ANALYZING THE EFFICIENCY OF AGENT-BASED WEB SERVICES: COMMUNITIES CONSIDERATION

MAHSA ALISHAHI TABRIZI

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE (INFORMATION SYSTEMS SECURITY) AT

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

OCTOBER 2012

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mahsa Alishahi Tabrizi**

Entitled: **Analyzing The Efficiency of Agent-based Web services: Communities Consideration**

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. A. Youssef — Chair

Dr. A. Ben Hamza — Examiner

Dr. J. Paquet — Examiner

Dr. J. Bentahar — Supervisor

Dr. R. Mizouni — Supervisor

Approved by _____

Chair of Department or Graduate Program Director

_____ 2012 _____

Dr. Robin A.L. Drew, Dean

Faculty of Engineering and Computer Science

Abstract

Analyzing The Efficiency of Agent-based Web services: Communities Consideration

Mahsa Alishahi Tabrizi

Recently, a number of frameworks have been proposed to aggregate rational agent-based web services having the same functionalities but different non-functional characteristics inside communities with the aim of enhancing their capabilities of providing the required services. For an agent-based web service, deciding about functioning alone or within a community is a challenging problem that still needs to be investigated. Moreover, most of the proposed frameworks are considering either only competitive or cooperative behaviors of web services within their communities. However, some of the proposed frameworks suggest that web services within these communities although they are competing to get requests from users, may also exhibit a cooperative behavior, so web services are said to be coopetitive. Deciding which strategy to adopt within a community, competing or cooperating, is still an open question.

In this thesis, we propose a game-theoretic-based framework addressing the impact of joining a community on the efficiency of web services. The objective is to help web services decide about joining communities or acting alone. We also introduce a mechanism web services can use to effectively choose coopetitive strategies within communities which bring maximum utility. In this direction, we investigate web services' characteristics and their expected utilities over different

strategies. We enable web services that are hosted in communities with reasoning capabilities to enhance their quality of strategic interacting mechanisms as decision making procedures. The ultimate objective is to measure the threshold that web services can use in order to decide about different interacting strategies. Moreover, we develop a simulated environment where we analyze different scenarios and verify the obtained theoretical results using parameters from a dataset of real web services.

Acknowledgments

I would like to express my appreciation to my supervisor, Dr. Jamal Bentahar, for providing me with the opportunity to be part of his research team. I would like to thank him for all his guidance, advice and support over the years during my graduate study and research at Concordia University.

I would also like to thank my co-supervisor, Dr. Rabeb Mizouni, for her accurate and kind advice on my thesis. I really appreciate her efforts through this process.

I would like to extend my special thanks to my friend and colleague Babak Khosravifar for all the priceless help, discussions, support, and experience we have shared, there are not enough words to describe his kindness.

My gratitude also goes to my friends, and lab mates who have provided endless inspiration during my stay at Concordia University.

Finally, I would like to express my sincerest appreciation and love to my parents and family for all their unconditional help and support through all this process.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the past decade, intelligent agents and multi-agent systems have extensively absorbed attention and turned out to be popular topics in computer science in general and artificial intelligence in particular. In this domain, investigations about web services as abstract intelligent and autonomous agents have attracted increasing interest in service computing [7, 21, 38, 48]. These agent-based web services are used as means of promising interoperable software applications. Coordinations and interactions among agent-based web services happen according to their incomplete knowledge about the environment where they reside. Agent-based web services in a multi-agent setting try to gain information from other agents or provide and request services from peers. These kinds of interactions make trust and reputation assessments important [23, 36]. Moreover, considering web services as rational agent-based systems give them the capability of making decisions about competing or cooperating with other services in their environments to enhance their reputation and efficiency in general. Thus, measuring and observing the efficiency parameters and behaviors of these web services require designing sound and accurate frameworks.

This chapter introduces the context of research, which is about analyzing the efficiency of agent-based web services, for example in terms of reputation and payoff. The analysis is conducted considering the concept of *community*, which is a group of web services having similar functionalities but probably different non-functional properties and the role of a controller agent named *the master web service* to observe and control the behaviors and interactions among web services inside the community to provide a trusted community environment [26, 29, 31]. The chapter also discusses the two main issues we address in this thesis: (a) analyzing the impact of joining a community on the efficiency of single web services; and (b) analyzing the efficiency of those web services within their communities by investigating their cooperative and competitive behaviors. Moreover, the chapter summarizes the contributions of the thesis and outlines the thesis organization.

1.1 Context of Research

Web services are software programs that are deployed to serve and maintain application to application interactions. These web services can be abstracted as (or represented by) software autonomous agents which are capable of engaging in rational interactions [21, 49]. This abstraction will enable web services to have social attitudes and characteristics and decide about behavioral strategies in terms of cooperating and competing with other peers [6]. In fact, agents are software systems that can sense the environment they reside in and are able to react to the changes in that environment following their own goals. Autonomous agents can exhibit selfish and thus malicious behavior [20], which raises the issue of managing and controlling such agents. In

autonomous agent-based web services environments, this behavior is managed by means of reputation assessment, which is considered as the basis for service selection. In fact, having better quality of service leads to higher reputation and thus more service requests.

Service request overloading and lack of responsiveness are two main issues for single web services. The process of finding and selecting suitable services is another challenge in service computing settings. Regarding these issues, introducing communities which gather several agent-based web services with the same functionality but different non-functional characteristics has been emerged [6, 26, 29, 31, 32]. The concept of communities of web services enhances the service reliability in general as services can replace each other. However, gathering agent-based autonomous web services in a community raises some management challenges such as inviting or accepting web services as new members of the community, firing or rejecting existing web services, maintaining or dismantling the community. These activities and other community management issues are handled by the master web service. Within communities, reputation is evaluated by the community's representative and controller, called *the master web service*, based on the feedback provided by service users on the received quality of service.

This thesis is about increasing the efficiency of agent-based web services within their communities. Increasing the efficiency in terms of overall payoff is the ultimate goal in every business setting. In service computing, efficiency can be defined in terms of the number of requests a service can receive and handle, and is function of several parameters such as reputation, market share and capacity. A particularly challenging issue in this context is the interactions and behaviors of web services inside a community. Different strategies can be adopted by web services to increase their efficiency, such as cooperating, competing, and combining cooperation and competition with other peers. Deciding about which strategy to choose is still an open issue and this

thesis is an attempt in this direction.

1.2 Problem Statement

In this thesis, two issues are addressed and investigated: (a) web services' efficiency by joining or leaving the communities of web services; and (b) web services' competitive/cooperative behaviors (simply called *coopetitive* behaviors), decision making about when to compete and when to cooperate inside the community, and the role of this decision making in increasing the efficiency and hence service income. Regarding the first issue, a model in which web services either act alone or cooperate with other web services within a community is presented. Each single web service or community of web services manages its efficiency parameters such as reputation, market share, and capacity. There are also service users that select services based on their reputation. Each entity seeks maximum efficiency following strategies of *joining*/*leaving* a community, *accepting*/*refusing* a request to join a community, and *inviting* to join a community. In different scenarios, we investigate the situation that maximizes players' efficiencies. The goal is to investigate strategies as rational behaviors that web services and communities can adopt to increase efficiency.

Regarding the second issue, a model is introduced to help web services in their decision making process when these web services function within communities. This model presents a mechanism where web services in the community could choose either to compete for an announced task, or to cooperate with other competing web services in the same community to accomplish some subtasks of the announced task. The objective of this model is to enable web services to reasonably evaluate and decide over their coopetition strategies, which means when to compete

and when to cooperate.

1.3 Contributions of the Thesis

The contributions of this thesis are as follows:

1. The first contribution is the design of a multi-agent environment as distributed network of web services and service users. A community infrastructure has been designed by extending the work done in [31] to host web services. In this contribution, the task allocation problem is regulated by a mechanism taking reputation, market share, and capacity into account. The attributes of rational services are formalized and a game-theoretic analysis has been conducted to investigate the stabilized situation within which entities achieve high efficiency. This work identifies a threshold allowing web services to decide about joining or leaving a community. In different scenarios, we investigate the situation that maximizes players' efficiencies.

2. The second contribution builds on and extends the first contribution and proposes a mechanism within which web services in the community could choose either to compete for an announced task, or to cooperate with other competing web services in the same community to accomplish some subtasks of the announced task. It introduces the strategic decision making procedures that enable web services to apply different techniques to constrain high efficiency and obtain the maximum utility. In the presented mechanism, web services decide to compete based on their level of confidence which we call *growth factor* and is a function of their reputation, quality of service and budget over time. If their level of confidence is higher than a threshold obtained by a game-theoretic technique, web services bid

for tasks. In this model, a reward and penalty mechanism is applied by the master web service towards web services that accomplish tasks, either alone through competition or by cooperation with other web services. The web services' expected payoffs and involved probabilities that are used to choose over the two interacting strategies are calculated and investigated in this contribution. As intelligent entities, web services are equipped with a reasoning technique that enhances their abilities over best acting strategies and the attitude they could exhibit to yield maximum utility.

1.4 Thesis Outline

This chapter discussed the context, motivation, and contributions of the research work done in this thesis. The rest of the thesis is organized as follows. Chapter 2 gives the background and some definitions about agents and multi agent systems, web services and communities of web services and ends by discussing the behaviors of web services inside a community following by some definitions in game theory. Chapter 3 presents the proposed game-theoretic model in which web services either act alone or cooperate with other web services within a community to investigate rational behaviors that web services and communities can adopt to increase efficiency. It also describes the proposed simulation framework and experimental results. Chapter 4 extends the idea proposed in chapter 3 by exploring in detail the type of interactions that web services can choose. We analyze the impact of deciding to compete or cooperate with peers within the same community by making use of a game-theoretic model. Simulation and experimental results are also presented in this chapter. Chapter 5 concludes this work and discusses further ideas and possible directions for future work.

Chapter 2

Background

2.1 Agents and Multi-Agent Systems

For many years, agent has been perceived as an important concept in artificial intelligence, computer science, and even in the business discipline. Several definitions have been proposed for agents in the literature and each of which follows different purposes. Some of them are very general definitions while others are specific so that they apply only for limited settings. In this thesis, we consider the definition inspired by Wooldridge and Jennings in [47, 49] according to which, an agent is an entity that enjoys the following properties:

1. **Autonomy.** Each agent has control over its actions without any direct human or other interventions.

2. **Social ability.** Using a kind of communication language, agents are able to interact with each other or with humans.

3. **Reactivity.** Agents can sense the environment in which they reside. They respond to this

environment changes in a timely manner.

4. **Pro-activeness.** Agents do not simply react based on environment changes, but they follow their goals by taking their own initiatives.

In a more comprehensive definition [50], agents are considered as intentional systems with human characteristics such as beliefs, desires, knowledge and commitments. Some of these characteristics are categorized under *Information Attitudes* [49] which generally include the information of the agent about its surrounding world and some of them are part of the category of *Pro-attitudes* [49] that are guidance for agents to take the right actions.

A Multi-Agent System (MAS) is a distributed system composed of multiple interacting autonomous agents [48]. The knowledge about the world or environment is distributed among agents locally. None of the agents has a global or complete knowledge about the other agents or the environment. For this reason, agents need to continuously interact with one another. In this regard, agents (are supposed to) complete each other's knowledge and capability to accomplish different types of tasks. This has an important influence on the interactions among agents, since each agent acts based on a partial vision about the other agents' states. Based on the type of the system and goal of the agents, different models of interactions are defined in multi-agent systems which we are going to discuss in Section 2.4.

2.2 Web Services

Web services are application-to-application programs or Service Oriented Applications (SOA) [34] which are used to model and accelerate solving real world business problems. These highly

abstracted loose-coupling language-independent interfaces describe the network-accessible interactions and operations through standardized XML messages. A set of formal XML messages define service descriptions that provide all the details necessary for service interactions. WSDL [10] which is an abbreviation for Web Service Description Language expresses these web service descriptions [16]. The Simple Object Access Protocol (SOAP) which is an XML-based protocol for messaging and remote procedure enables communications between web services [13]; and the Universal Description, Discovery and Integration (UDDI) [9] directory is a registry for web services descriptions. As fruitfully argued in [5, 6, 18, 30, 44], web services could be abstracted and represented by agents to equip them with intelligence and decision making abilities.

According to the definition of web services in [8], web service architecture provides the message exchange between software agents. This message exchange mechanism has three main components (Figure 2.1):

1. **Service requester agent.** This agent is the software agent that request services from some other software agents.

2. **Service provider agent.** This kind of agent is responsible to provide services for other software agents which are called service requesters.

3. **Service registry.** The service requesters can find the appropriate service providers descriptions in service registry.

Three operations are defined in web service architecture [16]:

1. **Publish.** A service provider publishes its service description in detail in the service registry.

2. **Find.** A service requester then finds its required service description in the service registry.

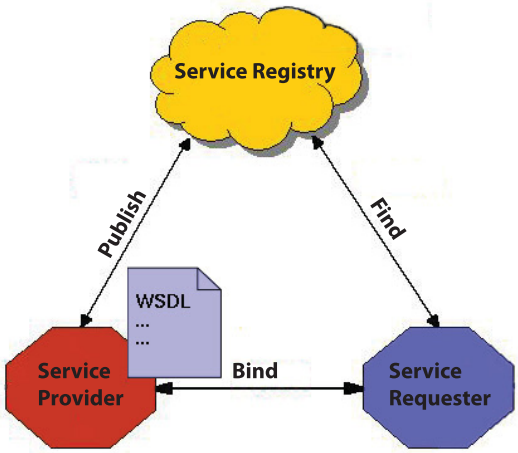

Figure 2.1: Web Service Architecture

3. **Bind.** The service requester agent binds to the found service provider agent at a particular web location and starts interacting with it.

2.3 Communities of Web Services

The idea of gathering web services with similar functionalities but different non-functional characteristics in a group called *community of web services* is proposed in [5,6,31]. The main purpose of the community in [31] is having better accessibility to web services and makes finding the appropriate service providers with specific functionalities easier.

