Trust and Reputation in Multi-Agent Systems

Babak Khosravifar

A Thesis

in

The Department

of

Electrical and Computer Engineering

Concordia University

Presented in Fulfillment of the Thesis Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

April 2012

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Babak Khosravifar

Entitled: Trust and Reputation in Multi-Agent Systems

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

DOCTOR OF PHILOSPHY (Electrical & Computer Engineering)

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

Signed by the final examining committee:

_____- Chair
Dr. P. Grogono

_____- External Examiner
Dr. P. Yolum

_____- External to Program
Dr. L. Kosseim

_____- Examiner
Dr. R. Dssouli

_____- Examiner
Dr. A. Hamou-Lhadj

_____- Supervisor
Dr. J. Bentahar

Approved by _____
Dr. J.X. Zhang, Graduate Program Director

April,11, 2012

Dr. Robin A.L. Drew, Dean

Faculty of Engineering & Computer Science

Abstract

Trust and Reputation in Multi-Agent Systems

Babak Khosravifar

Multi-Agent systems (MAS) are artificial societies populated with distributed autonomous agents that are intelligent and rational. These self-independent agents are capable of independent decision making towards their predefined goals. These goals might be common between agents or unique for an agent. Agents may cooperate with one another to facilitate their progresses. One of the fundamental challenges in such settings is that agents do not have a full knowledge over the environment and regarding their decision making processes, they might need to request other agents for a piece of information or service. The crucial issues are then how to rely on the information provided by other agents, how to consider the collected data, and how to select appropriate agents to ask for the required information. There are some proposals addressing how an agent can rely on other agents and how an agent can compute the overall opinion about a particular agent. In this context, the trust value reflects the extent to which agents can rely on other agents and the reputation value represents public opinion about a particular agent. Existing approaches for reliable information propagation fail to capture the dynamic relationships between agents and their influence on further decision making process. Therefore, these models fail to adapt agents to frequent environment changes. In general, a well-founded trust and reputation system that prevents malicious acts that are emerged by selfish agents is required for multi-agent systems. We propose a trust mechanism that measures and analyzes the reliability of agents cooperating with one another. This mechanism concentrates on the key attributes of the related agents and their relationships. We also measure and analyze the public reputation of agents in large-scale environments utilizing a sound reputation mechanism. In this mechanism, we aim at maintaining a public reputation assessment in which the public actions of agents are accurately under analysis. On top of the theoretical analysis,

we experimentally validate our trust and reputation approaches through different simulations. Our preliminary results show that our approach outperforms current frameworks in providing accurate credibility measurements and maintaining accurate trust and reputation mechanisms.

This thesis is dedicated to the memory of my father, Bahman Khosravifar. I miss him every day, but I am glad to know he saw this process through to its completion, offering the support to make it possible, as well as plenty of friendly encouragement.

I would also like to dedicate this thesis to my wife, Ayrin Tabibi. There is no doubt in my mind that without her continued support and counsel I could not have completed this process.

Acknowledgements

I would never have been able to finish my dissertation without the guidance of my committee members, help from friends, and support from my wife and family.

I would like to express my endless gratitude to my advisor, Dr. Jamal Bentahar, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I would like to thank him for his ever encouraging kind support. I would like to thank Dr. Philippe Thiran, who invited me to do collaborative research with university of Namur, Belgium. Special thanks goes to Dr. Pinar Yolum, who participated in my defense committee.

I would like to thank Maziar Gomrokchi, who as a good friend, was always willing to help and give his best suggestions. It would have been a lonely lab without him. Many thanks to researchers in the laboratory of Dr. Bentahar for helping me for different parts of research projects. My research would not have been possible without their helps.

I would also like to thank both my parents and my wife's parents. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to thank my wife, Ayrin Tabibi. Through all these years, she has always been there cheering me up and stood by me through the good times and bad. She gave me the most needed support and encouragement while I wrote this dissertation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context and Motivation

Artificial Intelligence has become one of the most fundamental research areas in computer science. One line of research in artificial intelligence is associated with coordination of intelligent agents [6,7,9,10,17]. Coordination takes place in multi-agent systems [9,31,44,87] where distributed agents have limited knowledge about their surrounding environment; therefore, they continuously ask other agents to obtain required information. They could also require services provided by other (reliable and well-reputed) agents. In general, the growing popularity of agents requires systematic coordination management that enables agents to decide about their interacting partners and overall acting attitude. This is a main reason behind the emergence of trust and reputation-based frameworks that facilitate agents' coordination in multi-agent systems. Considered as agents' beliefs about one another, reputation is highly significant in such settings. Thus, a carefully designed mechanism is required to maintain the accuracy of this parameter.

In this chapter, we introduce the context of our research, which is mainly about trust and reputation in multi-agent systems. We identify the motivations, problems, and research questions that we address in this thesis. Finally, we present our summarized hypotheses, objectives, and methodology.

1.1.1 Agent

Agent is an intelligent entity that is programmed and equipped with variety of methods and techniques, observes its surrounding environment, and acts according to the decisions that are made in its control system [13,22]. A particular agent uses the built-in capabilities to decide about the most appropriate real-time action(s). It applies reasoning techniques to analyze the outcome of its action, acts autonomous to achieve its predefined objectives, and could cooperate with other agents, which share some goals. In general, agents react to the environment using their reasoning capabilities, analyze the obtained data, and react towards achieving their goals [42]. Overall, agents are reactive, proactive and follow social interactions. They have limited capabilities, which constrain them to coordinate with other agents to accomplish complex tasks.

1.1.2 Multi-Agent Systems

A multi-agent system is composed of multiple interacting intelligent agents [13,19,21]. Multi-agent systems are widely used in distributed environments [11, 67]. They are also used as an alternative to centralized problem solving, either because problems are themselves distributed, or because the distribution of these problems between different agents reveals itself to be a more efficient way to organize the process of problem solving. Multi-agent systems give us the possibility to build artificial universes that are small laboratories for testing theories about individual and group behaviors. These artificial universes can be used to describe specific interaction mechanisms and analyze their impact at a global level. The entities that are represented are usually called animates, since they are mainly inspired by animal behaviors (hunting, searching or gathering habits).

The aim of research in multi-agent systems is to have societies of agents that are very flexible and autonomous (for example, when agent-based robots are sent within an expedition and they are required to be very independent from the instructions they could receive). Multi-agent approach can be used for the coordination of different mobile robots in a common space. This approach can also be seen as an efficient and modular way of programming as agents could be

designed and programmed with different abilities, behaviors and intentions. To this end, analyzing multi-agent systems is like analyzing human or animal behaviors in the sense that they are self-independent and their selfish actions may affect the environment in any unpredictable way. Multi-agent systems are designed mainly with the objective of finding best social situation for all the involving intelligent components. The main feature obtained by developing multi-agent systems is flexibility in proactive environments. These systems also tend to be rapidly self-recovering and failure proof, usually due to self managed features.

In multi-agent systems, a class of computational models for simulating the actions and interactions of autonomous agents is developed. In such systems, game theory [3, 51, 65, 80–82], and evolutionary programming [21, 85, 94] are used. These techniques are important because of random distribution of agents in multi-agent systems and their independent decision making processes. In fact, rational agents aim at reaching their predefined goals and accordingly choose their acting strategies using their limited knowledge and capabilities. This limitation encourages agents to coordinate their actions. There are different research directions that address aspects of coordination in multi-agent environments such as coalition formation [10,66,87,90,95] and clustering [27, 60, 78, 104]. To investigate the rationality behind these coordinations and how agents are involved in, we need to thoroughly address the problems of agents reliability and how this reliability is established among agents.

1.1.3 Trust Establishment and Reputation Formation

The unpredictable behavior of multi-agent systems raises different important questions. Two interesting issues that we are mainly interested in are the concepts of trust and reputation that have crucial impacts on agents' strategic decision-making. Trust has long been recognized as a vital concept in open multi-agent systems, where agents are self-interested, diverse and deceptive. Trust is the parameter that reflects agents' risk level when they want to rely on the provided information or service of other agents for the fulfilment of their own goals. Trust is established in a mutual way between agents. It is defined with variety of meanings, but as in [67], we denote trust as a belief an agent has that the other particular agent would accomplish

what it promises. To this end, we define the trust as a peer-to-peer reliability estimation in which different components are involved and might change over time and during different interaction experiences.

Reputation is a parameter that reflects agents public reliability level in the multi-agent network. Reputation is then a combination of different trust ratings which form a public opinion about a particular agent. To this end, an agent that wants to evaluate the reliability of a particular agent with no previous interaction can use the the agent's reputation as a reasonable reliability rating. As a matter of fact, in any environment populated with multiple (self-interested) components that act independently and follow self-dedicated goals, trust and reputation assessment becomes important in the sense that the inter-correlation of components is influenced by the level of trust that components manage to have in a mutual way. Furthermore, in large-scale environments (and mostly in market-based societies), reputation becomes an important parameter that reflects the image of a component (i.e. an agent) regarding other components in the system. Reputation provides a form of social control, which encourages honesty in cooperative environments. Since agents are acting in a cooperative network with dynamic behaviors, there would be diverse opinions about a particular agent raised from different perspectives. The reputation aggregates various impressions reflecting the public opinion about a particular agent. Obviously, this value would be more accurate when there is a small diversity of opinions about a particular agent.

The Trust Model

In the literature, the term 'trust' has been used with different meanings. But in the context of multi-agent systems, trust is mostly referred to as the expected reliability of a particular agent in cooperative networks [14]. In many agent-based frameworks, interactions take place when agents trust each other. The first attempt in computing the trust associated to agents is made by Marsh [53]. Since then, there has been a number of other models that advance the trust computing in different aspects. We mainly concentrate on the frameworks that consider multi-agent environments [17, 40, 61]. In most of these models, intelligent agents [4, 25] are equipped with

reasoning capabilities which influence their interacting strategies. However, these models are not adaptable for open multi-agent systems where agents freely join or leave the network. In ubiquitous multi-agent environment, we require more advanced trust frameworks that specifically tackle the trust evaluation problem with respect to continuous environment changes. In this context, there have been a number of models [32, 33, 83] that to some extent address the trust evaluation problem [84, 89, 96, 98]. We explore the characteristics of each one of these models in detail in Chapter 2. However, all these models have one common missing aspect which has not been addressed in any of them: the concept of effective trust adjustment after trust evaluation. This means agents do not reconsider trust evaluation process to analyze their evaluation accuracy. In multi-agent environments with random distribution of agents and dynamic changes of behaviors, trust adjustment is crucial for agents to adapt with environment characteristics. The belief set adjustment should be fast enough to avoid agents' misleading in interactive environments. This issue is the main distinguishing point that advances our proposed framework compared to the existing trust-based frameworks.

To this end, we characterize a well-founded trust framework (called "ideal trust framework" in this thesis) that is applicable to multi-agent systems to hold the following properties:

- Accuracy: The trust framework is accurate once two requirements are fulfilled: (a) the collected data should be relevant to the trust evaluation process; and (b) the necessary amount of data should be collected to facilitate trust evaluation.
- Rationality: Intelligent agents are supposed to be rational and an ideal trust framework establishes the trust in such a way that agents utilize the framework based on some reasoning techniques. For example, a new agent runs the trust evaluation system more frequently than the one that has a consistent belief set about the environment. However, these agents also periodically utilize the trust framework and update their belief sets.

  Rationality is considered in common agent-based architectures and therefore, all the models that we consider in our related work develop rational behavior. But still we highlight it as a characteristic for an ideal trust framework.

- Adaptability: Agents require to update their belief sets with respect to environmental changes to adapt with new network characteristics. This property is crucial in systems that maintain business interactions. In such systems, agents need to adapt with new changes to keep their efficiency high. For example, in business interactive networks with diversity of behaviors, an agent might form a belief regarding a particular service provider agent by trusting it as an optimal choice to provide a given service. The adaptable agent should be able to recognize the change in service provider's attitude if it is altered. Otherwise, the selfish service provider could increase the service fee and decrease the service quality while has no doubt that the agent still follows the same belief about its interactive network.
- Agileness: Beyond adaptability, a trust framework is required to effectively upgrade the belief set to the most recent information captured from the environment such that agents are always holding the information that refer to the most recent collected data and reflect the most recent environment changes. Such a framework is more effective in decision making process regarding acting strategies of agents.

1.1.4 Trust versus Reputation

Since we mainly focus on trust and reputation modelling in multi-agent systems, it is worth clarifying their roles in agent-based interactions. The trust is a parameter that is mutually established between components and imposes impact on their one to one cooperation. Trust represents the level of reliability that agents have regarding the type and quality of information or service that is provided by other agents [54]. Although the measured trust might not reflect the actual credibility of agents, agents still need to evaluate this parameter to make decisions [44, 53]. Moreover, agents utilize trust-oriented learning strategies [84] that take into account past interactive experiences.

The reputation parameter is a factor that an agent holds as a means to attract other agents

in order to communicate and coordinate with them. Reputation represents the level of popularity that an agent has and this value is obtained with respect to the agents truthful actions regarding other agents in the environment [97]. Similar to trust, the obtained reputation might not accurately reflect the real reputation of an agent in the system, but that is a valuable source of information that agents use to make decisions. This is the reason behind the need to enhance the quality of the trust and reputation assessments and provide sound trust and reputation mechanisms. As it is deduced from the definitions, we consider the reputation as the extended form of trust in group-wise decision making process. In fact, when we analyze the reputation evaluation and gather individuals' ratings, we skip the details of the provided rating evaluation. These ratings could be the trust values that agents have regarding a particulars agent. We still consider the four factors representing an ideal trust model in any reputation mechanism that collects the data and computes a reputation value associated to a particular agent. However, we mainly concentrate on the public aspects of such computation and study the conditions under which this data collection is performed and how the reputation mechanism functions in a truthful environment.

1.2 Problem Statement

1.2.1 Trust Assessment

In multi-agent environments, intelligent agents get involved in continuous interactions to achieve their goals. In an effort to realize their defined goals, agents need to make the best decisions in their moves and apply selective strategies to facilitate their progresses. Therefore, these agents should be equipped with a strong trust assessment protocol that is capable of estimating the trust level of other agents. The trust level should be accurate enough that allows each agent to identify the most reliable agents in its surrounding environment.

The trust assessment protocol should be complete to gather all the relevant factors which influence the trust that an agent has about other agents. Failure to gather those factors would

lead to compute a non-accurate trust value, which could explicitly influence agent's outcome. For instance, the incomplete trust evaluation process would miss the reliable interactive agents and bring relatively low outcome for the agent that is looking for accurate information or high quality of service. In systems where agents are capable of learning other's attitudes, selfish service provider agents could easily recognize such incomplete trust assessment procedure and could mislead the evaluating agents to interact with them at high price but with low service quality. Moreover, the protocol should dynamically update the agents' belief sets to capture new characteristics of the environment.

1.2.2 Reputation Formation

In a large-scale multi-agent system, a sound reputation mechanism is required to reflect the public reputation of agents. This mechanism should be flexible enough to regularly get updated and provide accurate reputation ratings. Also it should be strong enough not to get violated and the represented data turn out to be inaccurate. This means that selfish agents may try to violate the reputation mechanism to take advantage of the faked data that support self reputation. This action is expected in open multi-agent networks within which the reputation plays an important role in agents interactions.

Moreover, in interactive multi-agent environments, reputation is highly competitive because agents resort to this value as a means of ranking their interactive parties. In fact, the obtained information regarding ones' reputation would highly influence the decision making process of agents that are looking for high quality of service or accurate information. To this end, a sound reputation mechanism has to act rigorously against any sort of violation in the collected data. Furthermore, like trust frameworks, this mechanism has to be agile in updating the computed values to represent an accurate image of the environment.

1.3 Objectives

The goal of this thesis is to develop and maintain strong trust and reputation assessment procedures that optimally function in multi-agent systems with dynamic changes of environment attributes such as agent goals, credibilities, and population. The main objectives are categorized as follows:

- Designing and developing a flexible trust-based framework to accurately consider the involved factors and provide an optimum (in computation and accuracy) trust estimation process. Moreover, the agile adaptation of agents' goals and beliefs should be considered in this framework as the system is supposed to be highly dynamic.
- Proposing a sound reputation mechanism to discourage malicious actions of the agents trying to increase self-reputation level and take advantage of open multi-agent system environment.
- Proposing a mechanism that investigates the parameters yielding optimal performance for agents. We aim to study the cases where selfish agents could obtain best payoffs using their decision making procedure.

1.4 Basic Assumption

In this thesis, we consider trust and reputation in different chapters and analyze their details within different simulated environments. However, trust and reputation are strongly connected in the sense that reputation is based on collected ratings and the provided ratings could be considered as direct trust that agents have regarding a particular agent [6,8,9]. Therefore in this thesis, we start the discussions by proposing a trust framework and continue by generalizing it to a reputation system.

In the trust-based framework, we mainly focus on the details of mutual agent relationships.

The parameters that are involved in this framework reflect the type and strength of the relationship between two specific agents (the evaluating agent and the agent being evaluated). We carefully investigate the data related to the previous interactions between these specific agents. At some point, we also consider the collected data in the form of consulting reports. But overall, the main concentration is the reliability analysis, which is maintained by the evaluators.

In the reputation mechanism, we mainly focus on the details of global reputation value that agents hold and use as a means to represent self-status. This is a crucial factor that is being used in enterprize environments to attract consumers. To this end, we consider a special and concrete multi-agent system where the agents evolve as a structure hosting (web) services, which are abstracted by intelligent agents. In this environment, agents are categorized into consumers and providers of services. We analyze the reputation of these agents (also called agent-based (web) services) and investigate their attitudes in representing truthful actions in the system. The motivation behind using this specific case of multi-agent system is the need to use specific and concrete parameters when evaluating the reputation of the system, which are (web) services in my case. In this environment, the reputation value associated to a particular agent is in fact the (weighted) mean value of ratings from different agents that might have variety of impressions about this particular agent. Therefore, unlike trust, the reputation is not from an agent's point of view but is more general as it reflects public opinion. This value could be used by new agents that do not have previous interactions with a particular agent.

1.5 Thesis Overview

The organization of this thesis is represented in the following. Moreover, Figure 1.1 illustrates the sequence of chapters and their prerequisites for convenient reading.

- ❏ In Chapter 2, we present relevant literature review and related work. We split the related work into two sections, one for trust-based frameworks and one for reputation mechanisms. A discussion about these frameworks is included in this chapter. Reputation is the context of (web) service ends the chapter.

- ❏ In Chapter 3, we introduce our trust-based framework and discuss the algorithms to maintain the optimal trust assessment procedure. We also discuss the implemented system to observe the framework functionality. The proposed model is fully described and analyzed with respect to the ideal characteristics defined in the current chapter. Furthermore, we study the effectiveness of the proposed model against conventional systems in different environments. The objective is to clarify the cases where the accurate trust-based model could effectively function.

- ❏ In Chapter 4, we introduce a reputation mechanism to compute general reputation value with respect to different involving parameters. We consider a typical multi-agent system hosting agent-based (web) services and we define communities of (web) services as groups of (web) services with common functionalities. I consider community of (web) services because these entities form an enterprize system where continuous interactions are crucial for their survival in the environment. Therefore, we put the system under a competitive reputation mechanism within which rational agents aim to achieve their predefined goals. The objective of that is to maintain an accurate reputation system that reflects the most recent image of agents' opinions about service provider agents. We implement the proposed mechanism and investigate its accuracy in a variety of settings.

- ❏ In Chapter 5, we continue the discussions about the reputation model with a different perspective. In this chapter, we mainly concentrate on the sound reputation mechanism and the conditions under which the malbehavior of rational agents is minimized. We apply game-theoretic payoff analysis to increase the expected payoff as a result of truthful actions. The objective is to maximize accuracy and prevent collusion, which could temporarily increase ones reputation. We study in details the scenarios and consider different parameters that influence agents' reputation and identify constraints where the malicious acts are minimized.

- ❏ In Chapter 6, we introduce an application in context of (web) services where the proposed reputation mechanism could be utilized. In general, we use reputation as a means

to enhance decision making process of the (web) service agents to maximize their performance over long time interactions. We implement a multi-agent environment where agents interact based on the proposed reputation mechanism (Chapter 4) and seek for maximizing their payoffs. We study long-term performance of such agents where settings alter according to agents' actions and movements. We apply reputation mechanism in this model and enhance agents' capabilities to maintain long-term high performance.

- In Chapter 7, conclusions and future work are discussed and the contributions of this thesis are summarized.

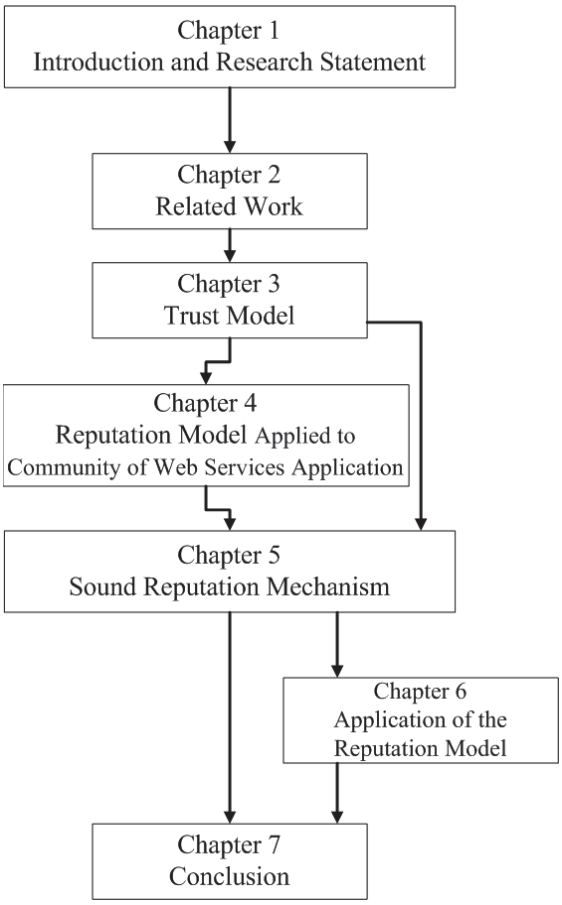


Figure 1.1: Thesis chapters listed in sequence of reading.

Chapter 2

Literature Review

2.1 Introduction

In multi-agent context, trust and reputation are very close concepts in the sense that most of the proposed frameworks analyze them both and in many cases without clear distinction. But in this thesis, we clarify our point of view in the following. Considering a multi-agent environment hosting rational agents, trust refers to peer-to-peer reliable interactions between two agents that are called the evaluating agent and the agent being evaluated. Different frameworks consider diverse approaches to address the trust evaluation problem. On the other hand, the reputation refers to public opinion about particular agent's reliability. The public opinion could be restored in a central unit and disclosed to agents upon requests. But there is no guarantee on the accuracy of the information. The obtained reputation value could be used as the expected trust value if an agent initiates the interaction with a particular agent with disclosed reputation value.

In this chapter, we discuss different frameworks and categorize them into two sections related to trust and reputation. Some of the reputation models consider trust assessment in their approaches, but for representing public opinion. According to our classifications, those models will be discussed in the reputation section.

2.2 Trust-based Frameworks

During the past few years, agent communication languages and protocols have been of much interest in multi-agent systems. Agents are distributed in large scale networks and mutually

interact to collaborate, coordinate and share services and resources with other agents. Therefore, trust is essential to ensure effective interactions within open multi-agent networks. In the Oxford dictionary, trust is defined as ”confidence in or reliance on some quality or attribute of a person or thing”. In multi-agent context, an agent's trust is a measure of the agent's willingness to actually do what it agrees to do. In fact, trust is a measurement of the complement of risk level that an agent can take when it wants to interact with another agent. The trust value is used as a means to estimate the reliability of the interactive party. This value could refer to different aspects such as accuracy, responsibility, or honesty. The estimated value might be wrong which misleads agents in interactive settings. Overall, trust represents the extend to which an agent relies on another agent.

In the literature, there are different proposed frameworks that consider variety of approaches to address the trust establishment problem in multi-agent systems. These models are aimed at developing trust-based models to enforce reliability in multi-agent environments. To survey these models, we need to categorize them into groups that have similar assumptions. We explore the details of these models in the rest and compare their characteristics against our proposed idealistic trust model using the four aforementioned characteristics in Chapter 1: accuracy, rationality, adaptability, and agileness.

The first group is based on direct interaction of two agents [71, 72, 88, 91]. In the frameworks belonging to this group, the trust is only computed by information obtained from direct interaction experience of two agents. The main idea of these frameworks is to explore details of previous interactions and obtain required information to compute the reliability of the agents being evaluated. The second group includes frameworks that consider the type of interaction as a means to estimate the reliability of agents [31, 32, 73]. These models consider the interaction domain and distinguish agents with respect to their capabilities in different domains. The third group includes frameworks that collect information from external parties to maintain the trust evaluation process [32, 33, 83]. Direct interactions (if any) are still used as a part of the process. The proposed model in this thesis (proposed in Chapter 3) is highly relevant to the models of this group. To this end, we explore more details about these frameworks and compare their

capabilities in different perspectives.

We continue our discussion with detailed analysis of the relevant models that include FIRE [32], REGRET [71], BRS [33] and Travos [83] that our model will be compared against in Chapter 3. We implement these models in our simulated environment to compare their performances with respect to different aspects. We mainly highlight their differences with respect to the ideal trust framework's characteristics explained in Chapter 1. For example, the BRS system filters out outlier ratings that are provided by other agents. This leads to the model's low accuracy in models where malicious agents collude to broadcast misleading information. In a setting where probability density functions are used to estimate the reputation of a selling agent, propagating ratings provided by multiple advisors requires careful analysis. This process of filtering and propagating ratings does not satisfy the adaptability and agileness factors of the ideal trust framework since a group of agents could collude and propagate some misleading ratings that distract the trust evaluation process. The Travos system uses the approach of discounting the ratings provided by less trustworthy advisors. We will show that this model also does not satisfy the agileness factor of the ideal trust framework. We highlight the advantages and disadvantages of these frameworks and clarify the motivation of the proposed framework. We also introduce a categorization of various features that have been introduced to make trust models robust, and discuss the types of systems in which they have been used. The categorization of different approaches provides a valuable perspective on the key challenges faced in designing an effective trust-based system that makes use of advice from other agents.

2.2.1 Beta Distribution Model (BRS)

One of the most complete trust frameworks that we survey in this chapter is the beta reputation system [33]. This framework is among witness trust models and computes the trust value of interactive agents trough a probabilistic model. In this probabilistic model, events occur with respect to beta probability distribution, which is parameterized by α and β. In the probability theory, beta distribution is a type of continuous distribution which is defined on the interval $[0, 1]$. Variable x denotes the probability ($x \in [0, 1]$). The probability density function

$f(x;\alpha,\beta)$, mean μ, and variance $Var(x)$ of this distribution are as follows:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du}$$

$$\mu = E(x) = \frac{\alpha}{\alpha+\beta}$$

$$Var(x) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

In [33], authors compute the reliability of an agent by collecting information provided by some other agents. The collected information represent other agents' impressions about the reliability of the agent in question. These ratings fall into discrete choices of 0 (negative impression) and 1 (positive impression). The collected data are counted in terms of number of positive ratings (m) and negative ones (n). In the beta distribution model used in [33], $\alpha = m + 1$ and $\beta = n + 1$ to ensure positive parameters. Therefore, the reliability of a particular agent is considered as the expected value μ ($\frac{\alpha}{\alpha+\beta}$), which reflects the most likely frequency value. In general, BRS satisfies the accuracy and rationality factors of the ideal trust model. The reason for accuracy is the approach to aggregate positive and negative impressions. Using BRS, an agent estimates the trustworthiness of another agent by relying on the obtained ratings. This model together with other trust frameworks are rational since they are used by intelligent agents that we assume rational. However, there could be some irrational models that are not agent-based and they are out of scope of our related work.

Although BRS considers accumulated ratings in terms of suggestions from other active agents, this approach is accurate when a certain portion of ratings points out the actual reliability of the agent to be evaluated. However, the model fails to accurately update the trust values once dramatic changes are applied in the system. This might occur when a group of malicious agents collude to distribute misleading ratings. Therefore, this model does not satisfy the adaptability and agileness factors of the ideal trust model. For example, the expected value μ could fail to represent the reliability due to misleading ratings collected from other agents, which might not have correct impression of the agent or provide fake information for purpose.

2.2.2 Travos

Travos [83] is another trust model that falls into witness trust category. This approach advances the beta distribution model in the sense that it attempts to discard the inaccurate ratings. Doing this, Travos is a sequence of two parts. In the first part, the reliability of the witness agent is computed via the direct experience of interaction between the evaluating agent and witness agent. This value is used to consider the accuracy of the provided rating in terms of witness report. In this model, n equal subintervals between 0 and 1 are considered. This is used to compare previous witness reports provided by the same witness agents to find out the similar ratings. Therefore, the current rating is accurate following the beta probability density function that is parameterized by the number of successful and unsuccessful ratings regarding previous reports. The Travos model mainly concentrates on the consulting reports and best ways to aggregate them for an accurate trust value. This model satisfies accuracy and rationality factors of the proposed ideal trust model.

The second part of Travos is to update the report data set to discard inaccurate ratings. The objective is also to rate the inaccurate witness agents in order to avoid collecting information from them in future. Following this approach, Travos outperforms the BRS model and also satisfies the adaptability factor. But, this model fails to recognize stochastic behavior of agents. A rational reliable agent may change its behavior to follow its goals and therefore act selfishly. The report provided by the witness agent influences the trustworthiness value of other agents. Therefore, the witness agent can easily mislead the evaluator agent by providing inaccurate reports. According to the system settings, it takes certain time for the evaluator agent to recognize the inaccuracy of the provided reports from the previously trusted witness agents. Moreover, the reliable witness agents might not have enough information to provide accurate ratings. But in Travos, the collected data is still used for trust computation. Consequently, the Travos model does not satisfy the agileness factor of the ideal trust model.

The BRS and Travos models are the main ones that our personalized model will be compared to in Chapter 3 because all the three approaches use the beta probability density function. In the rest of this chapter, we also introduce other related models. The categorization of these

approaches will provide a valuable perspective on the key challenges faced when designing an effective approach to cope with the problem of unfair ratings. All the remaining trust models studied in the rest of this chapter do not satisfy the agileness factor of the ideal trust model. They do not quickly capture environmental changes. Some of them also fail to adapt with new agents' attitudes and therefore do not satisfy the adaptability criteria as well. But all the selected trust models are aimed at computing an accurate trust value and rational agents could use them to estimate the reliability of the interacting agents.

2.2.3 FIRE

The FIRE model [32] is a witness-based trust model that collects the required information from other agents in the form of advisors. The collected data is used by the evaluator agent to compute the trustworthiness of a particular agent. The computed trust is used as a means for further interactions with the agent. Overall, the FIRE trust framework is simple and satisfies the accuracy criteria. However, there are a number of missing details, which leads to weak results in some cases. For instance, the data collection process, advisor selection, and the belief set update and its influence on advisor set management could simply fail to accurately function in highly dynamic multi-agent systems. All these operations influence the integrity of the model in developing a sound trust-based framework that is applicable to multi-agent environments.

In a model based on witnesses, there is a possibility for witnesses to refuse sharing their experiences. Therefore, the authors in [32] propose a method called certified reputation. This method consists in adding an additional factor for defining the trustworthy of referee agents which are introduced by the target agent. The most important aspect of this method is that an agent quickly evaluates the target agent's trust value, because of the small number of interactions needed while it does not create the trust graph. In some cases, target agents propose some colluding referee agents to mislead the evaluating agent. Thus, in these cases the final trust rate would be affected by non-reliable information about the target agent. Essentially, the agents beliefs about the target agent will not be true, therefore the evaluating agent has to evaluate the

referee agents, although it will cost an extra computational overhead for the method. Considering the characteristics of the FIRE model, equipped agents are rational and adaptable, but they fail to be accurate and agile in all situations. In our implemented environment explained in Section 3.6 (Chapter 3), we compare the FIRE model along with other models that take similar approach against the proposed trust framework.

2.2.4 Reinforcement Learning Model (RL)

The reinforcement learning model [84] is a trust model based on the reinforcement learning technique. This model falls into the interactive trust category, which is based on information collected from past direct experiences. This method of learning is used to select best interactive party in the multi-agent environment. From service consumer agent's perspective, the RL model enhances the obtained profit in terms of service quality, whereas from service provider agent's perspective, the RL model enhances the efficiency in balancing the service fee and quality that yield best outcome. The trust update procedure used in the RL model is inspired by a trust framework proposed in [100]. The update process is maintained after comparing the obtained information about the reliability of a particular service provider agent against the obtained service quality from the same agent. In the RL method, there are some certain thresholds set to categorize service provider agents to trustworthy and untrustworthy agents. The service consumer agents do not interact with the untrustworthy agents and among the trustworthy ones, agents with the highest values are selected as interaction option. The problem with this approach is that a particular service consumer agent can easily lose the trust in a particular service provider agent and avoid interacting with it afterwards. In the RL model, agents attempt to use the collected information at best to compute the trust values, but do not quickly recognize the environment changes and adapt with new settings. Moreover, the information is based on consumer agent's personal interaction experience, so the new entry agent has lack of knowledge about different service provider agents. To this end, the agents equipped with the RL model only satisfy the rationality characteristic of the ideal trust framework. This model is extended in [68] to include information exchange between evaluating agents. The collected

data is also categorized into trustworthy and untrustworthy using the same learning process. Another framework that uses reinforcement learning is the work done in [38]. In this paper, authors propose a model to identify the trustworthy agents. The model enhances the decision making performance of the agents so that the agents equipped with this model can selectively interact with the others. The novel approach in this framework is modelling trustworthy agents based on their outcomes from taking different actions while they interact with other agents. This is done instead of tracking different properties regarding prior interactions. Authors claim that the system accurately functions when agent's actions are clearly identified. But the precision metric that is built in this framework could be misplayed by colluding agents and deviate from the primary concern of trustworthy selections.

2.2.5 Other Conventional Trust Computation Models

In multi-agent networks, the trust is generally compared based on collected information either from past experience or from other agents. Most of the proposed frameworks have similarity in the approach they use to address the question. For instance SPORAS [101] is a trust-based system, which performs simple rating. This system suffers from rating noise because it treats all ratings equally. In addition, SPORAS is a centralized approach so it is not suitable for open systems. This system fails to consider the adaptability and agileness factors of the ideal trust system.

Yu and Singh [97] by applying social network concepts in multi-agent systems have proposed a trust model called Referral. In this framework, witness agents use message passing method for transmitting information. Doing so, they retrieve ratings through social networks. This aspect of the referral model is similar to the role of links that search engines use to obtain a web page while approaching another source of information.

The idea of witness reputation has been used by Sabater who has proposed a decentralized trust model called REGRET [71]. REGRET uses the reports from the witnesses in addition to the technique based on direct interaction experience. This work is sensitive to noise and thus vulnerable to fake information known as distractions made by some malicious agents. In [88],

Wang and Singh have developed an algebraic method for aggregating trust over graphs understood as webs of trust. They state that current approaches based upon combining trust reports tend to involve ad hoc formulas. In their work, they have developed a principled evidential trust model that would underlie any multi-agent system, where trust reports are gathered from multiple sources. Regarding ad hoc formulation, a work similar to Wang and Singh's one has been done by Velleso and his colleagues who assign trust levels in ad hoc networks [86]. The key aspect of their work is its reference to human concept of trust. They also use the recommendations by trustworthy agents in addition to self direct experience. They tried to balance the recommendations regarding recency relevance and relationship maturity. However, agents in this framework do not have reasoning capabilities. Moreover, they do not have policies for dealing with malicious agents.

2.2.6 Trust Model with Statistical Foundation (TMSF)

In the work done by Shi et al. [79], a trust model has been introduced to assist agents' decision making in order to predict the likely future behavior by analyzing the past behavior. The authors have mostly worked on the environment characteristics, for example the space of possible outcomes has been studied. They state that it is crucial to identify the space of possible outcomes, which determines the nature of the associated trust model. The notion of discrete categories used in this model gives more flexibility to the ratings as feedback in order to get more accurate direct interaction estimation. However, they have not taken into account the measurements, which would unbalance the trust estimation and decision making that are solely based on the previous interactions. Therefore, agents equipped with TMSF trust model only satisfy the rationality criteria.

2.2.7 Discussion on Trust Frameworks

In Chapter 1, we categorized four criteria that represent an ideal trust framework: accuracy, rationality, adaptability, and agileness. In this chapter, we surveyed a number of trust frameworks

that are developed in the context of multi-agent systems. We discussed their methodologies, highlighted their advantages, and criticized their weaknesses with respect to presented ideal trust framework's characteristics. For example, the FIRE model [32] is an accurate and rational framework, but it fails to be adaptable and agile in highly dynamic environments. This is due to the model's approach in computing trust and the aggregation technique. In FIRE, the recent dramatic changes do not impose big affect on the computed trust values and therefore, the equipped agent continues relying on an agent that has recently changed its goal and is not reliable anymore. This model recognizes the changes fairly late and the equipped agent undergoes a number of interactions with unreliable agents to update its belief set. Adversely, the REGRET model [71] is adaptable, but not accurate to some acceptable extent. The REGRET model satisfies the adaptability criteria thanks to the approach that the method takes by developing images that reflect agents' impression about other agents' attitudes. The images are transferred to the other agents and therefore, recent changes in the environment are recognized and properly broadcasted. However, this approach fails to accurately compute the trust values because the aggregation function does not consider all relevant data. In this model, the trustworthy agents that are not well-connected with other agents (because they are new comers) do not succeed to interact with many agents and therefore it takes relatively long time for them to become well-known. Adversely, agents that are well-connected with other agents can collude to propagate unified images regarding their reliability and stay well-known in the system.

