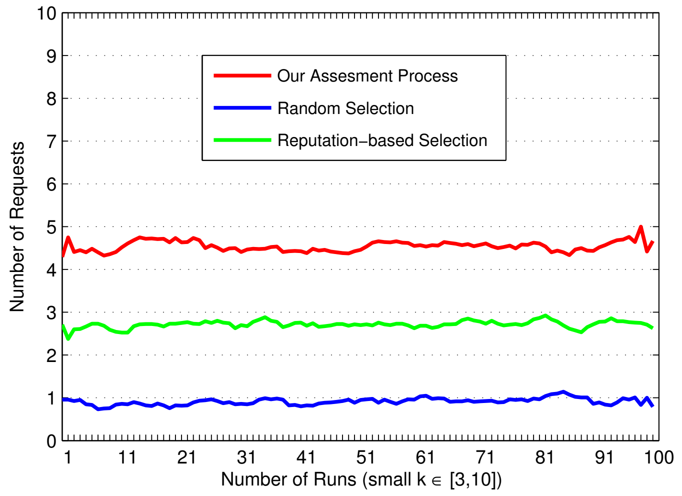


Figure 4.4 Comparison of Web Services Response Time

Furthermore, we compared the different web services from the number of requests received, which reflect the popularity of the web service. In Figure 4.5, we plot the simulation results with regard to this property. Again, the web services selected based on our approach shows better performance.

Another significant metric to consider is the contribution of the web service to its community as computed in Chapter 3. We report the contribution comparison in Figure



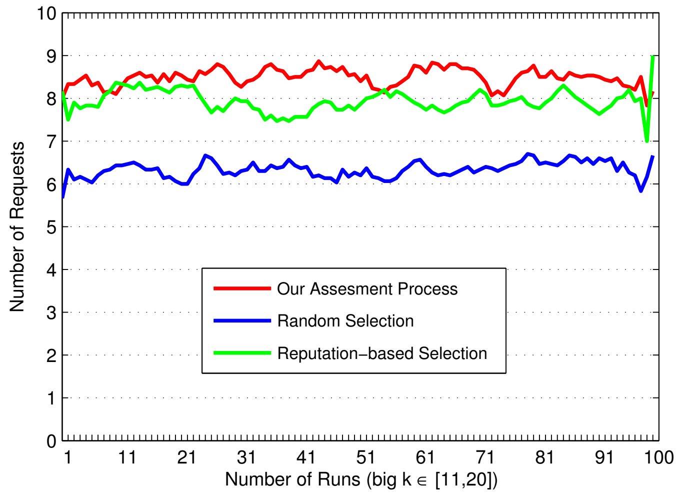


Figure 4.5 Comparison of Web Services Number of Requests

4.6. Unlike the random-based and reputation-based approaches, our assessment and clustering-based approach considers this factor when it comes to selecting web services for composition purposes. This justifies the high performance of our method over time as depicted in the graph.