The purpose of having a community structure as mentioned in [31] is to make it easier to discover web services with a specific functionality. This approach requires more discussions about rules and guidelines for managing the community. The following questions are addressed in this context: How to initiate and setup a community, how to manage web services in a community, how to reconcile conflicts inside a community and between communities, and when to dismantle a community. Web services may join, leave, become unavailable temporarily, or resume their actions

in a community. As mentioned in [5, 29], a community provides a way of responding to desired services without explicitly referring to any specific web service which is really providing the service at runtime. Communities of web services should have an added-value for both the service provider and the service requester. From the service requester's perspective, the provided quality of service (QoS) by a community should be higher than the one provided by a single web service, which would justify the use of such community instead of direct relation with the single service. From a service provider or a web service's point of view, service income is the important factor. The concept of community of web services enhances the service reliability in general since at the time of failure or overload of one of the service providers, other web services inside the community can be used as substituters. Also, web services may exhibit different attitudes. Sometimes they cooperate with each other to gain more benefits and sometimes they compete. They also can announce misleading information to enhance their participation opportunities in composites or gain better credit. So there should be a monitoring system to follow web services behaviors closely to avoid malicious actions. This monitoring component is called *Master Web Service* (Figure 2.2), while other web services inside a community are called slaves. The master web service can be a completely new type of agent whose responsibility is only observation of web services or it can be one of the web services inside the community that has better management capabilities comparing to others. In this thesis, we consider the master web service as an agent who is in charge of managing the community and observing web services interactions inside the community. In the next chapters, the master web service is considered as the representative of the community of web services.

In the architecture proposed in [31] which we are going to discuss more in Chapter 3, Section 3.1, service providers publish their web services properties in an UDDI registry. One of the

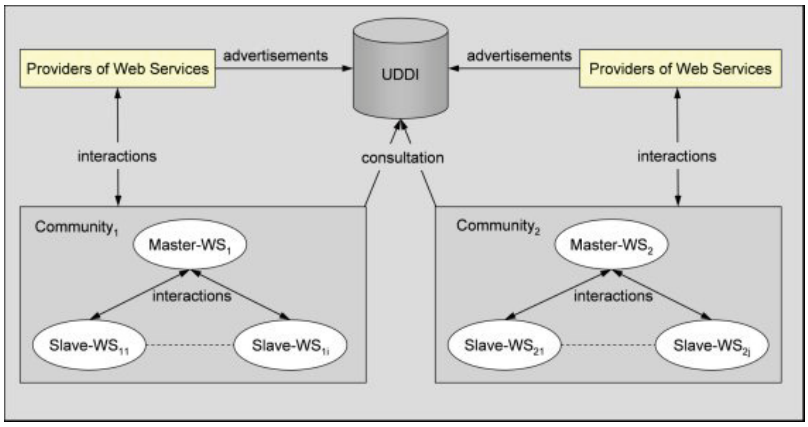

Figure 2.2: Community of Web Services Architecture [29]

responsibilities of master web service is to find web services by consulting with UDDI registries and attract them to join its community as well as to fire the web services that have not acted based on their advertised properties. Also, providers of web services can suggest their web services to master web service to invite them to join the community. The other responsibility of master web service is selecting web services to participate in a composite web service to perform a desired task. At any time, the master web service can analyze the performance of the community and decide to maintain the current status, enhance or dismantle the community which it is responsible for. Another role of master web service is to discover and punish the malicious web services inside the community. The ability of master web service in this latter role denotes the reputation of the community of web services and is of a high importance for both service requesters and service providers, because it shows how reliable is a community. Gathering web services with the same functionality but different non-functional properties develops the new interactions and behaviors among web services inside the community. We are specifically interested to investigate different types of interactions and strategies that web services choose when they subscribe into a community. Analysis of these strategies enables the master web

service to reach high performance while supervising the community. We will discuss two of the most important interaction strategies in the next section: cooperation and competition.

2.4 Cooperation and Competition

Previously in this section, we promoted the agents, multi-agent settings and abstracted web services that are hosted in communities. In a typical community of web services, a task could be accomplished with a single web service or with an aggregation of a group of web services' contributions. This is called *cooperation*. Cooperation happens when a web service cannot handle its service request because of overload or any kind of failure, and another web service substitutes the failed web service. Sometimes a complex request is accomplished by a cooperation among web services inside the community. In the context of communities, we distinguish web services cooperation from web services composition. By cooperation, we mean that the community aggregates web services capable of interacting with one another to manage allocated tasks, for example by allowing a web service to replace another that is incapable of executing a task. By composition, we mean the extension initiated by a web service to other web services to finalize a specific task that the initiator web service cannot perform alone. The cooperation concept is a well-known term in the sense that it represents group work and its requirements [14]. It is worth to note that we use terms *cooperation* and *collaboration* in this thesis interchangeably. However, inside a community there would be some additional constraints that are applied by the master web service. We discuss more about the effectiveness of cooperative behaviors among web services inside a community in Chapter 3. Besides cooperation, there is *competition*, which represents web services' tendency to obtain a task or to take part in compositions while they have other

competitors residing in the same community. Being more realistic and practical, web services intend to exhibit both cooperative and competitive behaviors in the same community for different purposes, simultaneously. This behavior is called *coopetition*, which is a very popular subject in business-related context [4, 11, 42, 52], but only recently a few coopetition frameworks have been proposed in the web service and community-related literature [32, 33]. We are interested in the study of these two behaviors and how they could be distinguished and correlated. In this thesis, we will further investigate the influence of each behavior on the performance of the community as a whole. Moreover, we will study the outcomes of these behaviors and the decision making procedures while residing in a community with continuous tasks in Chapter 4.

2.5 Game Theory

By a game we mean roughly a situation of conflict between two or more people, in which each contestant, player, or participant has some, but not total, control over the outcome of the conflict [43]. *Strategy* is the action that players choose and exhibit for playing in the game. In some cases players cannot deterministically choose from their strategies, so they randomly select among these pure strategies with certain probabilities [45]. This latter is called a *Mixed Strategy*. A game can be analyzed as a one-shot or a repeated game. One of the important solution concepts in game theory is *Nash Equilibrium* which is a strategy profile where the strategy of each player is a best response to the other strategies. In other words, a Nash equilibrium is a strategy profile so that no player can gain by changing his strategy unilaterally, that is, with the remaining strategies kept fixed [45].

Chapter 3

A Game Theoretic Approach for Analyzing the Efficiency of Web Services in Collaborative Networks

As mentioned earlier in Sections 2.1 and 2.2, web services cooperate in order to increase their efficiency, which is the ultimate goal in business settings. Efficiency in service computing can be defined in terms of the number of requests a service can receive and handle, which is a function of different parameters such as *service reputation*, *market share* and *capacity*. To address the efficiency issue, there have been efforts attempting to model and analyze collaboration among web services. Moreover, many frameworks [5, 6, 29, 31] have been proposed towards communities of web services (Section 2.3) and task allocation management, which we will discuss in this chapter and Chapter 4 in more details. In all the proposed frameworks on collaboration, the objective is to increase efficiency in distributed computing. However, in such frameworks, strategies web

services can follow to achieve this goal are very limited to simple cooperation case and more sophisticated strategies have not been thoroughly analyzed and investigated yet. Such sophisticated strategies can help communities and single web services achieve higher efficiency.

The aim of this chapter is to investigate strategies as rational behaviors that web services and communities can adopt to increase efficiency. We present a game-theoretic model in which web services either act alone or cooperate with other web services within a community. Each entity (single web service or community of web services) manages its reputation, market share, capacity, and efficiency parameters. There are also service users that select services based on their reputation. A game is defined between typical single web service and the representative of a typical community (master web service). Each entity seeks maximum efficiency following strategies of *joining*/*leaving* a community, *accepting*/*refusing* a request to join a community, and *inviting* to join a community. In different scenarios, we investigate the situation that maximizes players' efficiencies. Overall contributions of this chapter are threefold:

1. We provide a distributed network of web services and users where reputation, market share, and efficiency are taken into account.

2. We propose a game-theoretic analysis investigating the stabilized situation within which entities achieve high efficiency.

3. We identify a threshold allowing the web service to decide about joining/leaving the community.

In the rest of this chapter, we discuss relevant related work in Section 3.1. Then we define the efficiency metrics we take into account in our model (Section 3.2). After that, we present

our game-theoretic model in Section 3.3. Our proposed framework and experimental results are provided in Section 3.4.

3.1 Related Work

In many frameworks [3, 19, 22, 24, 35, 37] proposed in the literature, service selection and task allocation are regulated based on the reputation parameter. We discuss these proposals in Section 3.1.1. All these models address the reputation in environments where web services function alone. In such models, web services efficiency is not discussed in details and in general, balancing the market share with the capacity is not considered as an issue for web services besides their reputation.

There have been few models addressing the communities of web services. The objective is to facilitate and improve the process of web service selection and effectively regulate the process of request and task allocation. We will consider them more in detail in Section 3.1.2. In general, the recent aforementioned proposals motivate the existence of communities rather than single functional web services, but fail to systematically provide potential benefits and technically compare different scenarios that increase service providers' efficiency.

3.1.1 Reputation-Based Service Selection

To make interactions between agents and service selection in an intelligent system environment robust, there should be a technique or tool to let agents discover services automatically. To differentiate between these services, the proposed framework in [3] introduces an automated service discovery and selection of web services based on a user's trust policy. This framework is validated

by a case study of data mining web services. The issues addressed in this paper include describing and discovering web services semantically by the concentration on high level conceptual basis and not the details, and an automatic mechanism to select trustworthy services from the user's trust policy perspective. The proposed framework makes use of a case study identified in FAE-HIM project [3], and they show that the factors which play a role in the final decision making for service selection in this case study are dependent on the user's trust disposition. This framework has three main components that discover and match services in a reputation-based mechanism. A matchmaker semantically discovers services, and then a composer component gathers a combination of services with the same functionality as requested and a reputation manager which selects a service from the service combination using the reputation metrics. Resource providers periodically advertise their available services with their capabilities to one or more matchmakers. Users activate the matchmaker by sending a query containing their type of resource they want. Different algorithms are used for different service categories. By matchmaking the services in this framework, there is no need for the user to have low level knowledge about services or various data mining algorithms. And instead of just a random selection from available services, a trust-based service selection is done by a service selector in the framework.

According to the limited knowledge of the agents, they self-organize into community structures to share information locally. The contribution demonstrated in [19] tries to show how such locally shared information increases the service demand stability and leads into self-organized community structure. The authors in [19] propose a multi-agent model in which agents have a limited knowledge about the availability of the service providers. Meanwhile the demand for particular services changes over time. Hence to provide service more effectively and responding to the demand changes in a proper manner, agents organize and share their information locally with each

other. Three issues have been addressed in this model:

1. What service types should be offered by which providers in order to satisfy the current demand.

2. How to minimize the competition by utilizing what providers known to a customer.

3. How to organize the system to have a better balance between demand and supply for a particular service type.

In this model, consumer agents monitor the service provider they know locally. They use the gained knowledge for their future decisions about service requests and they share this knowledge once establishing a community. While the providers have the capability to offer different types of services over time to respond to changing in the demands requested by community of consumers, these providers are allowed to advertise and provide just one service at a time. The decision about which service to offer is made based on the providers previous experience of the consumers demands for each service. When the number of providers and consumers is equal and static, service supply may converge to satisfy demand. By this model which is an adaptive multi-agent system, the relationship between the quantity of information exchanged and the size of emergent neighborhoods and also the formation of stable local communities is discussed.

A new QoS attribute, termed verity, and an architecture to quantify this attribute is introduced in [24]. The authors try to prove that verity should be taken into account in quality-driven selections and compositions of web services. They believe that verity is a better and more intuitive indicator of the service provider's trustworthiness. Verity is measured by external components and is the ability to retain the difference between the proposed and provided service metrics in

their lowest level. In this context, reputation is defined as a vector of verity, user feedback or a rating system and compliance level. Compliance is the normalized difference between the proposed QoS attributes values and the real provided values. As a mean to measure verity, the variance in the compliance levels is being considered shows the ability of service provider in providing the same level of service as it had promised. Lower the variance, the more successful the service provider in delivering the consistent service. Verity is obtained from the past service invocations and the past calculated variances. This new attribute indicates how effectively QoS attributes have been delivered. Therefore the service selection in a quality-driven scenario should be done using verity along with some other well-known and widely used service attributes like response time, availability or performance and so on. Users and service brokers can have different values of verity of services for themselves.

A conceptual model for reputation has been described in [37]. This model explains how to organize and manage the information related to reputation and how this information and reputation, itself, can facilitate the service selection process. New service initiation and evolution over time in a system, how these new services can be recognized and discovered, the aggregation of right information of these services along with a formal definition of them as service quality are addressed in this contribution. The proposed model is a generic conceptual model that can be enhanced based on different domains.

A multiagent framework based on QoS and a new model of trust within which, the providers are able to advertise for their services, the users can express their preferences and the ratings of services can be gathered and shared has been developed in [35]. Service ratings in this system are based on quality properties and are obtained via automatic monitoring or user input. By sharing the ratings, the agents can form an environment in which they are able to help each

other by providing information regarding services which can be a help for other agents towards better service selections. Indeed, service selection can be rationally done only in an empirical basis, that is, how an agent or service really acts or behaves and not only how it advertises. An agent called explorer agent is being considered in this framework to monitor the different service implementations, specially the ones that are new or idle. It allows the customers to better find the services suitable to their required QoS, even if there is not adequate positive experience with such implementations available. In general, the proposed system works based on self-adjusting trust model.

Another framework which works based on service level agreement and considers the actual delivered QoS as a basis for price calculations is proposed in [22]. A reputation mechanism is introduced and used in this framework to monitor the promised QoS and the delivered one. The authors show that both providing the promised service by service provider and sending correct reports about the received service by users are optimal. The latter is provided by a scoring rule. Monitoring the services and transactions among agents is necessary to avoid any kind of malicious behaviors. Here the monitoring is done by customers themselves. So it raises two problems regarding honest feedbacks from clients and collusion between providers and clients. In this framework, service providers repeatedly offer the same service to the clients but with different QoS parameters, and then all the evaluations are done based on these parameters and the difference between the promised QoS parameters and the provided ones. The idea behind this framework is to discourage service providers from advertising untruthful and higher QoS parameters than they are really able to afford. To this end, a penalty will be applied to the untruthful service providers. This penalty is directly proportional to the difference between the promised and the delivered QoS such that the untruthful service provider will obtain less price than if it

had advertised the real QoS. Reputation information is used to compute the penalty value. The other enhancement is considered for the clients in this framework such that clients are got paid for submitting the honest feedback. Here the authors emphasize on the rational clients who can analyze the changes in the environment and can see the benefits of reporting the truth rather than a false feedback.

3.1.2 Communities of Web Services

Two protocols have been followed in the development of the community in [31], Contract Net Protocol (CNProtocol) and Web Services Community Development Protocol (WSCDProtocol). Since the main reason for developing a community in this approach is web services functionalities, first of all, for inviting web services, their functionalities are mapped into the community functionality, and after that the credentials of selected web services are checked by master. This latter has dual advantages: first, it avoids dealing with malicious web services in the future and second, it enhances the trustworthiness level of master web services towards slaves. This security level can enhance the opportunity of community to attract single web services to join the community and users to send requests to the community. However, there is always a chance for a web service to become lazy after joining a community or act maliciously. In these cases the master will decide to eject or punish the malicious web service. Meanwhile, if the master evaluates the configuration of the community as a poor configuration, it decides to dismantle the community. Task allocation is done based on CNProtocol. The master sends a call for bids. Slave web services which are available respond. The best bid is chosen and the selected web service is get notified. The rest of web services are notified as well. The processes of joining and leaving and

monitoring slaves are followed by CNProtocol, WSCDProtocol, or a combination of both.