Among the aforementioned trust frameworks, the Travos model [83] outperforms the other models since it accurately computes the trust values and the equipped agents rationally use the model. Moreover, this model attempts to adapt agents' belief sets with recent changes of the environment. The Travos model is relatively complete and considers all related parameters that influence the trust values of the agents. However, the model fails to recognize dramatic changes in the environment. For instance, consider a trusted service provider agent in a system. Due to some changes in the surrounding environment of the service provider agent, this agent is not able to provide the service with the same quality as before whereas the agent is still considered as a trusted service provider. Consequently, the agent can change its goal and based on its prior

Table 2.1: Trust models classifications with respect to characteristics of the ideal trust framework.

Models	Accuracy	Rationality	Adaptability	Agileness
BRS	√	√		
Travos	√	√	√	
FIRE		√	√	
RL		√		
SPORAS		√		
Referral	√	√		
REGRET		√	√	
TMSF		√		

trust impression that it has among other agents collects a high number of service fees and leaves the environment. This dramatic change of the agent is not recognized with other agents. In fact, the reluctance of the service provider agent in service response could be recognized by some interacting agents and they could consequently warn other agents about this particular agent's suspicious action.

Table 2.1 classifies the aforementioned trust models with respect to the ideal trust framework's criteria. As it is clear from the table, there is no any trust framework that satisfies all the criteria. That means there is no such a comprehensive framework that can accurately compute the trust, and adapt with the environment as well as quickly take actions towards updating the belief sets to keep most reliable impressions about the multi-agent environment. Consequently, there is a need to have a system that accurately functions in multi-agent system with dynamic changes in agents' goals. A robust system needs to be designed that can manage the trust computation of such environments in long term interactions. To achieve this objective, we propose a trust framework in Chapter 3 that is aimed at satisfying all the criteria and overcome the already mentioned problems.

2.3 Reputation-based Frameworks

In the literature, the reputation mechanisms are mostly applied to large scale multi-agent systems, which host numerous interactive agents that seek to find the highly reputed agents to interact with. As mentioned before, the trust and reputation are very close concepts in the sense

that in most frameworks both concepts are analyzed and addressed without a clear separation line. In general, the reputation refers to public opinion about a particular agent and the trust is a reliability that is measured by an agent regarding others. In the following, we continue the survey by presenting different models that compute agents' reputation values. Since the data aggregation processes are very similar, we do not repeat the same analysis in this chapter, but we investigate the system integrity and accuracy in long-term interactive networks. We mainly concentrate on the techniques that guarantee long-term effective reputation management system.

Unlike trust frameworks, we do not categorize the reputation models against ideal reputation system characteristics. The reason is simply because the same set of criteria that we mentioned for trust frameworks could be applied to reputation assessment systems. Our concern in reputation systems is mainly establishing a sound and long-term secure mechanism allowing active agents in a dynamic multi-agent environment to achieve their goals. We generally aim at advancing the reputation mechanisms to function accurately and safe in long term interactions. We go beyond the computation problem, which was our main concern in trust frameworks. We would like to maintain a sound and secure reputation system within which the accuracy is achieved via provided incentives to the interacting agents. This point of view is new and does not overlap with the target of the related works that concentrate on addressing the reputation mechanism. Consequently, we discuss different reputation frameworks as well as their characteristics. We continue with highlighting the needs of an efficient reputation mechanism that last long enough in a multi-agent system with dynamic behaviors.

2.3.1 Bayesian Network Model

The Bayesian network model [89] is an interactive/witness reputation model in which service provider's behavior is analyzed in different aspects such as download speed, quality, and type. In this model, a naïve Bayesian network represents conditional dependencies between the reliability of the service provider and the analyzed aspects. Therefore, each service consumer agent constructs a naïve Bayesian network regarding each service provider agent. The user might

get access to the naïve Bayesian network constructed by other agents in case it does not have enough information about the particular service provider. In this case, the evaluating agent considers the reliability of the recommender in its decision-making process. The reputation values are continuously updated using RL model's formulation [38].

The Bayesian network model is constructed based on the similarity of the service consumer and provider agents and their preferences in interactions. This approach in unsuitable in the cases where the preferences of the recommender and evaluator agents do not perfectly match and the collected information does not accurately represent the trustworthiness of the service provider agent. In fact, this model fails to accurately combine the obtained data when the recommenders had weak interacting relationship with the particular agent under study.

2.3.2 Weighted Majority Algorithm (WMA)

The weighted majority algorithm is a discrete algebraic method [98] that uses Dempster-Shafer theory [15, 76] to compute the trust value of agents. This approach falls into witness report category. The method is based on three parameters: belief (b), disbelief (d), and uncertainty (u) that sum up to 1. An orthogonal sum function is defined in this method that aggregates the parameters to combine impressions of different witness ratings. In WMA, there are assigned values for witness agents which reflect their reliability from evaluator agent's point of view. These weights are assigned by the evaluator agent and updated due to reliability changes of the witness agents.

In [97], the weights are increased and decreased based on their positive or negative influence in correct reputation evaluation maintained by the evaluator agent. This algorithm is inadequate since the evaluator agent might unfairly decrease the weights when it does not have enough information about the provider agent. There is no methodology defined to fix the mistaken weight updates. Additionally, the witness agent would not get the chance to increase its associated weight once been decreased. The method does not characterize a realistic multi-agent environment where agents dynamically change their acting strategies.

2.3.3 Cluster Filtering Approach

Cluster filtering approach [20] is another witness trust-based mechanism that mainly concentrates on discarding inaccurate ratings to compute a general overview regarding a particular agent's reputation. In this model, a collaborative filtering technique [1] is used to recognize the most trustworthy agents for the evaluating agent. Using cluster filtering approach, a particular evaluating agent divides its surrounding agents into high and low rating clusters. The high rating cluster represents reports with relatively high inaccuracy and therefore are discarded. This approach also considers most recent reports to avoid confusion in clustering. This model addresses the agileness of the reputation computation, but the whole technique does not accurately function in some situations where multi-agent network hosts agents with rapid change of behavior.

In [20], the author concludes that by controlling anonymity, the inaccurate witness reports cannot be minimized because of collusion that could be established between the agent to be evaluated and the witness agent. Moreover, this framework works with higher efficiency in large networks where agents' relocation is not highly considered. The concept of collusion emerges when agents act in an open environment and individuals by default do not have full knowledge about their surrounding environment and agents might expect better outcome by colluding with other agents. This issue of collusion is fully analyzed in Chapter 5.

2.3.4 Robust Reputation System for Mobile Ad-hoc Networks (RRS-MAN)

In [11], authors propose a robust reputation system for mobile Ad-hoc networks. This system is based on distributed individuals and the objective is to handle false disseminated information. In this model, agents have a belief set regarding the reliability and public reputation of other agents. The reliability is used as the probability that the agent will provide truthful information. The reputation reflects agent's influence in decision making maintained by the agent that holds this information. This model associates different ratings for the collected data and therefore,

categorizes the surrounding agents in its belief set. To handle false information provided by other agents, RRSMAN updates the reputation ratings with respect to their accuracy in comparison with other random trusted advisers. The collected data from an agent is considered accurate if the difference with the held reputation is less than a deviation threshold. In the RRSMAN model, the collected data are investigated in the order they are received. This could decrease the accuracy of the model in cases where a high load of non-relevant data needs to be analyzed where an important evidence of environment changes is reported. Furthermore, the time discount factor is not considered in this model and therefore, old information are treated the same as new ones. In open multi-agent environments with dynamic behavior changes, this approach quickly fails since the used approach in collecting data fails in stochastic environments that host agents with different ranges of behaviors.

2.3.5 Discussion on Reputation Frameworks

In this section, we categorized some reputation frameworks that mainly focus on reputation computation and the ways to keep it accurate. The attempts to compute such a value is similar to the ones we discussed in trust frameworks. This is the reason we do not go into further details about comparing different approaches of aggregating some collected data. We would like to highlight the fact that in multi-agent systems there are rational agents that by default follow their goals and they could take actions that benefit themselves whereas others undergo some payoff loss. For example, in an environment where agents exchange services based on some fees, the goal of the service provider agent could be to charge as many agents as possible. To achieve this goal, the agent needs to maintain a high reputation value to absorb other agents' attentions. In such a system, if there is a way to claim high reputation, the rational service provider agent would resort to any way to maintain it. Consequently, the reputation should be competitive and the reputation mechanism should be robust against malbehavior of agents to mislead the environment with fake reputation values. Moreover, the reputation mechanism should last long and this is fulfilled when such a mechanism can update itself with respect to dynamic change of environment.

Considering the related work, we feel the need of a robust mechanism that imposes incentives in such a way that malicious behavior of agents are minimized. In general, we need a mechanism to maintain accuracy of the reputation system by preparing a situation within which agents seek maximum payoff by acting truthfully and compete for high reputation rather than colluding for fake reputation values to temporary increase their reputations. Chapter 4 is associated to propose such a reputation mechanism that its objective is to tackle the incentive-based reputation assessment problem. Chapter 5 and 6 are also about a robust reputation mechanism which is claimed to be sound and secure in long term interactive interval. The combination of these three chapters develop a complete reputation model that could be implemented in multi-agent environments where there is a need to cope with malicious agents and constrain the accuracy of the network and safety of the interacting agents' transactions. The structure used in these chapters is the network of web services that could be also grouped together as community of web services. The rational behind the use of this structure is the enterprise system of web services and their rational activities while exchanging services between one another. Moreover, web services (that are attached to agents as web service agents) compete to serve their services and obtain utilities that could be in the form of service fee or any other sort of payoff. We explore more details about this structure in the following section.

2.4 Web Service Applications and Discussions

Networks of web service agents are typical examples of multi-agent environments that run continuous business interactions through service exchange. In this context, the reputation management has been intensively stressed [35,41,54,91] aiming to facilitate and automate the good service selection. In [77], the authors have developed a framework aiming to select web services based on the trust policies expressed by the users. The framework allows the users to select a web service matching their needs and expectations. In [52], the authors proposed to compute the reputation of a web service according to the personal evaluation of the previous users. In general, the common characteristic of these approaches is that the reputation of the

web service is measured by a combination of data collected from users. To this end, the credibility of the user that provides this data should be taken into account. There should be a mechanism that recognizes the biased rates provided from the users and accordingly updates their credibilities. If the user tries to provide a fake rating, then its credibility will be decreased and the rating of this user will have less importance in the reputation of the web service. In [57], authors have designed a multi-agent framework based on an ontology for Quality of Service (QoS). The ontology provides a basis that allows the providers to advertise their offerings, the users to express their preferences, and the ratings of services to be gathered and shared. The users' ratings according to the different qualities are used to compute the reputation of the web service. In [74, 75], authors believe that selection process should be consumer-oriented and context-dependent. But to avoid misleading from diversity in context and satisfaction criteria, authors propose a model that captures subtle details using ontology. Furthermore, liars could be detected by investigating their prior experiences and getting filtered during service selection process. In [35], service-level agreements are discussed in order to set the penalties over the lack of QoS for the web services. In general, in all the mentioned models, web services are considered to act individually and not in collaboration with other web services. In such systems, the service selection process is very complicated due to the relatively high number of services in the network. Furthermore, web services can easily rig the system by leaving and joining the network when they have incentives to do so. For example, when their reputation is fall off for some reason, which is a rational incentive for such web services that manage to start as new once they have shown a low efficiency. Meanwhile, it is hard to manage the huge number of data in web services settings. Considering these inefficiencies, authors in [49] highlight the concept of gathering web services together into communities. Communities are in general formed to get stronger and more publicized in the system, so they do not resign and register as new. In such a methodology, users interconnect with the community as the service provider and there would be a web service assigned through the community.

Regarding the aforementioned issues, there have been some proposals that try to gather web services and propose the concept of community-based multi-agent systems (CWSs) [16,26,42].

In [16], authors propose a reputation-based architecture for CWSs and classify the involved metrics that affect the reputation of a community. They derive the involved metrics by processing some historical performance data recorded in a run-time logging system. The purpose is to be able to analyze the reputation in different points of views, such as users to CWSs, CWSs to web services, and web services to CWSs. The authors discuss the effect of different factors while diverse reputation directions are analyzed. However, they do not derive the overall reputation of a CWS from the proposed metrics. Failing to assess the general reputation for the community leads to failure in efficient service selection. Moreover, authors assume that the run-time logging mechanism (the logging file, which holds the feedback submitted by the service consumers) is an accurate source of information. In general, in open reputation-feedback mechanisms, always the feedback file is subject to be the target by selfish entities. To this end, the feedback mechanism should be supervised and its precise assessment should be guaranteed. In [42], authors have proposed a framework that explores the possibilities that the active communities act truthfully and provide their actual information upon request. This method is related to the ideas proposed in this thesis in the sense that the communities are provided with the incentives that push them to act truthfully. In [26], a layered reputation assessment system is proposed mainly addressing the issue of anonymity. In this work, the focus is on the layered policies that are applied to measure the reputation of different types of agents, specially the new comers. Although, the proposed work is interesting in terms of anonymous reputation assessment, the layered structure, because of its rigid hierarchical organization, does not optimally organize a community-based environment that gathers autonomous web services. Moreover, as claimed by authors, ”the computational expenses seem to be relatively high”.

Chapter 3

Trust-based Framework

3.1 Background

As discussed in previous chapters, the trust issue is application-dependent. In most of multi-agent environments, there are numerous agents that continuously interact with one another with respect to the trust they have regarding each other. Although in repeated interactions, agents form a history of interactions that helps them address the trust evaluation problem (interactive trust category), in many cases agents lack enough information required to compute the trust value of others (witness trust category). We consider a comprehensive approach that is mixed of both cases (where agents have or lack the required information to compute trust value of other agents) in the sense that we carefully analyze the history feedback and collect the required data from environment in the form of witness reports. So we address the trust estimation problem with a combination of direct and indirect trust evaluations.

The proposed trust framework is based on collected information from inside (belief set and the interaction history) and outside (witness agents), which is mostly the core of the witness trust category in the literature. We implement a comprehensive trust assessment mechanism, which effectively aggregates the collected data and produces the most efficient results. Moreover, we apply a maintenance mechanism to reconstruct the belief set based on the obtained experiences. The maintenance part is new in trust models in the sense that it accounts for quick changes and modifies the belief set according to the obtained experiences via direct interactions. We investigate the characteristics of the proposed trust framework in different implemented environments.

3.1.1 Motivating Example

We motivate our approach with an example. Consider two agents Ag_a and Ag_b that are active in the environment. These agents have already interacted with each other in the past. These interactions could be named as information or service exchanges. After each interaction, agents obtain a feeling on the quality of the interaction. For example, if the provided information by Ag_b was useful, Ag_a obtains a positive view regarding Ag_b. Likewise, if the provided service by Ag_b was not as promised, Ag_a obtains a negative view regarding Ag_b. Over time and continuous interactions, Ag_a rates the credibility of Ag_b with respect to the direct experience that Ag_a have had with Ag_b. There is a similar process for Ag_b and any other agent.

Trust models using direct experience need long term of interaction to reach a stage that agents can evaluate trust level of others. To this end, if agent Ag_a thinks its previous interactions could not be the perfect source of information to make a decision about Ag_b's trust level, it might request some other agents to reveal their credibility rate (or estimated trust value) regarding Ag_b. This is done by moving to the second level of evaluation process [6, 32], asking other agents that are known to be trustworthy (these agents are called trustworthy agents). However, there is a problem if these trustworthy agents are not able to report on the agent being evaluated (Ag_b). Therefore, in our proposed framework, we use trustworthy agents together with referee agents that are proposed by the agent Ag_b. Figure 3.1 illustrates this situation by using agents in different groups. The consulting agents are either known by Ag_a to be trustworthy (we call these agents trustworthy agents) or known by Ag_b and have been introduced by this agent to report on its trust level based on their past experience (we call these agents referee agents). Consequently, we distinguish the community of trustworthy agents from the community of referee agents.

In the proposed framework, the consulting agents are supposed to reveal the trust value of the agent Ag_b with their own evaluation perspective. Although the provided information by the consulting agent Ag_c helps Ag_a analyze the trust value of the Ag_b, this is considered as an interaction done between agent Ag_a and the consulting agent Ag_c. In this case, Ag_a can obtain an overview on the credibility and accuracy of the consulting agent. In fact, this would help

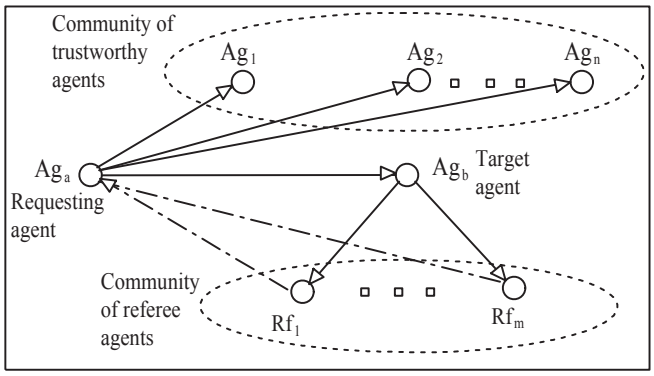


Figure 3.1: Overall trustworthy and referee agents network topology.

Ag_a to also stay updated on the credibility of its surrounding agents.

In our approach, we propose a maintenance algorithm that helps Ag_a update its belief set regarding the credibility of the consulting agents that provide information for some different reasons. In the maintenance process, the suggestions provided by other agents are compared with the actual behavior of the new agent in direct interaction.

3.1.2 Application Discussions on the Proposed Approach

Several trust-based architectures exist in the literature. In the context of service computing, a consumer agent maintains trust only based on its history, even there might be no interaction recorded. Using this model, the consumer agent selects some providers that look trustworthy. However, this can negatively affect the decision that the agent can make. Next, the consumer interacts with the provider agent and evaluates the actual quality of the provided service and accordingly records the evaluation ratings in its history. In an alternative scenario, the consumer consults with some other agents and evaluates the credibility of the evaluated agents by combining the collected information from consulting agents. In this case, the consumer requires applying learning methods to update the model it is maintaining for the trust evaluation. Our approach handles both of these scenarios.

3.2 Direct Trust Evaluation

During the past couple of years, agent communication languages and protocols have been of much interest in multi-agent systems, where agents are distributed in large scale networks and interact to collaborate, coordinate and share services and resources. Trust is then essential to make such interactions within open multi-agent systems effective [16, 77, 96]. An agent's trust is a measurement of the agent's possibility to actually do what it agrees to do. Attempting to maintain a trust-based approach, different frameworks have been proposed to represent and assess the trust agents have in one another. As discussed in Chapter 2, the most recent research proposals in trust models for multi-agent systems are as follows: (a) interaction trust, based on the direct interactions of two parties [88, 91]; (b) trust based on the type of prior interactions [31,32,73]; (c) witness reputation based on certified (and encrypted) references obtained by the agent to be evaluated after interacting with other agents. These references are then made public to any other agent that wants to interact with this agent [31, 33, 49, 71, 83]; and (d) certified reputation, based on references from other agents detailing a particular agent's behavior [6, 31, 32].

The proposed frameworks objectively focus on collecting some parameters that may contribute in the trust assessment procedure. The aim is to collect reliable information leading to an accurate trust assessment process. Since agents might be selfish, receiving fake information by particular agent(s) is always possible. This problem does exist even when a certified reputation [31] is provided by the agent to be evaluated. In this case, the final trust rate would be affected by non-reliable information and eventually the agents' perception of their surrounding environment will not be accurate. Generally, these frameworks are not suitable when the environment changes dynamically because they fail to quickly recognize the recent improvement or degradation of agents' capabilities as in dynamic environments these agents tend to change their goals and behaviors. To overcome this problem, some methods have been proposed to capture the recent changes in the environment [49]. In these frameworks, a retrospect trust adjustment mechanism is proposed to reconsider the trust evaluations that have been performed

in the past to learn how to select better witness agents. Although the mechanism is novel in this domain, its complexity is a considerable issue. Moreover, the applicability of the proposed framework is vague in the sense that the retrospect mechanism does not follow a systematic execution process that enhances the agents' accuracy.

The framework we propose is built upon a model in which a set of trust meta-data is introduced to define the trust level of contributing agents [6, 8, 16, 49]. The objective is to overcome the aforementioned limitations by proposing a comprehensive framework called *CRM* [46] (Comprehensive Reputation Model). The CRM model is aimed at satisfying the four criteria we mentioned in Chapter 1 to maintain a complete trust framework to be used in multi-agent environments. In this framework, agents interact and rate each other based on previous interactions (either satisfactory or dissatisfactory). The obtained ratings are collected to assess the trustworthiness of a particular agent. To be self-contained, we also consider how agents communicate to exchange ratings. Inter-agent communication is regulated by protocols (shared amongst agents and thus made public) and determined by strategies (internal to agents and thus private). Using this framework, agents are capable of evaluating the trust level of other agents that are not known (or not very well-known) by collecting some relative information, either from their interaction history or from consulting other agents that can provide their suggestions in the form of ratings. To express the efficiency of the proposed framework with respect to our aforementioned ideal trust framework's characteristics, we discuss in more details the performance of the CRM considering accuracy, scalability and applicability. We analyze the scalability of the system because CRM is capable of handling the large scale systems. But this cannot be considered as an ideal trust model's characteristics in the sense that a model could be highly efficient in an environment where scale is not a point of interest. Moreover, we explore more details about the applicability of the CRM model to highlight its strengthen as a trust model. More details about CRM's ability to satisfy ideal trust model's criteria are discussed in the implemented environment in Section 3.6.

CRM's Accuracy: In general, CRM is based on collecting information before making decisions. The idea of consulting other agents originates from the fact that in social networks,

agents assess diverse trust levels for other agents depending on their different experiences of direct and indirect interactions, and thus, an evaluator agent can balance the trust assessment process by considering different factors. In this model, the evaluator agent is referred to as *the trustor agent* and the agent to be evaluated is referred to as *the trustee agent*. In the evaluation process, the trustor may ask some other agents to report on the trustee. These interfering agents are basically divided into two groups: (1) well-known agents by the trustor agent (so-called *trustworthy agents*); and (2) those introduced by the trustee agent (so-called *referee agents*). CRM reaches acceptable accuracy because it collects the information from the agents that are considered the most appropriate sources. The potential aim is on updating the consulting agents to only keep the most accurate ones (i.e. the most trustful). The structure of information update approaches a stable situation wherein the trustor agent received accurate information from the surrounding agents and continuously updates its surrounding environment with respect to the changes in agents behaviors and goals.

CRM's Scalability: In general, a system is considered scalable when over the population expansion, the complexity does not affect accordingly. In the structure that defines the CRM framework, the scalability is considered at best. This is claimed due to the fact that enlarging the network does not affect the fact that the trustor agent uses a limited number of consulting agents. Agents use their historical information and do not initiate a new process of information search upon every request. Therefore, in case of increasing the agents to hundreds or thousands, the process of evaluation does not change. But the knowledge over the environment is reduced which makes the agents to maintain interactions with new agents more rapidly. The trustor agent in this system considers its history of interactions and accordingly rates the importance of the information provided by the consulting agents. Moreover, the extendable social network would increase the accuracy of agents by since the new source of information are needed.

CRM's Applicability: It is worthy to discuss the applicability of the proposed model. In fact, in distributed multi-agent system (for example distributed agent-based web services and trading agents in e-commerce settings) the proposed framework is applicable. However, what makes the proposed model essential in these environments is its sensitivity to obtain accurate

information and its capability to survive in dynamic environments. In fact, all the systems that involve multiple components, which require to exchange information need to establish a comprehensive and adaptable trust framework to guarantee the safety of information retrieval.

The off-line evaluation adjustment made by the trustor after a period of direct interaction with the trustee is the main contribution of our proposed framework. The trustor does this in order to adjust the accuracy of the *consulting agents* (i.e. trustworthy and referee agents). In the off-line process, the suggestions provided by other agents are compared with the actual behavior of the trustee through direct interaction. The trustor will update its beliefs about the consulting agent with respect to the accuracy and usefulness of the provided information through different trust evaluation procedures. By doing this, more accurate ratings about the other agents will be gradually propagated throughout the environment, which provides a better trust assessment in the CRM model. In off-line process, the maintenance mechanism is designed such that it prevents collusion performed between the trustee and the referee community. In the off-line the consulting agents encounter the trustor agent and for not being accurate are getting penalized. Therefore, to attract the trustor agent, they need to provide their accurate information. We have analyzed the impact of the off-line process from different points of view and compared the system's efficiency with some other models.

The remainder of this chapter is organized as follows. In Section 3.3, we present the specification of agents interaction system together with the trust computing mechanism. Section 3.4 focuses on the propagation of trust through a social network and defines our framework that combines trustworthy and referee agents as reporters. Afterwards, we describe and discuss the details of computing the trust in our combined framework. In Section 3.5, we perform the maintenance that typical agent makes after a period of time since the interactions have been initiated. In Section 3.6, we outline the properties of our model in the experimental environment, present the testbed and compare the simulation results of the CRM model with the results of other well-known trust models in terms of efficiency in reputation assessments. We also discuss the features of the CRM model and its efficiency, particularly in dynamically changing environments. Finally Section 5.9 concludes the comprehensive trust framework.

3.3 Trust Evaluation Environment

3.3.1 Interaction System Structure

In this section, we define the communication messages the agents exchange during the trust evaluation process along with the corresponding dialogue game rules.

Definition 3.1 *A communication message is a tuple* $\langle CType, CValue, Ag_x, Ag_y, M, t\rangle$*, where* $CType$ $(CType \in \{Request, Reply\})$ *indicates whether it is a request or reply communication message;* $CValue$ $(CValue \in \{Information, Refuse, Not\ Have\})$ *represents the type of the message as requesting information in case of initiating the communication* $(Information)$*, refusing to reveal information* $(Refuse)$*, or not having the information in case of replying to a request message* $(Not\ Have)$*; agents* Ag_x *and* Ag_y *are respectively the sender and receiver of the message;* M *is the content of the message and finally* t *is the time at which the message is sent.*

Let $\mathcal{T}_{Ag_a}$ be the set of all Ag_a's trustworthy agents and $\mathcal{T}s_{Ag_a}^{Ag_b} \subseteq \mathcal{T}_{Ag_a}$ be the selected trustworthy agents Ag_a (the trustor) uses to evaluate Ag_b (the trustee). The selection of trustworthy agents is upon need and thus would differ from evaluation to another with respect to the interaction history between the trustor and trustee. In general, the most trustworthy agents could be a reasonable idea (ranking the trustworthy agents and selecting the first 3 in case Ag_a would like to consider 3 in each consulting agents). The set of selected trustworthy agents is subject to continuous update with respect to environment changes. This issue is discussed in more details later in this section. To request information, Ag_a uses the communication message $\langle Request, Information, Ag_a, Ag_{t1}, Trust(Ag_b), t_0\rangle$, which means Ag_a at time t_0 sends to the trustworthy agent Ag_{t1} ($Ag_{t1} \in \mathcal{T}s_{Ag_a}^{Ag_b}$), a request for information $(Information)$ related to Ag_b's trust. Consequently Ag_{t1} replies to the message by one of the following choices:
1) $\langle Reply, Information, Ag_{t1}, Ag_a, Information(Ag_b), t_1\rangle$;
2) $\langle Reply, Not\ Have, Ag_{t1}, Ag_a, *, t_1\rangle$; or

3) $\langle Reply, Refuse, Ag_{t1}, Ag_a, *, t_1 \rangle$

where $t_1 > t_0$. In the first choice, Ag_{t1} replies by sending to Ag_a the relative information (trust rating and the number of direct interactions between Ag_{t1} and Ag_b) about the credibility of Ag_b. In the second choice, Ag_{t1} informs Ag_a that it does not have any information regarding the credibility of Ag_b ($*$ represents empty message). Finally, in the third choice, Ag_{t1} refuses to reveal the requested information to Ag_a. There is a chance that Ag_{t1} replies with $Not\ Have$ reply type in order to hide its refusal of providing information. Such cases are among the situations that Ag_a would consider while adjusting its beliefs about the accuracy of the provided information. Consequently, the non-accurate agents would be penalized in the sense that a trustworthy agent for Ag_a may not be considered in $\mathcal{T}_{Ag_a}$ anymore. These details are out of scope of this framework and here we only focus on recognizing and thus avoiding the non-accurate agents. The sequence of these request and reply messages represents a dialogue game that we formalize by the following rule, where $\Rightarrow$ is the implication symbol:

$$\begin{aligned}
&\langle Request, Information, Ag_a, Ag_{t1}, Trust(Ag_b), t_0 \rangle \ \Rightarrow \\
&\quad \langle Reply, Information, Ag_{t1}, Ag_a, Information(Ag_b), t_1 \rangle \\
&\quad \vee \ \langle Reply, Not\ Have, Ag_{t1}, Ag_a, *, t_1 \rangle \\
&\quad \vee \ \langle Reply, Refuse, Ag_{t1}, Ag_a, *, t_1 \rangle
\end{aligned}$$

Meanwhile, Ag_a uses the $\langle Request, Referee, Ag_a, Ag_b, Referee(NUM), t_0 \rangle$ communication message, which means Ag_a at time t_0 sends to Ag_b a request to introduce some referees ($Referee$). The content message $Referee(NUM)$ indicates the number of referee agents (NUM) that can recommend Ag_b. Ag_b is supposed to introduce the referee agents that support him in the trust evaluation done by Ag_a. Ag_b would rely on his best trustworthy agents in this exercise. Let $\mathcal{R}_{Ag_b}$ be the set of Ag_b's referee agents. Then, Ag_b after receiving the request communication message, chooses the appropriate referee agents from $\mathcal{R}_{Ag_b}$. The selected subset, which is introduced to

Ag_a at time t_2 ($t_2 > t_0$), is denoted by $\mathcal{R}s^{Ag_a}_{Ag_b}$, where $|\mathcal{R}s^{Ag_a}_{Ag_b}| = NUM$. This issue is formalized by the dialogue game represented by the following rule:

$$\langle Request, Referee, Ag_a, Ag_b, Referee(NUM), t_0\rangle \Rightarrow$$
$$\langle Reply, Referee, Ag_b, Ag_a, \mathcal{R}s^{Ag_a}_{Ag_b}, t_2\rangle$$

After obtaining the set of referee agents from Ag_b, Ag_a continues with requesting information from each introduced referee agent at time t_3. At t_4 ($t_4 > t_3$), the requested referee agent has three possible answers: replying by giving the relative information about the credibility of Ag_b; replying with no information; or refusing to reveal the information regarding the credibility of Ag_b. Let Ag_{r1} be a selected referee agent ($Ag_{r1} \in \mathcal{R}s^{Ag_a}_{Ag_b}$), the following dialogue game rule specifies the exchanged messages:

$$\langle Request, Information, Ag_a, Ag_{r1}, Trust(Ag_b), t_3\rangle \Rightarrow$$
$$\langle Reply, Information, Ag_{r1}, Ag_a, Information(Ag_b), t_4\rangle$$
$$\vee \langle Reply, Not\ Have, Ag_{r1}, Ag_a, *, t_4\rangle$$
$$\vee \langle Reply, Refuse, Ag_{r1}, Ag_a, *, t_4\rangle$$

It is rare that the referee agent does not have information regarding the trust level of Ag_b. This is because the referee has been chosen by Ag_b based on previous direct interactions. But this does not guarantee a positive rating regarding Ag_b's credibility. The chosen referee agent is in fact facing the trustor Ag_a and since there would be after interaction off-line mechanism, the referee agent would be penalized if provides inaccurate information. Therefore, if the referee agent is not satisfied with Ag_b's behavior, it is better to retrieve the correct information (bad rating) rather than hiding it (replying "$Not\ Have$"). To this end, in case Ag_b has changed its behavior, the referee would

rationally retrieve its accurate information to obtain better rate from the trustor agent.

3.3.2 Trust Computing Mechanism

To compute trust (i.e. credibility) in our model, we first introduce the trust function as follows:

Definition 3.2 *Let $\mathcal{A}$ be a set of agents, $\mathcal{D}$ a set of domains or topics, and T a set of time points. The trust function Tr associates two agents from $\mathcal{A}$, a domain from $\mathcal{D}$, and a time point from T with a trust value between 0 and 1:*

$$Tr : \mathcal{A} \times \mathcal{A} \times \mathcal{D} \times T \longrightarrow [0, 1]$$

Given some concrete agents Ag_a (the trustor) and Ag_b (the trustee) in $\mathcal{A}$, some concrete domain D, and a time point t, $Tr(Ag_a, Ag_b, D, t)$ stands for "the trust value associated to the trustee agent Ag_b in domain D at time t by the trustor agent Ag_a". To simplify the notation, in the remainder we will omit the domain and time from all the formulas. Given agents Ag_a and Ag_b in $\mathcal{A}$, we will represent $Tr(Ag_a, Ag_b)$ in short as $Tr_{Ag_a}^{Ag_b}$. The reason behind this simplification is that our main contribution in this framework is to equip the agents to efficiently evaluate the trust and get adapted with continuous environment changes. Although the domain is important in trust evaluation (as mainly considered in some trust-based frameworks [13]), but in this framework we only focus on the adaptation of agents with dynamically changing environment and on how agile the agent is while acting where the trust evaluation is crucial. Furthermore, although the time is omitted from the formulation, it is implicitly represented as the trust function is continuous over T.

We propose a probabilistic method by investigating the distribution of the random variable X representing the trustworthiness of the trustee agent Ag_b. Let us first consider the case where X takes only two values: 0 (the agent is not trustworthy) or 1 (the agent is trustworthy). Therefore, the variable X follows a *Bernoulli distribution* $\beta(1, p)$ so that $E(X) = p$ where $E(X)$ is the expectation of the variable X and p is the probability that the agent is trustworthy.

$$f(k; p) = p^k(1-p)^{1-k} \quad for\ k \in \{0, 1\}$$

$$E(X) = p; \qquad var(X) = p(1-p)$$

Here, p is the probability we are looking for. Therefore, it is enough to evaluate the expectation $E(X)$ to find $Tr_{Ag_a}^{Ag_b}$. However, when X is a continuous variable, this expectation is a theoretical mean that we must estimate. To this end, we can use the *Central Limit Theorem (CLT) and the law of large numbers*. The CLT states that whenever a sample of size n $(X_1, \ldots, X_n)$ is taken from any distribution with mean μ, then the sample mean $(X_1 + \cdots + X_n)/n$ will be approximately normally distributed with mean μ. As an application of this theorem, the arithmetic mean (average) $(X_1 + \cdots + X_n)/n$ approaches a normal distribution of mean μ and standard deviation $\sigma/\sqrt{n}$. Generally, and according to *the law of large numbers*, the expectation can be estimated by the weighted arithmetic mean.

Our random variable X is the weighted average of n independent variables X_i that correspond to Ag_b's trust level according to the point of view of trustworthy agents $\mathcal{T}s_{Ag_a}^{Ag_b}$ and referee agents $\mathcal{R}s_{Ag_b}^{Ag_a}$. These variables follow then the same distribution. They are also independent because the probability that Ag_b is trustworthy according to an agent Ag_{t1} is independent of the probability that this agent (Ag_b) is trustworthy

according to another agent Ag_{t2}. Consequently, the variable X follows a normal distribution whose average is the weighted average of the expectations of the independent variables X_i. In our model defined in depth in the following sections, the mathematical estimation of the expectation $E(X)$ is computed in two steps, on-line and off-line estimation. In the on-line estimation four main components are considered: direct trust, rates from referee and trustworthy agents, interaction strength and interaction recency. The off-line estimation, performed after the on-line process, is finalized. The estimation is formulated to modify the trust values of the agents that have provided information in the on-line process. We refer to this process as *maintenance*, which will be addressed in Section 3.5.

3.4 On-line Trust Estimation

In this section, we discuss the on-line evaluation process in which the trustor collects some information and combines them to assess the credibility of a trustee. Two approaches can be distinguished in this process. In the former one, the evaluator only relies on what it has from previous interactions with the trustee. In the later, the trustor prefers using the information provided by some other agents to get a more accurate assessment. In fact, the direct interaction assessment is combined with the suggested ratings by the consulting agents.