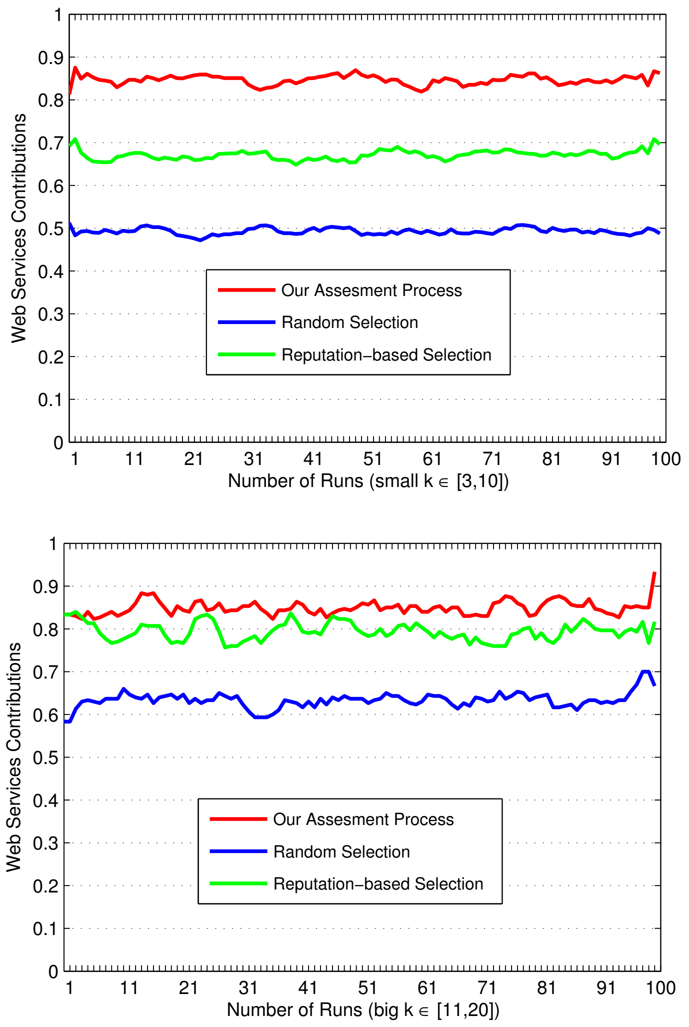


Figure 4.6 Comparison of Web Services' Contribution

4.3 Conclusion

In this chapter, we reported the results of our simulation and highlighted the effectiveness of our selection method. We clearly showed that our communities' evaluation outperforms the random-based and reputation-based approaches. The main reason is that the random-based approach is selecting communities randomly, so that sometimes the selection is not appropriate. Although the reputation-based selection makes use of reputation, which is a key factor, other performance factors are ignored. We observed that the community chosen based on the assessment process is showing a high number of requests, so high in-demand, and high satisfaction level. Moreover, we noticed that this community is taking less time to execute the users' requests. Our experiments showed that our method and the reputation-based selection remain consistently close when we chose big number of clusters, but the proposed method is always performing better when a small k is chosen. In the web services analysis part, we observed that our method shows a high number of requests and higher contribution with low response time. In all the cases, the simulation results showed that the selection based on our method belongs to the high performing class and outperforms the other two methods specially when we chose a small number of clusters. However, when we chose bigger number of clusters we found that the web services selected based on both the assessment process and the reputation-based selection are consistently close, although our assessment process is still performing better.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 Summery of Contributions

The main contribution of this thesis is the three-step assessment method of web services and their communities. This method helps users, providers, and masters choose the best candidate with regard to a set of relevant parameters. With the large number of web services providing the same functionality, the master of the community struggles with choosing the best web service to invite. In addition, as the number of communities providing similar functionalities increases, the competition between them also increases. This competition makes it difficult for users and providers to choose the best community that fits their needs.

To address this issue, the master, users, and providers need to perform a comprehensive assessment process of the existing communities and web services before making their choice. The approach we proposed in this thesis works as follows:

1. **Step 1:** Evaluating web services and communities considering different parameters from users and providers perspectives. Different equations have been introduced to compute those metrics.

2. **Step 2:** Clustering web services and their communities using the k-mean algorithm. The inputs of this clustering algorithm are vectors of dimension n for each web service and community considered, where n is the number of evaluated metrics in Step 1. The clustering algorithm runs until it converges. By convergence, we mean the agents in the k groups that are established are not moving to any other

cluster. Thus, when the identified clusters in iteration i are exactly the same as iteration $i-1$, the k-mean algorithm stops.

3. **Step 3:** Ranking the identified clusters in Step 2 using the goodness function.

To evaluate the effectiveness of our approach, we conducted simulations and compared the approach with two other methods against a classification benchmark of three identified classes. The results revealed that focusing on the requirements of the master, users, and providers during the assessment process helps chose the best candidate that satisfies the stakeholders' requirements.

5.2 Future Work

This work opens up a number of interesting avenues of research opportunities toward unanswered questions. This section describes some of the most interesting ones:

- The bootstrapping problem. What is the best way to assign an initial value to a new entity in a way to suit the context of web services and their communities? We also may consider developing game-theoretic incentives to encourage new developed communities and web services to perform well, so that they can get high initial values from users' and providers' points of view.

- In [19], Lim, Thiran, Maamar, and Bentahar consider what they call the 3-way satisfaction process, which is an optimization approach of selecting web services based on the satisfaction of communities, users, and web services. The selection method they propose is done by the master to select the appropriate slave to respond to the user request while considering different constriants of the stakholders. This approach can be extended to our selection method by selecting the best web service to invite, while focusing on the satisfaction of all the three parties.

BIBLIOGRAPHY

[1] A.S. Ali, S.A. Ludwig, and O.F. Rana. *A cognitive trust-based approach for web service discovery and selection*. In Proceedings of the European Conference on Web Services (ECOWS), pages 38?-40, 2005.