While speaking about the community of web services, the reputation of such communities becomes an issue. Analyzing the reputation of communities of web services and single web services and comparing them with each other are discussed in [25–27]. The reputation mechanism is mainly working based on the users' feedbacks. In [25], a model to rank different communities of web services and single web services is been proposed. The discussion then is extended by analyzing the single web services benefits and incentives of joining a community of web services. The service reputation is calculated considering two factors: satisfaction and inDemand. It means the reputation has been measured based on the obtained feedbacks and previous provided service. This model and the model proposed in [26] are extensions of the work done in [27] within which, the assessment of reputation of communities of web services is done based on the after-service feedbacks provided by the users and these feedbacks are get supervised and verified in order to avoid fake feedbacks. In [26], a time factor is added to the analysis of reputation, called recency. The whole idea in [25–27] is about maintaining a sound reputation mechanism by encouraging agents to avoid malicious behaviors like providing fake feedbacks or collusion and to discourage communities of web services to provide a unreliable service to users by penalizing the malicious communities and agents.

3.2 Efficiency Metrics

Feedback is a rating value that can be -1 or 1, posted by a user to express satisfaction about the provided service. Feedback are accumulated in the feedback file to compute and analyze the web service's reputation.

Reputation is a value between 0 and 1 computed from the feedback file. Web services, as rational entities, aim at increasing this value to maximize their income. We compute the reputation value of a single web service w_i ,R_{w_i}, and a community of n web services c_j ,R_{c_j}, in Equation 1. In this equation, PF_{w_i} denotes the number of positive feedback posted for the web service w_i and TF_{w_i} represents the total submitted feedback to this web service. These parameters could be evaluated during a fixed period of time such as a day or a week.

$$R_{w_i} = \frac{PF_{w_i}}{TF_{w_i}} \tag{1}$$

$$R_{c_j} = \frac{\sum_{w_i \in c_j} R_{w_i}}{n}$$

Market Share is a value between 0 and 1 denoting the portion of the total requests directed to the web service (or community) to all initiated requests. The value is computed in Equation 2. In this equation, Req_{w_i} denotes the requests sent to to the web service w_i and $TReq$ represents the total requests filed.

$$M_{w_i} = \frac{Req_{w_i}}{TReq} \tag{2}$$

$$M_{c_j} = \sum_{w_i \in c_j} M_{w_i}$$

Capacity is a parameter denoted by Cp_{w_i} that the web service w_i holds as an individual attribute. This value is fixed and only reflects the service total ability in handling simultaneous requests. For the community, the total capacity is simply the sum of its members' capacities (see Equation 3).

$$Cp_{c_j} = \sum_{w_i \in c_j} Cp_{w_i} \tag{3}$$

3.3 Game-Theoretic Model

In this section, we formalize the attributes of rational services. In general, all rational entities, including users and web services, tend to maximize their efficiencies. Here, we only consider the perspective of web services. Thus, we propose a heuristic (see Equation 4) for computing the efficiency E_x as a function f of R_x, M_x and Cp_x where $x \in \{w_i, c_j\}$

$$E_x = f(R_x, M_x, Cp_x) \tag{4}$$

The function f should satisfy the following properties.

Property 1. *f is continuous.*

This property says that at each moment the efficiency of a web service or a community can be evaluated with respect to the current attributes.

Property 2. *f is strictly increasing in R_X and M_X.*

This property says that the efficiency of the service increases if it holds high reputation and market share in the system. Consequently, services and communities will have incentive to do better to get their overall efficiency increased.

Property 3. *f is monotonically decreasing in $M_X - Cp_X$.*

This property says that the efficiency of a service or a community decreases if it fails to make a good balance between its capacity and the requests it should handle. Consequently, services and

communities will have incentive to analyze their capacities and manage to have acceptable market share. The idea is that the more service provider entity succeeds in making balance between its capacity and market share, the higher the efficiency would be.

Equation 5 gives a possible definition of f.

$$f = \frac{R_x \times M_x}{|M_x - Cp_x| + 1} \tag{5}$$

Theorem 1. *The function f satisfies Properties 1, 2 and 3.*

Proof. Satisfaction of Property 1 is straightforward as all the parameters are defined at each moment in time, so the function is continuous. Property 2 can be proved by computing the partial derivatives $\frac{\partial f}{\partial R_x}$ and $\frac{\partial f}{\partial M_x}$, which are clearly positive. Property 3 can be proved by considering $|M_x - Cp_x|$ as a variable, say v and compute the partial derivative $\frac{\partial f}{\partial v}$, which is manifestly positive. □

The other attribute that categorizes services is the risk factor S_X. This factor shows how flexible the service is in loosing its efficiency. For example, if the risk factor associated with w_i is $\%20$ ($S_{w_i} = 0.20$), then the web service w_i would consider any situation in its strategy analysis where estimated efficiency is more than $\%80$ of its current efficiency. $\overline{E_{w_i}}$ is defined as the estimated efficiency of the web service w_i after taking any strategy for updating its status ($\overline{E_{c_j}}$ would corresponds to the community c_j). To this end, the web service w_i would discard all the strategies (and choices of updating the current status) that yield an estimated efficiency less than $(1 - S_{w_i})E_{w_i}$. The reason behind using the provider risk factor is that web services or communities need to be flexible in choosing strategies. For the rest of this section, we discuss two different cases where the web service is outside and inside the community. In each case, we analyze the best strategies

that culminate in maximum efficiency level for both the web service and community. We calculate the efficiency in different strategies of *joining/leaving* a community, *accepting/refusing* a request to join a community using Equation 5.

3.3.1 Web Service Out of Community

In this scenario, the single web service w_i is facing the community c_j with different strategies that would end in either the single web service w_i joins the community c_j or not. This action could be initiated or ceased by the web service or community representative. Doing so, there are four different cases:

1. w_i attempts to join c_j, J_{w_i,c_j}, and the attempt is accepted, A_{w_i,c_j}.
2. w_i attempts to join c_j, $JR_{w_i}^{c_j}$, but c_j refuses the join request, $JR_{c_j}^{w_i}$.
3. c_j invites the web service w_i, $SI_{c_j}^{w_i}$, but w_i refuses the invitation, $SI_{w_i}^{c_j}$.
4. there is neither invitation from c_j nor join request from w_i

From the outcome perspective, the cases of "w_i attempts to join and c_j accepts" and "c_j invites and w_i accepts" are similar. However, refusal from any party would lead to different estimated efficiencies and this is why we consider them as two separated cases. In the following, we compute the estimated efficiency of each entity with respect to the taken action.

Case (1) The web service w_i takes the risk (S_{w_i}) of joining the community. It would update its reputation, market share and capacity parameters respectively in Equations 6, 7, and 8 where n denotes the current cardinality of the community set (since we distinguish between web services composition and cooperation in this context, see Section 2.4, the capacity stays unchangeable).

$$\overline{R_{w_i}} = \frac{n \times R_{c_j} + R_{w_i}}{n+1} \tag{6}$$

$$\overline{M_{w_i}} = \frac{M_{c_j} + M_{w_i}}{n+1} \tag{7}$$

$$\overline{Cp_{w_i}} = Cp_{w_i} \tag{8}$$

In this case, our assumptions are as follows:

1. The reputation of a web service would be updated to the average of the community reputation. To this end, each registered web service in the community holds its individual reputation, but broadcasts the public reputation of the community.

2. We consider the capacity as a fixed attribute. Therefore, the capacity of the web service stays unchanged, but the community accumulates the joined web service's capacity.

When it comes to the market share, the community simply accumulates the market share of the new web service. However, the joined web service is going to obtain a share of total market share from the community. The corresponding attribute updates regarding the community c_j are formulated in Equations 9, 10, and 11.

$$\overline{R_{c_j}} = \overline{R_{w_i}} \tag{9}$$

$$\overline{M_{c_j}} = M_{c_j} + M_{w_i} \tag{10}$$

$$\overline{C_{c_j}} = C_{c_j} + C_{w_i} \tag{11}$$

In this case, both entities consider the estimated parameters and compute their new efficiency values (see Equation 5). The case would take place when the following inequalities hold:

$$\overline{E_{w_i}} \geq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \geq (1 - S_{c_j})E_{c_j}$$

Case (2) In this case, w_i requests joining, but the community does not accept the request. The difference between the cases (1) and (2) is that in case (1) the join takes place, which brings actual updated efficiency for both entities. However, in case (2) the join does not take place, which keeps the analysis at the estimation level. The corresponding estimated efficiencies are characterized by the following inequalities:

$$\overline{E_{w_i}} \ngeq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \geq (1 - S_{c_j})E_{c_j}$$

Case (3) This case is similar to the case (2), except the fact that the refusal is caused by the web service. The corresponding estimated efficiencies are characterized by the following inequalities:

$$\overline{E_{w_i}} \geq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \ngeq (1 - S_{c_j})E_{c_j}$$

Case (4) In this case, both entities are not encouraged to attempt joining and therefore, the join does not take place. In this case, we have:

$$\overline{E_{w_i}} \ngeq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \ngeq (1 - S_{c_j})E_{c_j}$$

3.3.2 The Game Set up for Single Web Service

Upon the discussed cases, we develop a game-theoretic model consisting of the web service w_i as player 1 and community c_j as player 2. The player 1 follows the strategy profile of (join/not join) when is initiating the game (i.e. play first), and follows the strategy profile of (accept join/refuse join) when is reacting to the opponent's move (i.e. play second). Since for our analysis it is only important whether the join takes place or not, the order of playing does not matter when calculating payoffs (represented in terms of efficiency). Table 3.1 shows the assigned payoffs for both players in different cases. As shown in the table, the values of J_{w_i,c_j} and A_{w_i,c_j} are the generalized form of "join/accept" or "invite/accept join" cases. These values are actual differences in efficiency values after the join (E'_{w_i} and E'_{c_j}). The obtained payoffs could be either positive or negative. The negative payoff denotes a wrong decision the entity regrets. The payoffs obtained in the other cases are all upon estimations.

The developed game is only a one-stage game between a typical web service and a typical community. The game could be set up between any other two entities and is repeated over time when entities are active in the network. Moreover, rational entities consider the information obtained in one game in their further strategy analysis. We formalize the results we obtain from the set up

Web service (Player 1)	Community (Player 2): Accept/Invite Join	Community (Player 2): Refuse/Not Invite Join		
Join/Accept Invitation	J_{w_i,c_j}, A_{w_i,c_j}	$JR_{w_i}^{c_j}, JR_{c_j}^{w_i}$	$J_{w_i,c_j} = E'_{w_i} - E_{w_i}$ $A_{w_i,c_j} = E'_{c_j} - E_{c_j}$	$JR_{w_i}^{c_j} = E_{w_i} - \overline{E_{w_i}}$ $JR_{c_j}^{w_i} = E_{c_j} - \overline{E_{c_j}}$
Stay/Refuse Invitation	$SI_{w_i}^{c_j}, SI_{c_j}^{w_i}$	0 , 0	$SI_{w_i}^{c_j} = E_{w_i} - \overline{E_{w_i}}$ $SI_{c_j}^{w_i} = E_{c_j} - \overline{E_{c_j}}$	- - -

Table 3.1: Payoff regarding 2 players when web service is outside the community.

game between these entities in the following.

Proposition 1. *In one-stage game, there is no pure strategy Nash equilibrium.*

Proof. In the set up one-stage game, the payoff of web services regarding accepted join request (J_{w_i,c_j}) could be either more or less than that of refusing the invitation (as it refers to the actual efficiency evaluation). This is also the case for the master of the community. Consequently, there is no dominant strategy for any player. Therefore, no pure strategy Nash equilibrium can be found. □

As consequence of this proposition, there is no stable situation rational entities can try to achieve by playing the game. Both players should then consider the risk parameter in their strategy selections. To this end, we define web service and community's mixed strategy probabilities respectively as $w_i(S_{w_i}, 1 - S_{w_i})$ and $c_j(S_{c_j}, 1 - S_{c_j})$. Thus, we compute web services expected payoff α_{w_i} of join (or accept to join) versus the mixed strategy profile of the community in Equation 12. Equation 13 computes the related value regarding the refusal of join.

$$\alpha_{w_i}(join, c_j(S_{c_j}, 1 - S_{c_j})) = S_{c_j}(J_{w_i,c_j}) + (1 - S_{c_j})(JR^{c_j}_{w_i}) \tag{12}$$

$$\alpha_{w_i}(stay, c_j(S_{c_j}, 1 - S_{c_j})) = S_{c_j}(SI^{c_j}_{w_i}) + (1 - S_{c_j})(0) \tag{13}$$

The web service aims at maximizing its payoff. Therefore, for all adopted strategies, we need to consider the best response (to the other player) and discard the others. For instance, if the web service obtains a higher expected payoff with the joining strategy, it would change its probability profile to $(1, 0)$, so the join would be the dominant strategy.

Since each player in each stage game chooses between only two strategies, and since any of these strategies could be the best response in a particular situation, we analyze the case where the expected payoffs are equal. By so doing, we can compute a threshold (μ_{w_i}), which is used to identify which strategy is dominant. The threshold μ_{w_i} is used by the master to control the expected payoff of the web service in the sense that the web service adopts the master's desirable strategy as dominant. Thus, the master would pay the least possible cost to obtain its desirable control on the web services. This eventually would lead to the control mechanism of the master web service over cardinality of the community set. The threshold is computed in Equation 14.

$$\alpha_{w_i}(join, c_j(S_{c_j}, 1 - S_{c_j})) = \alpha_{w_i}(stay, c_j(S_{c_j}, 1 - S_{c_j}))$$

$$\Rightarrow S_{c_j}(J_{w_i,c_j}) + (1 - S_{c_j})(JR^{c_j}_{w_i}) = S_{c_j}(SI^{c_j}_{w_i})$$

$$\Rightarrow S_{c_j}(E'_{w_i} - E_{w_i}) + (1 - S_{c_j})(E_{w_i} - \overline{E_{w_i}}) = S_{c_j}(E_{w_i} - \overline{E_{w_i}})$$

$$\Rightarrow \mu_{w_i} = \frac{(S_{w_i})(E'_{w_i}) + E_{w_i} - 3(S_{w_i})(E_{w_i})}{1 - 2S_{w_i}} \tag{14}$$

The threshold μ_{w_i} obtained in Equation 14 is in terms of the estimated efficiency $\overline{E_{w_i}}$, which could be changed by the master c_j. So if the expected efficiency of join is computed to be more than μ_{w_i}, the web service w_i would adopt the join strategy or accept the invitation to join. We have then the following result.