3.4.1 Direct Trust Evaluation

Agents can compute the trust value of each other using their interaction histories. This would generate real numbers, which fall in the range [0,1] and thus, instead of just integer ratings (scores) 0 and 1, we would have more flexible real ratings representing the satisfaction or dissatisfaction degree of the interaction's outcome. In the general

case, agents can evaluate their interactions according to a scale of n types numbered from 1 (the most successful interaction) to n (the less successful interaction), such that the first m interaction types ($m < n$) are successful. Let ${NI_i}_{Ag_a}^{Ag_b}$ be the number of interactions of type i between Ag_a and Ag_b. Then, $Tr_{Ag_a}^{Ag_b}$ can be computed by Equation 1. This type of trust evaluation is not novel as seen in literature [13, 30]. In this Equation, the ratio of the "*number of successful outcomes*" to the "*total number of possible outcomes*" is computed, where w_i is the weight associated to the interaction type i to represent its importance and v_{ij} is the value of the interaction, which is particularly important in transactional settings to avoid two transactions with different values being treated equally. It is worthy to point out that the number of interactions ${NI_i}_{Ag_a}^{Ag_b}$ is only considered here as a mean to evaluate the strength of the connection between the agents Ag_a and Ag_b. In our approach, we do not consider the details of these interactions as it would increase the complexity of the trust evaluation.

$$Tr_{Ag_a}^{Ag_b} = \frac{\sum_{i=1}^{m}(w_i \times \sum_{j=1}^{{NI_i}_{Ag_a}^{Ag_b}} v_{ij})}{\sum_{i=1}^{n}(w_i \times \sum_{j=1}^{{NI_i}_{Ag_a}^{Ag_b}} v_{ij})} \quad (1)$$

In fact, there are two issues in weighting an interaction: 1) the importance of the interaction type (e.g., in some cases *fair* as an interaction's outcome is enough for the interaction to be counted as important, but in other cases, maybe *very good* is mandatory as an outcome type for the interaction to be counted as important enough); and 2) transaction importance (e.g, two transactions of the same type (say *good*) may have different values in terms of their actual entity). Let us consider the following example of two dissatisfactory transactions (e.g., outcome is *bad*) that have been weighted for $w_i = 3$. Basically the value 3 reflects the importance of this kind of transactions

(i.e., the weight of *bad* transactions), which could hold different values. v_{ij} is used to represent this value. For example, the first transaction has as value ($v_{i1} = 20000\$$ and the second has as value $v_{i2} = 200\$$). In this example, v_{ij} would reflect the extent to which the damage has been occurred. This idea will protect the model from attacks like reputation squeeze [19] in which one agent would obtain some positive ratings and make a bad interaction that actually makes a large damage.

Another factor should be considered to reflect the *timely relevance* of transmitted information. This is because the agent's environment is dynamic and may change quickly. The idea is to promote recent information and to deal with out-of-date information with less emphasis. The timely relevance could be represented as a coefficient when computing the agent's trust. In literature, there are similar approaches addressing this issue. For example, in [13], authors discuss about the limitations that are used in the freshness of the data to be evaluated. In our model, we assess this factor denoted by $TiR(\Delta t_{Ag_a}^{Ag_b})_{ij}$ by using the function defined in Equation 2 and we do not make the system so sensitive to the past data as it might bring up more confusion to the trustor agent. However, as later discussed in this chapter, we equip the CRM agent with an off-line mechanism that overcome this sensitivity. We call this function: the *Timely Relevance* function.

$$TiR(\Delta t_{Ag_a}^{Ag_b})_{ij} = e^{-\lambda\ ln(\Delta t_{Ag_a}^{Ag_b})_{ij}} \quad \lambda \geq 0 \tag{2}$$

The variable λ is application-dependent and $(\Delta t_{Ag_a}^{Ag_b})_{ij}$ is the time difference between the current time and time at which interaction j of type i took place. The intuition behind this formula is to use a function decreasing with the time difference.

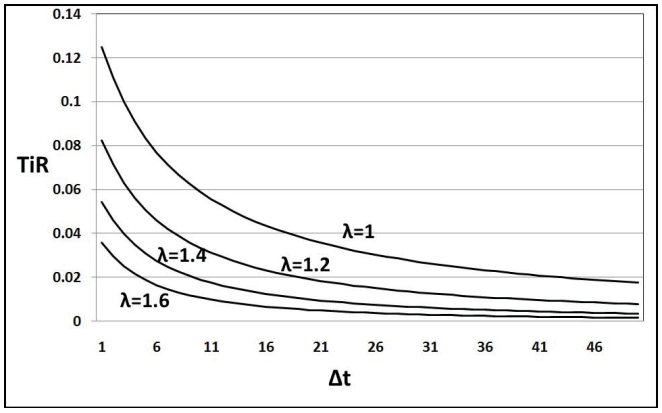


Figure 3.2: The timely relevance function with respect to different λ values.

Consequently, recent information makes the timely relevance coefficient higher. The graph of $TiR(\Delta t^{Ag_b}_{Ag_a})_{ij}$ using different λ values is shown in Figure 3.2. In some applications, recent interactions are more desirable to be considered when evaluating the trustee. In that case, the trustor uses a higher value for λ. In some other applications, even the old interactions are still valuable sources of information. In that case, the trustor assigns a smaller value to λ. Considering the involved issues, we recompute the direct trust in Equation 3. In fact, Ag_a rates each previous interaction with Ag_b in terms of its freshness, which privileges recent interactions because they are more valuable sources of information.

$$Tr^{Ag_b}_{Ag_a} = \frac{\sum_{i=1}^{m}(w_i \sum_{j=1}^{NI_i^{Ag_b}_{Ag_a}} v_{ij} \times TiR(\Delta t^{Ag_b}_{Ag_a})_{ij})}{\sum_{i=1}^{n}(w_i \sum_{j=1}^{NI_i^{Ag_b}_{Ag_a}} v_{ij} \times TiR(\Delta t^{Ag_b}_{Ag_a})_{ij})} \tag{3}$$

3.4.2 Consulting Reports: Indirect Trust Estimation

The other approach in trust estimation of the trustee consists of collecting some information in terms of suggestions from some other agents. As described before, consulting agents are divided into two groups: (1) trustworthy agents the trustor Ag_a can rely on to request information; and (2) referee agents introduced by the trustee Ag_b as recommenders. In this section, we address the selection process of the consulting agents and how to deal with the information they provide to support Ag_b.

As mentioned before, $\mathcal{T}s_{Ag_a}^{Ag_b}$ is the set of trustworthy agents selected by Ag_a for consultation. Another set to be involved in the evaluation process is the set of referee agents, which are introduced by Ag_b. Upon request from Ag_a, Ag_b replies by providing a list of the referee agents it knows. Ag_a consequently asks (some of) the referees to report on the credibility of Ag_b ($\mathcal{R}s_{Ag_a}^{Ag_b}$) and those referees reply according to their past experiences of direct interaction with Ag_b.

Assume there is a particular referee agent Ag_r that Ag_a does not know. In this case, Ag_a does not consider its suggestion about Ag_b, but it saves it anyway in order to compare it with the real behavior Ag_b performs after starting interacting with Ag_a. Thus, the referee is known by Ag_a from now on and its trust level is calculated by the adjustment of the Ag_b's real behavior and the referee's suggestion.

Let n be the total number of interaction types (see Equation 1) and $NI_{Ag_x}^{Ag_y}$ be the total number of interactions between two agents Ag_x and Ag_y, which is computed by Equation 4:

$$NI_{Ag_x}^{Ag_y} = min(\sum_{i=1}^{n} NI_{i\,Ag_x}^{Ag_y}, MV) \tag{4}$$

In this equation, MV, fixed by the system designer, is the maximum value that $NI_{Ag_x}^{Ag_y}$ can reach after a finite number of interactions. When the number of interactions goes

beyond MV, the old interactions are simply not counted, so that only the MV most recent interactions are considered. This restriction makes the model suitable for a large amount of real scenarios where agents have limited resources and computing capabilities. It is worthy to mention that the total number of interactions between Ag_t as a trustworthy agent (resp. Ag_r as a referee agent) and Ag_b, $NI_{Ag_t}^{Ag_b}$ (resp. $NI_{Ag_r}^{Ag_b}$) is an important factor because it promotes information coming from agents knowing more about Ag_b. The agents that had high number of interactions with Ag_b are considered as good sources of information about its trustworthiness in the sense that they are supposed to know Ag_b from relatively longer history of interactions. Considering this factor, Ag_a would penalize the agents with high interactions harder in the maintenance process.

Regarding the importance of the information provided by a consulting agent, we consider another factor, which reflects the confidence (in the range of $[0, 1]$) of the consulting agent on truthfulness of the provided information ($Cf_{Ag_t}^{Ag_b}$ for the typical trustworthy agent and $Cf_{Ag_r}^{Ag_b}$ for the typical referee agent). This factor has a twofold aim. First, the consulting agent would let the trustor agent Ag_a to have a better decision on the extent to which it can take this information into account. Second, the consulting agent would clarify the extent to which it can take the risk on contributing in the trust estimation process initiated by Ag_a. In the simulations, the confidence is randomly generated for each consulting agent using a Gaussian distribution with mean 0.5 and variance 0.2.

The trust equation $Tr_{Ag_a}^{Ag_b}$ we are interested in should take into account the aforementioned relevant factors: (1) the trustworthiness of trustworthy/referee agents according to the trustor Ag_a ($Tr_{Ag_a}^{Ag_t}$ and $Tr_{Ag_a}^{Ag_r}$); (2) the trustee Ag_b's trustworthiness according to the trustworthy/referee agents ($Tr_{Ag_t}^{Ag_b}$ and $Tr_{Ag_r}^{Ag_b}$); (3) the total number of interactions between these trustworthy/referee agents and Ag_b ($NI_{Ag_b}^{Ag_t}$ and $NI_{Ag_b}^{Ag_r}$), as

communicated by Ag_t/Ag_r to Ag_a following the dialogue games previously indicated in Section 3.3.1; and (4) the confidence of trustworthy/referee agents about the provided information ($Cf_{Ag_t}^{Ag_b}$ and $Cf_{Ag_r}^{Ag_b} \in [0, 1]$). Before defining this equation, let us discuss its desired properties. Some of these properties are inspired by [42].

Property 1 *Assuming that the trustee is known in the system by some agents, $Tr_{Ag_a}^{Ag_b}$ is continuous.*

This property says that at each moment the trustor Ag_a can evaluate the trustee Ag_b. This does not mean that agents are interacting every moment of time, but at every moment, the trustor can get the needed information to assess the trust value of the trustee.

Property 2 *Assuming that the trustee is known in the system by some agents, $Tr_{Ag_a}^{Ag_b}$ is strictly monotonically increasing in $Tr_{Ag_t}^{Ag_b}$ and $Tr_{Ag_r}^{Ag_b}$.*

This property says that the trust value of the trustee increases if it performs well in this environment. Consequently, agents always have incentives to do better to get their overall trust increased.

Property 3 *Assuming that the trustee is known in the system by some agents, $Tr_{Ag_a}^{Ag_b}$ is not monotonically increasing or decreasing in one of the followings: $Tr_{Ag_a}^{Ag_t}$, $NI_{Ag_t}^{Ag_b}$, $Cf_{Ag_t}^{Ag_b}$, $Tr_{Ag_a}^{Ag_r}$, $NI_{Ag_r}^{Ag_b}$, and $Cf_{Ag_r}^{Ag_b}$.*

This property says that the trust values of trustworthy agents and trustee are not necessarily correlated. The reason is that some of these agents support the trustee, but some of them do not. The same property holds for referee agents and for the number of interactions and confidence. Thus, for instance, by increasing the number of its interactions with some agents, the trustee cannot guarantee a growth of its trust value, because these agents are may not be supportive.

Property 4 *Assuming that the trustee is known in the system by some agents,* $Tr_{Ag_a}^{Ag_b}$ *is strictly monotonically increasing in one of the followings:* $Tr_{Ag_a}^{Ag_t}$, $NI_{Ag_t}^{Ag_b}$, $Cf_{Ag_t}^{Ag_b}$, $Tr_{Ag_a}^{Ag_r}$, $NI_{Ag_r}^{Ag_b}$, *and* $Cf_{Ag_r}^{Ag_b}$ *iff all* Ag_t *and* Ag_r *agents support* Ag_b.

This property gives the condition on the trustworthy and referee agents, so that increasing their trust value, number of interactions, and confidence will make the trust value of the trustee increasing. The opposite is given by the following property:

Property 5 *Assuming that the trustee is known in the system by some agents,* $Tr_{Ag_a}^{Ag_b}$ *is strictly monotonically decreasing in one of the followings:* $Tr_{Ag_a}^{Ag_t}$, $NI_{Ag_t}^{Ag_b}$, $Cf_{Ag_t}^{Ag_b}$, $Tr_{Ag_a}^{Ag_r}$, $NI_{Ag_r}^{Ag_b}$, *and* $Cf_{Ag_r}^{Ag_b}$ *iff all* Ag_t *and* Ag_r *agents do not support* Ag_b.

Property 6 *Let* X *be the set of all pieces of information that* Ag_a *uses to assess* Ag_b, *and* Y *the set of all pieces of information that* Ag_a *uses to evaluate another trustee* Ag_c, *i.e.* $X = \{Tr_{Ag_a}^{Ag_t}, Tr_{Ag_t}^{Ag_b}, NI_{Ag_t}^{Ag_b}, Cf_{Ag_t}^{Ag_b} | Ag_t \in \mathcal{T}s_{Ag_a}^{Ag_b} \cup \mathcal{R}s_{Ag_a}^{Ag_b}\}$ *and* $Y = \{Tr_{Ag_a}^{Ag_{t'}}, Tr_{Ag_{t'}}^{Ag_c}, NI_{Ag_{t'}}^{Ag_c}, Cf_{Ag_{t'}}^{Ag_c} | Ag_{t'} \in \mathcal{T}s_{Ag_a}^{Ag_c} \cup \mathcal{R}s_{Ag_a}^{Ag_c}\}$. *Suppose that there is an injective function* $f : X \to Y$ *such that for all* $x \in X$, $f(x)$ *is at least as good for* Ag_c *as* x *is good for* Ag_b; *then,* $Tr_{Ag_a}^{Ag_c}$ *is at least as great as* $Tr_{Ag_a}^{Ag_b}$.

Let us now define the trust equation $Tr_{Ag_a}^{Ag_b}$ (Equation 5) and then prove it satisfies the aforementioned properties. This equation is composed of two different terms representing the values obtained from two different consulting communities involved in trust evaluation. The functions Ω_T and Ψ_R are defined as the combination of the trust values estimated by the trustworthy and referee agents together with their related trustworthiness from Ag_a's point of view, timely relevance, confidence and number of interactions between the trustworthy and referee agents and the trustee Ag_b.

$$Tr_{Ag_a}^{Ag_b} = \frac{\Omega_T(\mathcal{T}s_{Ag_a}^{Ag_b}) + \Psi_R(\mathcal{R}s_{Ag_a}^{Ag_b})}{\Omega'_T(\mathcal{T}s_{Ag_a}^{Ag_b}) + \Psi'_R(\mathcal{R}s_{Ag_a}^{Ag_b})} \tag{5}$$

where

$$\Omega_T(\mathcal{T}s_{Ag_a}^{Ag_b}) = \sum_{Ag_t \in \mathcal{T}s_{Ag_a}^{Ag_b}} Tr_{Ag_a}^{Ag_t} \times Tr_{Ag_t}^{Ag_b} \times NI_{Ag_t}^{Ag_b} \times Cf_{Ag_t}^{Ag_b}$$

$$\Omega'_T(\mathcal{T}s_{Ag_a}^{Ag_b}) = \sum_{Ag_t \in \mathcal{T}s_{Ag_a}^{Ag_b}} Tr_{Ag_a}^{Ag_t} \times NI_{Ag_t}^{Ag_b} \times Cf_{Ag_t}^{Ag_b}$$

$$\Psi_R(\mathcal{R}s_{Ag_a}^{Ag_b}) = \sum_{Ag_r \in \mathcal{R}s_{Ag_a}^{Ag_b}} Tr_{Ag_a}^{Ag_r} \times Tr_{Ag_r}^{Ag_b} \times NI_{Ag_r}^{Ag_b} \times Cf_{Ag_r}^{Ag_b}$$

$$\Psi'_R(\mathcal{R}s_{Ag_a}^{Ag_b}) = \sum_{Ag_r \in \mathcal{R}s_{Ag_a}^{Ag_b}} Tr_{Ag_a}^{Ag_r} \times NI_{Ag_r}^{Ag_b} \times Cf_{Ag_r}^{Ag_b}$$

We notice that $Tr_{Ag_a}^{Ag_t} \neq 0 \; \forall Ag_t \in \mathcal{T}s_{Ag_a}^{Ag_b}$ and Ag_b is known for at least one Ag_t, which means $NI_{Ag_t}^{Ag_b}, Cf_{Ag_t}^{Ag_b} \neq 0$, so $\Omega'_T(\mathcal{T}s_{Ag_a}^{Ag_b}) \neq 0$.

We now show that Equation 5 satisfies Properties 1 to 6. To simplify the notation, we will omit the arguments of the functions Ω_T, Ω'_T, Ψ_R, and Ψ'_R. $Tr_{Ag_a}^{Ag_t}$ and $Cf_{Ag_t}^{Ag_b}$ are non-zero continuous functions on time, and $Tr_{Ag_t}^{Ag_b}$ is continuous on time, so by considering $NI_{Ag_t}^{Ag_b}$ as a coefficient for $Tr_{Ag_t}^{Ag_b}$ for each Ag_t, we conclude that Ω_T and Ω'_T are non-zero continuous functions. Similarly, Ψ_R and Ψ'_R are continuous, so the trust function is continuous. To show that Property 2 is satisfied, we need to prove that the partial derivative of the trust function with respect to $Tr_{Ag_t}^{Ag_b}$ is greater than zero, and the same thing with respect to $Tr_{Ag_r}^{Ag_b}$. To simplify the proof, but without loss of generality, let us consider a specific agent Ag_{t1}, then the same procedure can be applied to all other Ag_t agents. We have:

$$\frac{\partial Tr_{Ag_a}^{Ag_b}}{\partial Tr_{Ag_{t1}}^{Ag_b}} = \frac{Tr_{Ag_a}^{Ag_{t1}}.NI_{Ag_{t1}}^{Ag_b}.Cf_{Ag_{t1}}^{Ag_b}}{\Omega'_T + \Psi'_R} > 0$$

The same proof can be used for a specific referee agent Ag_{r1}, thus the satisfaction of Property 2. To show that Property 3 is satisfied, we need to show that the partial

derivative of the trust function with respect to the factors mentioned in this property is not always positive and not always negative. Here we only show the proof for the case $Tr_{Ag_a}^{Ag_t}$ and the same proof can be used for $Cf_{Ag_t}^{Ag_b} \times NI_{Ag_t}^{Ag_b}$ (the number of interactions is considered as a coefficient) and for the other factors. As we did for Property 2, we consider a specific trustworthy agent Ag_{t1} and the generalization follows. We have:

$$\frac{\partial Tr_{Ag_a}^{Ag_b}}{\partial Tr_{Ag_a}^{Ag_{t1}}} = \frac{(Tr_{Ag_{t1}}^{Ag_b}.NI_{Ag_{t1}}^{Ag_b}.Cf_{Ag_{t1}}^{Ag_b}).(\Omega'_T + \Psi'_R) - (\Omega_T + \Psi_R).(NI_{Ag_{t1}}^{Ag_b}.Cf_{Ag_{t1}}^{Ag_b})}{(\Omega'_T + \Psi'_R)^2}$$

The sign of this partial derivative depends then on the sign of the numerator, which could be positive or negative. Thus, to prove that Properties 4 and 5 are satisfied, we only need to analyze when the numerator is strictly positive, and when it is strictly negative. We have:

$$\begin{aligned}
&(Tr_{Ag_{t1}}^{Ag_b}.NI_{Ag_{t1}}^{Ag_b}.Cf_{Ag_{t1}}^{Ag_b}).(\Omega'_T + \Psi'_R) - (\Omega_T + \Psi_R).(NI_{Ag_{t1}}^{Ag_b}.Cf_{Ag_{t1}}^{Ag_b}) > 0 \\
&\textit{iff } \; Tr_{Ag_{t1}}^{Ag_b} > \frac{\Omega_T + \Psi_R}{\Omega'_T + \Psi'_R} \\
&\textit{iff } \; Tr_{Ag_{t1}}^{Ag_b} > Tr_{Ag_a}^{Ag_b}
\end{aligned}$$

Thus, the partial derivative is strictly positive iff Ag_{t1} is supportive (Property 4), and it is strictly negative iff Ag_{t1} is not supportive (Property 5). If it is equal to zero, the function is simply constant. Finally, to prove that Property 6 is satisfied, we define the injective function f as follows: $f(Tr_{Ag_a}^{Ag_t}) = Tr_{Ag_a}^{Ag_{t'}}$; $f(Tr_{Ag_t}^{Ag_b}) = Tr_{Ag_{t'}}^{Ag_c}$; $f(NI_{Ag_t}^{Ag_b}) = NI_{Ag_{t'}}^{Ag_c}$; and $f(Cf_{Ag_t}^{Ag_b}) = Cf_{Ag_{t'}}^{Ag_c}$. So, for all $x \in X$, $f(x)$ is at least as good for Ag_c as x is good for Ag_b iff $f(x) \geq x$. Consequently, from Property 4, we obtain $Tr_{Ag_a}^{Ag_c} \geq Tr_{Ag_a}^{Ag_b}$, which is the result we want to prove.

Equation 5 is used by the initial trustor Ag_a to evaluate the trustee Ag_b where each

consulting agent is supposed to forward its own estimation (together with its confidence level) for this trustee. Following the ideology that Ag_a could, to a certain extent, rely on its own history of interactions with Ag_b (direct trust evaluation approach) and partially use the second approach (indirect approaches), Ag_a gives a 100% trustworthy rate to his history and considers himself as a member of its trustworthy community. This aggregation method takes into account the proportional relevance of each approach, rather than treating the two approaches separately. Basically, the contribution percentage of each approach in the final evaluation of $Tr_{Ag_a}^{Ag_b}$ is defined regarding how informative the history is in terms of the number of direct interactions between Ag_a and Ag_b and their time recency. Therefore, consulting other agents is considered with less importance if the history represents a lower uncertainty. Doing so, the indirect evaluation approach is combined with the direct approach to end up with an accurate trust estimation of the trustor Ag_a for the trustee Ag_b. To be more precise, we aim to analyze the quality of the interactions of the trustee considering what is expected (final trust evaluation $Tr_{Ag_a}^{Ag_b}$) and what is actually performed. To this end, we have a retrospect trust evaluation, which is represented in Section 3.5.

3.5 Off-line Trust Estimation

To avoid exposing the reputation framework to dishonest ratings, two types of agents should be considered: (a) bad mouthers: agents that exaggerate by giving negative ratings; and (b) ballot stuffers: agents that exaggerate by giving positive ratings. Minimizing the effects caused by these two types of consulting agents is an important aspect in trust evaluation. Although the ratio of relationship strength can be certainly inserted as a measure of trust to increase the accuracy of referee agent's credibility, this technique is not generic as it depends on how this relationship strength is represented and

measured. To tackle this problem, we propose other parameters. First, we consider the number and time recency of interactions as factors that reflects the trustor's expectation of receiving accurate information. Second, we consider the confidence level provided by consulting agents as a mean to enable the trustor to update its friend list. To this end, we split the off-line trust estimation into two parts: Off-line Interaction Inspection and Maintenance.

3.5.1 Off-line Interaction Inspection

After each interaction, the trustor Ag_a performs an off-line interaction inspection process regarding each of the consulting agents role in the trust evaluation process. In this procedure, Ag_a considers the given rate provided by the consulting agent $Ag_c \in (\mathcal{T}s_{Ag_a}^{Ag_b} \bigcup \mathcal{R}s_{Ag_a}^{Ag_b})$, the number and recency of interactions done by the trustee agent Ag_b. The objective of this process is to assign a flag (useful/useless) for each involved consulting agent.

Since the off-line interaction inspection is a process performed after the interaction, Ag_a has a self opinion regarding the credibility of Ag_a. Therefore, we refer to $OTR_{Ag_a}^{Ag_b}$ as the actual credibility observed by the trustor Ag_a. This value is compared to the given rate provided by each consulting agent. Figure 3.3 is the off-line interaction inspector algorithm that takes the observed trust value ($OTR_{Ag_a}^{Ag_b}$), given rate of each one of the consulting agent Ag_c ($Tr_{Ag_c}^{Ag_b}$), their corresponding number of interactions ($NI_{Ag_c}^{Ag_b}$) and the provided information time recency ($TiR_{Ag_c}^{Ag_b}$) as input. This algorithm provides an array (called flag) of binary numbers about the usefulness of the information provided by each one of the involved consulting agents.

In this algorithm, first the average of the differences between the provided trust and observed one of all the consulting agents is evaluated. The rational behind this is explained by the fact that the public opinion affects the threshold of the accuracy of

credibility rating. This means if the average difference is relatively high, the trustor agent Ag_a would doubt that the trustee agent Ag_b is a consistent reliable agent, otherwise the public opinion about this agent would not achieve that divergency. Once the average difference is obtained, the consulting agents are checked one by one to be tagged either as useful or useless. The agents who provided relatively accurate ratings with an acceptable confidence level $Cf_{Ag_c}^{Ag_b} > \nu$ (ν is application-dependant and in the simulations we assume that $\nu = 0.5$) are not all tagged as useful. They are all good except the ones who do not have high number of interactions or time relevance (strong connection or holding fresh information). This is due to the fact that in credibility assessment, the ratings that are submitted at random (by chance) could not be considered as a means to evaluate the truthfulness of a consulting agent. In this algorithm, the number of interactions and time relevance of the consulting agents are compared with the ones about the trustor and trustee agents' connection. To this end, there is higher priority assigned to consulting agents that hold stronger relationship. This partition of consulting agents based on useful and useless flags is an operational way of obtaining the partition of agents as reliable and doubtful as proposed in the TRSIM framework [12].

3.5.2 Maintenance

The maintenance procedure is a periodic process initiated to update the information that the trustor agent Ag_a has about its surrounding environment. Before performing this process there are two questions that have to be addressed: (1) when does the trustor agent need to initiate the maintenance?; and (2) which agents have to be cleared in the maintenance? In the rest of this section, we answer these questions in more details.

(1) When to initiate the maintenance procedure?

function interactionInspector(Ag_a,Ag_b,$\mathcal{T}s^{Ag_b}_{Ag_a}$,$\mathcal{R}s^{Ag_b}_{Ag_a}$,$Tr^{Ag_b}_{Ag_a}$,$OTr^{Ag_b}_{Ag_a}$):[flag]

$D = 0; \qquad \overline{D} = 0$

for all $Ag_c \in \mathcal{T}s^{Ag_b}_{Ag_a} \bigcup \mathcal{R}s^{Ag_b}_{Ag_a}$

$D_{Ag_c} = |Tr^{Ag_b}_{Ag_c} - OTr^{Ag_b}_{Ag_a}|;$

$D = D + D_{Ag_c};$

$\overline{D} = \frac{D}{|\mathcal{T}s^{Ag_b}_{Ag_a} \bigcup \mathcal{R}s^{Ag_b}_{Ag_a}|};$

for all $Ag_c \in \mathcal{T}s^{Ag_b}_{Ag_a} \bigcup \mathcal{R}s^{Ag_b}_{Ag_a}$

if $D_{Ag_c} < \overline{D}$

if $NI^{Ag_b}_{Ag_c} > NI^{Ag_b}_{Ag_a}$ *flag[Ag_c]=useful*

else if $TiR^{Ag_b}_{Ag_c} > TiR^{Ag_b}_{Ag_a}$ *flag[Ag_c]=useful*

else *flag[Ag_c]=useless*

else *flag[Ag_c]=useless*

Figure 3.3: After interaction inspection algorithm for assigning usefulness flags to each involved consulting agent

There are three answers for this question:

- **Bad Performance:** When the performance of correctly evaluating agents is decreased below a predefined threshold ($1 - Tr_{Ag_a}$). Tr_{Ag_a} is in fact the reputation value that Ag_a has in the system as estimated by himself of its interactions with other agents. This value does not have to be known publicly as it is used by Ag_a to perform a type of internal maintenance. In the case of bad performance, the trustor agent realizes that its performance $P_t(Ag_a)$ in trust evaluations (regarding time t) is decreasing in almost a continuous manner. The performance in evaluation is always calculated since the most recent maintenance and is aggregated (in average) over the time elapse. Equation 6 computes the current performance ($P_t(Ag_a)$) of the trustor agent Ag_a that is regarding the current time period (t) and does

not include the agent's performance in trust evaluations that have already gone through the previous maintenance procedures.

$$P_t(Ag_a) = \frac{\sum_{Ag_b \in S(t)} |Tr_{Ag_a}^{Ag_b} - OTR_{Ag_a}^{Ag_b}|}{|S(t)|} \tag{6}$$

The trustee agent Ag_b is selected from the set of trustee agents of Ag_a ($S(t)$) which keeps all the trustee agents since the last maintenance. We assume in this set, we keep different references for the same trustee agent with a number of interactions. This would allow us to evaluate all the interactions since last maintenance. In this process, if $P_t(Ag_a) > 1 - Tr_{Ag_a}$, the trustor agent Ag_a realizes that is the time to apply a new maintenance process.

- **Huge Difference:** This is the case where Ag_a is disappointed with a noticeable low quality trust evaluation that is recently done. In this case, Ag_a realizes that the provided information is not satisfactory to the extent to which Ag_a can rely on to continue its upcoming evaluations. Therefore if the following inequality holds, the trustor agent will decide to run a new maintenance process as an exceptional case to update its belief set. The value z depends on how picky the evaluator is. In our simulations we assume $z = 0.5$. For instance, picky agents can consider $0.2 < z \leq 0.5$ and very picky agents can consider $0 < z \leq 0.2$.

$$|Tr_{Ag_a}^{Ag_b} - OTR_{Ag_a}^{Ag_b}| > z$$

- **After Certain Period of Time:** If during the evaluation process there was no problem that caused initiation of a maintenance procedure, the off-line

trust estimation system would run after a certain period of time the maintenance process to update the belief set. This would help to have a better adaptation in case of rapid changes in surrounding agents' behavior.

(2) Which agents have to be cleared in the maintenance?

In the maintenance process, Ag_a selects some agents so that applying the maintenance on them would enhance the adaptation of Ag_a with its surrounding environment. In fact, if in the process of trust evaluation, since the most recent maintenance, Ag_a's belief set has not been changed, Ag_a would consult with the same set of trustworthy agents. All these agents are then included in the maintenance process. Besides these agents, some referee agents probably were involved in some trust evaluations. Ag_a selects the referees that did provide the asked information (regarding different trustee agents) with relatively high confidence (say ν, which is set by Ag_a). The reason behind this is that the process of indirect trust evaluation is in fact a twofold aimed process. Besides obtaining accurate information, Ag_a would like to get to know new agents and to better know the previously known agents. Therefore, the truthfulness of the agents regarding the provided information could be considered as a mean to get their credibilities updated. However, Ag_a would not consider any type of referee agent. In the maintenance process, Ag_a only considers the referee agents with high confidence on their provided information. This would let Ag_a to apply the update in a more serious and reliable manner.

Let $UF_{Ag_m}^{t_1,t_2}$ and $UL_{Ag_m}^{t_1,t_2}$ be the set of useful and useless flags associated with a trustworthy or referee agent Ag_m from his interactions during the interval $[t_1, t_2]$ as computed by the algorithm given in Figure 3.3. Equation 7 gives the rate illustrating the performance of Ag_m at time t_2 considering t_1 as a point of reference. This performance is computed in terms of the number of useful and useless flags during $[t_1, t_2]$, where -1 reflects the worst performance (all the flags are useless), 0 the average performance

(the numbers of useful and useless flags are equal), and 1 the best performance (all the flags are useful). This rate is used to update the trust value of Ag_m at t_2 ($Tr^{Ag_m}_{Ag_a}(t_2)$) as illustrated by Equation 8. This update satisfies the properties that 1) if the performance is average ($\alpha_{Ag_m}(t_2) = 0$), then the trust is constant ($Tr^{Ag_m}_{Ag_a}(t_2) = Tr^{Ag_m}_{Ag_a}(t_1)$); 2) if the performance is the worst, then $Tr^{Ag_m}_{Ag_a}(t_2) = 0$; and 3) if the performance is very good, then $Tr^{Ag_m}_{Ag_a}(t_2)$ can achieve 1 depending on the value of ($Tr^{Ag_m}_{Ag_a}(t_1)$.

$$\alpha_{Ag_m}(t_2) = \frac{|UF^{t_1,t_2}_{Ag_m}| - |UL^{t_1,t_2}_{Ag_m}|}{|UF^{t_1,t_2}_{Ag_m}| + |UL^{t_1,t_2}_{Ag_m}|} \tag{7}$$

$$Tr^{Ag_m}_{Ag_a}(t_2) = \begin{cases} 1 & \text{if } Tr^{Ag_m}_{Ag_a}(t_1).(1+\alpha_{Ag_m}(t_2)) > 1 \\ Tr^{Ag_m}_{Ag_a}(t_1).(1+\alpha_{Ag_m}(t_2)) & \text{if } 0 < Tr^{Ag_m}_{Ag_a}(t_1).(1+\alpha_{Ag_m}(t_2)) \leq 1 \\ \alpha_{Ag_m}(t_2) & \text{if } Tr^{Ag_m}_{Ag_a}(t_1).(1+\alpha_{Ag_m}(t_2)) = 0 \text{ and } \alpha_{Ag_m}(t_2) > 0 \\ 0 & \text{else} \end{cases} \tag{8}$$

Figure 3.4 shows the pseudo-code of the maintenance process that computes $Tr^{Ag_m}_{Ag_a}(t_2)$. Ag_a initiates this process with respect to any of the three discussed answers to question 1. In this pseudo-code, $\mathcal{M}_{Ag_a}$ is the set of agents that are going to be selected for the maintenance and as mentioned before, all the trustworthy agents $\mathcal{T}_{Ag_a}$ are included. For all the interactions since the latest maintenance, the trustee is considered. For all the referees of the trustee in question, the selected ones are those who showed high confidence. Finally, with respect to their flags (useful $+UF$ and useless $-UL$), their update rates (α_{Ag_m}) are computed as shown in Equation 7 (with a notational simplification). Then the updated trust value is computed as illustrated in Equation 8.

Since between the variable maintenance periods the trustworthy agents of a particular trustor agent Ag_a are the same, there is a fixed number of agents that are involved

function maintenance(Ag_a,$S(t)$)
$\mathcal{M}_{Ag_a} := \mathcal{T}_{Ag_a}$;

for all $Ag_e \in S(t)$
for all $Ag_c \in \mathcal{R}s_{Ag_a}^{Ag_e}$
if $Cf_{Ag_c}^{Ag_e} > \nu$
$\mathcal{M}_{Ag_a} := \mathcal{M}_{Ag_a} \bigcup \{Ag_c\}$
end for all
end for all

for all $Ag_m \in \mathcal{M}_{Ag_a}$
consider all interactions since last maintenance
$\alpha_{Ag_m} = \frac{+UF-UL}{+UF+UL}$
if $Tr_{Ag_a}^{Ag_m} \neq 0$
$X := Tr_{Ag_a}^{Ag_m} \times (1 + \alpha_{Ag_m})$
if $X > 1$
$Tr_{Ag_a}^{Ag_m} := 1$
else $Tr_{Ag_a}^{Ag_m} := X$
end if
else
if $\alpha_{Ag_m} \geq 0$
$Tr_{Ag_a}^{Ag_m} := \alpha_{Ag_m}$
else $Tr_{Ag_a}^{Ag_m} := 0$
end if
end if
end for all
end function

Figure 3.4: The maintenance algorithm for updating trust rating performed by the trustor Ag_a

in the maintenance process. Moreover, there are some referee agents that are considered in this process and might be different with respect to different trustee agents. Because the number of involved agents in such a process is not high, the corresponding computations regarding their trust value update is negligible in the off-line trust estimation mechanism. Besides this, the trustor agent Ag_a takes the advantage of updating his trust values with respect to the referee agents that might not have high number of interactions. Furthermore, the maintenance algorithm is linear with both the number

of agents and the number of interactions (i.e. $O(|\mathcal{T}_{Ag_a}| + \prod_{Ag_e \in S(t)} |\mathcal{R}s_{Ag_e}^{Ag_a}|)$) where $|S(t)|$ is the number of interactions with different trustee agents (say Ag_e as a particular trustee agent), $|\mathcal{T}_{Ag_a}|$ is the number of trustworthy agents, and $|\mathcal{R}s_{Ag_e}^{Ag_a}|$ is the number of referee agents for a given trustee agent Ag_e. We notice that we need to compute all the interactions with referee agents even if some of them are common to different trustee agents, which justifies the product over those trustee agents $S(t)$. The linear complexity of the proposed maintenance process makes it computationally efficient.