[2] E. Al-Masri and Q.H. Mahmoud. Discovering the Best Web Service. In Proceedings of the 16th international conference on World Wide Web, Pages 1257–1258, 2007.

[3] M. Bell. *Introduction to Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley &Sons, 2008.

[4] . B. Benatallah, Q.Z. Sheng, and M. Dumas. *The Self-Serv Environment for Web Services Composition*. IEEE Internet Computing, 7(1):40–48, 2003.

[5] A. Benharref, M. Serhani, S. Bouktif, and J. Bentahar. *A New Approach for Quality Enforcement in Communities of Web Services*. In Proceedings of the 8th IEEE International Conference on Services Computing (SCC), pages 472–479, 2011.

[6] J. Bentahar, Z. Maamar, D. Benslimance, and P. Thiran. *An Argumentation Framework for Communities of Web Services*. IEEE Intelligent Systems, 22(6):75–83, 2007.

[7] J. Bentahar, Z. Maamar, W. Wan, D. Benslimane, P. Thiran, and S. Subramanian. *Agent-based Communities of Web Services: An Argumentation-driven Approach*. Service-Oriented Computing and Applications 2(4):219–238, 2008.

[8] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. *Web services architecture*. http://www.w3.org/TR/ws-arch/, February 2004.

[9] V. Buskens. *Social Network and Trust*. 2002.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web services description language (WSDL) 1.1*. http://www.w3.org/TR/wsdl, March 2001.

[11] F. Curbera, M. Duftler, R. Khalaf,W. Nagy, N. Mukhi, and S.Weerawarana. *Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI*. IEEE Internet Computing, 6(2):86–93, 2002.

[12] S. Elnaffar, Z. Maamar, H. Yahyaoui, J. Bentahar, and P. Thiran. *Reputation of Communities of Web services - Preliminary Investigation*. IEEE International Conference on Advanced Information Networking and Applications, pages 603–1608, 2008.

[13] R. Jurca and B. Faltings. *Reputation-based Service Level Agreements for Web Services* In Proceedings of the International Conference on Service Oriented Computing (ICSOC), pages 396–409, 2005.

[14] B. Khosravifar, M. Alishahi, E. Khosrowshahi Asl, J. Bentahar, R. Mizouni, and H. Otrok. *Analyzing Coopetition Strategies of Services within Communities*. In Proceedings of the International Conference on Service Oriented Computinf (ICSOC), pages 656–663, 2012.

[15] B. Khosravifar, J. Bentahar, A. Moazin, Z. Maamar, and P. Thiran. *Analyzing Communities vs. Single Agent-based Web Services: Trust Perspectives*. In Proceedings of the 7th IEEE International Conference on Services Computing (SCC), pages 194–201, 2010.

[16] B. Khosravifar, J. Bentahar, A. Moazin, and P. Thiran. *Analyzing Communities of Web Services Using Incentives*, International Journal of Web Services Research (IJWSR), 7(3):30–51, 2010.

[17] B. Khosravifar, J. Bentahar, P. Thiran, A. Moazin, and A. Guiot. *An approach to incentive-based reputation for communities of web services.* In Proceedings of the IEEE International Conference on Web Services (ICWS), pages 303?-310, 2009.

[18] Y. Kim, K. Doh. *Trust Type Based Semantic Web Services Assessment and Selection.* 10th International Conference on Advanced Communication Technology, pages 2048–2053, 2008.

[19] E. Lim, P. Thiran, Z. Maamar, and J. Bentahar. *On the Analysis of Satisfaction For Web Services Selection.* IEEE International Conference on Services Computing (SCC), pages 122–129, 2012.

[20] S.P. Lloyd. Least Square Quantization in PCM. IEEE Transactions on Information Theory, 28(2):129-?137.

[21] E. Lim, P. Thiran, Z. Maamar, and J. Bentahar. *Using 3-Way Satisfaction For Web Services Selection.* In Proceedings of IEEE International Conference on Services Computing (ICC), pages 731–732, 2011.

[22] E. Lim, P. Thiran, and Z. Maamar. *Towards Defining and Assessing the Non-Functional Properties of Communities of Web Services.* International Conference on Advanced Information Networking and Applications, pages 578–585, 2011.