Proposition 2. *In mixed strategy one-stage game, there is a threshold μ_{w_i} such that if $\overline{E_{w_i}} > \mu_{w_i}$, joining the community would be the goal of the web service. Otherwise, the web service w_i would not join the community.*

Corollary 1. *If the master web service considers the expected efficiency value computed by the web service and provides (broadcasts) a reputation value that let $\overline{E_{w_i}}$ exceeds μ_{w_i}, the master can control adopting strategy of the web service.*

3.3.3 Web Service in the Community

In the previous sections, we analyzed the case where the web service w_i was acting alone outside the community c_j. We also set up a game and analyzed the payoffs regarding different adopting strategies. In this part, we analyze the same system where the web service w_i is already acting in collaboration with other web services inside the community c_j. In this case, the web service chooses its actions from strategy profile of “leave/accept to leave" or “stay/refuse to leave" (we assume that any action that ends up in changing the status of the web service is being made upon agreements between the web service and master of the community). The community c_j also refers to the strategy profile of “accept of leave/fire" or “refuse the leave/not fire". Doing so, there are four different cases:

(a) w_i attempts to leave the community c_j, L_{w_i,c_j}, and the attempt is accepted, F_{w_i,c_j};

(b) w_i attempts to leave, $LR_{w_i}^{c_j}$, but c_j refuses the leaving request, $LR_{c_j}^{w_i}$;

(c) c_j encourages the web service w_i to leave, $SF_{c_j}^{w_i}$, but w_i refuses to leave the community, $SF_{w_i}^{c_j}$;

(d) there is neither firing from c_j nor leaving request from w_i.

Similar to the case where the web service was outside the community, we analyze the cases with their parameter updates.

Case (a) The web service w_i that takes the risk of leave (S_{w_i}) would update his reputation, market share and capacity parameters respectively in Equation 15

$$\overline{R_{w_i}} = R''_{w_i} \qquad \overline{Cp_{w_i}} = Cp_{w_i} \qquad \overline{M_{w_i}} = M''_{w_i} \tag{15}$$

In this case, our assumptions are as follows:

1. The reputation of a web service would be back to its previous individual reputation (R''_{w_i}). To this end, each registered web service in the community holds its individual reputation when joining a community. However, the community recalculates its average reputation.

2. We consider the capacity as a fixed attribute. Therefore, the capacity of the web service stays unchanged but the community reduces the left web service's capacity.

A similar analysis can be obtained for the market share where M''_{w_i} is the previous value. The corresponding attribute updates regarding the community c_j are formulated in Equation 16.

$$\overline{R_{c_j}}=\frac{n(R_{c_j}) - R_{w_j}}{n-1} \qquad \overline{C_{c_j}} = C_{c_j} - C_{w_i} \qquad \overline{M_{c_j}} = M_{c_j} - M_{w_i} \tag{16}$$

In this case, both entities consider the estimated parameters and compute their new efficiency values (see Equation 5). The case would take place when the following inequalities hold:

$$\overline{E_{w_i}} \geq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \geq (1 - S_{c_j})E_{c_j}$$

Case (b) In this case, w_i attempts to leave, but the community does not accept the leaving request. The difference between the cases (a) and (b) is the same as explained in the previous section. We have then the following inequalities:

$$\overline{E_{w_i}} \ngeq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \geq (1 - S_{c_j})E_{c_j}$$

Case (c) This case is similar to the case (b) except the fact that the refusal is caused by the web service:

$$\overline{E_{w_i}} \geq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \ngeq (1 - S_{c_j})E_{c_j}$$

Case (d) In this case, both entities are not encouraged to attempt leaving:

$$\overline{E_{w_i}} \ngeq (1 - S_{w_i})E_{w_i}$$

$$\overline{E_{c_j}} \ngeq (1 - S_{c_j})E_{c_j}$$

3.3.4 The Game Set up for the Joined Web Service

In this section, we also develop the game-theoretic analysis consisting of the web service w_i as player 1 and community c_j as player 2. The player 1 follows the strategy profile of (leave/not leave) when is the initiator and follows the strategy profile of (acceptance fire/refuse fire) otherwise. Table 3.2 shows the assigned payoffs for both players in different cases. As shown in the Table, the values of L_{w_i,c_j} and F_{w_i,c_j} are the generalized form of "leave/accept" or "fire/accept join" cases. These values are actual differences in efficiency values after the join (E'_{w_i} and E'_{c_j}). The obtained payoffs could be either positive or negative. We formalize the results we obtain from the set up game between these entities in the following.

Proposition 3. *In one-stage game, there is no pure strategy Nash equilibrium.*

Proof. The proof is similar to the one given for Proposition 1. □

Referring to the obtained payoffs shown in Table 3.2, we would have the same best response analysis that we did in the case for the single web service. To this, the obtained threshold μ_{w_i} is set the same.

Proposition 4. *In mixed strategy one-stage game, there is a threshold μ_{w_i} such that if $\overline{E_{w_i}} > \mu_{w_i}$, leaving the community would be the goal of the web service that is already member of the community. Otherwise, the web service w_i would not leave the community.*

Corollary 2. *If the master web service considers the expected efficiency value computed by the web service and provides a market share value that let $\overline{E_{w_i}}$ exceeds μ_{w_i}, the master can control the strategy of the web service.*

The market share offered to the web service by the community could cause dissatisfaction of the joined web services. Thus, this would generate a low E_{w_i} value, which would cause the web service to leave considering its previous individual efficiency value.

3.4 Proposed Framework and Experimental Results

We used a realistic multi-agent simulator in a `.net` platform and developed many agents with broad range of characteristics and capabilities. In the multi-agent based environment, we exposed dynamism in agents' actions and therefore, we could obtain results that are based on the performed realistic experiments. In the implemented environments, there are three types of agents:

1. user agents,
2. web service agents, and
3. master web service agents that represent communities

We do not emphasize the user agents for the sake of simplicity. However, in general, they look for best possible service (either from a single or a community of web services). During simulation runs, web services and users might leave or join the network.

For our implementation environment, we choose Visual Studio 2010 IDE [12] as our Integrated Development Environment, and `CSharp.net` as the language of programming. For each type of agents, we define an object oriented class. Each class of objects has its own properties and methods (see Figure 3.1). In addition to these classes, we have a constant class which holds basic and static values used for service initialization. This latter file makes the framework more flexible, because we can observe the different behaviors of the agents by changing the set up

values in the constant file very easily. As we mentioned earlier, we do not focus on the user agent class, but only on the web service and master web service classes.

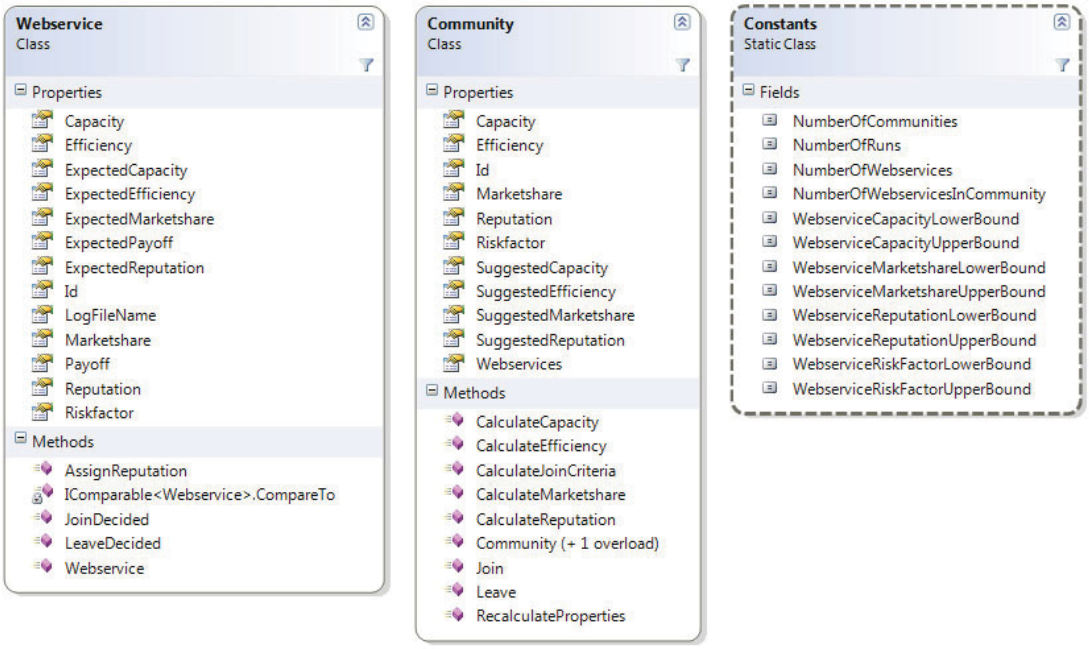

Figure 3.1: Web service, community and constant parameters classes

Web Service has a unique ID and some properties such as Capacity, Reputation, MarketShare, and RiskFactor. These properties are set at the beginning of the simulation. Web service has another important property, namely efficiency which is calculated and updated during the simulation in each run.

Community (Master Web Service) has also a unique ID and several properties such as Capacity, Reputation and MarketShare. Each community has a list of web services that have joined the community.

We tried to design and implement this framework as flexible as possible such that it can be extended for other purposes very easily. In fact, we extended these classes by adding some other properties and methods in Chapter 4, Section 4.4.

Table 3.4 provides the details regarding the implemented environment. We categorize the web services and masters based on the risk they take in adopting strategies. There are three classes of services that obtain different payoffs during the games.

In this section, we investigate the characteristics of the single web services that act alone outside the community. During the simulation runs, we set up a number of one-stage games analyzing the strategies that web services take in different situations. We repeat the same process using three different classes of the web services according to their risk attribute. Figure 3.2 illustrates 6 plots categorizing three different types of single web services that are involved in the one-stage game regarding joining the community. In plots (a), (b), and (c) the x-axis denotes community's public reputation that is broadcasted by the player 2 (c_j) in the game. The y-axis denotes the percentage of the web services that considered to join the community. In this experiment, the community is willing to accept joining web services since its market share is not balanced with its limited capacity ($M_{c_j} > C_{c_j}$). As it is shown in plots, there are different joining percentages regarding the situation that either encourages or discourages most of the web services. In Figure 3.2, plots (d), (e), and (f) illustrate the average efficiency comparison between the case where the web service was acting alone (the dotted curve) and the case where the web service joined the community (the bold curve). The updates in efficiencies clarify the extent to which the joining strategy is chosen wisely. In this experiment, the community adopts its strategies according to its individual efficiency analysis regardless of the threshold that could lead the web services to join. We also launch the experiment with the community representative that is capable of analyzing the threshold that would enhance the control of the master web service over the adopting strategies of the single web services willing to join and obtain higher efficiency. Figure 3.4 plots the same group of web services (categorized in plots (a), (b), and (c)) facing a community whose master

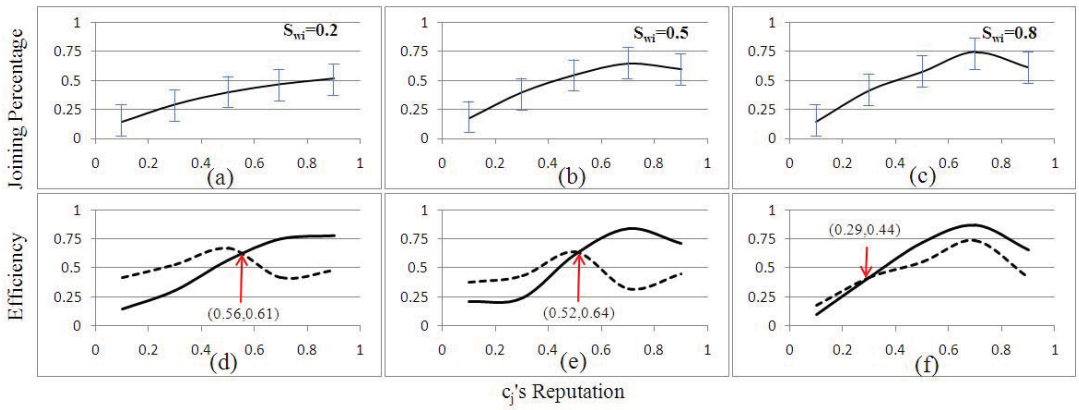

Figure 3.2: Efficiency of three categorized web services on joining a community.

web service analyzes the threshold that could encourage the web services to join. As shown in this figure, c_j is more successful in games with players that hold relatively high risk attribute. In lower risky web services, the community is more successful in absorbing the web services by advertising higher reputation. This fact is promising according to web services' desire to increase self- efficiency. However, the community facilitates the joining process and meanwhile, obtains the control on the strategies that the web services adopt. Thus, the master web service acts better compared to the case where the master web service considers self parameters in games.

We carry on the experiments with analysis on the efficiency updates regarding the joined web services that are involved in one-stage game facing the community representative. Figure 3.4 illustrates 6 plots categorizing three types of web services according to their risk attribute class. These plots illustrate the percentage of leaving the community together with their corresponding efficiency update. As it is clear in plots (a), (b), and (c), the web services with lower risk levels act more or less according to their satisfaction of joining the community. Therefore, the percentage of leaving is decreased by increasing the reputation of the community. Note that the public reputation of the community cannot be faked in this case as long as the web service is already

member of the community. The experiment shows the web services with higher risk level could adopt leaving strategy with weaker reasoning mechanism. Consequently, we observe a more chaotic behavior of the joined web service with higher risk level acting in a community with relatively low reputation value. This chaotic percentage is regulated while the reputation of the community is increased. In this case, the web services consider to refuse the leave.

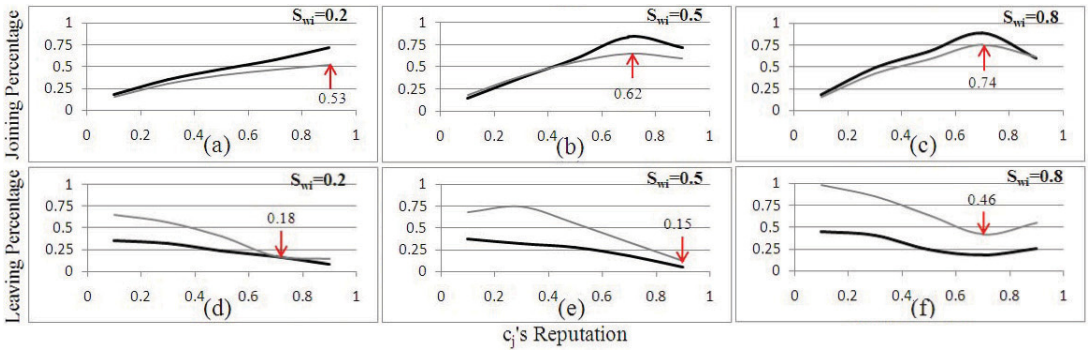

Figure 3.3: Efficiency of three categorized web services on joining and leaving a community while threshold being investigated.