3.6 Analysis and Experimental Simulation

3.6.1 Implemented Testbed

In this section, we assess the CRM model efficiency and describe the implementation of the tested. We also compare our model with three well known models as benchmarks: FIRE [31, 32], Referral [97, 99], SPORAS [101], Travos [83] and BRS [33]. All these models are explained in details and discussed in the related work chapter (Chapter 2). In the implemented testbed, agents are implemented as $Jadex^{©TM}$ agents, i.e. they inherit from the basic class $Jadex - Simulator^{©TM}\ Agent$. The agent reasoning capabilities are implemented as Java modules using logic programming techniques. As Java classes, agents have private data called *Belief Data*. The different dialogue games (presented in Section 3.3.1) are given by a data structure and implemented using tables and the different actions expected by an agent in the context of a particular dialogue game are given by a table called $data\_representative\_manager$. The different agents' reputation values that an agent has about other agents are recorded in a data structure called $data\_reputation$. Each agent also has a knowledge base about the reputation of other agents, called $table\_reputation$. Such a knowledge base has the following structure: $Agent - name$, $Agent - reputation$, $Total - interaction - number$ and

$Recent - interaction - time$. The visited agents during the evaluation process and the agents added in the reputation graph are recorded in two $Jadex^{©TM}$ $belief sets$ called: $table\_visited\_agents$ and $table\_graph\_reputation$.

The testbed environment (represented in Table 3.1) is populated with 200 agents categorized by two agent types: (1) service provider agents that are supposed to provide services (for simplicity, we assume that only one type of service is provided and therefore consumed); and (2) service consumer agents (equipped with the different trust models) that are looking for service providers to interact with and consume the provided service. As in FIRE and Travos, in the rest of this section we use the *gained utility* as a measurement for the quality of obtained service (QoS) in terms of satisfaction, response time, price, etc. Thus, the gained utility depends on the performance of service provider. We consider two service consumer groups to compare with our model CRM: (1) group1 (FIRE, Referral and SPORAS); and (2) group2 (Travos and BRS). The criterion used in this separation is the degree of sensitivity of the models to the environment and changes of behavior of the service providers. Group1 does not consider the continuous change of agents behaviors. The agents in this group tend to accurately maintain the trust process rather than putting effort on updating trust regarding the environment changes. Group2 takes action in response to such changes more rapidly. Generally, service providers are different and thus provide diverse range of service qualities. Furthermore, the consumer agents using these services obtain different gained utilities. In CRM, service agreements and generally intercation details such as expectations and contexts are abstracted. The focus is mainly on the numerical evluation of trust. But this model could be completed using the proposed model in [75] as the context and satisfaction criteria are taken into account. The objective is to capture subtle details regarding consumer-oriented selection process.

Each agent (either service provider or consumer) is located randomly in the environment and has been centralized and known by all other agents that are in its network of activity. The ordinary agents are given more populated network because they act normal, so in realistic environment they are more comparing to good providers or bad providers. The fickle agents are given the most populated network in order to distribute them in the environment to catch the capabilities of different trust models in treating them. In the network of agents, those that are close enough, have beliefs about each other. However, this does not exclude the fact that agents extend their activity areas and gradually get acquainted with other agents that are not in their activity areas. This allows agents to evaluate some other agents to interact with and update their belief sets based on the interaction output. The agents' belief sets are built and updated upon internal (previous history) and external (using consulting agents) information.

The simulation consists of a number of consequent runs in which agents are activated and build their private knowledge, keep interacting with one another, gain utility and enhance their overall knowledge about the environment. The more an agent knows the environment, the better it can choose service providers and thus, the more utility it gains. Agents are free to ask others for their beliefs about the service provider to be selected. Finally, each agent requests the service from the most trustworthy and reliable provider according to him. Table 3.1 represents the four types of service providers we consider in our simulation: *good*, *ordinary*, *bad* and *fickle*. The first three provide services according to the assigned mean value of quality with a small range of deviation. However, fickle providers are more flexible as their range of quality covers all possible outcomes. To put the system in a more tight situation, we use a high number of fickle agents.

Since the major difference between the considered models is the trust mechanism

Table 3.1: Testbed environment

Service Provider Agents (S.P.)	**S.P. Agent Type**	**Density in the S.P. Community**	**Provided Utility at Each RUN**		**Radius of Activity**
			Range	Standard Deviation	
	Good	15.0%	]+5, +10]	1.0	25
	Ordinary	30.0%	]-5, +5]	2.0	28
	Bad	15.0%	]-10, -5]	2.0	25
	Fickle	40.0%	[-10, +10]	-	30
Service Consumer Agents (S.C.) Group1	**S.C. Agent Type**	**Density in the S.C. Community**	**Number of Joining Agents at Each RUN**		**Radius of Activity**
	CRM	25.0%	6		35
	FIRE	25.0%	6		35
	REFERRAL	25.0%	6		35
	SPORAS	25.0%	6		35
Service Consumer Agents (S.C.) Group2	**S.C. Agent Type**	**Density in the S.C. Community**	**Number of Joining Agents at Each RUN**		**Radius of Activity**
	CRM	33.3%	10		35
	Travos	33.3%	10		35
	BRS	33.3%	10		35

they employ for credibility assessment, the utility gained by each model is considered as its efficiency in selecting reliable service providers. Doing so, we compare CRM with other models in two perspectives, honest (Section 3.6.2) and biased (Section 3.6.3) environments. In honest environments, agents are supposed be honest in the sense they reveal their beliefs with full accuracy. However, in biased environments, agents can reveal inaccurate information. Comparison is done first between CRM, FIRE [31] (a successful trust model with high performance), SPORAS [101] (a centralized approach), and Referral [97] (following the concept of reference in an honest environment). Travos [83] and BRS [33] are two other models that we compared CRM with in terms of how they survive in such biased environment where agents constantly change their behaviors. Like CRM, Travos and BRS are designed to take actions while agents are not fully trustworthy. These models differ from CRM in the trust assessment mechanism and analysis they perform in order to choose the best possible provider. In such an environment where agents have an intermittent attitude, a successful trust model is the one that adapts itself to new situations.

3.6.2 Honest Environment

Figure 3.5 depicts the overall comparison of different models. The testbed consists of a number of runs represented as the horizontal axis, and the ranking mean value for the utility gained of each group is represented in the vertical axis. As the runs are elapsing, each service consumer is using a particular model to find the most trustworthy service provider and thus, gain the most utility. First, the mean value of the gained utility by agents using the same trust model is computed. Then, the mean values obtained from different trust models are compared with each other using two sample t-test with 95% of confidence level to show the overall outperforming of CRM and FIRE compared to the other two models.

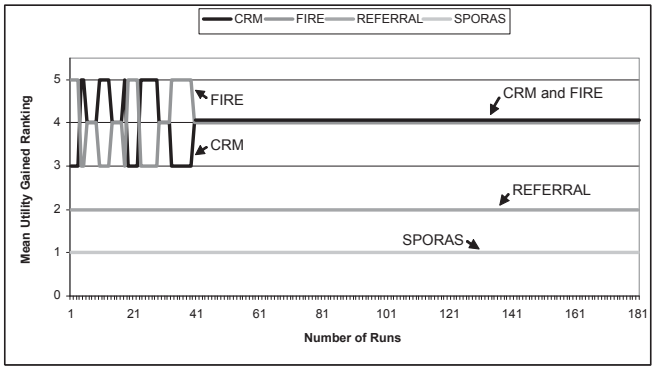


Figure 3.5: Comparison of CRM with FIRE, Referral and Sporas in terms of mean utility gained at each run in an honest environment

Groups reflect the performance of four different trust models we considered for comparison. SPORAS model evaluates the trust based on very recent interactions of each agent. Moreover, in this model, the credibility of highly interacted agents undergo a minor change compared to one with low number of interactions. Since SPORAS (generally used as benchmark in the literature) is a centralized model, it suffers from inconsistency of the trust values associated to agents while they register upon entrance in the system. Thus, this model would not perform well in situations when the good service providers are new to the system and remain unknown for a longer time

compared to others. Moreover, we still observe the problem of fake advertising to the central agent to get more benefit. Therefore, SPORAS performs weak in selecting the best service providers. Referral agents directly consider how to place trust in others and emphasize the key properties that affect the trust assessment. However, they do not restrict the suggestions of other agents, which lead them to assess the credibility of an unknown or partially known service provider. This may impact the selection of good providers from the beginning of simulation. FIRE agents [31] regulate the problem of collecting the required information by the evaluator to assess the trust of its partner. In addition, they apply certified reputation introduced by the trustee agent. As results of t-test illustrated in Figure 3.5, the commutative utility gained over the 500 elapsed runs by the FIRE and CRM agents are culminated to be the highest as both methods select good service providers, and therefore gain the highest possible utility (for space reasons, only the first 180 runs are shown in the figure). In this environment, the agents are considered honest and they reveal their beliefs with full accuracy. In the next section, we carry on comparison in the biased environment in which agents would not necessarily reveal their beliefs with 100% accuracy. As a result, the trustor can get confused in the trust assessment. Objectively, we discuss how the CRM agents cope with such a problem.

3.6.3 Biased Environment

Being more realistic, we exposed the same agents in a very biased environment in which agents, serving some certain goals, may reveal much less accurate information. Each agent employs its corresponding trust model to accumulate the utility gained through interactions. In general, the agents with more adaptable trust framework would be able to express more efficient performance and thus, obtain higher utility from the environment.

To prove the applicability of the proposed framework, we discuss the features allowing the CRM model to perform higher over the FIRE, Travos and BRS models in terms of efficiency. This discussion considers two perspectives. The former is in terms of balancing the trust assessment process by considering different involved agents. This comparison is done between FIRE and CRM, which is also highlighted with a detailed scenario. The latter discussion focuses on how agents are sensitive to the environment inconsistency and how it would be possible to gain more from diverse types of service providers. The CRM model is compared with the Travos and BRS models to show how these dynamic models act in an extensive intermittent environment. Our main objective is to investigate different models' abilities in satisfying the ideal trust model's criteria.

FIRE is a successful trust-certified reputation model, which addresses the problem of lack of direct history. Agents evaluate the trust of other agents as decentralized services. However, the FIRE agents do not recognize the agents that have got the good ratings and performed bad either in terms of inaccurate ratings provided for some others or the bad obtained utility. The CRM agents are equipped with a maintenance mechanism, which enables them to recognize change of behavior of others and respectively adjust their beliefs regarding the trust of some particular consulting agents. This mechanism is also effective in recognizing collusion behavior, by which agents intentionally reveal inaccurate information, aiming to gain more benefit at the end. This change of behavior should be recognized and the benefit of other agents should get adjusted. This process helps in quickly recognizing the fickle agents that may provide any quality of service.

Figure 3.6 shows a graph plotting fickle selection percentage versus number of runs. The graph highlights the difference of having and missing the maintenance regarding the behavior of the CRM and FIRE agents. In the first 80 runs, we observe that the CRM agents are reducing the selection of fickle agents as the time goes on. This is

because the CRM agents perform maintenance on the behavior of the fickle agents that provide a bad utility after the interaction and deduce their beliefs about them, which leads to less selection afterwards. The performance of the FIRE agents remain almost the same as they do not recognize the fluctuated behavior of the fickle agents. The picks of the CRM graph (P_1 and P_2) are simply because of a selection of few number of the CRM agents at each run, and therefore, the maintenance they perform generally has low effect on the consequent run until they are selected or distribute their ratings about the typical fickle agent they have done maintenance for. Hence, the curve goes down in a fluctuated manner until all the fickle agents lose their credibilities and never get selected, which happens in P_3. In a similar way, Figure 3.7 illustrates the good agent selection percentage versus the number of runs. This graph is the complementary of the one shown in Figure 3.6 as the less fickle providers are selected, the more good providers are recognized. As a result of maintenance, the CRM agents would then enhance their performance since good providers are always selected.

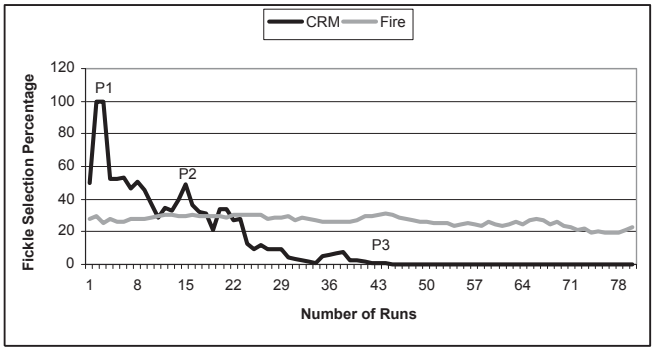


Figure 3.6: Comparison of CRM and FIRE in terms of selecting fickle service providers along the elapsing runs in a biased environment

In this section, we also analyze the CRM behavior compared with BRS and Travos, which are similar to CRM in the sense that they do consider other agents' suggestions while evaluating the trust of some specific agents (service providers) and discard inaccurate suggestions aiming to adapt themselves to the environment inconsistency

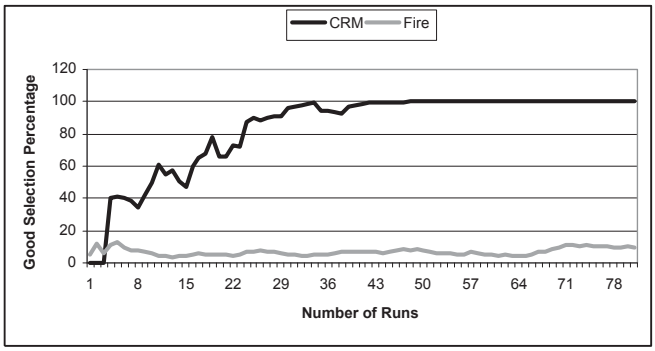


Figure 3.7: Comparison of CRM and FIRE in terms of selecting good service providers along the elapsing runs in a biased environment

attitude. In BRS, the trustor agent evaluates the recommender agents' suggestions using the beta distribution method and ignores the suggestions that deviate the most from the majority of ratings. BRS is in fact a relatively static trust method, which causes a low-efficient performance in very dynamic, open and biased environments. Cumulative gained utility vs. number of runs is shown in Figure 3.8. In this graph, all the agents consider the history of interactions in their selections. The BRS model is not sensitive to an agile behavior change. This means if a BRS agent decides to evaluate a new agent, it considers the majority of ratings, which are supposed to be truthfully revealed about the trustee agent. In the case where the trustee agent has just changed its strategy, the trustor agent would lose in trust assessment and does not maintain any action to verify the accuracy of the gained information. It may take as much time that other agents perform a number of direct interactions to start rating the spurious trustee agent. Therefore, as illustrated in Figure 3.10, the BRS agents would have a higher percentage of fickle providers selection and a relatively less percentage of good providers selection (illustrated in Figure 3.9). The peaks in Figure 3.10 are again as a result of rational agents learning the credibilities of their surrounding agents in the environment. It takes some while for the active agents to enhance the accuracy of their belief sets. Generally, it would take more time for the BRS agents to adapt themselves to the new

environment conditions. The simulation results outlined in this section are all based on 50% agent activation rate.

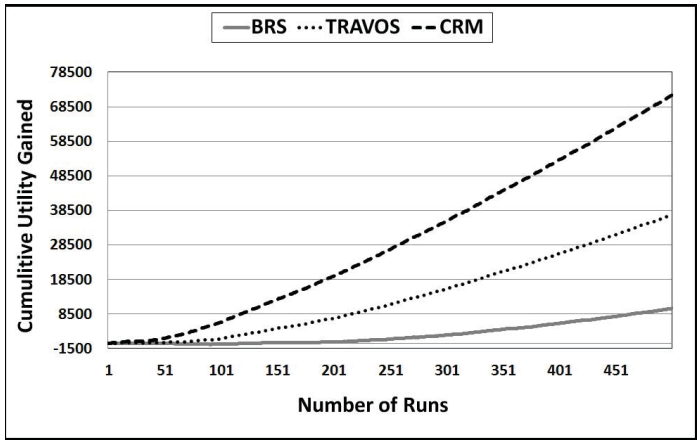


Figure 3.8: Comparison of CRM, Travos and BRS in terms of cumulative utility gained along the elapsing runs in a very biased environment

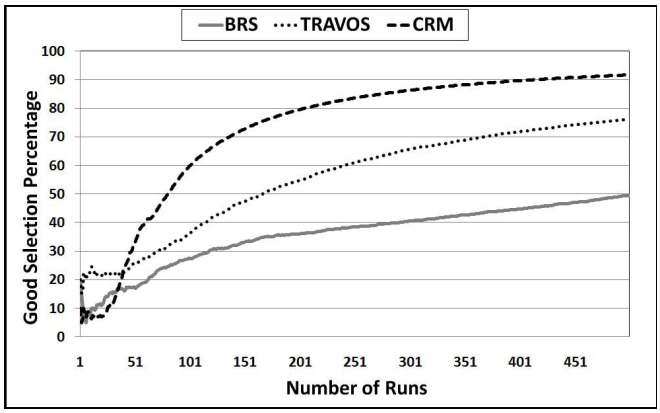


Figure 3.9: Comparison of CRM, Travos and BRS in terms of good provider selection percentage along the elapsing runs with 50% activation rate in a very biased environment

Travos [83] has a method similar to BRS. It also uses beta distribution to estimate the trustworthiness of an agent based on the previous interaction experience. The Travos model also does not have a partial rating. It gives the trustor agent the authority to merge its own experience with recommendations from other agents. However, unlike BRS, Travos filters the surrounding agents that are fluctuating in their reports about

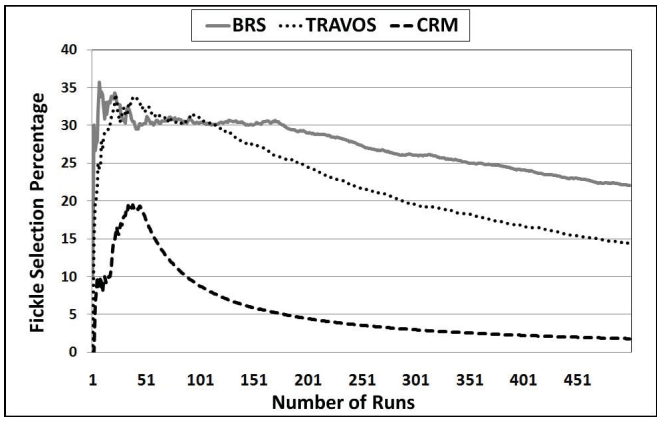


Figure 3.10: Comparison of CRM, Travos and BRS in terms of fickle provider selection percentage along the elapsing runs with 50% activation rate in a very biased environment

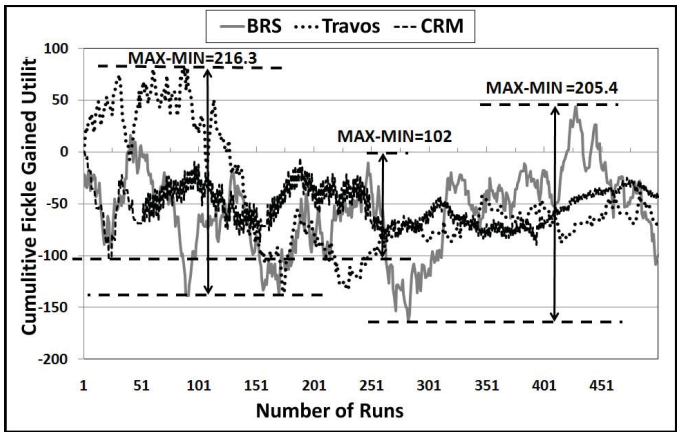


Figure 3.11: Comparison of CRM, Travos and BRS in terms of fickle gained utility along the elapsing runs with 50% activation rate in a very biased environment

a specific trustee agent. To some extent, this feature would implement a partial suggestion consideration and thus, the Travos agents would learn faster compared to the BRS agents. Ratings concerning the good and fickle selection percentage shown in Figures 3.9 and 3.10 reflect higher efficiency of Travos compared to BRS. The Travos agents are capable of preventing the concept of fake reputation in which a group of agents artificially increase their reputation by their collusive behaviors. However, the Travos model considers that agents do not change their behaviors during runs. This unrealistic assumption affects the accuracy of trust estimation in a very biased environment.

On the other hand, lack of agile learning ability for agents will weaken the protection against collusion and fake behaviors. This is the case when a surrounding agent is being discarded because of providing diverse reports about a particular trustee agent. In this case, the deviation would be filtered by mistake if the reports are reflecting the fickle attitude of that particular provider.

The Travos and BRS trust models enable agents to sense the environment and upgrade their beliefs along the elapsing time. Compared to the performance of FIRE, the Travos and BRS agents attempt to improve their best agent selection. However, these models have some aforementioned limitations that cause wrong direction to accurate trust estimation. In CRM, the aim is to improve the trust mechanism to deal with these limitations by enabling agents to adapt themselves while the environment is strictly intermittent. The CRM agents are equipped with the maintenance procedure by which they update their beliefs about the service providers together with the accuracy of the ratings provided by the neighbor agents in support or against a specific provider. Considering all the involved parameters, the agent that is doing maintenance balances its beliefs to be more accurate in terms of knowing the best provider and the best neighbors that can be consulted. Therefore, as shown in Figure 3.8, the CRM agents would gain more utility compared to the other two models. Figures 3.9 and 3.10 reflect the CRM agile reaction to increase its good selection percentage very fast, and thus, decrease the maximum possible its fickle selection percentage. To better analyze the affect of the fickle agents that we concern not to select them, we have shown the gained utility from fickle agents in each run in Figure 3.11. This figure elaborates the fact that the gained utility from selecting fickle agents is ideally minimized in the sense that there is no guarantee about the provided utility after selecting a fickle agent. Therefore the high performance agents would not rely on this utility and thus, they accumulate the obtained unitively from selecting the good providers.

The detailed simulation environment with different settings concludes the fact that CRM functions as a comprehensive trust framework that satisfies the aforementioned ideal trust framework's criteria (accuracy, rationality, adaptability and agileness represented in Chapter 1). This trust framework outperforms related work mainly because of its agile reaction to change of behaviors and its ability to reconstruct the accurate belief set. Considering the maintenance part of this model, a CRM agent is able to know its surrounding environment relatively faster than the one using similar trust frameworks.

3.7 Conclusion

Our contribution in this chapter is the proposing a new probabilistic-based model to secure multi-agent systems in which agents communicate with each other using dialogue games. The trust assessment procedure is composed of on-line and off-line evaluation processes. On-line framework is based upon trustworthy and referee agents as well as several other features. Objectively, this allows enhancing the accuracy for agents to make use of the information communicated to them by other agents. Off-line framework considers the communicated information to judge the accuracy of the consulting agents in the previous on-line trust assessment process.

Our model has the advantage of being comprehensive and taking into account four important factors: (1) the trust (from the viewpoint of the trustor agents) of the trustworthy agents; (2) the trust value assigned to trustee agents according to the point of view of trustworthy agents; (3) the number of interactions between trustworthy agents and the trustee agents; and (4) the timely relevance of information transmitted by trustworthy agents. Moreover, the original process of maintenance proposed in this framework enables agents to dynamically adjust their beliefs and their trustworthy community in a more efficient manner. The resulting model allows us to produce a

comprehensive assessment of the agents' credibility in a software system even if the environment is very biased. The proposed mechanism accuracy is compared with other related models and discussed in details to prove the capabilities of our framework. In conclusion, our proposed trust framework satisfies all the four factors regarding an ideal trust model.

Chapter 4

Reputation-based Framework

Applied to Agent-based Communities of Web Services

4.1 Background

In the previous chapter, we proposed a trust framework that is used to rate agents' reliability in interactive multi-agent system with dynamic environmental changes. The trust is individual's opinion regarding reliability of an entity. This could be generalized to public opinion that we refer to as reputation. The reputation then reflects public opinion regarding an entity's reliability. In this chapter, we propose a new framework to address reputation evaluation problem. This approach is built on the trust framework that we proposed before. So we mainly concentrate on the public aspect of the reliability computation. Moreover, we consider the network of web service agents as the infrastructure of the proposed model because such a network of web service agents is a suitable environment to discuss the public opinion regarding reliability of intelligent agents.

The proposed reputation framework is based on some relevant parameters that are inspired by the case study that we consider in this chapter. In literature, there are a number of related works that aggregate the relevant parameters to compute the reputation of an agent. But we distinguish our proposed framework from others by stressing the fact that the contribution of our framework in this chapter is to maintain truthful

feedback submission system. Upon reliable feedback pool, we aggregate the parameters utilized to compute the reputation. Regarding the truthful feedback submission, there are some relative models that encourage the feedback poster to act truthfully, but in this chapter we focus on malicious feedback detection, and the ways to discard fake feedback as well as penalizing the malicious poster agent. We generally investigate the scenarios where rational agents estimate better results via truthful actions. This type of reputation analysis is new and we have extended it into different directions in a number of publications [49, 50].

As one of the recent technologies for developing loosely-coupled, cross-enterprize business processes (usually referred to as B2B applications), a plethora of web services exists on the web waiting to receive users' requests for processing. Such requests are usually competitive in a reputation-driven manner. As pointed out in Chapter 2, we implement and apply our proposed reputation mechanism on the web services setting, where web services are supposed to be associated with agents that act on their behalf. One general way for such reputation assessment is collection of the after-interaction feedback that users provide with respect to the quality of the received service. However, in feedback-based reputation mechanisms, the precise reputation assessment needs to be verified. Selfish web services might manage to provide feedback that support them in the reputation mechanism. In general, online reputation mechanism is always subject to get violated with selfish web services.

Another way to address the selection problems is to gather web services having similar functionalities into a community. Community of web services (CWS) is a gathering of single and functionally similar web services that are aggregated to perform as one community while offering unique or variety of services [45,48]. The main property of a CWS is to facilitate and improve the process of web service discovery and selection and effectively regulate the process of user requests. There are underlying reasons for

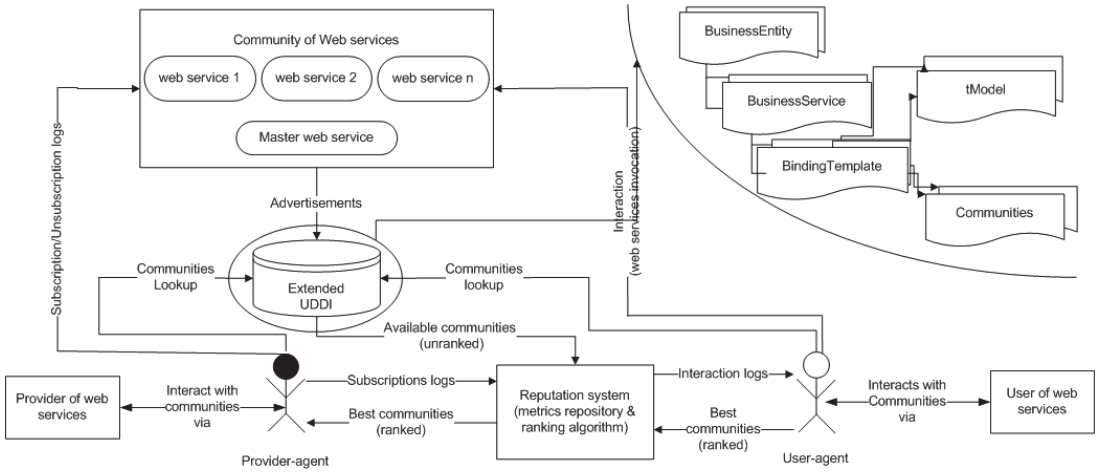


Figure 4.1: Architecture of reputation-based community of web services

this. In general, the individual web services fail to accept all the requests for them, and thus refuse to accept a portion of their concurrent requests. This would decrease their overall reputation in the environment and would lead to loose some users. In CWSs, the community gathers a set of functionally homogeneous web services. Given that some communities offer the same functionality (hotels booking, weather forecasting, etc.), there is a competition between different communities. In this case, reputation is considered as a differentiation driver of the communities. Moreover, reputation helps users to select the most reputable community, which would provide the best QoS, and helps providers to join the best community, which would bring them the most value. Users assess the reputation of the community and upon that request a service. Although the service selection process might be simplified, still communities might distract the reputation mechanism to support themselves. To this end, the reputation mechanism is needed to maintain a truthful service selection procedure.

4.2 Architecture of Reputation-Embedded Web Services Communities

In this section, we represent the CWSs architecture [16], which is designed to maintain the reputation of the communities. Here we assume that each web service is associated with a community and do not function alone. If a web service is not registered in a community, it could not be invoked by a user. Indeed, a web service can be registered in one of many communities. In Figure 4.1, we represent different components of the architecture, with their reputation and interactions. These components together with their detailed performance are explained as follows:

User agent. It is a proxy between the user and other interacting parties such as the extended UDDI, CWS and the reputation system.

Master agent. This agent is considered as the representative of the community in the sense that it manages the community requests in selecting the proper web service. Meanwhile, the master agent hires (or fires) some web services to join (or leave) the community. In general, the master of the community always tends to increase the community's performance and consequently, its reputation level.

Provider agent. Like the user agent, it relates the provider with the extended UDDI, CWS and reputation system.

Extended UDDI. The traditional UDDI XML schema is based on six types of information, allowing people to have information in order to invoke the web services [62]. In the UDDI registry, we restrict the access of the agents in the sense that user and provider agents only consult the list of masters, whereas the masters have access to the list of the web services in the UDDI registry. By adding this new information concerning the CWSs, we would clarify which CWS a web service belongs to.

Reputation system. Considering the fact that the CWSs could offer the same service, they always compete in order to obtain more requests. Therefore, evaluating CWSs is unavoidable for the users and providers. To be able to compute the reputation of these communities, the user and provider agents must gather operational data, reflecting different performance metrics, about the interaction between the user, provider and CWS. The user agents should intercept some logs like *Submission log*, *Response Time log*, *Invocation log*, *Success log*, *Failure log*, *Recovery log* and so on. It is important that the user and provider agents are independent parties in order to intercept trusted run-time data about each web service interaction.

The reputation system is the core component in this architecture. Its first functionality is to register the run-time logs; and the second functionality is to rank the communities based on their reputation by using a ranking algorithm. The ranking algorithm would maintain a restrictive policy, avoiding the ranking violation, which could be done by some malicious CWSs. The violation, which has not been considered in [16] could be done by providing some fake logging data (by some colluding users) that reflect positive feedback in support of the CWS, or by fake negative data that is registered against a particular community. To deal with this violation, we propose to assign a controller agent Cg. The task of this agent is to update the CWS reputation rankings in order to drop inaccurate registered data and thus enhance accuracy of the reputation system. The detailed discussion of this issue is provided in Section 4.4.

Controller agent. Cg is the assigned agent that takes the logging file under surveillance and updates the assigned reputations to the communities. Cg is mainly responsible to remove the cheated feedback that support particular communities. Investigating the recent feedback, Cg recognizes the fake feedback and accordingly analyzes the further actions of the community. In general, Cg may fail to accurately detect the fake feedback or similarly may recognize normal feedback as fake. Therefore, malicious

communities always consider this fake detection and analyze their chance of successful cheating.

4.3 Reputation Model

For simplification reasons, but without loss of generality, in the remainder of this framework, we only consider the users point of view (rather than users and providers) in reputation assessment. In order to assess the overall reputation of a CWS, the user needs to take some correlated factors into account. In Section 4.3.1, we present the involved metrics that a user may consider in this assessment. Consequently, in Section 4.3.2, we explain the methodology that the user uses to combine these metrics in order to assess the reputation of a CWS.

4.3.1 Metrics

Responsiveness Metric: Let C_i be the community that is under consideration by user U_j. Responsiveness metric depicts the time to be served by a CWS. Let $Res_{C_i}^{U_j,qs^t}$ be the time taken by the master of the community C_i to answer the request received at time t (qs^t) by the user U_j. This time includes the time for selecting a web service from the community and the time taken by that web service to provide the service for the user U_j. When it is understood from the context, C_i will be removed from the notations. Equation 1 computes the response time of the community C_i, computed with U_j during the period of time $[t_1, t_2]$ ($Res^{U_j,[t_1,t_2]}$), where n is the number of requests received by this community from U_j during this period of time.

$$Res^{U_j,[t_1,t_2]} = \frac{1}{n}\sum_{t=t_1}^{t_2} Res^{U_j,qs^t} \times e^{-\lambda(t_2-t)} \quad (1)$$

Here the factor $e^{-\lambda(t_2-t)}$, where $\lambda \in [0,1]$ is application-dependent and reflects the time recency of the received requests so that we can give more emphasize to the recent requests. If no request is received at a given time t, we suppose $Res^{U_j,qs^t} = 0$.

InDemand Metric: It depicts the users' interest for a community C_i in comparison with the other communities. This factor is computed in Equation 2.

$$InD^{[t_1,t_2]} = \frac{Req^{[t_1,t_2]}}{\sum_{k=1}^{M} Req_{C_k}^{[t_1,t_2]}} \tag{2}$$

In this equation, $Req^{[t_1,t_2]}$ is defined as the number of requests that C_i has received during $[t_1, t_2]$, and *M* represents the number of communities under consideration.

Satisfaction Metric: Let Sat^{U_j,qs^t} be a feedback rating value (which is supposed to be between 0 and 1) representing the satisfaction of U_j with the service regarding its request qs^t sent at time t to C_i. Equation 3 shows the overall satisfaction of the user U_j to community C_i.

$$Sat^{U_j,[t_1,t_2]} = \frac{1}{n} \sum_{t=t_1}^{t_2} Sat^{U_j,qs^t} \times e^{-\lambda(t_2-t)} \tag{3}$$

4.3.2 Metrics Combination

In order to compute the reputation value of a CWS (which is between 0 and 1), it is needed to combine these metrics in a particular way. Actually, the *Responsiveness* and *Satisfaction* metrics are the direct evaluations of the interactions between a user and a CWS whereas the *inDemand* metric is an assessment of a community in relation to other communities. In the first part, each user adds up its ratings of the *Responsiveness*

and *Satisfaction* metrics for each interaction it has had with the CWS. Equation 4 computes the reputation of the community C_i during the interval $[t_1, t_2]$ from the user U_j's point of view. In this equation, ν represents the maximum possible response time, so that if a community does not respond, we would have $Res^{U_j,[t_1,t_2]} = \nu$. In the second part, the *inDemand* metric is added. Therefore, the overall reputation of C_i from the users' point of view is obtained in Equation 5.

$$Rep^{U_j,[t_1,t_2]} = \eta(1 - \frac{Res^{U_j,[t_1,t_2]}}{\nu}) + \kappa Sat^{U_j,[t_1,t_2]} \tag{4}$$

$$Rep^{[t_1,t_2]} = \chi \frac{1}{m} \sum_{j=1}^{m} \left(Rep^{U_j,[t_1,t_2]}\right) + \phi \, InD^{[t_1,t_2]} \tag{5}$$

Where $\eta + \kappa$ = 1 and $\chi + \phi$ = 1.

In the rest of this chapter, we call $Rep^{[t_1,t_2]}$ as the reputation of the community C_i computed with respect to interacting users' points of views as well as the community's inDemand metric. But in next chapters, we discuss about the reputation of web service agents computed by the central reputation system and refer to this parameter by R_i reflecting web service i's reputation value. In context, these two parameters are same, but considering the time interval, we differently refer to the parameter in this chapter.

4.4 Feedback Logging Mechanism

Without loss of generality, in a network composed of CWSs, master agents (as representatives of communities) are selfish and may alter their intentions in order to obtain more benefits (in terms of popularity). This could happen by improving one's reputation level or by degrading other's reputation level. We respectively refer to these cases

as fake positive/negative alteration. Violating the logging feedback (distracting the reputation levels) could lead to system inconsistency in the sense that low quality CWSs may obtain more users or high quality communities may loose some users. Therefore, it is important to avoid such attacks and keep the logging mechanism accurate. In the rest of this section, we explain how to perform fake positive/negative correction (recognition and adjustment) and thus effectively maintain a reputation adjustment.

In the proposed architecture for the CWS, the reputation is computed based on the information obtained from the logging system that over the elapsing time, users leave their feedback. Thus, it is essential to keep such logging file accurate and discourage malicious actions. It is the responsibility of the controller agent Cg to maintain an accurate attack-resilient logging file. As a part of the UDDI system, Cg has the authority to update information such as overall reputation level of any CWS. In this framework, we assume that this agent is highly secured in order to avoid being compromised. However, if Cg gets compromised with a given community, then inconsistent actions of Cg could be recognized by some other communities, given the fact that they are competing with one another. But this issue is out of the scope of this chapter.