[23] Z. Maamar, S. Sattanathan, P. Thiran, D. Benslimane, and J. Bentahar. *An Approach to Engineer Communities of Web Services: Concepts, Architecture, Operation, and Deployment.* International Journal of E-Business Research 5(4):1?-21, 2009.

[24] Z. Maamar, P. Thiran, and J. Bentahar. *Web Services Communities: From Intra-Community Coopetition to Inter-Community Competition.* E-Business Application for Product Development and Competitive Growth: Emerging Technologies, pages 333–343, 2011.

[25] J.B. MacQueen. *Some Methods for classification and Analysis of Multivariate Observations.* Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281-?297, 1967.

[26] Z. Malik, and A. Bouguettaya. *RATEWeb: Reputation Assessment for Trust Establishment among Web services.* The VLDB Journal, 18(4):885–911, 2009.

[27] E. Maximilien and M. Singh. *A Framework and Ontology for Dynamic Web Service Selection.* IEEE Internet Computing 8(5):84–93, 2004.

[28] E. Maximilien, and M. Singh. *Toward Autonomic Web Services Trust and Selection.* In Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC), pages 212–221, 2004.

[29] M. Maximilien and M. Singh. *An Ontology for Web Services Ratings and Reputations.* In Proceedings of the Workshop on Ontologies in Agent systems (OAS), pages 25–30, 2003.

[30] B. Medjahed and Y. Atif. *Context-based Matching for Web Service Composition.* Distributed and Parallel Databases, 21(1):5–37, 2007.

[31] B. Medjahed and A. Bouguettaya. *A Dynamic Foundational Architecture for Semantic Web Services.* Distributed and Parallel Databases, Kluwer Academic Publishers, 17(2):179–206, 2005.

[32] S. Ran. *A Model for Web Services Discovery With QoS.* ACM SIGecom Exchanges, 4(1):1–10, 2003.

[33] F. Ruggeri and S. Sivaganesan. *On Modeling Change Points in Non-Homogeneous Poisson Processes*, Statistical Inference for Stochastic Processes 8(3):311-329, 2005.

[34] A. Shaikh Ali, S. Majitiha, O. Rana, and D. Walker. *Reputation-based Semantic Service Discovery.* Concurrency and Computation: Practice and Experience, 18(8):817–826, John Wiley & Sons, 2006.

[35] R.G. Smith. *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.* IEEE Transactions on Computers, C-29(12):1104–1113, 1980.

[36] R. Sreenath, and M. Singh. *Agent-Based Service Selection.* Web Semantics: Science, Service, and Agents on the World Wide Web, 1(3):261–279, 2004.

[37] Y. Wang and J. Vassileva. *A Review on Trust and Reputation for Web Service Selection.* In Proceedings of the International Conference on Distributed Computing Systems and Workshopps (ICDCSW), page 25, 2007.

[38] W. Wan, J. Bentahar and A. Ben Hamza. *Modeling and Verifying Agent-based Communities of Web Services.* In Proceedings of the 23rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA-AIE), volume 6704 of Lecture Notes in Artificial Intelligence, pp. 68-78, 2010.

[39] X. Wu, V. Kumar, J. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand, and D. Steinberg. *Top 10 Algorithms in Data Mining.* Knowledge Information Systems, 14(1):1–37, 2007.

[40] G. Wu, J. Wei, X. Qiao, and L. Li. *A Bayesian Network-based QoS Assessment Model for Web Services* IEEE International Conference on Services Computing (SCC), pages 498–505, 2007.

[41] Z. Xu, P. Martin, W. Powley, and F.Zulkernine. *Reputation-Enhanced QoS-based Web Services Discovery*. In Proceedings of the IEEE International Conference on Web Services (ICWS), pages 249–256, 2007.

[42] H. Yahyaoui. *A Trust-based Game Theoretical Model for Web Services Collaboration*. Knowledge-Based Systems, 27:162–169, 2011.

[43] Z. Zheng, and M. Lyu. *WS-DREAM: A Distributed Reliability Assessment Mechanism for Web Services*. IEEE International Conference on Dependable Systems & Networks, pages 392–397, 2008.

[44] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng. *Quality Driven Web Services Composition*. In Proceedings of the 12th International Conference on World Wide Web, pages 411–421, 2003.

[45] *IEEE Standard for a Software Quality Metrics Methodology*. IEEE Std 1061-1998 , vol., no., pp.i, 31 Dec. 1998 doi: 10.1109/IEEESTD.1998.243394