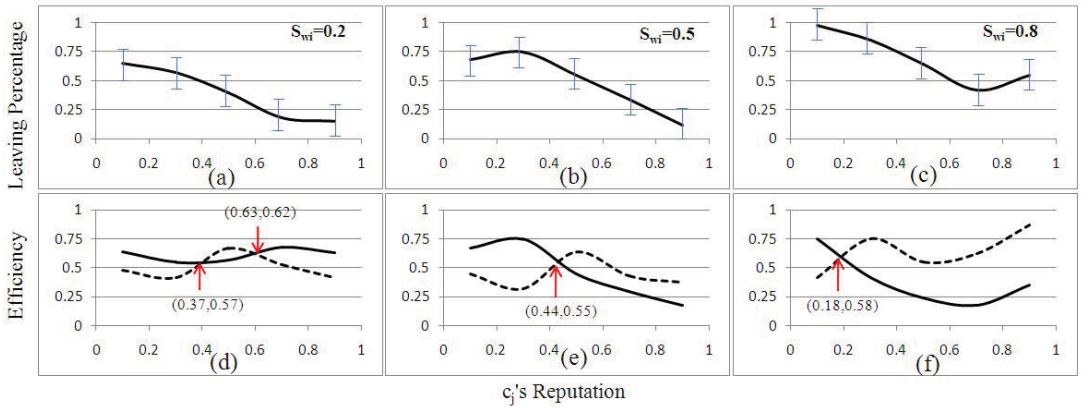

Figure 3.4: Efficiency of three categorized web services on leaving a community.

Figure 3.4 plots (d), (e) and (f) illustrate the leaving percentage of the web services in the same experiment but facing a community that manages to recognize the threshold μ_{w_i}. In these plots we observe a better handling of the web services, which reflects community's success in controlling

the adopting strategies of the web services.

In Figure 3.5, we compare the total efficiency of different communities categorized based on their efficiencies ($S_{c_j} = 0.2$, 0.5, and 0.8). In these plots, the bold curves represent the efficiency of the community when the threshold μ_{w_i} is taken into account and the dotted ones represent the community when the threshold is not taken into account. As shown in the plots, the efficiency of communities are enhanced when they consider the computed threshold. The results show us that the efficiency of the web services is higher when the threshold μ_{w_i} is being considered.

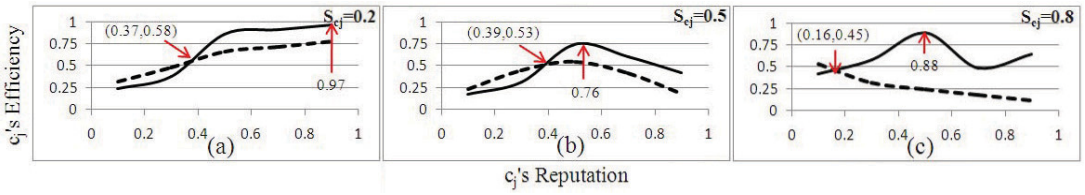

Figure 3.5: Efficiency of three categorized communities of web services.

Web service (Player 1)	Community (Player 2): Accept Leave/Fire	Community (Player 2): Not Fire/ Refuse Leave
Leave/Accept Leaving	L_{w_i,c_j}, F_{w_i,c_j}	$IR_{w_i}^{c_j}, IR_{c_j}^{w_i}$
Stay/Refuse Leaving	$SF_{w_i}^{c_j}, SF_{c_j}^{w_i}$	0 , 0

$L_{w_i,c_j} = E'_{w_i} - E_{w_i}$ $F_{w_i,c_j} = E'_{c_j} - E_{c_j}$	$IR_{w_i}^{c_j} = E_{w_i} - \overline{E_{w_i}}$ $IR_{c_j}^{w_i} = E_{c_j} - \overline{E_{c_j}}$
$SF_{w_i}^{c_j} = E_{w_i} - \overline{E_{w_i}}$ $SF_{c_j}^{w_i} = E_{c_j} - \overline{E_{c_j}}$	- - -

Table 3.2: Payoff regarding 2 players when web service is inside the community.

Entity Type	Number	Risk Level
User	1000	—
Web service	50	0.2
	50	0.5
	50	0.8
Community	5	0.2
	5	0.5
	5	0.8

Table 3.3: Environment characteristics

Chapter 4

Analyzing The Efficiency of Coopetition Strategies of Services within Communities

In the previous chapter, we proposed a community infrastructure designed to host web services as a multi-agent environment. We focussed on web services' efficiencies in their continuous interacting strategies. Using game theory, we analyzed the best acting strategy and the ways a community can constrain high overall performance. In this chapter, we extend this framework by exploring the types of interactions that web services can choose while they reside in a given community. We go further to analyze the impact that each one of these types could make on the overall efficiency of web services.

Deciding about which strategy to choose and when web services are competing but still need to cooperate to accomplish complex tasks is still an open issue. We propose analyzing those different strategies to help web services in their decision making process when these web services function within communities. The objective is to enable web services to reasonably evaluate and decide over their coopetition strategies, which means when to compete and when to cooperate.

More precisely, we propose a mechanism within which web services in the community could choose either to compete within their community for an announced task, or to cooperate with other competing web services in the same community to accomplish some subtasks of the announced task. We explore details behind the strategic decision making procedures and enable web services to apply different techniques to constrain high efficiency and obtain the maximum utility. We investigate web services' expected payoffs and the involved probabilities that are used to choose over the two interacting strategies.

As intelligent entities, agent-based web services require a reasoning technique that enhances their abilities over best acting strategies and the attitude they could exhibit to yield maximum utility. In this chapter, we obtain some theoretical results about the decision making process that could be used by a typical web service.

In the rest of this chapter, we discuss about related work in Section 4.1. Then we define the metrics we use in our investigations in Section 4.2. we extend this chapter by providing a model for web services decision making procedure in Section 4.3. We close this chapter by providing experimental results in Section 4.4.

4.1 Related Work

Our contribution in this chapter is the proposition of a model to analyze the efficiency issue of active services within their communities. The proposed framework considers the best response strategy to maximize the expected payoff and determine the threshold, so that a decision over competing or cooperating can be made.

In many frameworks [3, 19, 22, 24, 35, 37] proposed in the literature, service selection and task

allocation are regulated based on the reputation parameter. In some other frameworks [2, 5, 25] the cooperative behaviors of web services are discussed and observed. The proposed frameworks mostly aim to facilitate the coordination mechanism between web services. However, the opposite strategy of cooperating is not analyzed where web services might be more successful when competing within a same group. In fact, web services are not always willing to cooperate even if they have some common goals, particularly when they operate within groups such as communities. In such a context, web services can follow different interacting strategies and have to decide when to compete and when to cooperate so that their ultimate goal, maximizing their incomes, can be better achieved. In Chapter 3, we discussed about some of the related work; in this chapter, we will address other related work on interactions between web services in communities of web services and the coopetition strategy.

Engineering of communities of web services with the focus on the social interactions between the web services inside the community is the subject of research in [33]. In the proposed framework five types of interactions are defined, which are supervision, competition, substitution, collaboration and recommendation. And the attributes considered for web services are selfishness, fairness and unpredictability. Competition happens when web services try to take part in a composition inside a community and at the same time they cooperate with each other by substitution in the time of failure. This is a simple form of coopetition. However, this type of relation is not discussed in this paper. Here two types of web services are defined: master web service and slave web service. Supervision is master web service task and its goal is to assign requests to the most appropriate slave web services inside the community. For this purpose, a weighted unidirectional edge between master web service and the slave web service is defined. The weight of the edge is dependent on the three parameters, which are the functionality similarity level between master

web service and the slave web service, the trust level which is about how confident the master web service is in the capability of the slave web service to fulfill and satisfy the user's request, and the response level that shows the acceptance rate of the slave web service to the demand of the master web service to satisfy the user's requests. This weighted edge is called Supervision Level. The more the supervision level is close to one, the more similar is the slave web service in functionality to the master web service and hence, is more appropriate to be selected for a user's request. For competition relation, a weighted bidirectional edge is considered between two slave web services. To evaluate the weight of this edge which is referred to as Competition Level, two values are considered, functionality similarity level and non-functionality similarity level. The more closer is the value of competition level to one, the more similar two slave web services are, therefore, the first slave web service threatens the competitiveness capacity of its peer. Another type of collaboration relation is defined in this framework, which is between different groups of slave web services inside a community. These groups are interacting with each other through a specific slave web service named "focus" slave web service. Again in this type of relation, a weighted unidirectional edge is drawn between focus slave web service and slave web services in each group. The weight of this edge is referred to as Collaboration Level and is calculated based on the number of times that both slave web services in different groups of web services participated in joint compositions. Recommendation relation is also defined when at least a good number of compositions are completed and is dependent on the number of times a slave web service accepted to take part in joining a composition and denotes how much that slave web service likes to work with a specific joint composition. The selfishness of a slave web service is when it does not show a positive behavior towards its master web service or other peers in the community. The master web service rewards a slave web service either if that slave web service accepts to

participate in a joint composition or substitute another slave web service when needed. On the contrary, the master web service will penalize the slave web service who does not participate in a joint composition at run-time despite its prior acceptance or if it continuously declines to act as a substitution. After all, the authors discuss about the unpredictability property of a slave web service. This happens when a slave web service accepts the composition request for example, but it acts in opposite.

A heuristic algorithm to evaluate cooperation and competition behaviors of agents towards each other in the environment is proposed in [46]. In this context, only those tasks that need combination of multiple agents efforts to be completed are considered. To this end, both cooperative and competitive approaches are taking place. A two step procedure is proposed. In the first step, the agents have to evaluate the tasks and in the second step, they need to determine the agents which can help them complete the tasks. Four different agent types are defined here. No single agent possesses all the required skills to complete a task individually, so coalitions must be formed. These coalitions are formed in a way that maximizes the payoff received by the agents. The agents are rational and evaluate and compare their payoff in both cases of doing a task individually or participating in a coalition. In this framework, for a particular skill, the base fee is a non-decreasing function of skill level and is publicly known. The fee for a given skill-level pair is taken from a normal distribution with known mean and standard deviation. Four heuristics have been indicated in this work for task selection by agents:

1. Individual profit task selection in which the agent selects the task that maximizes its own profit.

2. Global profit task selection that will maximize global profit.

3. Best fit task selection within which the sum of under-utilization for the agent is minimized.

4. Coopetitive selection that selects the best fit task as long as it is within $P\%$ of the maximum individual profit, for a given P fixed by the designer.

And there are two coalition selection heuristics:

1. Best profit coalition selection in which for each subtask, the agent selects the supporting agent with the minimum fee.

2. Best fit coalition selection within which for each subtask, the supporting agents with the minimum under-utilizations are selected.

The authors hypothesize that if the environment is very competitive or the number of tasks to be done is less than the number of agents, an agent will be more interested in accepting the other agents' proposals.

Based on the previous works done in coalition formation area, there are three main activities in the coalition formation process; determining the value of each coalition, finding the group of agents for collaboration, and dividing the payoffs among the collaborated agents. In [39], a general framework has been introduced for the study of constrained coalition formation. In this framework a class of settings has been proposed to determine which agents should or should not collaborate with each other. This class of constraints is then transformed to a structure to be able to be used in coalition formation algorithms. This latter reduces any redundant computations while identifying all the feasible coalition formations. Another challenge addressed in this paper is about how to combine the feasible coalitions to have a more efficient coalition structure. The authors have tried to use simple but practical expressive constraints in this work. The constraint transformation algorithm which is a procedure to transform a set of defined constraints

into another isomorphic set is based on the *Divide and Conquer* approach. Building upon this algorithm, an algorithm for optimal coalition structure generation in basic coalition formation games is proposed.

The behavior of web services with the same functionalities and the architecture and interactions between communities of web services within which these web services reside are discussed in [15, 28, 32] . The proposed frameworks are all working based on reputation and QoS metrics. In [15], mostly the architecture of the communities of web services and the performance metrics as perceived by users and service providers are considered. The proposed framework focuses on how to structure and update reputation, how to make it accessible and how to keep it up-to-date. This reputation is important from two perspectives, users' perspective so they can decide to select which community to send their requests to and providers' perspective so that they can better decide whether they should let their web services join a community or should not. The communities of web services are discussed from another point of view which is about the interactions between web services inside a community and the communities with each other in [32]. The first one exhibits coopetition that is the simultaneous cooperative and competitive behaviors between web services and the second one is the competition between different communities of web services. Inside a community, web services compete with each other to participate in web service compositions since all of them offer the same functionality but with different non-functional properties such as QoS metrics, and when a failure occurs, they cooperate with each other for the substitution. Again for communities of web services both users and providers perspectives should be taken into account. For service providers, the high participation rate of communities is important and for users, their provided service and users satisfaction from the service is the main factor for selecting a community. The process and the important factors in selecting web services

inside a community which satisfies all three user, web service and community parties is the main contribution in [28]. This selection is done by the master web service. The satisfaction of web service is important because a satisfied web service will stay in the community and will offer more services and it will affect the QoS of the community; contrarily, if a web service with an average or high QoS leaves a community it can badly affect the promised QoS of the community to the user.

The rest of the related works in this chapter are related to the coopetition in industrial ecosystem. One of the industrial economy models that has been proposed as the multi agent architecture is [42] which concerns the investigation of a life cycle of a network in which the agents change dynamically. Here the agents are enterprises that can join or leave a network during the life cycle of the network while the enterprises that operate outside the network are capable of evaluating the participation in the network.

An explorative case study of two Swedish and one Finnish industries, within which both competitive and cooperative relationships can be found at the same time is used in [4]. The concept of coopetition and how to manage the different logics of cooperation and competition between actors in industries and how to divide and manage these two different activities are investigated. The authors show that each individual in a firm can just show one type of behavior at a time and that, this division should be done between individuals or can be controlled by an intermediate actor.