4.4.1 Fake Positive Correction

Fake positive recognition. One of the main responsibilities of the controller agent Cg is to perform fake positive correction. To this end, initially Cg should recognize a malicious behavior from one or a set of user agents (that could possibly collude with a particular community). This recognition is done based on the recent observable change in the reputation of a community. To this end, Cg would always check the recent feedback of the communities. So Cg would consider the reputation that is computed for a specific period of time $[t_1 - \epsilon, t_1]$, where t_1 is the current time. The value ϵ is set by the controller agent regarding to the system inconsistency in the sense that if the

network is inconsistent, so Cg would need to check most recent feedback (ϵ as relatively small amount). Otherwise, Cg would take even older feedback into account (ϵ as relatively large amount). Thus, $Rep^{[t_1-\epsilon,t_1]}$ is the reputation of the community C_i obtained from data measured from $t_1 - \epsilon$ to t_1. Different values of ϵ will be used in the simulation to observe the effect of the considered period on the overall recognition. Let $U^{[t_1-\epsilon,t_1]}$ be the set of users that during this time interval have provided a feedback for the community C_i, and t_b be the beginning time of collecting feedback. Cg would consider the positive feedback to be suspicious if the reputation improvement ($Rep^{[t_1-\epsilon,t_1]} - Rep^{[t_b,t_1]}$) divided by the number of users that caused such improvement is greater than the predefined threshold ϑ, i.e:

$$\frac{Rep^{[t_1-\epsilon,t_1]} - Rep^{[t_b,t_1]}}{|U^{[t_1-\epsilon,t_1]}|} > \vartheta$$

The number of users ($|U^{[t_1-\epsilon,t_1]}|$) is bounded by two factors: 1) communities cannot manage more than a maximum number of users by time unit considering their sizes (i.e. the number of web services populating the communities); and 2) in case of a malicious community, it is very unlikely that this community manages to collude with more than a certain number of users. This will prevent malicious communities from violating the feedback without being recognized by maximizing $|U^{[t_1-\epsilon,t_1]}|$. In that case, it is assumed that community C_i had a drastic reputation increase in the recent ϵ time. The value ϵ is set with respect to the controller agent's success in fake feedback detection. Interacting in the environment, Cg would update this value in the sense that the most efficient value is figured out. The detail algorithms on how to learn this value is out of scope of this chapter.

Fake positive Adjustment. Exceeding the threshold ϑ, Cg would figure out that a particular community is receiving consequent positives. Then Cg, in order to reload

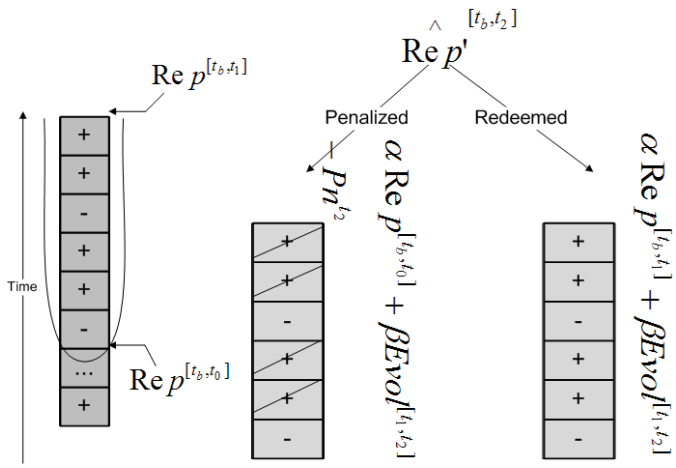


Figure 4.2: Fake positive correction cases

the previous and actual reputation level, would freeze the recent positive logs and notifies the corresponding community of such suspending. So, Cg would observe the upcoming behavior (in terms of satisfaction and responsiveness) of the community in order to match the actual efficiency with the suspended enhanced reputation level. During this period, the community is encouraged to behave in such a way that reflects the suspended enhanced reputation level. As it is shown in Figure 4.2, the community's feedback is recognized as suspicious at time t_1. Feedback from time t_0 are freezed to investigate the further behavior of the suspicious community C_i. At time t_2 controller agent Cg would decide whether to penalize community C_i or to redeem the freezed feedback. If the community shows the real improved performance, the suspended reputation level would be redeemed and considered for its reputation. But if the community fails to do so, the previous reputation level will be decreased by some applied penalties. In this case, the community would be in such a situation that either has to outperform its past in order to improve the enhanced reputation level, or would loose its current reputation, which is not wanted. Therefore, we form an incentive that communities would not risk their current reputation level and thus they do not by any

means (colluding with users or providers) provide fake positives in support of themselves. Let $Evol^{[t_1,t_2]}$ be the evolutionary reputation value for the community C_i that is measured by the Cg during specified time interval $[t_1, t_2]$ (investigation period). This value is computed in Equation 6, where δ is a small value such that the reputation is measurable within $[t - \delta, t]$.

$$Evol^{[t_1,t_2]} = \frac{\sum_{t=t_1+\delta}^{t_2} Rep^{[t-\delta,t]}}{t_2 - t_1} \tag{6}$$

Also, let Pn^t be the general penalty value that is assigned by Cg to C_i at a specific time t. Equation 7 computes the adjusted reputation level of C_i ($\widehat{Rep}^{[t_b,t_2]}$). This equation reflects the incentive we propose, so that CWSs in general would be able to analyze their further reputation adjustments upon fake action.

$$\widehat{Rep}^{[t_b,t_2]} = \begin{cases} \alpha Rep^{[t_b,t_1]} + \beta Evol^{[t_1,t_2]}, & \text{if redeemed;} \\ \alpha Rep^{[t_b,t_0]} + \beta Evol^{[t_1,t_2]} - Pn^{t_2} & \text{if penalyzed.} \end{cases} \tag{7}$$

where $\alpha + \beta = 1$.

As discussed before, Cg will decide to redeem the community C_i if the evolutionary value for the reputation is more than C_i's previous reputation value, i.e.: $Evol_{C_i}^{[t_1,t_2]} \geq Rep_{C_i}^{[t_b,t_0]}$. If Cg decides to redeem the community C_i, then the previous reputation value (from time t_b to investigation time at t_1) would be considered together with the evolutionary reputation value as a result of investigation during $[t_1, t_2]$. If Cg decides to penalize the community C_i, then the previous reputation is considered regardless of the improved reputation obtained in the period of $[t_0, t_1]$. In addition to the evolutionary reputation, a penalty Pn^{t_2} would also be assigned at time t_2.

False alarm detection. It is worth to discuss more about alternatives of Cg's fake positives recognition. Consider the two cases that Cg falsely, and truly recognizes the

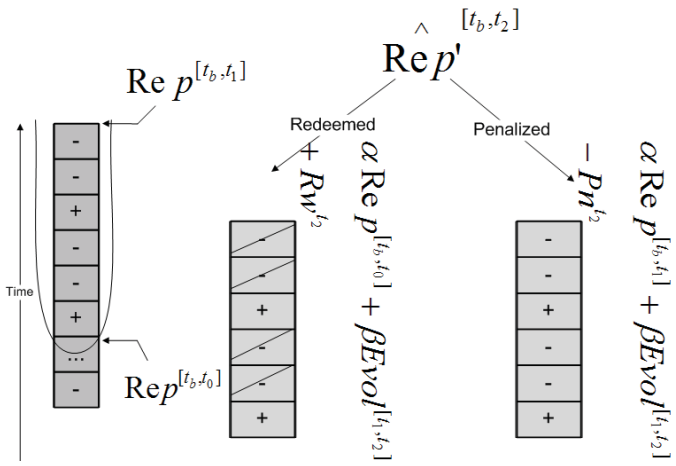


Figure 4.3: Fake negative correction cases

fake positives. In the former case, the positives are real, therefore, they reflect the actual performance of the community. Then even being suspended, the community can easily prove the quality level as it continues as before and basically would not loose anything. In the later case, the positives are fake, so the community needs to improve its actual quality level to prove suspended enhanced reputation level. If the community failed to fulfill such reputation, Cg would decrease its previous reputation level.

4.4.2 Fake Negative Correction

Similar to the fake positive case, there might be some fake negatives in order to decrease the reputation level of a particular community (see Figure 4.3). This could happen when a community or a set of communities would like to weaken a particular community (by dropping its reputation level) hoping not to compete with them. However, one unique case should not be excluded in which, a particular community would mal-behave and after certain number of providing services and obtaining negative feedback, claims that the feedback were fake and do not reflect its actual reputation level. To avoid such a situation, each community is responsible to recognize a change in its reputation level and consequently report the case to Cg. Upon received report, Cg

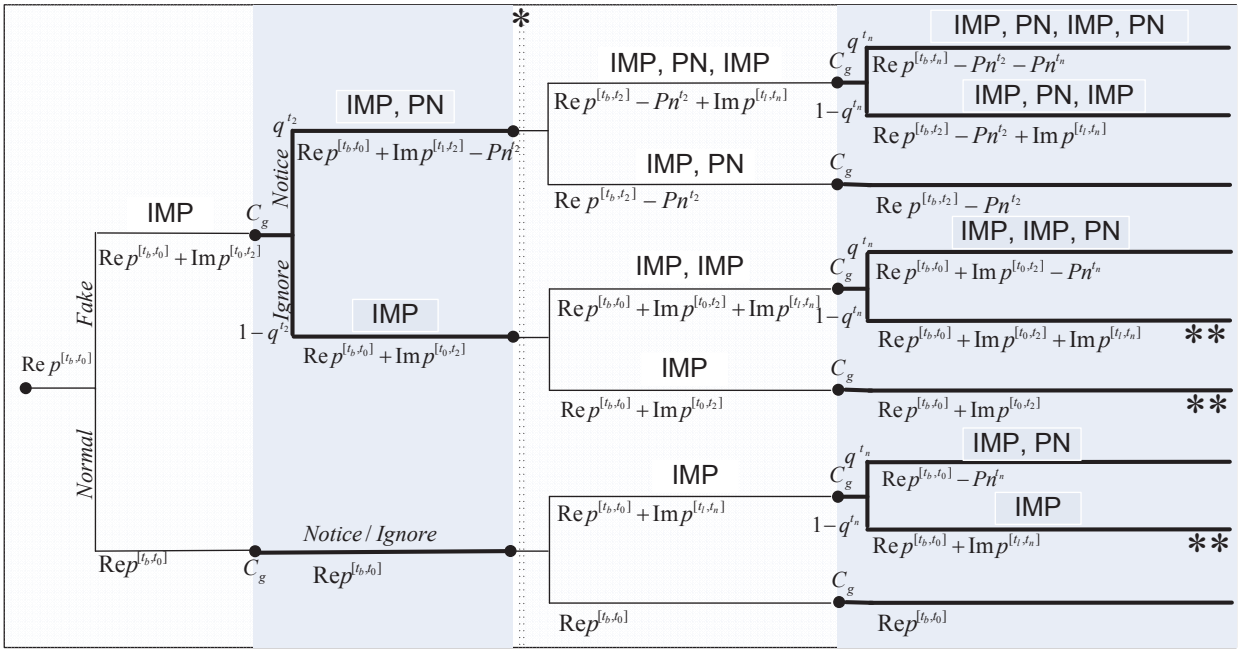
would decide whether the negative feedback were really as a result of the mal-behavior of the community or as a result of some other parties fake negatives. If Cg initiates the investigation at time t_1, after a period of evolutionary time, Cg would decide for the reputation adjustment at time t_2. In case of redeeming the community C_i that was suspected to have fake negative feedback, the negatives are discarded ($Rep^{[t_0,t_1]}$ is not considered), and a reward Rw^{t_2} is assigned at time t_2. The reason is to discourage the opponent communities not to cause a fake negative feedback for C_i and hope to degrade its reputation level. However, if after evolutionary investigation, Cg decides to penalize C_i, then the negative feedback are also considered (by considering $Rep^{[t_b,t_1]}$), and a penalty Pn^{t_2} is assigned to the community. Equation 8 computes the updated reputation value of the community C_i ($\widehat{Rep}^{[t_b,t_2]}$).

$$\widehat{Rep}^{[t_b,t_2]} = \begin{cases} \alpha Rep^{[t_b,t_0]} + \beta Evol^{[t_1,t_2]} + Rw^{t_2}, & \text{if redeemed;} \\ \alpha Rep^{[t_b,t_1]} + \beta Evol^{[t_1,t_2]} - Pn^{t_2} & \text{if penalyzed.} \end{cases} \tag{8}$$

There is also a case that a malicious community tries to mislead controller agent Cg with the fake feedback that it managed to provide for himself and tries to act better than usual in the evolutionary time to get the reward Rw^{t_2}. All such false detections reflect diverse situations in which Cg needs to recognize the source of submitted feedback (colluded users). For sake of simplicity, in this framework we do not talk about these cases and consider such cases of false detection out os scope.

4.4.3 Theoretical Analysis

In this section, we will discuss in details the updates of reputation level when a particular community C_i causes fake feedback that is eventually beneficiary for itself. To this end, we follow the steps over this reputation updates and elaborate Cg's actions on them. For simplicity reasons, we only analyze the case of self-positive feedback and generalize our discussion to fake negative feedback. We objectively assume that



Properties:
1) White area C_i's turn, gray area Cg's turn.
2) Upper branch reflects fake action for C_i, and Notice for Cg.
3) * denotes start of a general time period of $[t_l,t_m,t_n]$.
4) IMP reflects improvement in one's reputation. PN reflects the assigned penalty via Cg.
5) ** highlights the cases that faked reputation is more than actual reputation.
6) For simplicity, at each step improvements and penalties are declared.

Figure 4.4: The tree of backward induction reasoning

penalizing a community is relative to the reputation improvement that community had obtained. In this section, we use backward induction reasoning technique to show that CWSs loose interest in doing malicious acts that cause extra (fake) positives for themselves or extra (fake) negatives for some others.

To better analyze the decisions the communities could take, we calculate the expected reputation value of a particular community in the case that the community acts maliciously to provide fake positive feedback for itself and the case that the community acts as normal and performs its actual capabilities. By comparing the two expected values, the typical community C_i will decide either to act maliciously or as normal. As discussed earlier, this decision is made based on the probability that C_i estimates to

have a successful act. Being malicious, C_i always looks for the cases that could possibly cheat to increase its current reputation. Let q^t be the probability that the controller agent Cg notices the real intention of the community C_i and take actions with penalizing C_i at time t. We compute the expected reputation of C_i as a result of a malicious action in Equation 9 and as a result of normal action in Equation 10. In these equations, the expected value of the reputation for community C_i is measured under two assumptions. In the case that C_i has faked the feedback ($E(\widehat{Rep}^{[t_b,t_2]}|C_i\ faked)$), the community decides to fake at time t_0 (therefore, the reputation till t_0 is considered as normal), the biased feedback are recognized by Cg at time t_1, and the investigation is finalized at time t_2. To this end, by penalizing C_i, its previous reputation till t_0 is considered together with the investigation period $[t_1, t_2]$ with its penalty. If the controller agent Cg does not recognize C_i's malicious act, all the feedback are taken into account. In this analysis, we consider a very low possibility that Cg warns false negatives, which is the case that Cg falsely recognizes a malicious act. To this end, we assume that if the community C_i acts as normal, the reputation value would be measured as normal.

$$\begin{aligned} E(\widehat{Rep}^{[t_b,t_2]}|C_i\ faked) = \\ q^{t_2}(\alpha Rep^{[t_b,t_0]} + \beta Evol^{[t_1,t_2]} - Pn^{t_2}) \\ + (1 - q^{t_2})(\alpha Rep^{[t_b,t_1]} + \beta Evol^{[t_1,t_2]}) \end{aligned} \tag{9}$$

$$E(Rep_{C_i}^{[t_b,t_2]}|C_i\ notfaked) = Rep_{C_i}^{[t_b,t_2]} \tag{10}$$

Figure 4.4 is the tree representing the backward induction reasoning through actions of the community C_i and corresponding reactions made by the controller agent Cg in two steps. In this Figure, IMP refers to the fact that the community's reputation is getting improved thanks to fake positives the community has provided. We also refer in this Figure to PN as the state that the community's fake action is detected and

thus penalized by Cg. As it is illustrated, the community that provides fake positives, obtains an improvement, which could be followed by a penalty. Here we state that the probability of Cg's detection given the fact that C_i has faked before is high. Therefore, if C_i has been already penalized, it is so hard to retaliate and improve again. There is a slight chance that C_i fakes and Cg ignores, which comes with a very small probability. Thus, we compute the expected reputation level of both cases and compare them.

Definition 4.1 *Let $Imp^{[t_b,t_2]}$ be the difference between the adjusted reputation (in the case where the community is under investigation) and normal reputation (in the opposite case) within $[t_b, t_2]$, i.e:*

$$Imp^{[t_b,t_2]} = \begin{cases} \widehat{Rep}^{[t_b,t_2]} - Rep^{[t_b,t_0]}, & \textit{investigated by } Cg; \\ Rep^{[t_b,t_2]} - Rep^{[t_b,t_0]}, & \textit{otherwise.} \end{cases}$$

The following proposition gives the condition for the penalty to be used, so that the communities will not act maliciously.

Proposition 4.2 *If $Pn^{t_2} > \frac{1}{q^{t_2}} Imp^{[t_b,t_2]} - \alpha Rep^{[t_0,t_1]}$, then communities obtain less reputation value if they act maliciously and provide fake feedback for themselves.*

Proof: To prove the proposition, we should consider the condition true and prove that $E(\widehat{Rep}^{[t_b,t_2]}|C_i\ faked) < E(Rep^{[t_b,t_2]}|C_i\ Not\ faked)$. By simple calculation we get:

$$E(Rep^{[t_b,t_2]}|C_i\ Not\ faked) - E(\widehat{Rep}^{[t_b,t_2]}|C_i\ faked) =$$
$$Pn^{t_2} - \tfrac{1}{q^{t_2}} Imp^{[t_b,t_2]} + \alpha Rep^{[t_0,t_1]}$$

The obtained value is positive, so we are done.

In the previous proposition, we talked about the incentive that a rational community has to avoid fake feedback. Now we would like to discuss the general incentive of a

malicious act in multiple times to generalize the ultimate reputation adjustment of bad communities that in general prefer to cheat on the logging system. To this end, we extend our analysis by discussing about a particular community C_i that has previously made malicious act (for the first time action made at time t_{l1}, detection made at time t_{m1}, and decision made at time t_{n1}). In this analysis, we would like to investigate the community's further acts (made at general time t_l) in distracting the logging file and thus, its reputation treatment via the controller agent (detection at time t_m and decision at time t_n such that $t_n > t_m > t_l > t_{n1}$). Basically, as a result of the previous act, C_i could have been penalized (which means the community is less likely to act maliciously again) or have gained a reward (which means the community is very likely to act maliciously again). In the following, we study the penalty Pn^{t_n} that should be assigned to these types of communities to avoid their multiple malicious acts.

Assume that C_i has made its malicious act at time t_{l1}. For the performed action, there is a chance ($q^{t_{n1}}$) that the controller agent Cg noticed the act at time t_{n1} and thus, penalized the community by $Pn^{t_{n1}}$. We also consider the chance ($1 - q^{t_{n1}}$) that the controller agent ignores the act and thus, the community has obtained the improvement $Imp^{[t_{l1},t_{n1}]}$ through the feedback without any penalty from the controller agent. Considering the probabilities of different strategies that the controller agent may take, as we discussed earlier, there is a small chance that Cg ignores the malicious act. This basically means the probability of notice (for the first time) ($q^{t_{n1}}$) is normally high and that is because the sensitivity of the controller agent in investigating the list of feedback for each particular community. However, once recognized, the controller agent becomes more sensitive to the recognized community's further actions. Therefore, the probability of missing the second fake action is less than the first one and so on (($q^{t_{n2}} > q^{t_{n1}}$)). Generally speaking, the community would be more interested to continue its malicious behavior when it has never been recognized via Cg and thus penalized. However, there

is always a high possibility for this community to be recognized later (for the first time).

Considering the aforementioned cases, the expected reputation $E(Rep^{[t_b,t_n]})$ for a community that fakes the feedback again (for the second time or more) can be decomposed by the cases that Cg has previously (t_{nj}) noticed the community's malicious act ($Cg\ noticed|C_i\ faked$) with the probability $q^{t_{nj}}$ ($nj < n$) and Cg has previously ignored such action ($Cg\ ignored|C_i\ faked$) with the probability $1 - q^{t_{nj}}$. We study each case by analyzing the strategy that Cg has previously took in response to such fake action.

$$E(Rep^{[t_b,t_n]}|C_i\ fake\ again) =$$
$$(q^{t_{nj}})E(Rep^{[t_b,t_{lj}]}|Cg\ noticed)+$$
$$(1 - q^{t_{nj}})E(Rep^{[t_b,t_{lj}]}|Cg\ ignored)$$

Consider the first case that Cg notices the current fake behavior of C_i. We expand this case to the cases that Cg noticed C_i's previous act and the case that Cg ignored C_i's previous malicious act. This basically influences the control of Cg over the feedback of the community C_i since being recognized as malicious community.

$$E(Rep^{[t_b,t_n]}|Cg\ noticed) =$$
$$(q^{t_n})E(Rep^{[t_b,t_n]}|Cg\ noticed\ before)+$$
$$(1 - q^{t_n})E(Rep^{[t_b,t_n]}|Cg\ ignored\ before)$$

Basically the probability of notice for a community that has faked before is more than ordinary community without previous fake action. To this end, q^{t_n} is higher than $q^{t_{nj}}$ such that $q^{t_n} \times \alpha = q^{t_{nj}}$. The value α is a generic value ($0 < \alpha < 1$), but to be consistent we always use this value in order to apply the degradations.

Considering the case that Cg ignored the current fake behavior of the C_i, we expand this case to the case that Cg noticed C_i's previous malicious act and the case that Cg ignored C_i's previous malicious act. For simplicity, here we assume $q'^{t_n} = 1 - q^{t_n}$.

This means that if the previous fake action is recognized, the current fake action would be recognized as well with the probability of q^{t_n}. Likewise, if the previous fake action is ignored, the current fake action is made with the probability of q^{t_n}.

$$
\begin{aligned}
&E(Rep^{[t_b,t_n]}|Cg\ ignored) = \\
&(q'^{t_n})E(Rep^{[t_b,t_n]}|Cg\ noticed\ before)+ \\
&(1-q'^{t_n})E(Rep^{[t_b,t_n]}|Cg\ ignored\ before)
\end{aligned}
$$

The value q'^{t_n} would be a very small value in the sense that if Cg noticed the previous act of C_i, now the possibility of ignore would be very small. In general, the controller agent would become very sensitive to the acts of malicious communities. Considering the updates made by Cg over the reputation values of communities, the following proposition holds.

Proposition 4.3 *If communities fake again, they make a drastic degradation in their reputation value.*

Proof: Given the fact that Cg noticed previous fake action of C_i, it would be more restrictive for C_i's further performance, therefore, the probability of noticing the new fake action is higher than before ($q^{t_n} > q^{t_{nj}}$). In this case Cg increases the checking accuracy for such community and we defined this improvement by the factor of $1+\alpha$, which is multiplied to the previous notice probability value. Consequently, we rewrite the expected value as following. In Equation 11, the first line represents the case that fake action has been noticed before and now (so there is two penalties applied and no reward). Second line represents the case that fake action is noticed now but has been ignored before (so there is a current penalty but previous reward). Third line represents the case that fake action is ignored now but has been recognized before (so there is current rewards but previous penalty). Last line represents the case that fake

action been ignored in both previous and current time (so there are just rewards and no penalties).

$$
\begin{aligned}
&E(Rep^{[t_b,t_n]}|C_i\ faked\ again) = \\
&q^{t_n}(q^{t_{nj}})(Rep^{[t_b,t_{lj}]} - Pn^{t_{nj}} - Pn^{t_n}) \\
&+q^{t_n}(1-q^{t_{nj}})(Rep^{[t_b,t_{lj}]} - Pn^{t_{nj}} + Imp^{[t_l,t_n]}) \\
&+(1-q^{t_n})(q^{t_{nj}})(Rep^{[t_b,t_{lj}]} - Pn^{t_{nj}} + Imp^{[t_{lj},t_{nj}]}) \\
&+(1-q^{t_n})(1-q^{t_{nj}})(Rep^{[t_b,t_{lj}]} + Imp^{[t_{lj},t_{nj}]} + Imp^{[t_l,t_n]})
\end{aligned}
\tag{11}
$$

Following the ideology that the expected value of faking again should be (strictly) less than not faking, we simplify the obtained value in Equation 11 to the following:

$$
\begin{aligned}
&E(Rep^{[t_b,t_n]}|C_i\ fake\ again) < \\
&E(Rep^{[t_b,t_n]}|C_i\ not\ fake\ again) \Rightarrow \tfrac{1-q^{t_n}}{q^{t_n}} Imp^{[t_l,t_n]} < Pn^{t_n}
\end{aligned}
\tag{12}
$$

Generalizing the case $\frac{1-q^{t_n}}{q^{t_n}} Imp^{[t_l,t_n]} < Pn^{t_n}$ *to be valid in all* t_n*, it is shown that the required amount for the penalty for time* t_n *is less than the required amount for any previous time. This clarifies the incentive for faking again is less than the incentive for the first fake.*

$$
\begin{aligned}
&Pn^{t_n} < Pn^{t_{n'}} \\
&n' < n
\end{aligned}
\tag{13}
$$

Therefore, the probability of faking again is decreasing over time, so we are done.

4.5 Experimental Results

In this section, we describe the implementation of a proof of concept prototype. In the implemented prototype, CWSs are composed of distributed web services ($Java^{©TM}$ agents). The agent reasoning capabilities are implemented as Java modules. The

Table 4.1: Simulation summarization over the obtained measurements.

CWS Type	WS Density	WS Type	WS QoS
Ordinary	[25.0%, 35.0%]	Good	[0.5, 1.0]
Faker	[25.0%, 35.0%]	Bad	[0.0, 0.5]
Intermittent	[25.0%, 35.0%]	Fickle	[0.2, 0.8]

testbed environment is populated with two agent types: (1) agent-based web services that are gathered in a community (we assume only one type of service is provided and therefore consumed); and (2) user agents that are seeking for the best service provided by a web service. In general, the simulation consists of a series of empirical experiments tailored to show the adjustment of the CWS's reputation level. Table 4.1 represents three types of CWSs we consider in our simulation: ordinary, faker and intermittent. Ordinary community acts normal and reveals what it has, the faker community is the one that provides fake feedback in support of itself, and the intermittent community is the one that alternatively changes its strategies over the time. As it is shown in Table 4.1, the QoS value is divided into three ranges.

In each RUN, a number of users are selected to search for the best service. Strictly speaking, users are only directed to ask CWSs for a service and thus, user would not find out about the web service that is assigned by the master of the community. In order to find the best community, the requesting user would evaluate the CWSs regarding their reputation level. Some times, the users are in contact with some communities that are very good for the user, so the users re-select them. The selected community might be overloaded and consequently rejects the user requests. If the user agent is rejected from the best selected community, it would ask the second best community in terms of reputation level (and so on). After getting a response from a community, the user agent would provide a feedback relative to the quality of the obtained service and the community responsiveness. The feedback are logged in the logging mechanism that

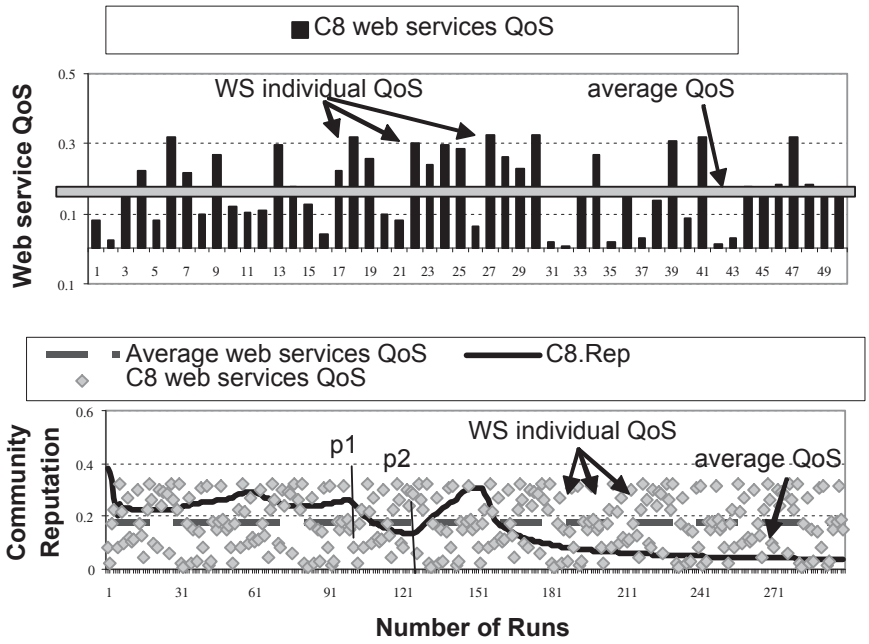


Figure 4.5: Communities overall quality of service vs. the number of simulation RUNs

is supervised by Cg. The accumulated feedback would affect the reputation level of communities. In other words, the communities would loose their users if they receive negative feedback, which will cause drop in their reputation level.

Considering the general incentive of CWSs to attract maximum possible users, communities in general, compete to increase their reputation level. Cheating on reputation level is done by colluding with a user (or a small group of users) to provide consecutive positive feedback in support of the malicious (faker) community. In the empirical experiment, we are interested observing the over-RUN reputation level of different types of communities and how fast and efficient the adjustment is performed by Cg. Figure 4.5 illustrates the plot of reputation level for a faker community $C8$. The upper plot represents the individual QoS for the community's assigned web services. In this plot the gray line defines the average QoS for the web services. The most prominent feature of the plot is the comparison of the reputation level with the average of the community web services QoS. The average value is assumed to be the actual QoS

for the community and thus, community's reputation level. In general, there would be convergence to such value if the community is acting in an ordinary manner (for $C8$ is 0.173). The lower plot illustrates the reputation level of this community over the elapsing RUNs. Here we notify that the master of a community is responsible to assign the web services to the user requests. To this end, normally the high quality web services are assigned first until they become unavailable, which forces the master agent to assign other lower quality web services. Thus starting the RUNs, $C8$ gains reputation value (up to 0.313), which is better than its individual average quality of service. In Figure 4.5 the peek $P1$ defines the RUN in which the community $C8$ is out of high quality web services. After passing this point, the reputation level of this community is decreased.

Figure 4.6 illustrates community $C8$ reputation level in comparison with an ordinary community $C6$. $C8$ at point $P3$ decides to provide fake positive feedback for itself to increase self reputation level. For the interval of 30 RUNs, this community gains higher reputation level up to the point $P4$. The controller agent Cg, periodically verifies the feedback logs, in order to recognize the malicious actions. At $P4$ the controller agent Cg notices the malicious act of $C8$ and freezes the obtained feedback for investigation. Peek $P2$ is the point in which the community $C8$ is penalized in its reputation level. After $P2$, a drastic decrease in reputation value is seen, which goes underneath $C8$'s average quality of service (up to 0.112). There is also a continuing but slower increase in the reputation of the faker community $C8$ that persists long after the first fake action recognition. There are then strong restriction effects, which cause loosing the users by the faker communities. However, there is also an ongoing effect of social influence, which leads users to have doubt in communities that have drastic decrease in their reputation level.

We continue our discussion by analyzing some parameters related to the controller

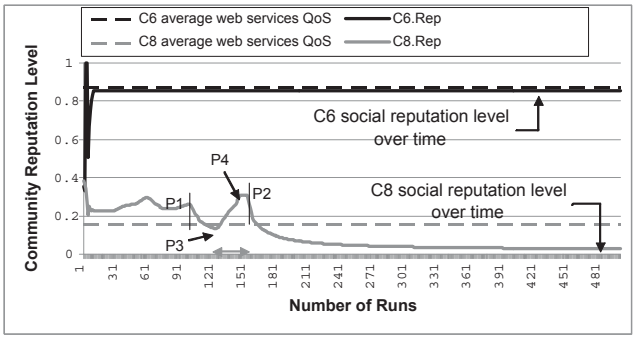


Figure 4.6: Communities overall quality of service vs. the number of simulation RUNs

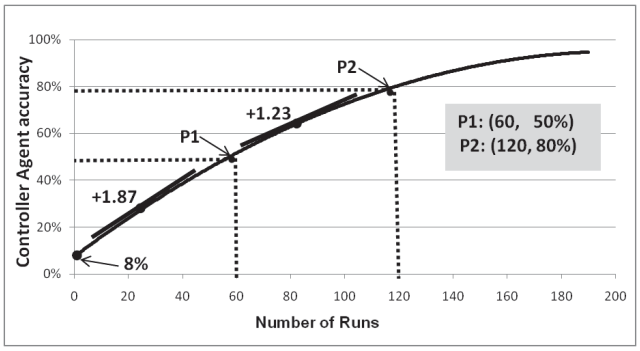


Figure 4.7: Controller agent Cg's accuracy in detection vs. the number of simulation RUNs

agent's performance and accuracy. One of the main factors in such a system is the accuracy of the controller agent in fake detection. The controller agent is supposed to investigate the feedback and recognize the malicious acts while the requesting users provide their rates. However, there are two possibilities for Cg to fail to accurately

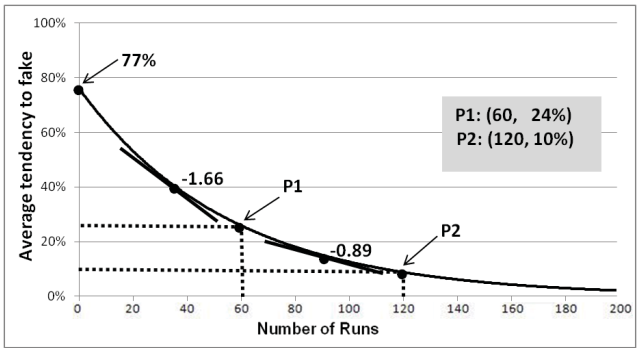


Figure 4.8: Communities' tendency to fake vs. the number of simulation RUNs

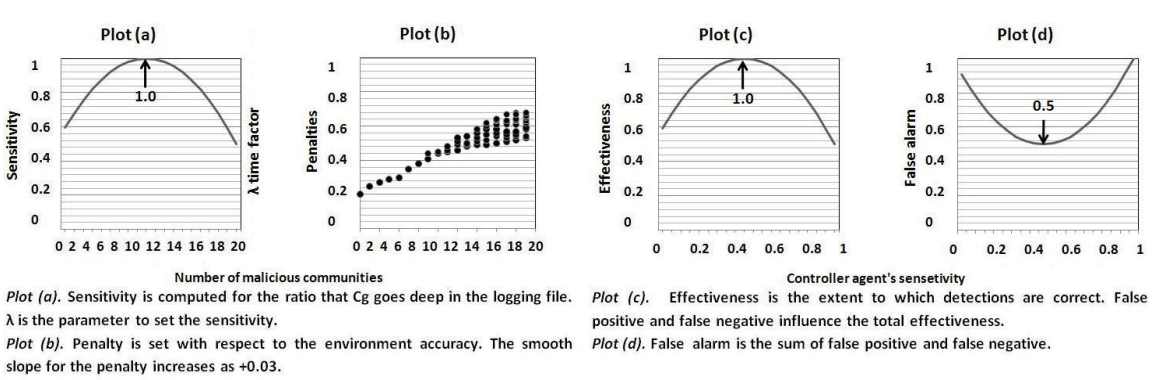


Figure 4.9: Controller agent's characteristic analysis

detect such actions. The false detections are detecting a non-fake action as fake, and ignoring a fake action as non-fake. The former case is called false positive (or α-error in statistics), which is rejecting the null-hypothesis when it is true. The later case is called false negative (or β-error), which is accepting the null-hypothesis when it is actually false. The false positive is the case that the controller agent would ignore a malicious act and thus, would not investigate it more closely. Since the controller agent is not re-acting to the initially detected action, there is a chance to recover the initial false alarm. Over the further investigation, the false negative (initially warned by Cg) is most likely corrected once the investigation is done, but the other cases, which have been ignored are not recognized as there is no further investigation over the detection.

To this end, one of the main objectives is to enhance the efficiency of the controller agent to decrease the false alarm ratio and strength the logging feedback crawling algorithm. Figure 4.7 shows the controller agent's accuracy over the elapsing RUNs while the recognized communities are penalized and thus, discouraged to redo the fake actions. As shown in this figure, the controller agent is relatively less accurate during the initial RUNs. Basically, detection weakness would highly encourage the faker and intermittent communities to perform fake actions. Mostly as a result of the reward that they obtain without the penalty. Basically, the accuracy of Cg is increased while

Cg acts successfully in detecting and thus, penalizing faker communities. Cg would act better over the Runs since previously detected communities are investigated more carefully and thus, the chance of failing to detect is decreasing.

In Figure 4.8 we discuss this issue as we observe the tendency of the communities to provide fake feedback in support of themselves. In this figure, the vertical axes plot the average percentage of the intermittent communities that might be encouraged to fake and the horizontal axes plot the RUNs, which reflect the elapse of time. In this figure, the average tendency to fake is decreasing as the number of intermittent agents that are penalized are increasing.

We take a narrower analysis on the characteristics of the controller agent Cg and their impacts that eventually influence the incentive of different communities to act maliciously. To this end, we study the aforementioned issues towards the network condensity and the extent to which the controller agent is crawling the feedback. In the former study, the idea is to observe how dealing with different malicious communities make the controller agent sensitive to get suspicious while crawling the feedback. Basically, the controller agent sets the threshold ϑ in Section 4.4.1 by observing the number of malicious communities in the environment. This means the controller agent tries to get more though when the number of malicious communities is increasing (see $Plot\ (a)$). However, this harsh manner could not be kept on since Cg cannot keep tracking all communities at the same time. On the other hand, by getting suspicious for any community, the false positive ratio would be going up, which reflects the low efficiency of Cg in terms of detection performance. Following the idea that Cg tries to avoid the increase of the malicious communities, we observe that this agent increases the average penalty value assigned to malicious communities while their number is increasing. $Plot\ (b)$ assigns a dot point to each community that gets penalized. The dot points are getting more condense, which shows their high number.

In the second part of the Figure 4.9, we study the efficiency of the controller agent versus its sensitivity. Since we analyzed the threshold that is set to indicate Cg's sensitivity, here we study how well Cg can act with different thresholds. $Plot\ (c)$ sketches a graph that shows a parabola for the effectiveness of Cg. In this graph, there is a tradeoff between the false positive and false negative errors. At a low sensitivity period, there are high number of false negatives. This basically encourages the malicious communities to highly redo their malicious acts as they distract in the logging file and increase their reputation and do not get penalized afterwards. To this end, the observed slope for the effectiveness is relatively small. There is a maximum point for the effectiveness, but this is not always true and may change depending on the environment and surrounded communities. Therefore, we cannot finalize the controller agent's efficiency to a specific value. $Plot\ (d)$ is depicting the same problem from another point of view. Indeed, in this plot we study the false alarm in spite of effectiveness. The false alarm is computed as the sum of false positive and false negative ratios. In this plot, the total false detections is minimized once the controller agent reaches its maximum efficiency. Likewise, the decreasing slope is so slow.

Chapter 5

Sound Reputation Mechanism

5.1 Background

In the previous chapter, we proposed a reputation-based framework that is used to compute agents' reliability in interactive multi-agent systems according to public opinion. The computed reputation value is accurate upon calculation but might lose its importance once the system has undergone dynamic changes of agents' strategies. A sound reputation mechanism is the one that maintains its accuracy over time and discards inaccurate information. Following this claim, in this chapter we continue our analysis of the proposed reputation mechanism and concentrate on its soundness in order to minimize the malicious strategies that might be adopted by selfish agents. A part of this chapter is dedicated to preliminaries that refer to the concepts that are already stated in the reputation model proposed in the previous chapter, but there are some clear differences that we explore in the following. The previous reputation model was based on the user agents and their points of views. The main concentration was on the aggregation of relevant parameters and maintenance of accurate feedback pool. We computed reputation of communities of web services (referred to Rep_i) using the aggregated feedback as well as other relevant parameters. In that case, the reputation parameter was computed within time intervals and was the value from consumer agents' points of views. But in this chapter we have a different point of view regarding this parameter and we compute it considering some new parameters. Moreover, we discard the concept of

interval and mainly focus on the way that this parameter is computed and updated. Considering these changes, we refer to the reputation by R_i in the rest of this Thesis. This parameter (R_i) represents the reputation associated to an entity i that could be a web service agent or a community of web services. The reputation model proposed in this chapter mainly concentrates on the soundness issue and how a reputation system could constrain its accuracy while the interacting agents are free to choose any sort of acting strategies. The reputation mechanism is aimed at imposing some incentives in the multi-agent environment so that essentially truthful strategies are adopted by majority of agents.

Regarding the related work, there are a number of reputation models that all aim at maintaining accurate systems [35, 54, 89]. But the dynamism of the environment is not well-studied yet. In this proposed model, we explore game-theoretical analysis of the expected payoffs obtained via different acting strategies [23, 24]. In general, game theory is a method of investigating different strategic decision making procedures that intelligent agents maintain. More formally, it is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers" [59]. We use game theory to focus on agents' preferences and the formation of their beliefs to find and develop tactics that impose our idealistic state (truthful environment) as a common goal between agents. We provide incentives to rational agents to investigate their risky alternative actions and estimate their expected outcomes. We investigate the specific thresholds by which the controller system can approach sound reputation system. Furthermore, we have implemented multi-agent environment systems with flexible parameters to observe impacts of the adopted strategies of the controller agent.

5.2 Overview and Motivation

Web services are deployed to maintain continuous interactions between loosely coupled applications. Abstracting web services using knowledge-empowered agents will benefit them from flexible and intelligent interactions that those agents are able to manage [28, 93]. However, because agents are autonomous and selfish, a key issue in agent-based environments is reputation, which is a significant factor that regulates the process of service selection. As discussed in Chapter 2, during recent years, there have been extensive work addressing the reputation in multi-agent and service environments [41], [52], [57], [35, 37]. Many of the proposed models are based on data collected from different sources that are considered reliable. However, this might not be the case in many concrete situations.

There is a different point of view in addressing the reputation mechanism, which is maintaining an incentive-based sound reputation mechanism [102, 103]. In this perspective, the ideal case is the situation in which rational agents have incentives to act such that ultimately the whole environment turns into a truthful network of agents. Maintaining this mechanism requirers designing a reputation framework with some defined characteristics that establish incentives and penalties along the direction towards a sound reputation mechanism. The concept of sound reputation assessment is being considered in very few attempts. The reputation model we propose in this chapter aims to advance the state-of-the-art by addressing this open issue (promoting truthful actions).

The general idea of collusion-resistent reputation mechanism is inspired by a previous framework we proposed in [50], in which we developed a game-theoretic analysis to maintain accurate reputation assessment mechanism for agent-based web service systems. In this reputation assessment framework, web services are ranked using users' feedback posted with respect to the quality and satisfaction of the received service. The

goal is to investigate the payoffs obtained through different situations and propose solutions that allow building collusion-resistent reputation mechanisms. In this chapter, we extend this framework by expanding the reputation management, considering more collusion scenarios, and providing more theoretical and simulation results and analysis. Moreover, we discuss in detail the system implementation and simulation environment. More details regarding the contributions of this proposed model are provided in the following subsection.

Contributions. In this model, we consider agent-based web services and address the aforementioned problems by providing accurate reputation assessment in open environments in which web services are selfish and utility maximizers. The reputation is accurately assessed mainly as a result of incentives provided to participating agents in order to act truthfully and avoid malicious actions. We aim to advance the-state-of-the-art by analyzing the system's parameters using game theory. We investigate the incentives to cheat that malicious web services can have and incentives to act truthfully while being aware of the possible penalties assigned by the *controller agent* (see Chapter 4 for the definition of this agent). In fact, we theoretically and empirically analyze the obtained payoffs according to the agent's followed strategy (i.e. acting truthfully or maliciously). In our simulations, we discuss the obtained results that enable us to elaborate on the outcome of different strategies that participants (or players) might choose. We conclude with incentives for web services to act truthfully and identify the state that is socially acceptable for all the participants.

5.3 Preliminaries

In Section 4.2 (Chapter 4), we pointed out the preliminaries related to the multi-agent system hosting web services agent as service providers and service consumer agents.

In this Section, we refer to these entities and highlight the most important concepts that are used in this model.

Service consumers are intelligent agents that continuously seek for the services provided by some other agents. Each service consumer agent is equipped with a purchase mechanism that facilitates its request initiation process. Moreover, this mechanism analyzes the received quality of service (QoS) and generates the corresponding feedback. Each service consumer c holds an acceptable quality threshold QT_c that is compared against the received QoS to decide about posting positive or negative feedback. Service consumer agents rationally follow their predefined goal, which is obtaining the most satisfactory QoS over time. However, some of them could be encouraged by some web services to temporarily support them by reporting false feedback, which could be temporarily compatible with the goals of these service consumer agents. This issue will be discussed in details later in this chapter.

Web services are agent-based services engaged in answering the service consumers' requests. As mentioned before, web services might initiate some collusion with consumers that might be beneficial for both parties. Each web service agent i is equipped with a selling mechanism that enables the agent to approach its predefined goal. This goal is to have a maximum reputation (which results in maximum market share). The reputation R_i of the agent i is a value, which is computed as a result of feedback aggregation kept in the feedback file, which is supervised by the controller agent (both the feedback file and controller agent are explained later as preliminaries). Each web service agent holds parameters of *QoS* Q_i, and market share M_i that are used by the selling mechanism to reach the predefined goal. The market share M_i is a metric inspired by the *inDemand* metric represented in the previous chapter. In fact, here we use M_i instead if *inDemand* because market share is partially declared by the web service agent and is more consistent with the quality and reputation computations.

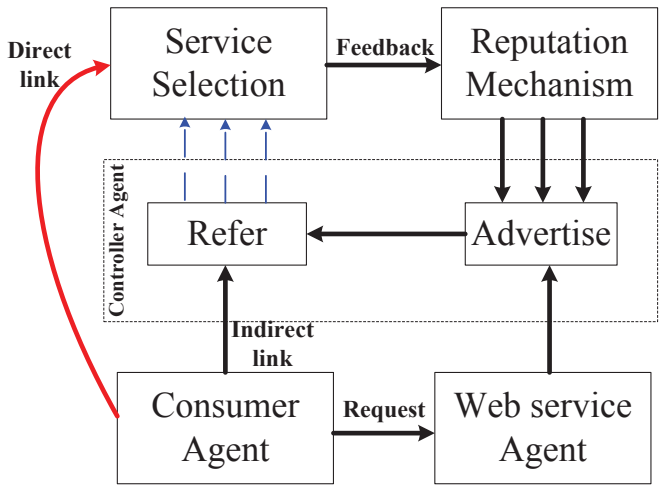


Figure 5.1: Architecture of the proposed framework

Feedback file is used in the proposed system to gather the submitted feedback from the service consumers. Consumers' feedback are aggregated to reflect the total credibility of web services. The feedback file is required to be supervised against malicious actions maintained by some selfish agents in the environment (selfish consumer agents and web services). Malicious actions mean violating the feedback file by posting some false feedback, which results in falsely increasing the reputation of some web services.

As discussed in Chapter 4, controller agent Cg is the assigned agent that takes the feedback file under surveillance. Cg is responsible of removing the false feedback that support particular web services. Cg is equipped with an investigation mechanism that enables this agent to investigate the recent feedback aggregated in the feedback file and recognize the faked ones by investigating further actions of the benefitted web service. In general, Cg might fail to accurately detect the fake feedback (false negative error) or similarly might recognize truthful feedback as fake (false positive error). Therefore, Cg holds a parameter regarding its accuracy A_{Cg}.

Figure 5.1 illustrates the links among the different entities in the proposed framework. Consumer agents take the initiative by looking for services using a service selection mechanism. These agents might contact previously known web services (direct

link in the figure) or refer to the controller agent to get updated with the most recent reputation ranking of the web services (indirect link and discontinue arrows in the figure). Once the web service is selected (i.e. the request is sent) and the corresponding service is provided, the consumer agent posts a feedback to the feedback file through the service selection mechanism. The controller agent updates the reputation ranking by aggregating the accumulated feedback. In this process, active web services would ask the controller agent for advertisement, which means they require to be considered in the reputation rankings provided to the consumer agents.

5.4 Reputation Mechanism

To maximize the reputation accuracy, we advance the reputation mechanism proposed in Chapter 4 by computing the reputation parameters in a different way and proposing some new parameters. We therefore, maintain a sound reputation mechanism. This is a mechanism that enables the service consumers to evaluate the credibility of the web services they want to invoke. In this system, Cg updates its surveillance algorithm and web services learn from their surrounding environment to make good decisions. The main result of this model is that over time, agent-based web services will get encouraged to act truthfully and discouraged to increase self reputation level with fake feedback. In the assessment process, there are key factors that we need to measure from the feedback. These factors, which reflect the health of a typical web service i are: quality (Q_i), and market share (M_i).

In the rest of this part, we explain each factor, then, we formalize the reputation of a typical web service as aggregation of these factors.

5.4.1 Reputation Parameters

Quality Q_i is used to measure the mean rate that is given to the web service i representing its quality in handling the users' requests in a timely fashion. Q_i is computed by collecting all the rates given to the web service to be evaluated. For simplicity reasons, but without affecting the main results of this model, we consider discrete feedback having the form $(+)$ for positive and $(-)$ for negative feedback. Let $\mathcal{P}_i$ be the set of positive feedback a web service i has received and $\mathcal{T}_i$ be the set of all the feedback i has received since published in the web. Thus, the acceptance factor would be simply computed in Equation 1.

$$Q_i = \frac{|\mathcal{P}_i|}{|\mathcal{T}_i|} \tag{1}$$

where $|\mathcal{P}_i|$ and $|\mathcal{T}_i|$ are the cardinality of $\mathcal{P}_i$ and $\mathcal{T}_i$ respectively.

Time Discount. In the trivial way of calculating Q_i in Equation 1, only the number of positive feedback is compared with the total number of feedback. This calculation is not highly effective when the environment is equipped with selfish agents that dynamically change their behaviors. We need then to consider the interactions history in a more effective way by giving more importance to the recent information. This can be done using a timely relevance function. In this model, we consider the following function similar to the one used in [31] and [50]: $e^{-\lambda \Delta t_k}$, where Δt_k is the time difference between the current time t and feedback k submission time t_k and λ ($\lambda \in [0, 1]$) is the recency scaling factor (i.e. scaling time values). Therefore, $e^{-\lambda \Delta t_k}$ is a weighted feedback. Consequently, the quality factor Q_i of web service i can be measured as shown in Equation 2.

$$Q_i = \frac{\int_{k\in\mathcal{P}_i} e^{-\lambda\Delta t_k} dt_k}{\int_{k\in\mathcal{T}_i} e^{-\lambda\Delta t_k} dt_k} = \frac{\int_{k\in\mathcal{P}_i} e^{-\lambda(t-t_k)} dt_k}{\int_{k\in\mathcal{T}_i} e^{-\lambda(t-t_k)} dt_k}$$

$$\Rightarrow Q_i = \frac{\frac{1}{\lambda} e^{-\lambda t} e^{\lambda t_k}\big|_{k\in\mathcal{P}_i}}{\frac{1}{\lambda} e^{-\lambda t} e^{\lambda t_k}\big|_{k\in\mathcal{T}_i}} \tag{2}$$

We notice that:

$$\lim_{|\mathcal{P}_i|\to\infty} \int_{k\in\mathcal{P}_i} e^{-\lambda\Delta t_k} dt_k = \int_0^{\infty} e^{-\lambda\Delta t_k} dt_k = \frac{1}{\lambda}$$

Consequently:

$$\lim_{\substack{|\mathcal{P}_i|\to\infty \\ |\mathcal{T}_i|\to\infty}} Q_i = \frac{\frac{1}{\lambda}}{\frac{1}{\lambda}} = 1$$

Intuitively, this means when the number of (positive) feedback is huge, the quality converges towards 1, which reflects the popularity of the concerned web service.

Market Share M_i is a parameter that indicates the extent to which the web service is active in the providers' network. This basically affects the popularity of the web service in the sense that the high service load together with high provided quality bring higher number of consumers (as a successful web service). We call this property the popularity property. In the proposed reputation mechanism, a successful web service is the one that receives high number of positive feedbacks while maintaining a good response to its consumers, which reflects its high request number. Equation 3 defines the market share for the web service i, which satisfies the popularity property. In this equation, the numerator represents the total feedback received for i, whereas the denominator is the integrated value for all recorded feedback ($\mathcal{G}$) for all active web

services controlled by Cg. As in Equation 2, the time discount is also considered.

$$M_i = \frac{\int_{k \in \mathcal{T}_i} e^{-\lambda \Delta t_k} dt_k}{\int_{k \in \mathcal{G}} e^{-\lambda \Delta t_k} dt_k} = \frac{\int_{k \in \mathcal{T}_i} e^{-\lambda (t - t_k)} dt_k}{\int_{k \in \mathcal{G}} e^{-\lambda (t - t_k)} dt_k}$$

$$\Rightarrow M_i = \frac{1}{\frac{1}{\lambda} e^{-\lambda t} e^{\lambda t_k} \big|_{k \in \mathcal{G} - \mathcal{T}_i}} \tag{3}$$

where $\mathcal{G} - \mathcal{T}_i \neq \emptyset$

5.4.2 Reputation Assessment

Taking the aforementioned parameters into account, we propose the estimated total reputation for the web service i that is crucial for its selection process and overall survival in the environment. First, we weight each parameter with a coefficient ($\beta_1 + \beta_2 = 1$). The value of each coefficient reflects the importance of the associated parameter. Therefore, we obtain the estimated reputation value r_i regarding the web service i in Equation 4. The reputation value r_i is only deduced from the feedback posted on the feedback file. However, at some point this value might not be the one that is publicly announced to the service consumers. These agents refer to the controller agent for the most accurate information regarding web services' reputation value and use the obtained value as a measure of reliability.

$$r_i = \beta_1 Q_i + \beta_2 M_i \tag{4}$$

In the proposed reputation mechanism, Cg is dedicated to manage the reputation assessment and make it sound. Therefore, on top of the rates that a web service i receives from collecting the consumers' feedback (r_i), Cg is eligible to offer a rate

reflecting its own point of view regarding the web service's reputation. The rate (C_i) that is given by Cg affects the web service i's total reputation. If C_i is so low (lower than r_i), that means the web service i has a bad-reputed history that might encourage users to avoid him. If the rate is relatively high (higher than r_i), the consumers rely more on what they have evaluated from the files. Equation 5 gives the formula of computing the total reputation R_i, which is defined so that it satisfies the conservative property. Such a property consists in giving higher weight to the lowest feedback. This is achieved because if $r_i > C_i$, then $\gamma_2 > \gamma_1$ where γ_1 is the weight of r_i and γ_2 is the weight of C_i.

$$R_i = \gamma_1 r_i + \gamma_2 C_i \text{ such that: } \begin{cases} \gamma_2 - \gamma_1 = & r_i - C_i \\ \gamma_1 + \gamma_2 = & 1 \end{cases} \tag{5}$$

5.5 Reputation Alteration

5.5.1 Collusion (Web Service Perspective)

In an open environment populated with agents who are aimed to achieve their predefined goals, some agents may choose strategies that only benefit some of them and in general are not good strategies for the whole system. In a multi-agent system of web services and service consumers, selfish web services might desire to increase self reputation to a level that have not been ranked to. The faked reputation level would temporarily bring extra service requests towards the malicious web service. However, the goal of the malicious web service is to keep the faked reputation as much as possible. A web service would collude with some consumer agents to provide continuous positive feedback supporting him. These consumer agents have to be encouraged to collaborate in the collusion by obtaining some privileges, such as low service fee, outstanding QoS, etc.

To discuss the collusion concept in more details, consider the web service i, which aims at increasing its quality report Q_i and market share M_i. In a collusion process, the malicious web service i faces a major risk reflected by the rate C_i submitted by the controller agent in the sense that if the malicious action is being detected, C_i would be fairly small reflecting bad history of the web service. The submitted rate via the controller agent affects the reputation value of the web service to some certain extent. In case of acting truthfully, the web service would obtain a better reputation compared to the case where its fake reputation is being recognized and thus, a low rate is submitted. To this end, a malicious web service, that is aiming to increase self reputation level, has a main challenge, which is the decision of acting maliciously. This means that even though the web service is capable of colluding with some consumers, there might be some reasons that prevent the agent from initiating such an action. Thus, to account for the web service's willingness to act maliciously, we introduce a willingness parameter w_i. In this case, the expected reputation values of acting 1) truthfully ($Exp(R_i|Truth)$) and 2) maliciously ($Exp(R_i|Mal)$) should be compared. A web service is willing to act maliciously when the expected reputation value of colluding is more than the one of acting truthfully. The parameter w_i is set as follows:

$$w_i = 1 \quad if \quad Exp(R_i|Mal) > Exp(R_i|Truth)$$

$$w_i = 0 \quad if \quad Exp(R_i|Mal) \leq Exp(R_i|Truth)$$

The expected values of reputation in different cases are computed in Equations 6 and 7. In Equation 6, q and $1 - q$ are respectively probabilities of being detected and ignored by the controller agent. The parameter $\overline{r_i}$ is the altered reputation as a result of collusion and the parameter $\overline{C_i}$ is the rate the controller uses if the collusion is detected. The value of $\overline{r_i}$ is greater than r_i thanks to the submitted faked feedback

by the colluding consumers. Likewise, the value of $\overline{C_i}$ is less than C_i as long as the controller agent would penalize more in case of collusion being detected.

$$Exp(R_i|Mal) = q(\gamma_1\overline{r_i} + \gamma_2\overline{C_i}) + (1-q)(\gamma_1\overline{r_i} + \gamma_2 C_i)$$

$$\Rightarrow Exp(R_i|Mal) = \gamma_1\overline{r_i} + \gamma_2(q\overline{C_i} + (1-q)C_i) \quad (6)$$

$$Exp(R_i|Truth) = \gamma_1 r_i + \gamma_2 C_i \quad (7)$$

In general, a normal web service that is acting truthfully, expects the actual reputation level when there is nothing wrong in the feedback file. Later in this section, we also consider the false positive cases where the truthful action also gets penalized and thus, the expected reputation rank should be updated. The malicious web service also has some other challenges, which are beyond the scope of this model: 1) when to act maliciously; 2) who to collude with; and 3) how many fake feedback to provide. To be focussed, in this model we only consider the malicious actions consisting of providing positive feedback. The fact of providing negative feedback, for example a web service can (indirectly) provide continuous negative feedback to a concurrent web service, is also important to be considered in future work.

5.5.2 Collusion Scenario

The collusion scenario could be initiated by either the consumer agent or web service agent. In this reputation model, we assume that this procedure is initiated by the malicious web service that is already willing to collude. This means that for the web service, the expected reputation rank with respect to collusion is more than the one of following a truthful action. Before discussing the collusion scenario, we also need to provide some insights regarding the consumer agent. In the proposed framework,

the consumer agents are aimed at obtaining the best service quality and therefore, they need to seek for the best reputed web service. In order to find the best web service, the consumer agent is required to refer to the controller agent to obtain the most updated web services' ranks. Otherwise, the consumer has to consider its history of service selection and accordingly, requests the most reliable web service. To this end, on top of the eagerness of high quality service, the consumer agent requires from the controller agent to be updated. Therefore, if the consumer agent accepts the malicious web service's invitation for collusion, the corresponding risk of reaction from the controller agent needs to be taken into account. If the controller agent recognizes the collusion, the recognized consumer would not benefit from the controller agent's services for some certain time, which affects the consumer agent's expected service quality. To be focussed, in our collusion analysis, we skip the details of collusion willingness regarding consumer's point of view and mainly consider the collusion process initiated by the web service and collaborated by the consumer agent. This limitation does not affect the obtained results.

In the collusion scenario, the malicious web service and consumer agent agree on a collusion that bring some benefits to both colluding parties. The web service i gets extra positive feedback that increase its reputation value out of the feedback file. The enhanced reputation value $\overline{r_i}$ is computed in Equation 8.

$$\overline{r_i} = \beta_1 \overline{Q_i} + \beta_2 \overline{M_i} \tag{8}$$

$$where \quad \overline{Q_i} = Q_i(1 + f_Q) \quad and \quad \overline{M_i} = M_i(1 + f_M)$$

$$f_Q = f(|\mathcal{P}_i|, |\mathcal{T}_i|, |F_i|) \quad and \quad f_M = g(|\mathcal{T}_i|, |\mathcal{G}|, |F_i|)$$

In Equation 8, the factors f_Q and f_M respectively represent the update factor regarding web service i's quality and market share parameters. These factors are functions

of current status of the web service i in the feedback file ($|\mathcal{P}_i|$ denotes the number of positive feedback for i, $|\mathcal{T}_i|$ the number of all feedback for i, $|F_i|$ the number of faked submitted feedback for i, and $|\mathcal{G}|$ the number of all recorded feedback for all active web services). The functions f and g are monotonically increasing with respect to $|\mathcal{P}_i|$ and $|\mathcal{T}_i|$ respectively (note that $F_i \subseteq \mathcal{P}_i \subseteq \mathcal{T}_i \subseteq \mathcal{T}$). Overall, the evaluated rate of reputation of web service i would be increased after collusion. The colluding consumer obtains higher quality of service with low fees, which exceeds its expectations if it acts truthfully. To this end, if the collusion is not recognized by the controller agent, the web service gains higher reputation value and the colluding consumer obtains better deal.

5.5.3 Detecting Malicious Actions

In the proposed framework, the controller agent serves as representative of the reputation system. Therefore, this agent is aimed to seize malicious acts and maintain a sound reputation system. In fact, Cg's challenges are: 1) how cautious to be (how to set the certainty parameter C_i explained earlier, which is proposed by the controller agent to measure the confidence this agent has on the web service i); 2) always being careful not to generate false alarms (detections); and 3) setting proper penalties to avoid detection failures [84]. Failing to detect malicious acts leads to false alarms, which are composed of two cases: the case of penalizing truthful agents (web service and consumer) by mistake (false positive), and the case of ignoring malicious agents by mistake (false negative). When a web service i is under investigation by Cg for a possible malicious action, a reputation value during the investigation time is calculated as shown in Equation 5 and denoted by μ_i. This means only the feedback received during this period are considered.

In the penalizing scenario, the controller Cg applies a penalty that affects the penalized web s.ervice with respect to its reputation value. Also the colluding consumer is penalized in the sense that it will not profit from the controller's services, for instance in terms of getting updated regarding the most recent reputation ranking. To this end, Cg analyzes the applied penalty to minimize malicious acts in the network. One clue would be applying a relatively high penalty to maintain a strong control over the feedback file. Such (harsh) manner does not necessarily imply a high performance for Cg because penalizing truthful agents imposes negative influence on its accuracy level. Therefore, Cg always looks for an optimum penalty value, which minimizes malicious acts and maximizes self-performance level. To detect malicious actions, Cg is then required to be equipped with a mechanism to analyze the interactions of the web services with the consumers. During the investigation, Cg aims to make the best decisions to update its significance level, which affects the accuracy of the rate C_i. Also, Cg needs to learn from the current penalties the information that is used in further detections. In our framework, we suggest using the t-statistic as a measurement of error and detection criteria that Cg uses to capture suspected behavior of the web services. Equation 9 shows this detection criteria where σ_i is the standard deviation of the reputation of i during the investigation period. The threshold ν is set by the controller agent and is application-dependant. The t-statistic is used because the mean and standard deviation of a sample reflecting the investigation time are to be considered, instead of the parameters of the whole periods since the activation of the web service. In fact, this error computes an estimate for the number of standard deviations the given sample (reflecting the behavior of the web service i during the investigation time) is from the mean reputation value of i.

$$|\frac{R_i - \mu_i}{\sigma_i}| > \nu \tag{9}$$

5.5.4 Suspecting Phase

The controller agent initiates a secret suspecting phase about the web service i when equation 9 is satisfied. In this stage, the behavior of the web service i is under closer investigation. The controlled web service is better off doing its best not to get penalized. If the web service did not act according to the raise in its reputation level ($\Delta R_i = R_i - \mu_i$), Cg might penalize the agent for faked feedback. If not, Cg would ignore the investigation and consider the raised reputation level as a normal improvement.

Although Cg uses its history of investigations together with the learned information collected from the environment, always there is a chance of mistake that would cause wrong decision. In general there are four cases: (c_1) the web service acts maliciously and accordingly gets penalized by Cg; (c_2) the web service acts maliciously, but gets ignored by Cg; (c_3) the web services acts truthfully, but gets penalized by Cg; and (c_4) the web service acts truthfully and Cg considers its action normal. Cases (c_1) as true positive and (c_4) as true neutral represent the fair situations. However, cases (c_2) as false negative and (c_3) as false positive are failures, which decrease Cg's performance. In the following, we analyze the scenario for each case and conclude with a general payoff gained by each involved party.

The concept of reputation update is the fact about changing ones reputation level by which social opinions could be influenced. Adversely, the reputation is updated once Cg applies some penalties to detected malicious acts. In general, the feedback file is subject to be modified by some non-authorized agents or an authorized controller agent. The interaction between a selfish web service and the controller agent can be modelled as a repeated game over time. The game consists of actions (made by the web service) and reactions (made by Cg). Here we consider the aforementioned four cases and obtain the corresponding payoffs of each case. The obtained reputation value

for web service i after web service i's action together with Cg's reaction (which could be estimated by $Exp(R_i|Mal)$ or $Exp(R_i|Truth)$ that are computed in Equations 6 and 7) is denoted OR_i. We use R'_i to denote the actual (or fair) reputation that has to be set for web service i. However the current set value (OR_i) might be different from R'_i because of false positives or negatives. In the rest of this chapter, we consider the effect of collusion and penalties on the reputation of web services.

According to the decision made by the controller agent, four outcomes are to be considered and we categorize them as follows: false negative (FN), false positive (FP), true positive (TP), and true neutral (TN). Hereafter, we explain and analyze each one of them.

Malicious Act not Penalized (FN). This is the case where the web service i acts maliciously for instance by colluding with some users and Cg does not recognize it. Thus, web service i increases its reputation level. We refer to this improvement as Imp_i. Imp_i is in fact the increased reputation that is obtained by increasing r_i value. We also refer to the assigned penalty value as Pn_i. This value is set by Cg considering the past history of i and is updated through time elapse. Equation 10 gives the corresponding values for the obtained reputation level OR_i and the actual (fair) reputation value R'_i.

$$OR_i = R_i + Imp_i; \qquad R'_i = R_i - Pn_i \tag{10}$$

$$OR_i - R'_i = Imp_i + Pn_i = \omega \tag{11}$$

The difference between the actual (fair) and current reputation values reflect the payoff that we can use in our game-theoretic analysis (Equation 11). We use this difference to be able to compare the possible scenarios in terms of reputation level. For simplicity, we set $Imp_i + Pn_i$ to ω. The difference here is positive, which means the web service gets benefit of $+\omega$.

Truthful Act Penalized (FP). This is the case where the web service i acts normal, but Cg decides to penalize him. In this case, i would lose its actual reputation value as shown in Equation 12. Equation 13 shows the obtained payoff, which is a negative value in this case. This reflects the fact that the web service i loses ω. This basically affects Cg as well in the sense that a wrong decision is being made, so there is a negative effect applied to its accuracy level.

$$OR_i = R_i - Pn_i; \qquad R'_i = R_i + Imp_i \tag{12}$$

$$OR_i - R'_i = -\omega \tag{13}$$

Truthful Act not Penalized (TN). This is the ideal case where i acts normal and Cg refuses to penalize. In this case the current reputation is the same as the actual reputation ($OR_i = R'_i$). Thus, the payoff assigned to i is zero ($OR_i - R'_i = \omega = 0$).

Malicious Act Penalized (TP). This is also the fair case where web service i acts maliciously hoping to increase self reputation level. Cg detects the action and thus, applies the penalty. In this case, i loses both the penalty and improvement ($-Pn_i - Imp_i = -\omega$).

In the cases considered here, we also need to discuss the obtained payoff for the consumer and controller agents. However, in this reputation model we only focus on the controller agent and skip the details of the penalizing procedure regarding consumer agents. Nevertheless, we assume that the penalized user would not be able to get the controller agent's services, for instance receiving information about the reputation ranking of web services. Therefore, the colluding consumer would be also influenced. Regarding the controller agent's payoff, one basic idea that we use in the rest of this chapter is to consider the accuracy of Cg in detecting the malicious acts and according to the performed reaction, we set the payoff. Therefore, in the first two cases where

the detections are wrong, Cg obtains a negative payoff (say $-\pi$), and in the second two where the decisions are correct, Cg obtains the positive payoff (say $+\pi$). The payoff is not received immediately after Cg's reaction, but after a period of time. The main question is then who is going to pay the controller agent and how? Different scenarios can be applied and in this model we assume that web services and consumers contribute together to the Cg's payoff by paying a fee to the controller for making the system secure and fairly competitive, which is of a great significance for both the consumers and web services. In such a setting, $-\pi$ means less income for Cg because some web services and users stop paying the fees. Here we analyze the different cases according to the four outcomes discussed earlier.

Malicious Act not Penalized (FN). In this case, some bad web services get promoted and ranked high. This can quickly be recognized by the competitors (web services) and some users who had previous experiences with those bad web services. Therefore, those competitors and users will refuse paying the controller as the system is no more secure for the users and fairly competitive for honest web services.

Truthful Act Penalized (FP). In this case, some honest web services are unfairly penalized, which make them stop paying the controller. Other honest competitors and some users who know the reputation of the penalized web services will feel the system unfair and insecure. They can consequently decide to stop contributing in the payment of Cg and get its services.

Truthful Act not Penalized (TN). This is the situation where all the web services and users are satisfied as the system seems secure and working correctly, which brings more competition for the benefit of the users. Web services and users will then continue supporting Cg and requesting its services.

Malicious Act Penalized (TP). In this situation, some users and competitors who know the penalized web services will feel satisfied as the system is getting more secure

and fairly competitive. This will encourage them to increase the Cg's payment to counterbalance the system against the loss caused by the penalized web services who will probably cease participating in the payment of the controller.

The reason behind this payoff assumption is the fact that we consider the interaction between the web service and controller agent as a repeated game. The repeated game theory brings the concept of learning in detection and penalizing process. Such a repeated game would rationally help web services to obtain experiences from the past interactions with Cg and thus, know whether to act maliciously or truthfully. The objective of the repeated game is to maintain a sound reputation mechanism in which the controller agent is getting stronger in reputation updates, and the web services are discouraged to act maliciously.

5.6 Game Theoretic Analysis and Simulation

This section is dedicated to analyze the incentives and equilibria of reputation mechanism using the feedback file. Since the challenge is on the reputation (from web service's point of view, either to act maliciously, i.e. fake F or act truthfully, i.e. act normal N) and accuracy of the feedback file (from Cg's point of view), we model the problem as a two-player game. The active web services are of type good S_G or bad S_B ($P[S_G]$ and $P[S_B]$ represent the portion of each in the environment, e.g. 0.7 and 0.3). Web services of type good are more reliable and likely to act honestly, while the bad ones are more likely to act maliciously. The types are labelled with Cg's opinion imposed by web service's general reputation in the system. Let $Pr[N|S_G]$ (resp. $Pr[N|S_B]$) be the probability that a web service of type good (resp. bad) acts normal. In general, Cg's expected value for normal action from a typical web service is:

$$Pr[N] = P[S_G]Pr[N|S_G] + P[S_B]Pr[N|S_B] \tag{14}$$

where $Exp(R_i|Truth)$ and $Exp(R_i|Mal)$ that are computed in Equations 6 and 7 are good estimators of $Pr[N|S_G]$ and $Pr[N|S_B]$ respectively. For instance, the probability that a web service of type good to act truthfully can be estimated to be the expected reputation of this web service given that it acts truthfully.

The set of pure strategies for web services is defined as $st = \{F, N\}$. This chosen strategy imposes the behavior that the web service shows and thus, the controller agent observes after the action is occurred. Cg also chooses between two strategies: penalizing (P) and not penalizing, which means ignoring (I) the event. We consider a payment function χ associated to the sequence of actions performed by web services. The payment mechanism is defined as follows: $\chi : st \times st^{M-1} \mapsto [-\omega, +\omega]$, where M is the number of actions performed during the past and current periods and $-\omega$ and $+\omega$ are explained and computed in equations 11 and 13. Thus, $\chi(O_i, O_{-i})$ represents the assigned payoff to web service i when it selects $O_i \in st$ at current moment and $O_{-i} \in st^{M-1}$ represents its $M-1$ previous chosen strategies during $M-1$ periods. There is a similar payoff function for Cg that assigns values in the range $[-\pi, +\pi]$. In the rest of this section, we start by analyzing the one-shot game, which is then extended to continuous game.

Proposition 5.1 *In one-shot game, penalizing a fake action is the unique Nash equilibrium.*

Proof: Clearly acting fake by web service i, controller agent Cg would have a best utility if penalizing strategy is chosen rather than ignoring. On the other hand, if Cg chooses to penalize, i would not change its chosen strategy since in both cases i will lose $-\omega$. Consequently, penalizing a fake action is a Nash. Adversely, the normal act by i would let Cg to ignore. However, if the strategy is to ignore (by Cg), the best strategy for i is to act fake. Therefore, there is no Nash in ignoring the normal act.

Therefore, the obtained Nash is unique.

In one-shot game, players only consider the present information and they rationally tend to stay in fake-penalized state. This unique Nash is a good situation for Cg, but not for i. We need to study a socially better situation for both players when they learn the best strategies over time. This can be done by considering the repeated game. If i can estimate the expected payoff with respect to Cg's response, it might prefer acting normal. In fact, this issue is how to make agents (i and Cg) converge to a Pareto-Optimal [5], which is the best situation for both players. We call this situation Pareto-Optimal Socially Superior.

Definition 5.2 ***Pareto-Optimality.*** *A situation in a game is said to be Pareto-Optimal once there is no other situation that makes at least one player better off without making any other player worse off.*

In the following, we extend the one-shot game to the repeated game over periods of time. Therefore, following different strategies in time intervals will generate the corresponding payoffs to the players. At a given moment, Cg would decide whether to continue or stop investigating. To this end, e_0 is referred to as the case of doing no investigation effort and basically ignoring all actions. Otherwise, the best effort is made by Cg doing investigation. Cg has to decide about a proper strategy and obviously, if it chooses e_0 and i plays fake, the controller agent would lose right away. For simplicity, we analyze the game during fix intervals of time and a strategy of acting in each interval needs to be decided. We apply a weight to each interval to reflect the payoff portion during this interval. For instance, if 2 intervals are considered, μ would be the payoff coefficient for the acts done in $[t_0, t_1]$ and $1 - \mu$ the payoff coefficient for the acts done in $[t_1, t_2]$.