Negotiations were always characterized only as competitive transactions; a coopetitive negotiation process has been provided in [11] that leads to a business insurance policy. Negotiation is modeled as an iterative questions and answers which generate a common knowledge base of client's requirements and supplier's quality of service. The characteristics of underwriting of the

insurance policy are defined based on this common knowledge base. The process of the common knowledge base creation is a cooperative process.

The scheduling problem is addressed in [40, 41], i.e. the problem of assigning resources to a production task in a distributed and dynamic environment. Coopetition approach is used to solve the problem in two steps. In the first step which is cooperation step, a Request Session is opened and committed to a Request Session Agent. In the second step that is competition step, the participating Product Agents compete with each other to gain their individual goals. This latter is solved by means of the game theory.

A multi agent model to simulate coopetitive behaviors between companies in the supply chain in a single market is proposed in [52]. By comparing two extreme behaviors of cooperation and competition with coopetition, they show that the latter is a more effective strategy in long-term. In this simulated environment, the firms select their partners randomly and based on the strategy they adopt. Then a comparison between some statistical properties like degree distribution, weight distribution and degree correlation is done and shows that strictly competitive strategies dominate over strictly cooperative strategies in the markets. But yet in the long term, the coopetitive strategy makes more business friends and less production costs.

Matching potential benefits of web services while cooperating with one another as a framework is proposed in [17]. The interesting idea is to consider the benefits under four categories: innovation and learning, internal business process, customer, and financial benefits. This conceptual framework uses the theory of balanced scorecard concept to provide a guideline for identifying and deploying web services to enhance a firm's competitive advantage. The balanced scorecard is used as a performance management mean to keep track of the activities inside corporations. Based on [17], unless the Web services technology is used to create or enhance an IT strategy

that is aligned with the competitive strategy of the firm, the strategic benefits of implementing web services cannot be realized.

A dependable (i.e., intrusion-tolerant) infrastructure for cooperative web services coordination that works based on the tuple space coordination model is presented in [2]. This infrastructure processes interact through a shared memory abstraction where generic data structures are stored and retrieved. The proposed infrastructure implements several security mechanisms that make the system tolerate accidental faults as well as malicious attempts to disable the operation of system components. This architecture provides the integration of a dependable tuple space on the web services and allows the services to solve important problems like leader elections.

Another multi-agent model which is a knowledge-empowered agent information system is proposed in [51]. This system uses a payoff mechanism as a means of rewarding consumers for sharing their personal information with online businesses. In this system several agents employ various knowledge and requirements for personal information valuation and interaction capabilities that most users cannot do on their own. The agents work on behalf of consumers to categorize their personal data objects, report to consumers on online businesses' trust and reputation, determine the value of their compensation using risk-based financial models, and finally negotiate for a payoff value in return for the dissemination of users' information. The aim of this incorporation is to maximize the consumers' benefit.

To summarize, although many interesting frameworks have considered the coopetition strategies, no one has considered the best response analysis, which we used to calculate the threshold in our proposed framework, as a technique to decide when to compete and when to cooperate, particularly in the context of communities, were agent-based web services are supposed to compete and cooperate in different situations, which makes our proposal different from all the others.

4.2 System Metrics

Task QoS (QoS^t_{task}) is the required QoS metric for a specific task at time t. Users define tasks with specific QoS requirements such as response time, availability, and successability (or accuracy) [28]. We aggregate and normalize these metrics to a value between 0 and 1.

Web service QoS (QoS^t_w) is the QoS provided by the web service w after performing a task at time t. Again the metrics that contribute in computing this QoS are aggregated and normalized to a value between 0 and 1. The offered quality might or might not meet the required task quality QoS^t_{task}. In the latter case, the service user would be disappointed and a negative feedback is expected. In our proposed system, both cases are considered when calculating the web services' reputation.

Budget (B^t_w). Here we should mention that in our model, each web service has to pay a fixed fee as membership fee to the community to be able to reside in it. B^t_w is the amount of money the web service w has in its disposal at time t, which helps pay for the community membership fees (ϵ) and is one of the parameters that the web service considers when deciding about getting involved in a competition or not which we are going to discuss more when calculating the confidence of the web service in Equation 19. This parameter has been used in other service computing settings such as [28].

Reputation is a factor in any online community where trust is important. Without any trust enabling mechanism, users cannot differentiate between services, specially the ones which offer the same type of service. Reputation mechanisms usually aggregate users' experiences and in our case it strongly depends on QoS that each web service provides. Users define tasks with specific quality QoS^t_{task}, therefore after performing a task with QoS^t_w, the reputation of w gets evaluated

by the master web service. Rep_w^t refers to the reputation of w at time t.

In Equation 17, we compute the reward, a value between 0 and 1, that the master web service computes considering the task QoS compared with the web service provided quality QoS_w^t. In case the provided quality meets user expectations, the reward value would be positive. In this system, we consider a small value as default reward η which the master considers together with the proportional level of satisfaction as a weighted value (by υ). In this case, the higher the provided quality is, the more weighted reward is. In case the provided quality does not meet the user expectations, the reward will be negative. In this case, we also have a default penalty value ρ (where $\rho > \eta$) together with the weighted proportional difference. The idea is to harshly penalize the web services rather than rewarding them. To this end, rational web services should carefully analyze their capabilities once the available tasks are announced. In our proposal, web services have the goal of increasing their budget, which is directly related to their reputation. Thus, they have to decide strategically how to maximize this value.

$$reward_w^t = \begin{cases} \eta + \frac{QoS_w^t}{QoS_{task}^t + QoS_w^t} * \upsilon & \text{if } QoS_{task}^t \leq QoS_w^t; \\ -(\rho + \frac{QoS_{task}^t}{QoS_{task}^t + QoS_w^t} * \upsilon) & \text{otherwise.} \end{cases} \tag{17}$$

The assigned reputation value is updated by the computed reward value. The computed reputation of web service is bounded by the minimum and maximum reputation values 0 and 1. Let $\Gamma = Rep_w^t + reward_w^t$. The updated reputation value is then computed as follows:

$$Rep_w^{t+1} = \begin{cases} \Gamma & \text{if } 0 \leq \Gamma \leq 1; \\ 0 & \text{if } \Gamma < 0; \\ 1 & \text{if } \Gamma > 1. \end{cases} \tag{18}$$

Growth Factor is a parameter which declares web services' performance based on their recent strategies and activities. Growth factor is relative to web services' reputation and QoS. This parameter is the main variable a typical web service uses to decide which strategy to adopt. The details about the decision making process is described in Section 4.3. We use Equation 19 to compute the growth factor G_w^t of the web service w at time t. The growth factor function should be monotonically increasing in QoS_w^t, Rep_w^t and B_w^t, which is satisfied by the equation and this could be easily proven by calculating the partial derivatives of this function in 1) QoS_w^t; 2) Rep_w^t; and then 3) B_w^t and show that they are all positive. The contribution of the budget B_w^t in the calculation of the growth factor should be proportional to the ideal budget $\beta_w \times t$ where the web service receives all the offered tasks during the periods t. The parameter β_w denotes the profit obtained considering the mean received service fee μ_w and the cost of community membership ϵ. The mean service fee depends on the strategy adopted by the web service because a competitive service receives higher fees $\mu_{w,CM}$ compared to a cooperative one $\mu_{w,CO}$ ($\mu_{w,CM} > \mu_{w,CO}$). The motivation behind this is that a competitive web service for a given task is considered as the leader web service for that task while other cooperative web services are performing specific subtasks as asked by the leader. For this reason, while considering to reward or penalize web services after their service, the reward or penalty will be divided proportionally among cooperated web services and the leader web service will receive more reward or penalty comparing to the cooperative ones.

Table 4.1: List of proposed system parameters.

Notation	Definition
QoS^t_{task}	Required task QoS at time t
QoS^t_w	Web service w QoS at time t
Rep^t_w	Reputation assigned for w at time t
$Reward^t_w$	Reward to update the reputation
G^t_w	Growth factor of w at time t
τ^t_w	Coopetitive threshold of w at time t
U^t_w	Utility of w at time t
B^t_w	Budget associated to web service w at time t
ϵ	The community membership fee
$\mu_{w,CM}$	Mean service fee for competing w
$\mu_{w,CO}$	Mean service fee for cooperating w
β_w	Profit of w
COF^t_w	Cooperation fee of w at time t
$\pi^t_{w,CM}$	Competition payoff of w
$\pi^t_{w,CO}$	Cooperation payoff of w
$p^t_{w,CM}$	Competition probability of w at time t
$p^t_{w,CO}$	Cooperation probability of w at time t

$$G^t_w = \frac{Rep^t_w + QoS^t_w + \frac{B^t_w}{t \times \beta_w}}{3} \qquad with \begin{cases} \beta_w = \mu_w - \epsilon \\ \\ \mu_w \in \{\mu_{w,CM}, \mu_{w,CO}\} \end{cases} \tag{19}$$

The above explained parameters and other additional parameters which we will use in the rest of this chapter are listed in Table 4.2.

4.3 Web Service Decision Making Procedure

Figure 4.1 illustrates the architecture of a typical community aggregating a number of web services with different interactive strategies. Some of them compete for the task where they directly deal with the master. Some others cooperate in the associated task where they only deal with the

competed web service as the task leader and do not directly interact with the master (the master deals only with the web service that has bid for the task, which is responsible of choosing its collaborative network). In both sets, some web services are for certain moments out of any collaboration network. We highlight details of the interactive strategies in the rest of this section.

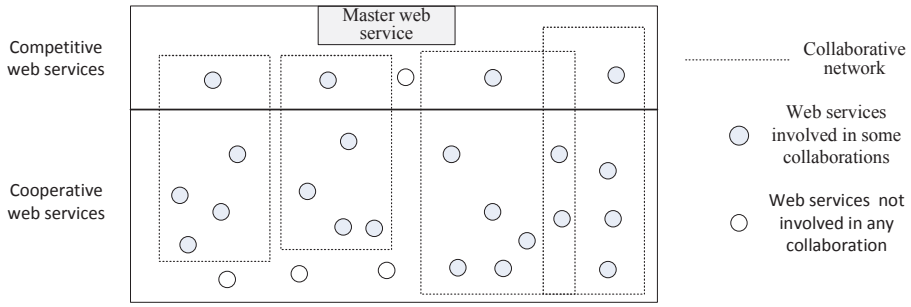

Figure 4.1: A typical community of web services with competitive members and their collaboration networks.

In the proposed system, the master recognizes the competing web services by checking their competing flags and sorts them based on some parameters (such as reputation) that we explain in the rest of this section, and selects the top ranking one. If the web service is busy or unwilling to take the task, the master allocates the task to the second competing web service in the list. There is a chance that some tasks could not be assigned to any web service. These tasks are accumulated to the task pool to be allocated in the next task allocation round. Upon allocation of the task, the web service is responsible to provide the required QoS that is stated in the task being generated by a service user. Afterwards, the master rewards or penalizes the competing web service by upgrading or degrading its reputation according to the provided QoS compared with the required one. This comparison influences the sorting mechanism used by the master to allocate the tasks in further task allocation rounds.

The main goal of each individual web service is to increase its income (payoff). This income

can be earned from tasks (or requests) done by this web service. In our model, web services can decide to compete to get a task from the master web service or to cooperate with other web services in a given collaborative network (the way a collaborative network is set by a leader is based on the cooperative web services reputation and their QoS parameters that should coincide with the required QoS). Therefore we define two types of web service strategies; when a web service has higher level of confidence based on its growth factor, it can compete to get a task from the master and adopts the competitive strategy. On the other hand, when it has a lower level of confidence that it does not feel it can compete with other web services to get a task, the web service waits for some other web services to cooperate with for completing a task and thus it adopts the cooperative strategy. It is worth mentioning that this web service is not providing any service alone, but only cooperates with other services. This means that over all provided service is maintained at the certain level of quality. Web services estimate the outcome of all the strategies and choose one of them accordingly. This decision is not static but can change over time so web services can switch from one strategy to the other and this dynamic attitude is referred to as coopetition. We discuss about this decision making process in more details in the rest of this section.

The main part of web services' decision making procedure falls into their growth factor analysis. In fact, the growth factor and its comparison to a particular threshold is the main reason that influences the web service's decision to follow either competitive or cooperative behavior. Web services initially compute this value and compare it with their computed threshold. Generally the main challenge is the threshold computation and we cope with this issue in the rest of this section. We additionally use the obtained results in the implemented environment and analyze their effectiveness on web services' strategic decision making procedures.

Figure 4.2 shows the decision making process that is followed by a typical web service. In case the web service is ready to compete, there is a chance that it bids for a task if it has the required capabilities to accomplish that task, or stays silent and returns to the cooperative status. But in case the web service is willing to cooperate, it has to wait for a cooperation opportunity that could be triggered by another web service that competed and obtained the task, so both web services will be part of the same collaborative network. Notice that we are not talking about another possible strategy of the web service which is not getting involved in any task accomplishment. Because in this case, the web service will have decrease in budget without any gaining, considering that each web service has to pay membership fees to be able to stay inside the community. In the decision making process presented in Figure 4.2, we assume that the competing web service might get the task (denoted as $Bid/obtainedTask$) or not get the task in case of being rejected by the master web service, or do not even bid for the task (denoted as $Silent/rejectedTask$). For simplicity reasons and without loss of generality, we group the two cases of $Bids$ and $obtainedTask$ together as well as $Silent$ and $rejectedTask$. The rational behind this aggregation is the fact that our main concentration is web services' status (competitive or cooperative) over different decision making rounds which could be caused by internal factors (the web services) or by the external factor (the master web service).

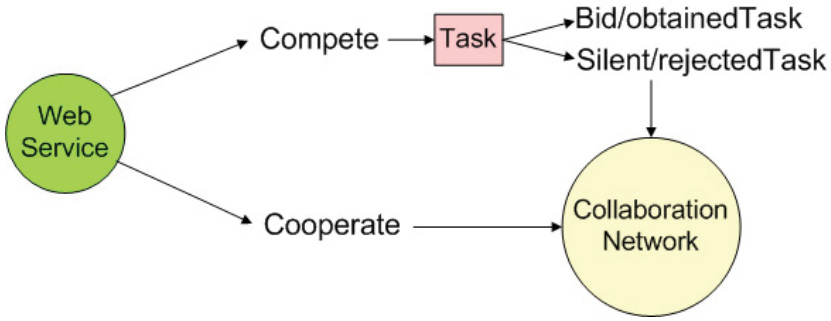

Figure 4.2: Decision making process over competitive and cooperative strategies.