For simplicity and illustration purposes but without loss of generality, we consider the repeated game with two shots. The general case with n shots ($n \geq 2$) will follow. In such a game, web service i as player 1 has to make two decisions over over fake F and act normal N, one in the first decision time spot (weighted by μ), and the other in the second decision time spot (weighted by $1-\mu$). Since i is the game starter, and Cg initially decides whether to stop or continue the game, we consider two continuous actions that reflect our game the best. An example of these actions is faking the first time spot (denoted here by F^{μ}) and the second time spot ($F^{1-\mu}$), which is denoted by $F^{\mu}F^{1-\mu}$. Therefore, i's set of pure strategies is $A_i = \{F^{\mu}F^{1-\mu}, F^{\mu}N^{1-\mu}, N^{\mu}F^{1-\mu}, N^{\mu}N^{1-\mu}\}$. In n-shot game, the set of pure strategies is: $A_i = \{F^{\mu_1} \dots F^{\mu_n}, F^{\mu_1} \dots N^{\mu_n}, \dots, N^{\mu_1} \dots N^{\mu_n}\}$ where $\sum_{i=1}^{n} \mu_i = 1$. Considering the choice of efforts, Cg's set of pure strategies (penalizing P or ignoring I) is $A_{Cg} = \{e_0, P^{\mu}P^{1-\mu}, P^{\mu}I^{1-\mu}, I^{\mu}P^{1-\mu}, I^{\mu}I^{1-\mu}\}$. Table 5.1 represents the payoff table of the two players over their chosen strategies. We continue our discussions in the rest of this section on this table.

In this game, the idea is to give the highest possible payoff $+\omega$ to the case in which i decides to fake the most and gets ignored by Cg. The more Cg recognizes the malicious act of i, the highest assigned negative value weighted by the payoff portion of the time spot (μ or $1-\mu$). For instance, if the web service i decides to fake during the first time spot but gets penalized, i's payoff would be $-\mu\omega$, and if it decides to fake again, but gets ignored this time, it will gain $(1-\mu)\omega$, which makes the final payoff $\chi(O_i, O_{-i}) = (1-2\mu)\omega$ (see line 3 column 1 of Table 5.1). There is a similar payoff assignment for Cg in the sense that its accurate detection is under investigation. For example, a correct detection in the first time spot would bring $+\mu\pi$, and if the second detection is wrong, this first portion will be added to the negative payoff of the second time spot $-(1-\mu)\pi$, which makes the final payoff equal to $(2\mu-1)\pi$ (see

Table 5.1: Two-shot game between web service i and controller agent Cg with obtained payoffs

Controller agent Cg \ Web service i	$F^{\mu}F^{1-\mu}$	$F^{\mu}N^{1-\mu}$	$N^{\mu}F^{1-\mu}$	$N^{\mu}N^{1-\mu}$
e_0	ω,-π	μω,-μπ	(1-μ)ω,-(1-μ)π	0,0
$P^{\mu}P^{1-\mu}$	**-ω,π**	-ω,(2μ-1)π	-ω,(1-2μ)π	-ω,-π
$P^{\mu}I^{1-\mu}$	(1-2μ)ω,(2μ-1)π	-μω,π	(1-2μ)ω,-π	-μω,(1-2μ)π
$I^{\mu}P^{1-\mu}$	(2μ-1)ω,(1-2μ)π	(2μ-1)ω,-π	-(1-μ)ω,π	-(1-μ)ω,(2μ-1)π
$I^{\mu}I^{1-\mu}$	ω,-π	μω,(1-2μ)π	(1-μ)ω,(2μ-1)π	0,π

line 3 column 1 of Table 5.1). The crucial key to survive in the environment for both players is to consider the previous events and moves. In the following, we elaborate on different cases while web services do or do not consider Cg's behavior in the game.

Proposition 5.3 *In repeated game, if i is not aware of Cg's previous chosen strategies, then faking all the time and penalizing all fake actions is the unique Nash equilibrium.*

Proof: (We illustrate the proof for two-shot game from which the general case follows.)

Nash. *It is clear from Table 5.1 that in both faking intervals, Cg receives the maximum payoff by penalizing both cases. In this case, i would not increase its payoff ($-\omega$) and thus, would not prefer any other strategy. In any other case, by choosing the maximum received payoff for any player, the other player has a better strategy to increase its payoff.*

Uniqueness. *We prove that theis Nash point is the only Nash with respect to the following cases. In the first row of Table 5.1, there is no Nash because Cg makes no effort, so the maximum received payoff is zero and thus, it can be increased by changing the status. In the third and forth rows, still there is no Nash since in these rows there are choices of P and I in the sense that for any of these choices, i would be better off*

changing to a strategy that maximizes its assigned payoff. In the last row, the payoff assignment is similar to the first one, so that Cg prefers to change its chosen strategy to apply penalty to fake actions.

We also have the following propositions generalized from the two-shot game. We motivate the fact that if the penalty assigned by Cg is clear, the strategy chosen by i would be different. The proofs are straightforward from the two-shot game as shown in Table 5.1.

Proposition 5.4 *In repeated game, if i is not aware of Cg's previous chosen strategies, then faking all the time is dominant strategy for i.*

Proposition 5.5 *In repeated game, if i is not aware of Cg's accuracy level, then acting normal by i and ignoring by Cg all the time is Pareto-Optimal Socially Superior .*

To analyze the reasons behind encouragement to act truthfully, we need to measure some expected values. In the repeated game, the probability that exactly n normal acts out of M acts are done in the past and current moment ($Pr[n, M]$) can be computed using binomial distribution as follows:

$$Pr[n, M] = \binom{M}{n} Pr[N]^n (1 - Pr[N])^{M-n} \tag{15}$$

where $Pr[N]$ is calculated in Equation 14. We use this probability in measuring the expected cumulative payoff denoted by $V(O_i, O_{-i})$ for web service i in the sense that in the chosen strategies (O_i, O_{-i}) n actions were normal as follows:

$$V(O_i, O_{-i}) = \sum_{n=0}^{M} Pr[n, M] \chi(O_i, O_{-i}) \tag{16}$$

As the objective of a rational web service i is to maximize the expected cumulative

payoff, it would select the current strategy O_i^* that maximizes $V(O_i, O_{-i})$:

$$O_i^* = argmax_{O_i \in st} V(O_i, O_{-i}) \tag{17}$$

This would be achieved when the following inequality is satisfied:

$$\sum_{n=0}^{M} Pr[n, M]\chi(O_i^*, O_{-i}) > \sum_{n=0}^{M} Pr[n, M]\chi(\overline{O_i^*}, O_{-i}) \tag{18}$$

where $\overline{O_i^*}$ denotes the opposite strategy of O_i^*, which means:

$$V(O_i^*, O_{-i}) > V(\overline{O_i^*}, O_{-i}) \tag{19}$$

Recall that q is the probability of correct recognition via Cg that impacts the strategy that i adopts in the repeated game. Therefore, in the repeated game, these probabilities of Cg are labelled as $q^{t_0}, \ldots, q^{t_M}$, which reflects the evolution of Cg's accuracy over time. Indeed, Cg's accuracy has impact on the expected cumulative payoff that web service i estimates given the penalty and improvement it makes. Therefore, Cg applies such penalty that discourages i to act maliciously.

Proposition 5.6 *At a given moment t_n, If $Pn_i > \frac{1-2q^{t_n}}{q^{t_n}} Imp_i$, then web service i receives less cumulative payoff $V(O_i, O_{-i})$ if it acts maliciously.*

Proof: To prove this proposition, we can simply assume that all the previous strategies O_{-i} are known as normal, and prove that if the condition $Pn_i > \frac{1-q^{t_n}}{q^{t_n}} Imp_i$ is true, then $O_i^* = N$. As $V(O_i, O_{-i})$ is defined in terms of $\chi(O_i, O_{-i})$ (Equation 16), which in turn is defined in terms of i's reputation, we simply need to prove that if the condition is true, then i will have less reputation value. To do that, we need to prove that:

$$Exp(R_i|N^{\mu_1} \ldots N^{\mu_{n-1}} F^{\mu_n}) < Exp(R_i|N^{\mu_1} \ldots N^{\mu_n})$$

By simple calculation, we expand the expected values to their possible cases together with their probabilities, so we get:

$$Exp(R_i|N^{\mu_1}\dots N^{\mu_{n-1}}F^{\mu_n}) =$$
$$(q^{t_n})(R_i - Imp_i - Pn_i)$$
$$+(1-q^{t_n})(R_i + Imp_i)$$

$$Exp(R_i|N^{\mu_1}\dots N^{\mu_n}) = R_i$$

The first equation gives the expected reputation value given that a fake action is made at the moment t_n, and the second one shows the expected reputation value given that no fake action is made. Assuming that $Pn_i > \frac{1-2q^{t_n}}{q^{t_n}} Imp_i$, it is easy to see that:

$$(q^{t_n})(R_i - Imp_i - Pn_i) + (1-q^{t_n})(R_i + Imp_i) < R_i$$

so we are done.

Theorem 5.7 *q^{t_n} is increasing with respect to Imp_i and decreasing with respect to Pn_i*

Proof: From Proposition 5.6, we obtain the lower bound of Cg's accuracy q^{t_n}: $\frac{Imp_i}{Pn_i+2Imp_i}$ ($q^{t_n} > \frac{Imp_i}{Pn_i+2Imp_i}$). Let B_i denote this lower bound. We have:

$$\frac{\partial B_i}{\partial Imp_i} = \frac{Pn_i}{(Pn_i + 2Imp_i)^2}$$

As $Pn_i \geq 0$, $\frac{\partial B_i}{\partial Imp_i} \geq 0$, which means B_i is increasing with respect to Imp_i. Consequently, q^{t_n} is also increasing with respect to Imp_i.

Table 5.2: Implemented environment details

Agent Type	Number	Agent Subtype	Percentage	Tendency
Controller	1	-----	-----	-----
Web service	100	Fixed	Truthful 20% Malicious 20%	0% 100%
		Random	30%	50%
		Observation	30%	$f(p_1,p_2,...,p_n)$
Consumer	1000	Fixed	Truthful 20% Malicious 20%	0% 100%
		Random	30%	50%
		Observation	30%	$g(p_1,p_2,...,p_n)$

On the other hand, we have:

$$\frac{\partial B_i}{\partial Pn_i} = \frac{-Imp_i}{(Pn_i + 2Imp_i)^2} \leq 0$$

B_i is then decreasing with respect to Pn_i. Consequently, q^{t_n} is also decreasing with respect to Pn_i.

This theorem is important and very intuitive as it tells us if the improvement in the web service's reputation is very high, then the controller should be very cautious as probably the improvement is a result of some malicious actions. On the other hand, if the agent is less cautious, then this should be balanced by making the penalty high, so that web services will be discouraged to act maliciously. This theorem is inline with the result found in [84] according to which "buying agents will not be harmed infinitely by dishonest selling agents and therefore will not incur infinite loss, if they are cautious in setting their penalty factor".

Theorem 5.8 *In n-shot repeated game, if $Pn > \frac{1-2q^{t_n}}{q^{t_n}} Imp_i$, acting normal and being ignored is both Nash and Pareto-Optimal.*

Proof: From Proposition 5.5, we know that ignoring normal acts in all the shots is Pareto-Optimal. On the other hand, from Proposition 5.6, we deduce that i would

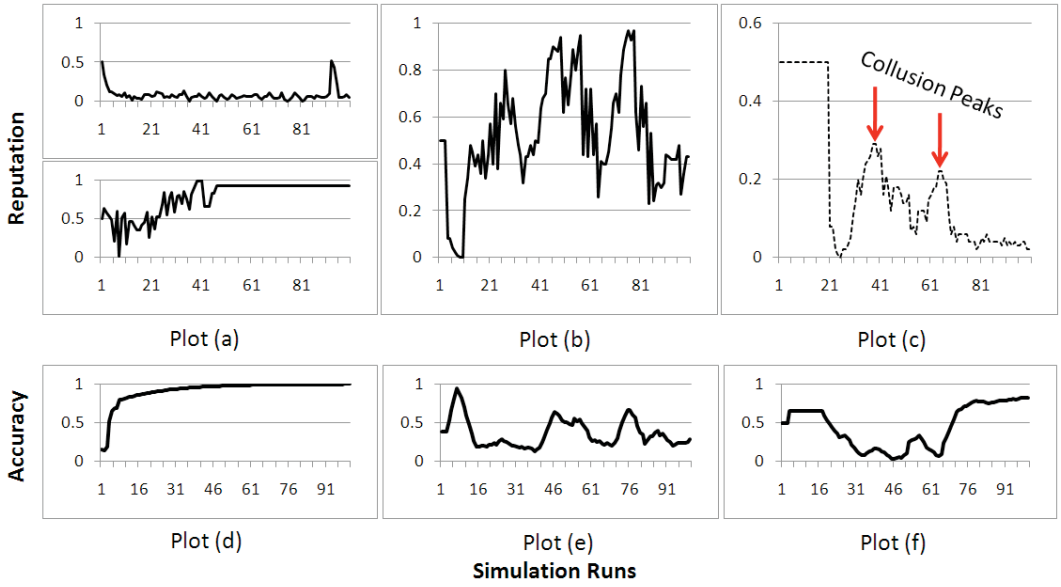


Figure 5.2: Overall reputation and accuracy assessment regarding different types of web services

have less cumulative payoff if it fakes given that it is aware of the assigned penalty Pn_i *and* Cg*'s accuracy. Therefore, the dominant strategy for* i *would be acting* N*. If* i *plays* N *as its dominant strategy, the best response from* Cg *would be* I *in all shots (see Table 5.1). Therefore, if the condition* $Pn > \frac{1-2q^{t_n}}{q^{t_n}} Imp_i$ *holds, then playing* N *and* I *is Nash, where* $N^{\mu_1} \ldots N^{\mu_n}$ *and* $I^{\mu_1} \ldots I^{\mu_n}$ *are dominant strategies for* i *and* Cg*, which completes the proof.*

This theorem shows that if the web services are aware of the penalties and the controller's accuracy, then the system will achieve a secure and healthy state.

5.7 Simulation and Experimental Results

We developed a simulator in a java-based platform hosting different agents having broad range of characteristics and capabilities. Three types of agents are implemented: controller agent, web service agents, and consumer agents. During the simulation runs, web services and consumers might leave or join the network if they wish so. Table

5.2 provides detailed information regarding the implemented environment. We categorized the consumer and web service agents into three classes with respect to their acting strategies through simulation runs: (1) acting strategies using fixed opinions; (2) acting strategies using random movements; and (3) acting strategies using environment observations. Acting using fixed opinions means agent are completely (100%) truthful (0% tendency to act maliciously) or completely malicious (100% tendency to act maliciously). Acting using random movements means agents randomly decide to act truthfully or maliciously and can change their decisions continuously. This type of agents, which represents 30% of the population with 50% tendency to act maliciously, makes the environment more realistic with presence of noise. Finally, acting through observations means agents are strategic and change their behaviors based on their observations of Cg's performance and their tendency to act maliciously is function of previous and current observations $p_1, \ldots, p_n$. The objective of this simulation is to analyze the outcome and performance of these agent types in different scenarios.

The first group of agents follow their predefined strategies regardless of the environment changes. The agents following this strategy fall into two groups of malicious and truthful. Figure 5.2 plot (a) illustrates two graphs reflecting the accumulated reputation of two typical web services (truthful and malicious) over the simulation runs. The truthful web service (lower graph) gradually maintains its actual reputation value, which converges to its publicly announced quality of service. This is the normal case in the implemented environment as the active web service collects the feedback with respect to the offered quality of service and thus, the accumulated reputation would reflect the actual quality value. The malicious web service (upper graph) eventually loses its accumulated reputation because based on its fixed strategy, it will continuously be involved in collusion scenarios. The controller agent recognizes the collusion made by web services following fixed malicious strategies. As consequence, the reputation

dramatically decreases at a certain time. Figure 5.2 plot (d) illustrates the overall reputation mechanism efficiency (i.e. Cg's accuracy) with respect to all the actions made by the web services and consumers and the reactions maintained by the controller agent. As shown by the graph, the controller agent acts accurately. In fact, recognizing the malicious actions maintained through fixed strategies is easy to learn for the controller. Therefore, the accuracy obtained by the controller agent is relatively high. We would carry on illustrating the efficiency graph in the rest of this section in order to compare the impacts on the reputation mechanism imposed by diverse parameters in the environment.

Figure 5.2 plot (b) represents the same results according to the observed reputation of a typical agent following random behavior as acting strategy. As represented in Table 5.2, agents of this type are developed with 50% chance of acting maliciously. Observed in different simulations, the controller agent is capable of recognizing these agents from time to time and penalizes them as it keeps the information regarding the past detections and web services with history of being detected are investigated more carefully. Hence, the web services which do not consider the controller's existence in their acting strategies would fail to accumulate a stable reputation value. In this figure, plot (e) illustrates the corresponding reputation mechanism efficiency with respect to the maintained actions. In fact, the unpredictable behavior of this type of web services confuses the controller agent because the agent that has maintained some collusion attempts, might act truthfully at some periods of time, where the controller agent has got very suspicious about the web service agent (because of a number of detected collusion attempts). The unpredictable and random behavior of this agent would generate a number of false detections for the controller agent, which brings about an oscillating efficiency.

Figure 5.2 plot (c) illustrates the results regarding a typical web service that considers the environment parameters (the controller agent's accuracy) in its acting strategy. The behavior of this type of agent is more dynamic compared to the previously discussed agents. In this plot, the considered web service maintains collusion attempt twice, which in both, the controller agent recognizes the attempt. Overall, controlling this type of agents is easy, but takes some time for the controller to completely learn from their behaviors and as illustrated by plot (f), the corresponding reputation mechanism efficiency increases once the behavior is being learnt, which reflects the controller agent's overall capability to manage the detections. In the rest of this section, we analyze the reputation assessment and reputation alteration in no collusion, collusion, and collusion-resistent environments. The exposed graphs are upon observed data from different experiments to avoid unpredicted randomization effects.

5.7.1 Reputation Assessment with No Collusion

We ran the simulation in a safe environment within which, web services act truthfully and the accumulated feedback reflect the actual reputation of the web services. The rational behind this experiment is to emphasize the fact that based on truthful actions, the accumulated reputation of a web service would approach its actual quality of service.

Figure 5.3 illustrates different curves obtained from separated simulation runs regarding only one typical web service i holding a quality Q_i. As shown by the figure, the overall reputation of this web service approaches its actual quality of service QoS_i over different experiments. This fact is analyzed via the reputation assessment procedure that is formalized in Equation 5 in Section 5.4. The reputation value regarding web service i is computed by aggregating web service's quality Q_i with the web service's market share M_i. In the simulations, M_i follows a normal distribution $\mathcal{N}(Q_i, 0.2)$.

According to a truthful web service i, Q_i percent of services are satisfactorily

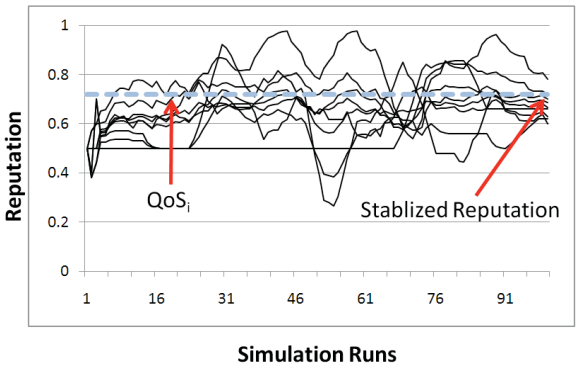


Figure 5.3: Reputation assessment with no collusion

offered and thus, web service i expects positive feedback for "Q_i" percent of total posted feedback. However, there is another parameter that comes to play, which is the probability of posting unbiased feedback k from a consumer agent upon reception of a service ($Pr(unb(k))$). Therefore, the probability of receiving positive feedback ($Pr(k \in \mathcal{P}_i)$) for web service i is computed in Equation 20.

$$Pr(k \in \mathcal{P}_i) = Q_i \times Pr(unb(k)) \tag{20}$$

The value $Pr(unb(k))$ would be different in experiments according to the reaction of the consumers. This value is out of control and is completely based on the distribution that the consumer uses to produce accurate feedback regarding the received service. In Figure 5.3, different curves are shown reflecting a number of experiments in which, the consumers use dynamic probabilities of providing unbiased feedback. However, overall in all of the graphs, the total reputation of the web service approaches its general quality of service (QoS_i) value. This means that, in a honest environment in which agents do not perform collusion, one's quality of service overall reflects its accumulated reputation. The reputation mechanism efficiency in this case is pretty high and very similar to the one shown in Figure 5.2 plot (d). The controller agent can easily manage the system control (as long as there is no collusion and web services act

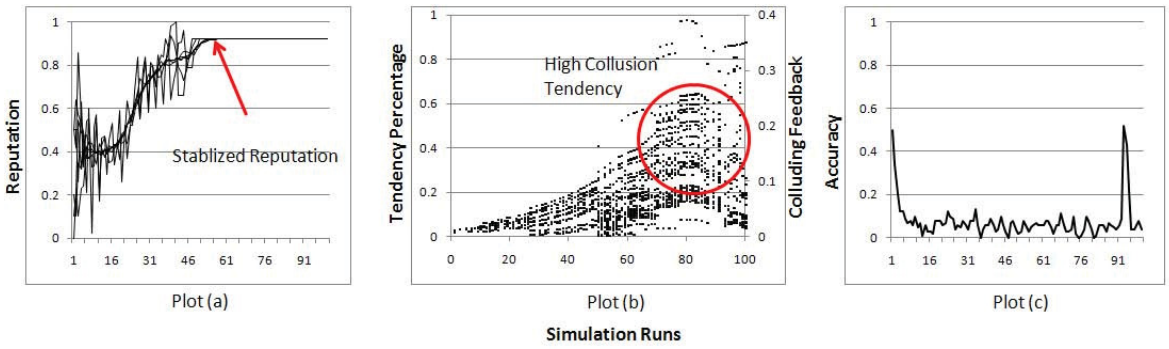


Figure 5.4: Reputation assessment through collusion

truthfully) and quickly recognize that the active users do not attempt to collude.

5.7.2 Reputation Assessment Through Collusion

In this simulation, we investigate the collusion impacts on the malicious agents' reputation values in a scenario where the controller agent imposes no penalty during simulation runs. Figure 5.4 plot (a) illustrates one typical malicious web service's reputation value extracted from different experiments. As depicted by the curves, the malicious web service performs collusion in all of them. This is due to the fact that the web service at the earlier collusion experiments recognizes that the controller agent is dormant and therefore, there would be no penalty applied after a performed collusion. This results in a dramatic increase of the probability of colluding.

Figure 5.4 plot (b) represents the collusion tendency of the whole network involving all the web services that are capable of acting maliciously, which represents 80% of the population ($20\% + 30\% + 30\%$, see Table 5.2). The X-axis of this plot denotes the elapse time over the simulation runs. The left Y-axis denotes the percentage of colluding web services (obtained from whole active web services in the network). This value is increasing over time, which expresses the increasing tendency of the web services to act maliciously. The right Y-axis denotes the number of colluding feedback, which

reflects the amount of increase in faked positive feedback set in the collusion agreement between the colluding web service and consumer. The dots in plot (b) show the extent to which the colluding web service maintains faked feedback in the collusion process. It is observed that overall, the amount of faked feedback is increasing over time when the malicious action is widespread in the environment. In such a chaos system, the performance of reputation mechanism (controller agent's accuracy) decreases dramatically as shown in Figure 5.4 plot (c)).

5.7.3 One-shot Game and Penalty Impact on Reputation Assessment

In this part, we expose the results obtained after one-shot game between the web service and consumer agents. Figure 5.5 plots (a), (b), and (c) represent respectively the reputation graphs obtained after a series of experiments. We study this result on three different types of web services (acting upon fixed opinions, acting randomly, and environment observers). In plots (a) and (b) typical agents follow strategies within which the controller agent's action is not considered. However, plot (c) shows agents which follow a strategy which considers controller agent's action. All these agents adopt malicious actions and get penalized via the controller agent, which confirms the theoretical result discussed in Proposition 5.1 that represents the Nash equilibrium. As shown in plots (d), (e), and (f), the controller agent expresses accurate collusion detection system and thus, the efficiency graph is increasing over time. However, the social situation depicted by the Nash is not well-accepted since the collusion is maintained regardless of the controller's accuracy.

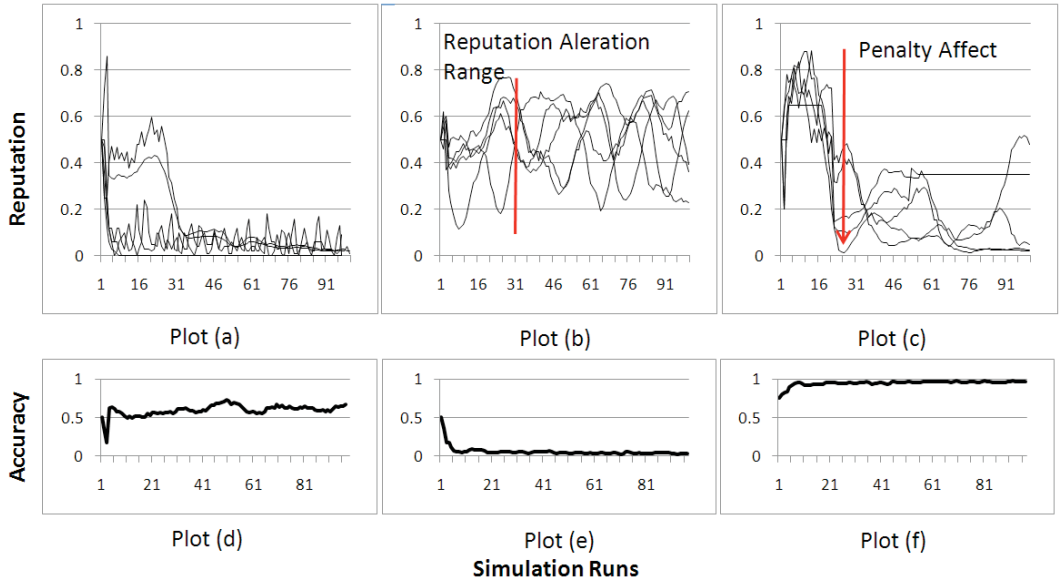


Figure 5.5: Reputation assessment and penalty impact in one-shot game

5.7.4 Repeated Game and Penalty Impact on Reputation Assessment

To simulate the repeated game case, we ran the simulation with agents capable of analyzing the history of interactions in order to adopt the most appropriate strategy. The web services, which belong to categories of fixed and random opinions are not considered in these experiments as they carry on the same behavior shown in Figure 5.5 plots (a) and (b). The repeated game and history analysis only affect the agents, which consider the environment characteristics. To this end, we run many experiments considering these agents with tendency to maintain malicious actions over the time. Figure 5.6 shows different simulations running different web services capable of observing the environment characteristics and analyzing the history of previous interactions with the controller agent. In these simulations, the controller agent adopts different detection and penalty settings, which imposes some impacts on the behavior and convergence of web services to a truthful reputation mechanism. In these experiments, the involved web services are all capable of collusion attempts. However, as shown in all the graphs,

the collusion attempt is being detected and the corresponding penalty is applied, which results in decreasing the reputation value. As mentioned in Theorems 5.7 and 5.8, the assigned penalty exceeding the specified threshold brings about truthful actions made by malicious web services and affects controller agent's accuracy to some certain extent. Obviously for the sake of true detections, the controller agent cannot increase the assigned penalty with no limit.

The graphs shown in plots (a), (b), and (c) are representative of reputation management regarding different penalty settings that the controller agent imposes in a repeated game to a set of malicious web services. Figure 5.6 plot (d) shows overall reputation management efficiency reflected by the accuracy of the controller agent in detecting malicious actions. The analysis of the reputation management efficiency shows that obtaining high efficiency does not necessarily make web service adopt the truthful strategy as shown in Figure 5.2 Plot (d). However, it is crucial to obtain this efficiency where web services also tend to act truthfully. The results in plots (a), (b), and (c) show that reputation values are affected through the collusion and penalties assigned by the controller agent. Figure 5.6 plot (e) shows the overall tendency of malicious web services to attempt collusion. Over simulation runs, the tendency of these agents is decreasing, which reflects the trustful action as Nash equilibrium.

5.8 Related Work

Reputation is measured in open systems using different methodologies [29]. In the literature, the reputation of web services have been intensively stressed [41]. In [77], the authors have developed a framework aiming to select web services based on the reputation policies expressed by the users. The framework allows the users to select a web service matching their needs and expectations. In [52], authors have proposed a

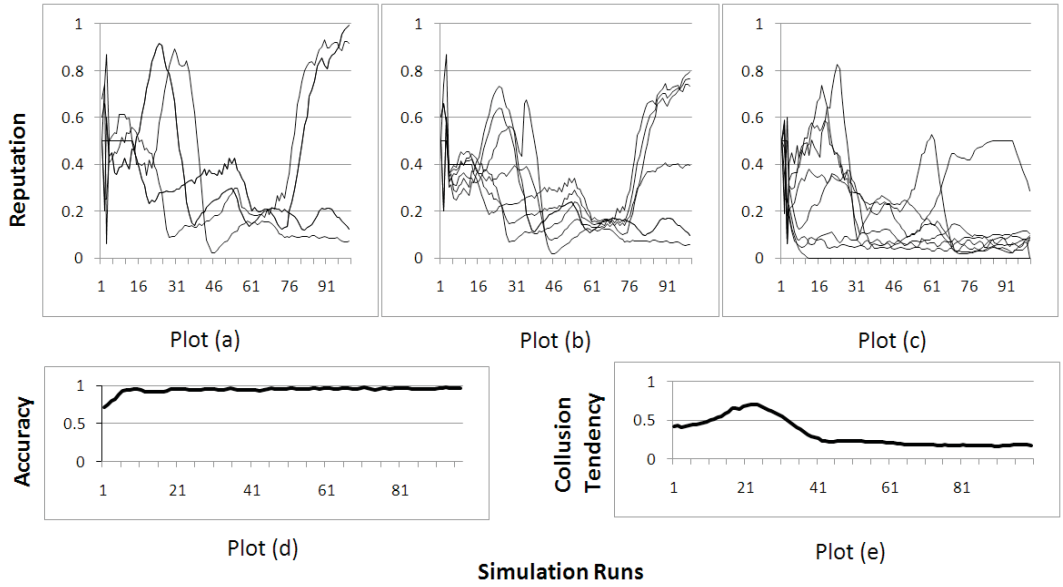


Figure 5.6: Reputation assessment and penalty impact in repeated game

model to compute the reputation of a web service according to the personal evaluation of the previous users. These proposals have the common characteristic of measuring the reputation of web services by combining data collected from users. To this end, the credibility of the user that provides the data is important. In [49], authors have designed a sound mechanism to address the credibility of the collected data from users. In [57], a multi-agent framework based on an ontology for QoS has been designed. The users' ratings according to the different qualities are used to compute the reputation of the web service. In [35, 37], service-level agreements are discussed in order to set the penalties over the lack of QoS for the web services. In [26], a layered reputation assessment system is proposed mainly addressing the issue of anonymity. In this work, the focus is on the layered policies that are applied to measure the reputation of different types of agents, specially the new comers. Although, the proposed work is interesting in terms of anonymous reputation assessment, the layered structure does not optimally organize a community-based environment that gathers web services and users, and also the computational expenses seem to be relatively high.

In all the aforementioned frameworks, the service selection is based on the data that could not be reliable. The main issue we addressed in this model, and which makes it different from the existing proposals is that web services are selfish agents and utility maximizers. Thus, if those agents are not provided with an incentive to act truthfully, they can violate the system to maliciously increase their reputation level. Analyzing the relationship between the payoffs and systems efficiency is another issue that has not been addressed in related proposals.

5.9 Conclusion

The contribution of this reputation model is the theoretical analysis and simulation over the reputation-based infrastructure that hosts agent-based web services as providers, users as consumers, and controller agent as reputation manager of the system. In the deployed infrastructure, web services can act maliciously to increase self reputation. Meanwhile, controller agent investigates user feedback and penalizes malicious web services. Controller agent may fail to accurately function, which is an incentive for some web services to act maliciously. The discussion is formed in terms of a game that is analyzed in one-shot and then repeated cases. This analysis is concluded by denoting the best social state in which selfish services are discouraged to act maliciously and increase self reputation. The analysis is accompanied by empirical results that highlight reputation system's parameters. In experimental results, malicious services are observed and their characteristics are measured over time. In general, the Pareto-Optimality is observed to be a stable state for both web services and the controller agent.

Our plan for future work is to advance the game theoretic analysis such that web services that risk the malicious act deploy a learning algorithm that enables them to

measure their winning chance. To this end, a continuous game can be extended, so that both players update their selected policies. Similarly, we need to discuss more about the different false detection cases that distract the reputation management.

Chapter 6

Long-term Performance Mechanism

Applied to Community of Web Services

6.1 Background

In the previous chapter, we investigated situations where agents are encouraged to act truthfully and the inaccurate information is discarded according to controller agent's collusion detection. In this chapter, we propose a model that is based on the infrastructure proposed in Chapters 4 and 5 and maintains long-term performance for the agents in interactive multi-agent systems. The contribution of this chapter is the analysis of long-term interactive strategies that constrain high performances for agents involved in dynamic environments hosting rational and selfish agents. In the model proposed in this chapter, we mainly focus on the efficiency of rational entities in the form of single web service agents or communities of web services. Unlike previous chapter, we do not consider controller agent to impose incentives to interacting agents to maintain sound reputation mechanism. Using game theory, we mainly highlight the states where rational agents obtain high performances while they are active for long-time in a dynamic multi-agent system.

6.2 Overview and Motivation

A multi-agent system is composed of multiple intelligent agents that according to their goals play different roles and follow different strategies of acting. Each agent is in fact a decision maker that seeks to effectively accomplish its goals. A typical active agent in a multi-agent system environment is potentially limited due to its own observations and domain knowledge. This is the main reason behind agent communication in an environment composed of multiple entities, which are functionally distributed. As motivated in previous chapters, the network of web services with consumers is one example of multi-agent system design as it represents distributed cooperation in IT networks. A typical web service abstracted as an intelligent agent is capable of providing some services in some certain domains. Doing this, the web service agent maintains some interactions and compositions aiming to enhance its productivity in enterprize networks. A typical service consumer is also an intelligent agent, which is capable of comparing different service qualities and based on its domain knowledge attempts to enhance the obtained quality in an enterprize network.

In IT networks with cooperative settings, each web service acts individually, but it is the resulting joint action that produces the outcome. Cooperation is therefore a crucial aspect that improves performance, robustness and scalability in such settings. The goal of cooperation is to result in optimal outcome for the group as a whole. In multi-agent systems composed of web service agents, the key goal is to maximize individuals outcome and performance. Considering the service quality as a built-in individual characteristic, we distinguish the performance from the quality in the sense that the performance is defined as the extent to which web service agent is successful in accomplishing some of its goals, whereas the quality is the ability of the agent to provide the required service. Exploring this further, web service agent is successful (highly efficient) when it can effectively use its resources and abilities. Adversely,

a web service agent is unsuccessful (poorly efficient) when it fails to manage its resources due to either high service demand or no demand. To this end, the key goal of a rational intelligent web service agent is to maximize its performance (with respect to its individual capabilities) rather than its service quality. We believe that performance increase is a systematic quality assurance procedure whereas the built-in quality increase needs fundamental enhancement of web service capabilities.

As discussed in Chapter 4, to address web service cooperation, there have been efforts attempting to model and analyze collaborations with communities of web services [50, 69, 70]. Recall that communities (introduced in Chapter 4) are frameworks gathering functionally similar web service agents that share a common goal [49]. In the context of communities, we distinguish web services collaboration from web services composition. By collaboration, 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 finalize a specific task. In all these proposed frameworks, the objective is to increase performance in distributed computing. However, in such frameworks, strategies web services can follow to achieve this goal are just limited to aggregation and different types of collaborations. In this context, more sophisticated strategies are yet to be investigated and analyzed. Such sophisticated strategies can help communities and individual web services achieve higher performance in using their resources.

The aim of this chapter is to investigate strategies as rational behaviors that web services and communities can adopt to increase performance. We present a game-theoretical 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 score rate, market share, capacity, and performance parameters. Using our proposed framework, interactive agents are capable of efficient decision making mechanism which yields maximum performance in multi-agent environments with diverse characteristics. A game is defined between typical single web service and the representative of a typical community (called *master web service*). Each entity seeks maximum performance 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' performances. Overall contributions of the proposed model are threefold: (1) we provide a distributed network of web services and consumers where the task allocation problem is regulated by a mechanism taking score rate, market share, and performance into account; (2) we propose a game-theoretic analysis investigating the stabilized situation within which, entities achieve high performance; and (3) we identify thresholds allowing the master web service to identify the optimal number of web services associated to the community. We also provide experiments that show and uphold the impact of our game-theoretic analysis on the behavior of rational web services.

6.3 The Model

In terms of notations, in the previous chapter we referred to an agent by letter i. For instance, R_i represented the reputation associated to agent i. In this chapter, we consider two types of intelligent entities: individual web services and communities of web services. To this end, here we refer to individual web services by w and communities of web services by c. Therefore, w_i and c_j respectively represent single web service i and community j.