Consider a web service w that is willing to compete (that means the computed growth factor is more than the analyzed threshold τ_w^t). This web service can estimate the expected payoff associated to this decision, called competition payoff. Equation 20 computes this expected payoff for web service w ($\pi_{w,CM}^t$) considering the $Bid/obtainedTask$ probability of $p_{w,CM}^t$ and $Silent/rejectedTask$ probability of $1 - p_{w,CM}^t$.

$$\pi_{w,CM}^t = p_{w,CM}^t(\mu_{w,CM} - COF_w^t - \epsilon) + (1 - p_{w,CM}^t)(-\epsilon) \tag{20}$$

In Equation 20, $\mu_{w,CM}$ is the mean service fee that is assigned by the master web service to w. This means that a competing web service w directly obtains this fee from the master web service. Moreover, the competing web service w expects a cooperation fee (COF_w^t) that it gives to its collaborators in case w needs to cooperate with other web services (cooperative web services in its collaboration network). In any case, the competing or cooperating web service pays a fixed amount of membership (ϵ) to the master web service as the coordinator of the community. This fee would be taken into account when a web service decides to leave to a cheaper community or act alone. But to concentrate on the main concerns of this chapter, we skip these small details.

Similar to the competitive web service case, if a web service w declares cooperative status, its expected cooperation payoff ($\pi_{w,CO}^t$) is computed in Equation 21. In this equation, $p_{w,CO}^t$ is the probability of getting involved in a cooperative task with other web services and $1 - p_{w,CO}^t$ is the probability of failure to find such a cooperation opportunity. These two probabilities are set when w decides to compete. We recall that $\mu_{w,CO}$ denotes the mean cooperation fee that is directly obtained from the leader (i.e., the competitive) web service of the underlying collaborative network. Compared to $\mu_{w,CM}$, $\mu_{w,CO}$ is relatively smaller since the competitive web service

generally dedicates a portion of its obtained income to pay other cooperative web services.

$$\pi^t_{w,CO} = p^t_{w,CO}(\mu_{w,CO} - \epsilon) + (1 - p^t_{w,CO})(-\epsilon) \tag{21}$$

To analyze the expected payoffs obtained from different strategies, web services need to compute the estimated probabilities that distinguish subcases in each behaviorial status ($p^t_{w,CM}$ for competitive and $p^t_{w,CO}$ for cooperative). To estimate these probabilities, we should notice that they are functions of web services' reputation values (Rep^t_w). Furthermore, $p^t_{w,CM}$ is also function of the difference between the provided QoS (QoS^t_w) and the requested one (QoS^t_{task}); and $p^t_{w,CO}$ is function of the reputation of other web services in the community because the leader is supposed to be selective when it comes time to choose the collaborators. To this end, we first discuss the desirable properties of an estimation function of each of these probabilities, and show that the proposed ones satisfy those properties. The desired properties of $p^t_{w,CM}$ are as follows:

Property 4. *$p^t_{w,CM}$ is continuous with regard to Rep^t_w, QoS^t_w, and QoS^t_{task}.*

Property 5. *$p^t_{w,CM}$ is monotonically increasing/decreasing in Rep^t_w and $QoS^t_w - QoS^t_{task}$ while $QoS^t_w - QoS^t_{task}$ is positive.*

Property 6. *$p^t_{w,CM}$ is zero if $QoS^t_w - QoS^t_{task}$ is negative.*

Property 7. *The increase slope of $p^t_{w,CM}$ is higher when the reputation Rep^t_w increases in the interval $[0, 0.5]$ than when it increases in the interval $[0.5, 1]$.*

Property 4 simply says that the probability of successful competition $p^t_{w,CM}$ can be always estimated as far as Rep^t_w, QoS^t_w, and QoS^t_{task} are available. Property 5 says that the reputation

and QoS are two key factors that influence the value of $p^t_{w,CM}$ in the sense of positive correlation. Property 6 indicates that the probability $p^t_{w,CM}$ is zero if the provided QoS is less than the expectation. Property 7 promotes the increase of the reputation for new comers and imposes higher increase rate at the beginning of the reputation curve because it is always hard to build the reputation, but once it is built, its maintenance is less challenging.

The desired properties of $p^t_{w,CO}$ are as follows:

Property 8. *$p^t_{w,CO}$ is continuous with regard to Rep^t_w and the reputation of other web services in the community.*

Property 9. *$p^t_{w,CO}$ is monotonically increasing in Rep^t_w and monotonically decreasing in the community's average reputation.*

Property 10. *The increase slope of $p^t_{w,CO}$ is higher when the reputation Rep^t_w increases in the interval $[0, 0.5]$ than when it increases in the interval $[0.5, 1]$.*

Property 8 is similar to Property 4. Property 9 says that w has more chance to get involved in a cooperation if it has high reputation compared to the other members. This chance decreases if other web services have higher reputations. Property 10 is similar to Property 7.

Equations 22 and 23 respectively compute the estimated success probability in cases where the web service w is competing and cooperating. These values are computed considering web service's reputation value (Rep^t_w computed by the master), web service's provided QoS (QoS_w), the task required QoS (QoS^t_{task}) which is the mean required QoS computed from previous tasks, the maximum provided QoS (QoS^t_k, which is provided by another competitive web service k), and the cooperative factor CL^t_w of the web service w at time t, which is computed as the portion of web service's current reputation on the average reputation of the community.

$$p^t_{w,CM} = \begin{cases} \sin(Rep^t_w \frac{\pi}{2}) \frac{QoS^t_w - QoS^t_{task}}{Max_k(Qos^t_k - QoS^t_{task})} & \text{if } QoS^t_w \geq QoS^t_{task}; \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

$$p^t_{w,CO} = sin(Rep^t_w \frac{\pi}{2}) CL^t_w \tag{23}$$

$$CL^t_w = \frac{Rep^t_w}{\sum_{k \in Community} Rep^t_k / |Community|}$$

It is easy to prove that Equation 22 satisfies Property 4. The partial derivative $\frac{\partial p^t_{w,CM}}{\partial Rep^t_w}$ is positive as the function $\cos$ (the derivative of $\sin$) is positive on $[0, \frac{\pi}{2}]$ and $Rep^t_w \in [0, 1]$. The partial derivative $\partial p^t_{w,CM}$ with respect to $QoS^t_w - QoS^t_{task}$ is also positive, which proves the satisfaction of Property 5. Property 6 is straightforward. Finally, the increase slope of the function $\sin$ on $[0, \frac{\pi}{2}]$ proves Property 7.

Similarly, we can easily prove that Equation 23 satisfies Property 8. Property 9 can be shown by calculating the partial derivative $\partial p^t_{w,CO}$ first with regard to Rep^t_w and second with regard to the community average reputation $\sum_{k \in Community} Rep^t_k / |Community|$. The first partial derivative is positive and the second is negative, which proves the satisfaction of Property 9. The proof of satisfaction of Property 10 is similar to the one of Property 7.

In this part, we compute the coopetition threshold that a typical web service could use to adopt reasonable interacting strategies and we empirically verify the effectiveness of the obtained results in the next section. In fact, to decide which strategy to adopt, we let the web service w compare its growth factor (G^t_w) with the coopetition threshold it holds at current time t (τ^t_w) and choose to compete with probability P^t_w that we compute in Equation 24. Based on this probability, we calculate the total utility U^t_w in Equation 25.

$$P_w^t = \begin{cases} \frac{G_w^t}{\tau_w^t} & \text{if } G_w^t \leq \tau_w^t; \\ 1 & \text{otherwise.} \end{cases} \tag{24}$$

$$U_w^t = P_w^t(\pi_{w,CM}^t) + (1 - P_w^t)(\pi_{w,CO}^t) \tag{25}$$

The key factor in the computation of the probability P_w^t and the associated utility is the threshold value. To compute the threshold, we use the game theoretic best response technique. A typical web service w will follow the best response strategy to maximize its expected aggregated payoff. The idea is to equalize the expected payoffs of the two acting strategies: compete and cooperate. The objective behind equalizing payoffs is to explore conditions under which the web service w could react with best response to further decision making procedures. We use the obtained conditions to compute the threshold τ_w^t at time t. By equalizing $\pi_{w,CM}^t$ and $\pi_{w,CO}^t$, we obtain:

$$\pi_{w,CM}^t = \pi_{w,CO}^t \rightarrow$$

$$p_{w,CM}^t(\mu_{w,CM} - COF_w^t - \epsilon) + (1 - p_{w,CM}^t)(-\epsilon) = p_{w,CO}^t(\mu_{w,CO} - \epsilon) + (1 - p_{w,CO}^t)(-\epsilon)$$

The fixed membership fee ϵ could be taken out and canceled from both sides of the equation so we obtain the following:

$$p_{w,CM}^t(\mu_{w,CM} - COF_w^t) = p_{w,CO}^t(\mu_{w,CO}) \rightarrow$$

$$sin(Rep_w^t \frac{\pi}{2}) \frac{QoS_w^t - QoS_{task}^t}{Max_k(Qos_k - QoS_{task}^t)}(\mu_{w,CM} - COF_w^t) = sin(Rep_w^t \frac{\pi}{2}) CL_w^t(\mu_{w,CO})$$

By simplifying the $sinus$ variable from both sides and substituting the cooperation factor CL_w^t of the web service w we obtain the following:

$$\frac{QoS_w^t - QoS_{task}^t}{Max_k(Qos_k^t - QoS_{task}^t)}(\mu_{w,CM} - COF_w^t) = \frac{Rep_w^t}{\sum_{k \in Community} Rep_k^t}(\mu_{w,CO})$$

We use the obtained equation to obtain the cooperation fee COF_w^t that is assigned by the web service w. This fee represents the amount that w spends to cooperate with other web service(s) to accomplish the task. By so doing, we obtain the maximum amount of cooperation fee that the web service w can use to constrain the positive payoff out of competing. Otherwise, the web service stays as cooperative entity. The cooperation fee is computed in Equation 26.

$$COF_w^t = \mu_{w,CM} - \frac{Rep_w^t}{\sum_{k \in Community} Rep_k^t} \frac{\mu_{w,CO} Max_k(Qos_k^t - QoS_{task}^t)}{QoS_w^t - QoS_{task}^t} \tag{26}$$

We use the maximum cooperation fee that a web service considers to constrain the positive expected payoff when the competitive strategy is adopted to update the threshold for the consequent time interval $(t + 1)$. We compare the maximum cooperation fee with the required fee $(ReqF_w^t)$ that the web service indicates to accomplish the task. The outcome of this comparison is a factor that uses the current threshold τ_w^t to compute the consequent threshold τ_w^{t+1}. As in online learning, the idea is to compute iteratively the threshold until the fixed point is achieved, which indicates the threshold's conversion, where the initial value is randomly chosen (in the simulation different initial values are used). Equation 27 shows this computation.

To investigate the effectiveness of this threshold on the outcomes of the web services that follow this reasoning technique, in the next section, we compare the results of different agents with diverse strategic reasoning techniques.

$$\tau_w^{t+1} = \begin{cases} \Theta & \text{if } 0 \leq \Theta \leq 1 \\ 1 & \text{if } \Theta > 1; \\ 0 & \text{if } \Theta < 0. \end{cases} \qquad with \quad \Theta = \tau_w^t \times \frac{COF_w^t}{ReqF_w^t} \tag{27}$$

4.4 Experimental Results

In this section, we provide an empirical analysis over the observed results regarding the characteristics of intelligent web services hosted in different communities of web services. In the implemented system, we simulate the behaviors of service users as request generators, web services as service providers, and master web services as community representatives. These entities are developed with respect to what is explained in Section 4.2. The objective is to investigate the effectiveness of the proposed strategic system on intelligent web services' overall budget. We study the overall performance of the community hosting the reasoning-empowered web services compared to the ones hosting stochastic and purely competitive services.

The simulation application is written in `CSharp.Net` using `Visual Studio.Net 2010` [12]. Developed web services are initialized with values taken from a real dataset [1][1]. This dataset represents 2507 real web services that exist on the web. It includes the QoS values of 9 parameters including availability, throughput and reliability. These QoS values were determined by monitoring the web services over a 6 day period. We equip some of those web services with our proposed strategic decision making procedure and compare the performance of the equipped services against other ordinary web services.

[1]Our implemented framework uses the QWS dataset provided by Eyhab Al-Masri and Qusay H. Mahmoud that is freely available at: www.uoguelph.ca/ qmahmoud/qws.

We start our discussions with cumulative budget comparison regarding different communities within which services follow different reasoning techniques. Figure 4.3 part (a) illustrates three graphs for three different communities. Each community hosts web services that follow different reasoning techniques:

1. a community that follows the interactive reasoning techniques presented in this chapter (referred to as coopetitive),

2. a community that follows a random reasoning technique, so decisions about selecting competitive or cooperative strategies are totally random (referred to as random coopetitive), and

3. a competitive community where all services follow the competitive strategy (referred to as competitive)

The proposed model's reasoning mechanism enables services to effectively select their interacting strategies and the obtained budget represents the best outcome over the strategic decision making procedure they run all the time. This procedure avoids cases where a service selects the competitive strategy but gets refused to obtain a task from the master. The procedure allows services to make decisions that maximize their utilities, so that if the web service cannot compete, the procedure would suggest to cooperate, which is better than competing and failing to obtain the task. In this case (i.e., competing and not getting the task), the service stays idle but still pays the community membership fee, which means losing utility. The developed strategic decision making mechanism leads some web services to follow cooperative strategies that overall maintain an optimal community budget. In the same Figure, we observe the cumulative budget of a community where services follow random interacting strategies. The outcome is clearly lower because

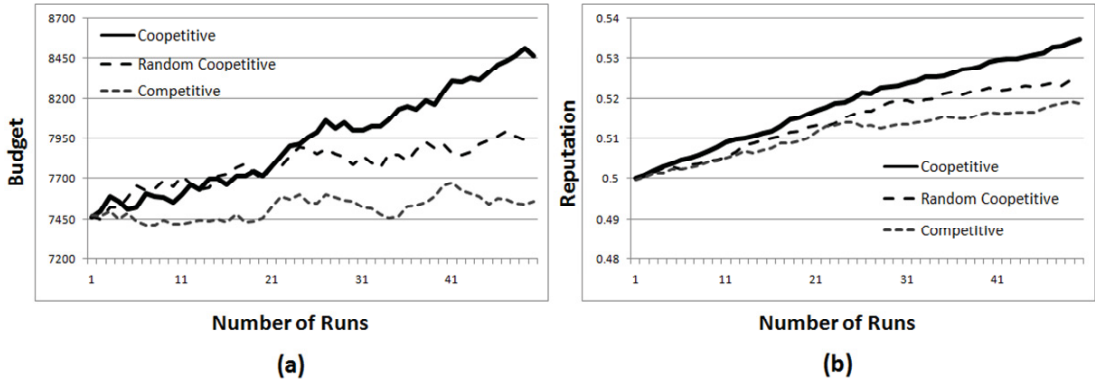

Figure 4.3: (a): Cumulative community budget comparison. (b): Average community reputation comparison over different strategic decisions.

services at each run randomly decide over their acting strategies. This potentially influences the community budget because a low quality service if randomly selects to follow the competitive strategy, it will fail to get a task. This kind of strategy selection is totally stochastic while the task allocation algorithm follows a logical path. The ideal system is the one that analyzes the optimal strategic path and consistently follows strategies that bring maximum outcome. The result regarding the community that follows the random strategy shows how stochastic decision making degrades the community budget, but still this result outperforms the one of the purely competitive community. In the last case, all the services follow the competitive strategy and the frequency of getting rejected from a task is relatively high. In a community with limited number of tasks, the competitive strategy for all services highly influences the community budget because a potential group of services are losing at every run.

The results illustrated in Figure 4.3 (a) verify the importance of the strategic decision making procedure to logically decide over the possible competitive and cooperative choices. Figure 4.3 (b) illustrates average reputation of involved web services in communities. The graphs represent the influence of the rewards that the master web service imposes to encourage highly capable

web services to compete for a task. As for the cumulative budget, we observe that the coopetitive community outperforms the random coopetitive and competitive communities in terms of average reputation. The proposed model's average reputation increases because web services follow optimal strategies where they can perform better so obtain higher rewards. For the same reasons as for the cumulative budget, the average reputation of the random coopetitive community outperforms the one for the competitive community.

In Figure 4.4 (a) and (b), we observe the competitive and cooperative probabilities of four different web services where two of them (w_1 and w_3) are following optimal strategies (competitive for w_1 and cooperative for w_3) and the two others (w_2 and w_4) are following non-optimal strategies. Over elapsing runs, web services that follow optimal strategies bring best budget. In fact, the master web service rewards the high quality web service that chooses the competitive strategy, cooperates with other web services and successfully accomplishes the task. In this system, the reputation regarding such a web service is increasing over time and the possibility of allocating further tasks is increasing as well. By increasing the growth factor, such a web service (here shown as w_1) increases the probability of selecting the competitive strategy. On the other hand, the other web service (here shown as w_2) that is incapable of competing is penalized by the master web service because the provided quality might not meet the required task quality. Thus, w_2 degrades its growth factor by following the competitive strategy. As intelligent entity, this web service is encouraged to change its strategy to the cooperative one and thus, its probability of selecting the competitive strategy is decreasing over time. We have similar results in Figure 4.4 (b) regarding web services w_3 and w_4 where unlike w_4, w_3 is strategically following the cooperative strategy. Therefore, w_4 is more seeking the competitive strategy where it can increase its growth factor.

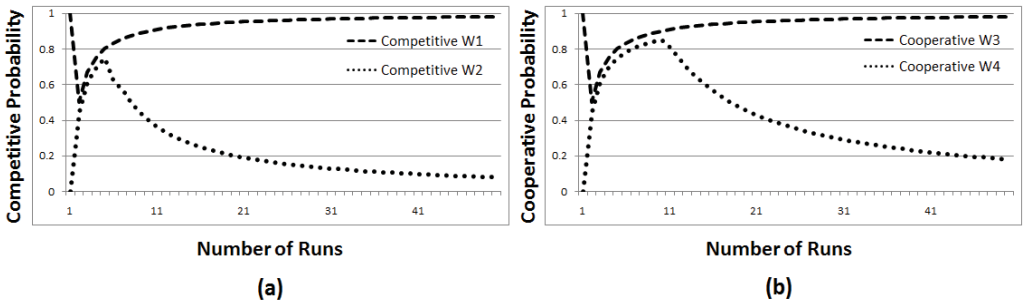

Figure 4.4: Competitive and cooperative probabilities regarding four different web services.

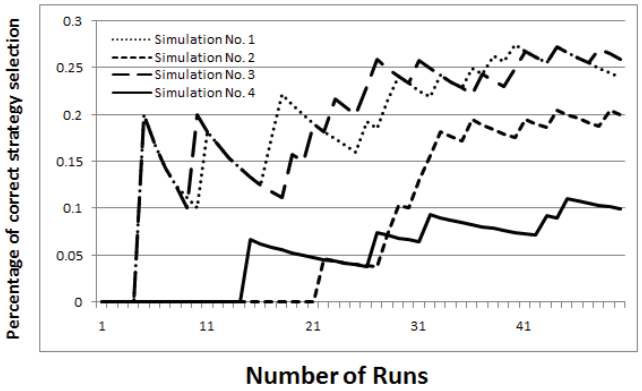

Figure 4.5: Percentage of correct strategy selection in different runs using different initial threshold values.

We conclude our analysis by discussing the percentage of correct strategy selection (Figure 4.5). By this we mean the average of web services that can get a task when the coopetitive strategy is followed. We observe that in different runs, this percentage is generally increasing. This shows that over runs the web services get to learn the threshold τ_w^t that allows them to select the good strategy. We should notice that in different runs different initial values of the threshold are used ranging from 0.3 for run 1 to 0.6 for run 4.

Chapter 5

Conclusion and Future Work

5.1 Summary of Contributions

In this thesis, we proposed a game-theoretic model to study and analyze the efficiency characteristics of agent-based web services that can dynamically join and leave communities. We also introduced a model incorporating a task allocation mechanism and decision making procedures to help web services choose the best action over time. The proposed frameworks measure the efficiency of web services considering their reputation, market share and capacity as well as service users' feedback. We proposed games where web services have to decide about joining and leaving a community. We analyzed the existing Nash equilibria and situations where the maximum payoff is obtained. We also computed a threshold that web services can use to adopt the best strategy and decide about competing or cooperating when they reside in a given community. All the theoretical results were backed and supported by simulations where different parameters are considered.

The resulting models show that the efficiency of the community and web services is increasing

once the game-theoretic analysis is considered to impose parameters to control the decision of joining and leaving the community and making decisions about the coopetition strategy.

5.2 Future Work

The analysis of efficiency and behaviors of web services inside the communities we conducted in this thesis opens the door to many questions and research opportunities in the field of web service communities. Some of these opportunities can be listed as follows:

1. Considering reputation assessment for communities of web services and the role of user agents as rational service consumers and their feedback for this assessment. Therefore, reputation of communities of web services can play a significant role in user's service selection.

2. Advancing the model in the second contribution to analyze the coopetitive interactions and behaviors among communities of web services. Communities can exchange services or tasks between each other and hence a new type of social behavior will be defined among these rational agents.

3. Enhancing the task sharing mechanism proposed in the second contribution to consider the history of cooperation between web services inside the cooperative networks for selecting the enthusiasm and more effective cooperative peers in the future interactions.

4. Considering malicious agents like the ones that provide false information to gain more reputation, and their acts in the games and frameworks presented in this thesis. These malicious behaviors can result in a bad reputation in cooperative networks modeled in the

second contribution, so a new detection and prevention mechanism should be considered from the master web service perspective as the controller of the community.

Bibliography

[1] E. Al-Masri and Q.H. Mahmoud. Discovering the best web service. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, pages 1257–1258, 2007.

[2] E.A.P. Alchieri, A.N. Bessani, and J.S. Fraga. A dependable infrastructure for cooperative web services coordination. In *Proceeding of the International Conference on Web Services (ICWS), 2008*, pages 21–28, 2008.

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

[4] M. Bengtsson and S. Kock. "Coopetition" in business networks–to cooperate and compete simultaneously. *Industrial Marketing Management*, 29(5):411–426, 2000.

[5] J. Bentahar, Z. Maamar, D Benslimane, and P. Thiran. An argumentation framework for communities of web services. *IEEE Intelligent Systems*, 22(6):75–83, 2007.

[6] 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.

[7] M. Bichier and K.-J. Lin. Service-oriented computing. 39:99–101, 2006.

[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] P. Brittenham, F. Curbera, D. Ehnebuske, and S. Graham. Web services description language (WSDL) 1.1. http://www.ibm.com/developerworks/webservices/library/ws-wsdl/, September 2001.

[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] M. Comuzzi and C. Francalanci. Agent-based negotiation in cooperative processes: automatic support to underwriting insurance policies. In *Proceedings of the 6th international conference on Electronic commerce (ICEC)*, pages 121–129, 2004.

[12] Microsoft Corporation. Microsoft visual studio 2010. http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express, 2010.

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

[14] M. d'Inverno, M. Luck, and M. Wooldridge. Cooperation structures. In *Proceedings of the Internationbal Joint Conference on Artificial Inteligence*, pages 600–605, 1997.

[15] S. Elnaffar, Z. Maamar, H. Yahyaoui, J. Bentahar, and P. Thiran. Reputation of communities of web services - preliminary investigation. In *In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications and Workshops.*, pages 1603–1608, 2008.

[16] K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to web services architecture. *IBM Systems Journal*, pages 170–177, 2002.

[17] C.D. Huang and Q. Hu. Integrating web services with competitive strategies: The balanced scored approach. In *Communications of the Association for Information Systems*, pages 57–80, 2004.

[18] M.N. Huhns. Software agents: The future of web services. In *Agent Technology Workshops*, volume 2592 of *LNAI*, pages 1–18, 2003.

[19] M. Jacyno, S. Bullock, M. Luck, and T.R. Payne. Emergent service provisioning and demand estimation through self-organizing agent communities. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMA)*, pages 481–488, 2009.

[20] N.R. Jennings. Agent-based computing: Promise and perils. In *In Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1429–1436, 1999.

[21] N.R. Jennings. On agent-based software engineering. *Artificial intelligence*, 117(2):277–296, 2000.

[22] R. Jurca and B. Faltings. Reputation-based service level agreements for web services. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC)*, volume 3826, pages 396–409, 2005.

[23] R. Jurca and B. Faltings. Obtaining reliable feedback for sanctioning reputation mechanisms. *Journal of Artificial Intelligence Research*, 29(1):391–419, 2007.

[24] S. Kalepu, S. Krishnaswamy, and S.W. Loke. Verity: a qos metric for selecting web services and providers. In *Proceedings of the 4th International Conference and Workshops onWeb Information Systems Engineering*, pages 131–139, 2003.

[25] B. Khosravifar, J. Bentahar, A. Moazin, Z. Maamar, and P. Thiran. Analyzing communities vs. single agent-based web services: Trust perspectives. In *Proccedings of the IEEE International Conference on Services Computing (SCC)*, pages 194–201, 2010.

[26] 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.

[27] 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 onWeb Services (ICWS)*, pages 303–310, 2009.

[28] E. Lim, P. Thiran, Z. Maamar, and J. Bentahar. On the analysis of satisfaction for web services selection. In *Proccedings of the IEEE International Conference on Services Computing (SCC)*, 2012.

[29] Z. Maamar, M. Lahkim, D. Benslimane, P. Thiran, and S. Subramanian. Web service communities: Concepts and operations. In *Proceedings of the 3rd International Conference on Web Information System and Technology (WEBIST)*, March 2007.

[30] Z. Maamar, S.K. Mostefaoui, and H. Yahyaoui. Toward an agent-based and context-oriented approach for web services composition. *IEEE Trans. Knowledge and Data Eng.*, 17(5): 686–697, 2005.

[31] 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 (IJEBR)*, pages 1–21, 2009.

[32] Z. Maamar, P. Thiran, and J. Bentahar. Web services communities: From intra-community coopetition to inter-community competition. In *Web Services Communities. E-Business Applications for Product Development and Competitive Growth: Emerging Technologies. Ed. In Lee. Hershey: IGI Global*, pages 333–343. 2011.

[33] Z. Maamar, H. Yahyaoui, E. Lim, and P. Thiran. Social engineering of communities of web services. In *In Proceedings of the IEEE/IPSJ. 11th International Symposium on Applications and the Internet (SAINT)*, pages 100–109, 2011.

[34] Q. H. Mahmoud. Service-oriented architecture (soa) and web services: The road to enterprise application integration (eai). `http://www.oracle.com/technetwork/articles/javase/soa-142870.html`, April 2005.

[35] E.M. Maximilien. Multiagent system for dynamic web services selection. *The 1st Workshop on Service-Oriented Computing and Agent-based Eng.*, pages 25–29, 2005.

[36] E.M. Maximilien and M.P. Singh. Reputation and endorsement for web services. *SIGecom Exch.*, 3(1):24–31, December 2001.

[37] E.M. Maximilien and M.P. Singh. Conceptual model of web service reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[38] M.P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10):25–28, 2003.

[39] T. Rahwan, T. Michalak, E. Elkind, P. Faliszewski, J. Sroka, M. Wooldridge, and N. Jennings. Constrained coalition formation. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, pages 719–725, 2011.

[40] J. Reaidy, D. Diep, and P. Massotte. Management and control of complex production systems: co-opetition through game theory principles and agents based information systems. In *Proceedings of the IEEE International Conference on Industrial Informatics*, pages 217–223, 2003.

[41] J. Reaidy, P. Massotte, and D. Diep. Comparison of negotiation protocols in dynamic agent-based manufacturing systems. *International Journal of Production Economics*, 99:117–130, 2006.

[42] P. Renna. Dynamic co-opetitive network organization supported by multi agent architecture. In *Business Organizations and Collaborative Web: Practices, Strategies and Patterns*, pages 165–183, 2011.

[43] P. R. Thie and G. E. Keough. *An Introduction to Linear Programming and Game Theory*. 2008.

[44] H. Tong, J. Cao, S. Zhang, and M. Li. A distributed algorithm for web service composition based on service agent model. *IEEE Trans. on Parallel and Distributed Systems*, 22(12): 2008–2021, 2011.

[45] B. von Stengel. *Game Theory Basics*. 2008.

[46] K. Westwood. Heuristics for co-opetition in agent coalition formation, 2006.

[47] M. Wooldridge. Intelligent agents: The key concepts. *Multi-Agent Systems and Applications II*, pages 151–190, 2002.

[48] M. Wooldridge. *An introduction to multiagent systems*. Wiley, 2009.

[49] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995.

[50] M. Wooldridge and N.R. Jennings. Agent theories, architectures, and languages: a survey. In *Proceedings of the Workshop on Agent Theories, Architectures, and Languages on Intelligent Agents*, pages 1–39, 1995.

[51] A. Yassine, A.A. Shirehjini, S. Shirmohammadi, and T. Tran. Knowledge-empowered agent information system for privecy payoff in ecommerce. In *Knowledge and Information Systems, DOI: 10.1007/s10115-011-0415-3, 2011*, 2011.

[52] H. Yihong, J. Houde, and D. Tam-Kien. A multi-agent model of cooperative and competitive strategies in supply chain. In *In the Proceedings of the IEEE International Conference on Automation and Logistics (ICAL)*, pages 2908–2913, 2008.