In our multi-agent design, we consider feedback pool where consumer agents post

their satisfaction ratings regarding their past experiences with a particular service provider. In this system, we assume every consumer agent provides post-interaction feedback. Our assumption in rational behavior of the consumer agents does not deviate from our main contributions and therefore, we skip details regarding consumer agents' variation of accuracy.

In our feedback system, posted feedback are accumulated to compute and analyze service providers' score rate in multi-agent environment. A typical web service agent's reputation (R_{w_i}) could be written as a function of its score rate (Sr_{w_i}). But we skip the details of this function as it is discussed in Chapter 5. The rate is simply computed based on satisfaction rates obtained from other interacting agents. In the proposed model in this chapter, the score rate of a service provider (Sr_{w_i} for the web service w_i and Sr_{c_j} for the community c_j) as a value between 0 and 1. Web services and communities as rational agents aim at increasing this value, which imposes positive impact on their outcomes. However, increasing the score rate brings more requests which might impose negative impact if requests are not systematically handled. In this framework we use a simple and conventional scoring mechanism like the one used in e-bay with three forms of $+1$ for satisfied experience, 0 for no response to the request, and -1 for dissatisfied experience. This mechanism adds the value of all provided feedback for particular service provider and divides by their number. In Equation 1, PF_{w_i} and NF_{w_i} respectively denote the number of positive and negative feedback posted for web service w_i. NR_{w_i} denotes the number of no responded requests associated to web service w_i. In Equation 1, the score rate of the community c_j is computed as the average of the score rates of all the web services (n) that are associated to the community c_j.

$$Sr_{w_i} = \begin{cases} \frac{PF_{w_i} - NF_{w_i}}{PF_{w_i} + NF_{w_i} + NR_{w_i} + 1} & if \quad PF_{w_i} > NF_{w_i} \\ 0 & otherwise \end{cases} \tag{1}$$

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

The parameters PF_{w_i}, NF_{w_i}, and NR_{w_i} are updated on regular basis that could be daily (in fact depending on the multi-agent design and how interactive the transaction system is). Therefore, upon request, the updated score rate value is provided. Similar to the individual web services, the community of web services also holds updated score rate value (as an average of score rates of all the involving web services). In service selection algorithm used by service consumer agents, the score rate of the community is taken into account, but master web service would decide how to cope with the service request. This means that the consumer agent cannot select the specific web service in the community to be served. However, upon task allocation, the consumer agent provides the post-interaction feedback regarding the corresponding web service(s) that provided the service. In this case, active web services in the community still update their individual score rates and influence the mean score rate value associated to the community as a whole.

We continue formalizing the attributes of rational services. In general, all rational entities, including users and web services, tend to maximize their efficiencies. To make this chapter focussed, we only consider the perspective of web services. Thus, we propose a heuristic (see Equation 2) for computing the efficiency E_x as a function f of Sr_x, M_x (market share introduced in Chapter 5) and Cp_x (capacity introduced in Chapter 4) where $x \in \{w_i, c_j\}$

$$E_x = f(Sr_x, M_x, Cp_x) \tag{2}$$

The function f should satisfy the following properties.

Property 7 *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 8 *f is strictly increasing in Sr_X and M_X.*

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

Property 9 *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 3 gives a possible definition of f.

$$f = \frac{Sr_x \times M_x}{|M_x - Cp_x| + 1} \tag{3}$$

Theorem 6.1 *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 $\frac{\partial f}{\partial Sr_x}$ 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, so we are done.

The other attribute that categorizes services is the risk factor S_X. This factor is denoted as how flexible the service is in loosing its efficiency. For example, if the risk

factor associated to 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 to an estimated efficiency less than $(1 - S_{w_i})E_{w_i}$.

The reason behind using the provider risk factor is the fact 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.

6.3.1 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: (a) w_i attempts to join c_j and the attempt is accepted; (b) w_i attempts to join, but c_j refuses the join request; (c) c_j invites the web service w_i but w_i refuses the invitation; and (d) 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 estimating 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 (a) The web service w_i that takes the risk of join (S_{w_i}) would update its score rate, market share and capacity parameters respectively in Equations 4 and 5, where n

denotes the current cardinality of the community set.

$$\overline{Sr_{w_i}} = \frac{n \times Sr_{c_j} + Sr_{w_i}}{n+1} \tag{4}$$

$$\overline{Cp_{w_i}} = Cp_{w_i} \qquad \overline{M_{w_i}} = \frac{M_{c_j} + M_{w_i}}{n+1} \tag{5}$$

In this case, our assumptions are as follows: (1) the score rate of a web service would be updated to the average of the community score rate. To this end, each registered web service in the community holds its individual score rate, but broadcasts the public score rate of the community; and (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 6 and 7.

$$\overline{Sr_{c_j}} = \overline{Sr_{w_i}} \tag{6}$$

$$\overline{C_{c_j}} = C_{c_j} + C_{w_i} \qquad \overline{M_{c_j}} = M_{c_j} + M_{w_i} \tag{7}$$

In this case, both entities consider the estimated parameters and compute their new efficiency values (see Equation 3). The case would take place when the following inequalities hold:

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

Case (b) In this case, w_i requests joining, but the community does not accept the request. The difference between the cases (a) and (b) is that in case (a) the join takes

place, which brings actual updated efficiency for both entities. However, in case (b) the join does not take place, which keeps the analysis at the estimation level. The corresponding estimated efficiencies are characterized by the following inequalities:

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

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

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

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

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

6.3.2 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 6.1 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

[h]

Table 6.1: Payoff regarding 2 players when web service is outside the community.

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}	$JR_{w_i}^{c_j}, JR_{c_j}^{w_i}$
Stay/Refuse Invitation	$SI_{w_i}^{c_j}, SI_{c_j}^{w_i}$	0 , 0

$$J_{w_i,c_j} = E'_{w_i} - E_{w_i}$$
$$A_{w_i,c_j} = E'_{c_j} - E_{c_j}$$
$$JR_{w_i}^{c_j} = E_{w_i} - \overline{E_{w_i}}$$
$$JR_{c_j}^{w_i} = E_{c_j} - \overline{E_{c_j}}$$
$$SI_{w_i}^{c_j} = E_{w_i} - \overline{E_{w_i}}$$
$$SI_{c_j}^{w_i} = E_{c_j} - \overline{E_{c_j}}$$

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 the 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 game between these entities in the following.

Proposition 6.2 *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 a 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 8. Equation 9 computes the related value regarding the refusal of join.

$$\alpha_{w_i}(join, c_j(S_{c_j}, 1-S_{c_j})) = S_{c_j}(J_{w_i,c_j}) + (1-S_{c_j})(JR_{w_i}^{c_j}) \tag{8}$$

$$\alpha_{w_i}(stay, c_j(S_{c_j}, 1-S_{c_j})) = S_{c_j}(SI_{w_i}^{c_j}) + (1-S_{c_j})(0) \tag{9}$$

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 10.

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

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

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

$$\Rightarrow \mu_{w_i} = \frac{(S_{w_i})(E'_{w_i}) + E_{w_i} - 3(S_{w_i})(E_{w_i})}{1 - 2S_{w_i}} \tag{10}$$

The threshold μ_{w_i} obtained in Equation 10 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 or accept the invitation to join strategy. We have then the following result.

Proposition 6.3 *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 6.4 *If the master web service considers the expected efficiency value computed by the web service and provides (broadcasts) a score rate that let $\overline{E_{w_i}}$ exceeds μ_{w_i}, the master can control adopting strategy of the web service.*

6.3.3 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 the 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 and the attempt is accepted; (b) w_i attempts to leave, but c_j refuses the leaving request; (c) c_j encourages (fires) the web service w_i, but w_i refuses the invitation; and (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 its score rate, market share and capacity parameters respectively in Equation 11

$$\overline{Sr_{w_i}} = Sr"_{w_i} \qquad \overline{Cp_{w_i}} = Cp_{w_i} \qquad \overline{M_{w_i}} = M"_{w_i} \tag{11}$$

In this case, our assumptions are as follows: (1) the score rate of a web service would be back to its previous individual score rate ($Sr"_{w_i}$). To this end, each registered web service in the community holds its individual score rate when joining a community. However, the community recalculates its average score rate; (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 12.

$$\overline{Sr_{c_j}}=\frac{n(Sr_{c_j}) - Sr_{w_j}}{n-1} \quad \overline{C_{c_j}} = C_{c_j} - C_{w_i} \quad \overline{M_{c_j}} = M_{c_j} - M_{w_i} \tag{12}$$

In this case, both entities consider the estimated parameters and compute their new efficiency values (see Equation 3). The case would take place when the following inequalities hold:

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

Case (b) In this case, w_i attempts to leave, but the community does not accept the

[h]

Table 6.2: Payoff regarding 2 players when web service is inside the community.

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

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

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:

$$\overline{E_{w_i}} \not\geq (1 - S_{w_i})E_{w_i} \qquad \overline{E_{c_j}} \geq (1 - S_{c_j})E_{c_j}$$

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

$$\overline{E_{w_i}} \geq (1 - S_{w_i})E_{w_i} \qquad \overline{E_{c_j}} \not\geq (1 - S_{c_j})E_{c_j}$$

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

$$\overline{E_{w_i}} \not\geq (1 - S_{w_i})E_{w_i} \qquad \overline{E_{c_j}} \not\geq (1 - S_{c_j})E_{c_j}$$

6.3.4 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 6.2 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 6.5 *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 6.2, 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 6.6 *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 6.7 *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.*

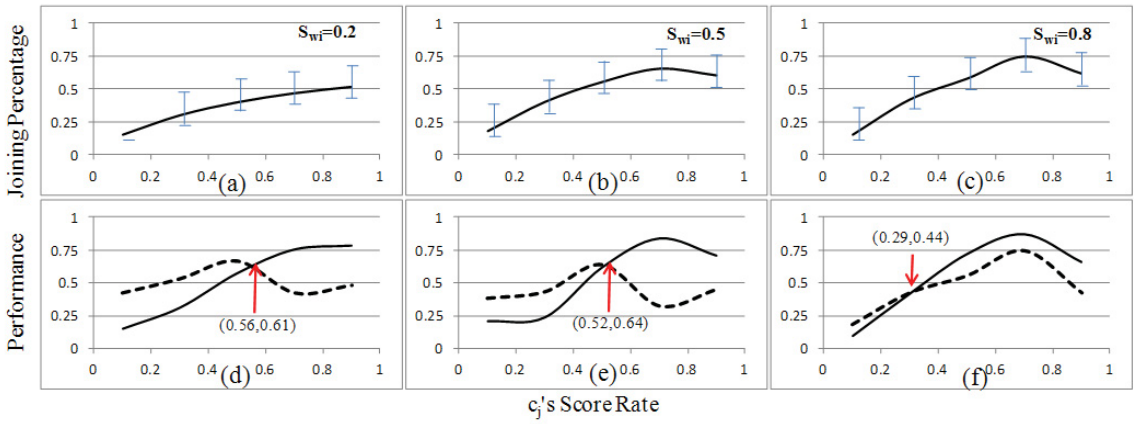


Figure 6.1: Efficiency of three categorized web services on joining a community.

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.

6.4 Empirical Analysis

We used a realistic multi-agent simulator in a java-based 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: (a) user agents; (b) web service agents; and (c) 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 web service (either from a single or a community of web services). During simulation runs, web services and users might leave or join the network. Table 6.3 provides the details regarding the implemented environment. We categorize the web services and

[h]

Table 6.3: Environment Characteristics.

Entity Type	Number	Risk Level	Environment Percentage
User	1000	---	86
Web service	50	0.2	4
	50	0.5	4
	50	0.8	4
Community	5	0.2	0.4
	5	0.5	0.4
	5	0.8	0.4

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 6.3.4 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 score rate 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 6.3.4, 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.

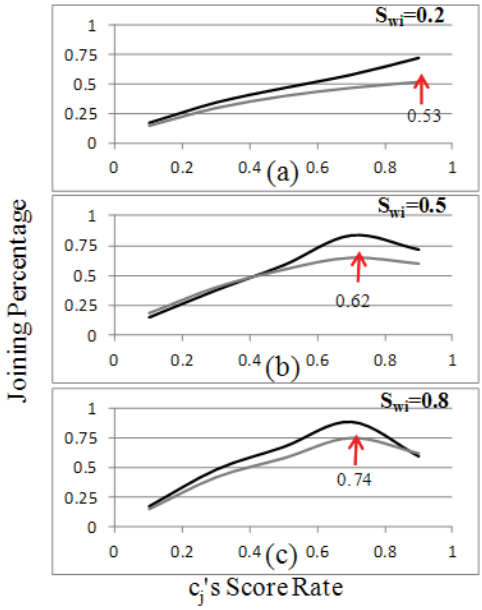


Figure 6.2: Efficiency of three categorized web services on joining a community while threshold being investigated.

We also lunch 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 6.4 plots the same group of web services (categorized in plots (a), (b), and (c)) facing a community whose master 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 score rate. 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

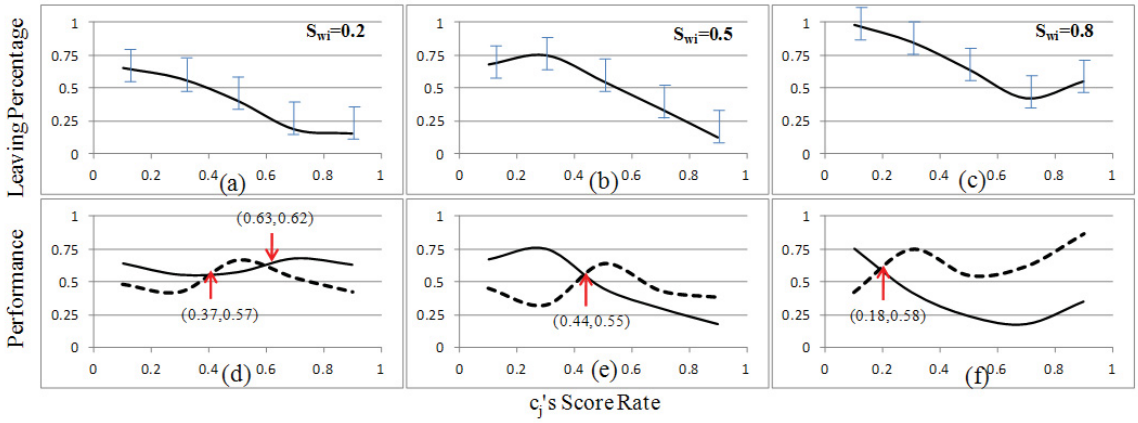


Figure 6.3: Efficiency of three categorized web services on leaving a community.

joined web services that are involved in one-stage game facing the community representative. Figure 6.4 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 score rate of the community. Note that the public score rate 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 score rate value. This chaotic percentage is regulated while the score rate of the community is increased. In this case, the web services consider to refuse the leave.

Figure 6.4 illustrates 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

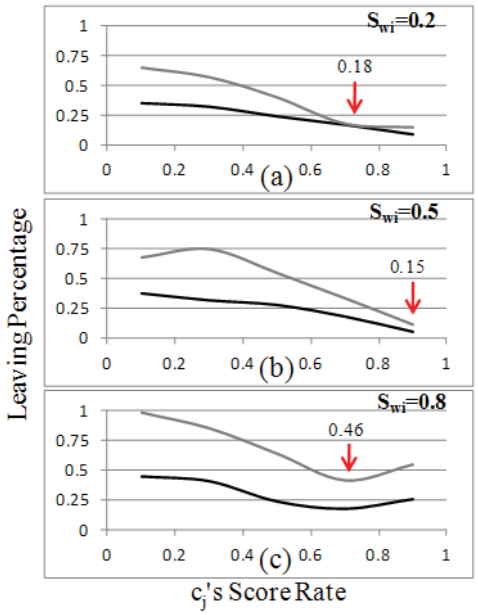


Figure 6.4: Efficiency of three categorized web services on leaving a community while threshold being investigated.

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 6.4, 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.

6.5 Related Work

In many frameworks proposed in the literature, service selection and task allocation are regulated based on the reputation parameter [69, 70]. In [2], the proposed framework regulates the service selection based on the trust policies expressed by the service

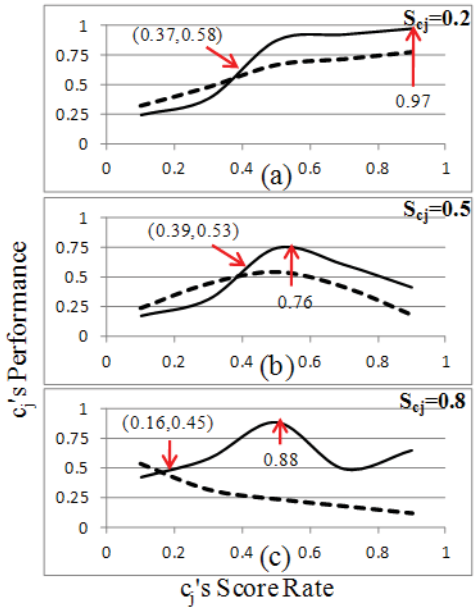


Figure 6.5: Efficiency of three categorized communities of web services.

users. In [54], authors propose ontology for quality of service. Users compute the web services' reputation using ratings. The frameworks proposed in [41, 57] address effective reputation mechanism for web services. All these models address the reputation in environments where Web services function alone. In such models, web service efficiency is not discussed in details and in general, balancing the market share with the capacity is not considered as an issue for web service besides its reputation.

There have been few work addressing the communities of web service. The objective is to facilitate and improve the process of web service selection and effectively regulate the process of request and task allocation [28]. In [49], authors propose a reputation-based architecture for communities and investigate the collusion scenarios that might falsely increase communities' reputation in the network. In [50], the authors mainly address the overall assessed reputation that is used as a main reason for service selection. The authors do not consider efficiency as a parameter that impacts service selection in future. 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.

6.6 Conclusion

The contribution of this model is the proposition of a game-theoretic based model to analyze the best efficiency characteristics for the active services in open networks. The proposed framework measures the efficiency of the web services considering a number of involved factors. The proposed game measures the threshold that lead to a control of strategies adopted by the single web service.

Our model has the advantage of being simple and taking into account four important factors: (1) rational services seek better efficiency in the environment; (2) in service computing the collaboration concept is well defined if the maximum efficiency is posed as the main goal; (3) rational web services might meet higher performance either by joining a community (for the sake of collaboration) or acting alone (for managing the task alone); and (4) the community is capable of managing the number of involving web services. The resulting model shows that the efficiency of the community is increasing once the game-theoretic analysis is considered to impose parameters to control the cardinality of the community set.

Chapter 7

Conclusions and Future Work

7.1 Summary

This thesis is about studying and analyzing trust and reputation in systems of autonomous agents. In Chapter 3, we proposed a probabilistic-based trust framework to secure multi-agent systems in which agents continuously communicate with each other. The trust assessment procedure is composed of on-line and off-line evaluation processes. The on-line process is based upon trustworthy and referee agents as well as several other features. Objectively, this allows enhancing the accuracy for agents to make use of the information communicated to them by other agents. The off-line process considers the communicated information to judge the accuracy of the consulting agents in the previous on-line trust assessment procedure using a maintenance process implemented as our optimization protocol.

Our trust model has the advantage of being computationally efficient and of taking into account four important factors: (1) the trust (from the viewpoint of the trustor agents) of the trustworthy agents; (2) the trust value assigned to trustee agents according to the point of view of trustworthy agents; (3) the number of interactions between trustworthy and trustee agents; and (4) the timely relevance of information transmitted by trustworthy agents. Addition process of maintenance enables agents to dynamically adjust their beliefs and their trustworthy community in a more efficient manner. The resulting model allows us to produce a comprehensive assessment of the agents'

credibility in a software system even if the environment is very biased. The proposed mechanism efficiency is compared with other related models and discussed in details to prove the capabilities of our framework. The proposed CRM model in Chapter 3 fulfils the first objective we had in Chapter 1: CRM is a flexible trust-based framework that accurately considers the involved factors and provides an optimum trust estimation process. Moreover, the agile adaptation of agents' goals and beliefs are considered in this framework as the system is supposed to be highly dynamic. We extended the CRM model proposed in [46] to a model that using trust, extends its connectivity in the social network [47]. In this model, we provide a detailed discussion over the network formation by taking into account the edge creation factors.

In Chapter 4, we proposed an incentive-based reputation model for open multi-agent systems modelled as communities of web services gathered to facilitate dynamic users requests. The reputation of the communities are independently accumulated in binary feedback reflecting the satisfaction of the users being served by the communities. The model represents a sound logging mechanism maintaining effective reputation assessment for the communities. The controller agent investigates the logging feedback released by the users to detect the fake feedback as a result of collusion between a community and a user (or a group of users), which are provided in support of the community. Upon detection, the controller agent maintains an adjustment in the logging system, so that the malicious community would be penalized by decreasing its reputation level.

Our reputation mechanism has the advantage of providing suitable metrics used to assess the reputation of a community. Moreover, having a sound logging mechanism, the communities would obtain the incentive not to act maliciously. The proposed mechanism efficiency is analyzed through a defined test-bed. The proposed reputation mechanism in Chapter 4 mainly addresses our general thesis goal which is to develop

and maintain strong reputation assessment procedures that optimally function in multi-agent systems with dynamic changes of environment attributes such as agent goals, credibilities, and population.

In Chapter 5, we continued the discussion on sound reputation mechanism by concentrating on agents' acting strategies and their willingness to obtain highest positive payoffs. The controller agent as the representative of the reputation system applies different penalties to constrain rational agents to adopt malicious actions as their dominant acting strategies. In this chapter, we investigated scenarios within which the controller agent overcomes the malicious activities and discourage agents to act maliciously. We studied the obtained results and analyzed the impact of the controller agent's imposed values on the payoffs associated to the interacting agents. The mechanism proposed in this chapter fulfils another objective of the thesis mentioned in Chapter 1: the sound reputation mechanism discourages malicious actions of the agents trying to increase self-reputation level and take advantage of open multi-agent system environment. This model is also extended to another work where we model and analyze the arrival of requests and study their impacts on the overall reputation [48]. The web services may be encouraged to handle the peak loads by joining to a group of web services.

In Chapter 6, we utilized game theory to analyze the best performance characteristics for active web service agents in open networks. The objective is to measure thresholds within which the control of adopting strategies by web service agents could be maintained. The model considers four important assumptions: (1) rational web service agents seek better performance in the environment; (2) in service computing the collaboration concept is well-defined if the maximum performance is set as the main goal; (3) rational web service agents might meet higher performance either by joining a community (for the sake of collaboration) or acting alone (for managing the task alone); and (4) the community is capable of managing the number of involving

web service agents. The resulting model shows that the performance of the community is increasing once the game-theoretic analysis is considered to impose parameters to control the cardinality of the community set. The game-theoretic analysis in this chapter mainly fulfils our last objective mentioned in Chapter 1: the proposed mechanism investigates the parameters yielding optimal performance of agents. Using this mechanism, we study the cases where selfish agents could obtain best payoffs using their decision making procedure. This model is also extended to another work where we discuss a mechanism which web services can use to join existing group of web services [45]. Moreover, we analyze the scenarios where the community is overloaded with web services that lied about their capabilities before joining.

7.2 Future Work

7.2.1 Trust Framework

Our objective for future work is to advance the assessment procedure to enhance the model efficiency using a comprehensive approach we developed in [44], which considers the trust issue as an optimization problem. We plan to enhance the efficiency of the trust framework in different aspects. The maintenance process is in general a learning methodology that updates the agents' beliefs with respect to environment changes. The information provided by the consulting agents reflect their behaviors and honesty and could be used in learning methodology to update the belief set about the surrounding environment. This helps agents quickly adapt with the environment changes and recognize the honest agents around them.

In the maintenance process, we can use game theory and mechanism design approaches to analyze the incentives agents can have to encourage them to be more accurate. In this framework, we need to investigate the cases where the consulting agents

do not accurately provide the information they have. In fact, the agents' incentives for truthful action should be analyzed. The purpose of the game theoretic analysis of this framework is to provide a methodology that agents can use for estimating their expected benefits of their further actions. This analysis would let agents identify the best strategies and act accordingly.

7.2.2 Reputation Mechanism

The second plan for future work is to advance the reputation mechanism developed in [49]. In the logging system, we need to optimize detection process, trying to formulate it in order to be adaptable to diverse situations. In the proposed reputation mechanism, the detection policy of the controller agent plays an important role, as the malicious actions are discouraged by accurate detection of the controller agent. We aim at designing a game with three players in which the consumer is also considered as a separate player together with the controller and provider agents.

In reputation mechanism, we are mainly aimed at establishing a sound reputation mechanism. Following this aim, we can extend our work in different directions listed in the following.

1. Consumer agents could be encouraged to only post truthful feedback and loose their payoffs in case of collusion or any misleading action that leads to temporary increase of one's obtained payoff. This could be maintained by applying game theory to analyze the behavior of consumer agents during interacting interval.

2. The controller agent could apply different learning methods to use the experiences obtained from previous malicious action detections to decrease the possibility and chance of false detection in future interactions.

3. The reputation control system could be re-defined as a Markov decision process

to systematically cope with observations and rationally use them to enhance the reputation system performance.

We can extend our framework to enhance the performance of the web service agents to obtain best payoffs in interactive networks. Furthermore, we plan to promote the concept of communities of web services by analyzing the performance of these communities with respect to their handling abilities of service consumers compared to that of web services that act lonely. In this analysis, we would like to provide a game-theoretic analysis of the benefit of a single web service that is capable of serving limited number of consumers and the incentives that encourage the web service to join a group of web services to increase self-performance. Examples of issues that still need to be investigated are: when to join, which community to join if more than one choice is available, which web service to hire/fire, and etc.

List of References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734-749, 2005.

[2] A.S. Ali, S.A. Ludwig, and O.F. Rana. A cognitive trust-based approach for web service discovery and selection. Proceeding of the 3'rd European Confference on Web Services, pp. 38-40, ECOWS 2005.

[3] C. Archibald, A. Altman, M. Greenspan and Y. Shoham. Computational pool: A new challenge for game theory pragmatics. AI Magazine 31(4):33-41, 2010.

[4] T. Barner-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, May 2011.

[5] D. Banerjee and S. Sen. Reaching pareto-optimality in prisoners dilemma using conditional joint action learning. Autonomous Agents and Multi-Agent Systems, 15(1):91-108,2007.

[6] J. Bentahar, F. Toni, J-J. Ch. Meyer and J. Labban. A Security framework for agent-based systems. The International Journal of Web Information Systems, 3(4):1102-1115, 2007.

[7] J. Bentahar and J-J. Ch. Meyer. A new quantitative trust model for negotiating agents using argumentation. The International Journal of Computer Science and Applications, 4(2):1-21, 2007.

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

[9] J. Bentahar, B. Khosravifar. Using trustworthy and referee agents to secure multi-agent systems. In Proceeding of the International Conference on Information Technology: New Generations, pp. 477-482, 2008.

[10] S. Breban. A coalition formation mechanism based on inter-agent trust relationships. First International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 306-307, AAMAS 2002.

[11] S. Buchegger and J-Y. L. Boudec. A robust reputation system for mobile ad-hoc networks. Technical Report IC/2003/50, EPFL-IC-LCA, 2003.

[12] A. Caballero, J. Botia, and A. Gomez-Skarmeta. A new model for trust and reputation management with an ontology based approach for similarity between tasks. Multi-Agent System Technologies, LNCS, 4196: 172-183, 2006.

[13] A. Danek, J. Urbano, A.P. Rocha, and E. Oliveira, Engaging the dynamics of trust in computational trust and reputation systems. In Proceeding of the 4'th International KES Symposium on Agents and Multi-Agent Systems Technologies and Applications, Gdynia, Poland, pp. 22-31, 2010.

[14] P. Dasgupta. Trust as a commodity. In Diego Gambetta, editor, Trust: Marking and Breaking Coopertaive Relations, Department of Socialogy, University of Oxford, electronic edition, pp. 49-72, 2000.

[15] A. P. Dempster. A generalization of Bayesian inference. Journal of the Royal Statistical Society, Series B (30)205-247, 1968.

[16] S. Elnaffar, Z. Maamar, H. Yahyaoui, J. Bentahar, Ph. Thiran. Reputation of communities of web services - preliminary investigation. Proceeding of the 22'nd IEEE International Conference on Advanced Information Networking and App., pp. 1603-1608, AINA 2008.

[17] A. Cheyer and D. Martin. The open agent architecture. Journal of Autonomous Agents and Multi-Agent Systems, 4(1):143-148, 2001.

[18] R.K. Dash, S.D. Ramchurn, and N.R. Jennings, Trust-based mechanism design. In Proceeding of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 748-755, 2004.

[19] C. Dellarocas. The digitization of word-of-mouth: promise and challenges of online feedback mechanisms. Management Science 49(10):1407-1424, 2003.

[20] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In Proceedings of the 2'nd ACM Conference on Electronic Commerce, pp. 150-157, EC 2000.

[21] H. Dong, J. He, H. Huang, and W. Hou. Evolutionary programming using a mixed mutation strategy. Information Sciences, 177(1):312-327, 2007.

[22] P.M. Dung, P. Mancarella and F. Toni. Computing ideal sceptical argumentation. Artificial Intelligence, 171(10-15):642-674, 2007.

[23] C. Ferguson and C. Gawargy. U(0,1) Two-person poker models. Game Theory and Applications, 12:17-37, 2007.

[24] T. Ferguson, L. Shapley and R. WeberNotes. Notes on a stochastic game with information structure. International Journal of Game Theory, 31, 223-228, 2003.

[25] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: enabling scalable virtual organization. The International Journal of High Performance Computing Applications, 15(3):200-222, 2011.

[26] E. Fourquet, K. Larson, and W. Cowan. A reputation mechanism for layered communities. ACM SIGecom Exchanges, 6(1):11-22, 2006.

[27] J. V. Gael and X. Zhu. Correlation clustering for crosslingual link detection. Proceedings of the 16'th International Joint Conference on Artificial Intelligence IJCAI07; pp. 1744-1750, 2007.

[28] M. Jacyno, S. Bullock, M. Luck, T.R. Payne. Emergent service provisioning and demand estimation through self-organizing agent communities. 8'th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 481-488, 2009.

[29] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. Decision Support Systems, 43(2):618-644, 2007.

[30] R. Jurca, F. Garcin, A. Talwar and B. Faltings, Reporting incentives and biases in online review forums. ACM Transactions on the Web (TWEB), 4(2):1-27, 2010.

[31] T.D. Huynh, N.R. Jennings and N.R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems 13(2):119-154, 2006.

[32] T.D. Huynh, N.R. Jennings and N.R. Shadbolt. Certified reputation: how an agent can trust a stranger. In Proceeding of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 1217-1224, 2006.

[33] A. Jesang and R. Ismail. The beta reputation system. In proceeding of 15'th Bled Electronic Commerce Conference e-Reality: Constructing the e-Economy, June 2002.

[34] I.A. Junglas, N.A. Johnson, D.J. Steel, D.Ch. Abraham and P. MacLoughlin. Identifying formation, learning styles and trust in virtual worlds, The DATA BASE for Advances in Information Systems, 38(4):90-96, 2007.

[35] R. Jurca and B. Faltings. Obtaining reliable feedback for sanctioning reputation mechanisms. Journal of Artificial Intelligence Research, 29:391-419, 2007.

[36] R. Jurca, B. Faltings, and W. Binder. Reliable QoS monitoring based on client feedback. In Proceeding of the 16'th International World Wide Web Conference, pp. 1003-1011, WWW 2007.

[37] R. Jurca and B. Faltings. Reputation-based service level agreements for Web services. In Proceeding of the International Conference on Service Oriented Computing (ICSOC 2005), Lecture Notes in CS, Volume 3826, pp. 396-409, 2005.

[38] O. Kafali, P. Yolum. Adapting reinforcement learning for trust: effective modeling in dynamic environments. Proceeding WI-IAT Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, 2009.

[39] O. Kafali, P. Yolum. Action-based environment modeling for maintaining trust. In 11'th International Workshop on Trust in Agent Societies, AAMAS, pp. 2332, 2008.

[40] M. L. Kahn and C. D. T. Cicalese. The CoABS grid. Goddard/JPL Workshop on Radical Agent Concepts, 2002.

[41] S. Kalepu, S. Krishnaswamy, S. W. Loke. A QoS metric for selecting Web services and providers. In Proceeding of 4'th International Conference on Web Information Systems Engineering Workshops, pp. 131-139, 2003.

[42] G. Kastidou, K. Larson, and R. Cohen. Exchanging eeputation information between communities: a payment function approach. Proceedings of the 18'th international joint conference on Artificial Intelligence IJCAI09; pp. 195-200, 2009.

[43] R. Kerr. Toward secure trust and reputation systems for electronic marketplaces. M.Sc Thesis, University of Waterloo, Canada, May 2007.

[44] B. Khosravifar, M. Gomrokchi, J. Bentahar, P. Thiran. Maintenance-based trust for multi-agent systems. In Proceeding of 8'th International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 1017-1024, AAMAS 2009.

[45] B. Khosravifar, J. Bentahar, K. Clacens, C. Goffart, and P. Thiran. Game-theoretic analysis of a web services collaborative mechanism. In Proceedings of the 9'th International Conference on Service Oriented Computing, pp. 549-556, ICSOC 2011.

[46] B.k Khosravifar, J. Bentahar, M. Gomrokchi, and R. Alam. CRM: an efficient trust and reputation model for agent computing. In Knowledge-Based Systems, Elsevier. DOI: 10.1016/j.knosys.2011.01.004, 2011.

[47] B. Khosravifar, J. Bentahar, and M. Gomrokchi. Declarative and numerical analysis of edge creation process in trust-based social networks. In M. Baldoni, J. Bentahar, J. Lloyd, and M.B. van Riemsdijk editors, Declarative Agent Languages and Technologies VII. Volume 5948 of Lecture Notes in Artificial Intelligence, pp. 141-161, Springer.

[48] B. Khosravifar, J. Bentahar, and A. Moazin. Analyzing the relationships between some parameters of web services reputation. In Proceedings of the 8'th IEEE International Conference on Web Services, Application and Industry Track, pp. 329-336, IEEE Press, ICWS 2010.

[49] B. Khosravifar, J. Bentahar, P. Thiran, A.Moazin, and A. Guiot. An approach to incentive-based reputation for communities of web services. In Proceeding of IEEE 7'th International Conference on Web Services, pp. 303-310, ICWS 2009.

[50] B. Khosravifar, J. Bentahar, A.Moazin, and P. Thiran. On the reputation of agent-based web services. In Proceeding of the 24'th AAAI Conference on Artificial Intelligence (AAAI), pp. 1352-1357, 2010.

[51] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness. Journal of AI Research (JAIR) (in Press), 2011.

[52] Z. Malik and A. Bouguettaya. Evaluating rater credibility for reputation assessment of web services. Proceeding of 8'th International Conference on Web Information Systems Engineering, pp. 38-49, WISE 2007.

[53] S. P. Marsh. Formalising trust as a computational concept. PhD thesis, University of Stirling, 1994.

[54] E.M. Maximilien, and M.P. Singh. Conceptual model of web service reputation. SIGMOD Record 31(4):36-41, 2002.

[55] E.M. Maximilien, M.P. Singh. Toward autonomic web services trust and selection. In Proceeding of the 2'nd International Conference on Service Oriented Computing, pp. 212-221, ICSOC 2004.

[56] E.M. Maximilien. Multiagent system for dynamic web services selection. The 1'st Workshop on Service-Oriented Computing and Agent-based Eng., pp. 25-29, SOCABE 2005.

[57] E.M. Maximilien, and M.P. Singh. Reputation and endorsement for web services, ACM SIGEcom Exchanges, 3(1):24-31, 2002.

[58] P. McBurney, S. Parsons and M. Wooldridge. Desiderata for agent argumentation protocols. In Proceeding of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 402-409, 2002.

[59] R. B. Myerson. Game theory analysis of conflict. Harvard University Press, 1991.

[60] F. Nie, Z. Zeng, I.W. Tsang, D. Xu, and C. Zhang. Spectral embedded clustering: a framework for in-sample and out-of-sample spectral clustering. IEEE Transactions on Neural Networks, 22(11):1796 - 1808 , 2011.

[61] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Agent-based formation of virtual organisations. Knowledge-Based Systems, 14(2-4):103-111, 2004.

[62] Organization for the advancement of structured information standards. Introduction to UDDI: Important features and functional concepts, October 2004. www.oasis-open.org.

[63] D.Ch. Parkes. Iterative combinatorial auctions: achieving economics and computational efficiency. Ph.D. Thesis, University of Pennsylvania, 2001.

[64] S. Parsons, P.J. Gmytrasiewicz, and M.J. Wooldridge (Eds.). Game theory and decision theory in agent-based systems. Springer, 2002.

[65] P. Tang and F. Lin. GUARDS Innovative application of game theory for national airport security. Proceedings of the 22'nd international joint conference on Artificial intelligence IJCAI11, pp. 2710-2716, 2011.

[66] T. Rahwan, T. Michalak, E. Elkind, P. Faliszewski, J. Sroka, M. Wooldridge, and N. Jennings. Constrained coalition formation. In The 25'th Conference on Artificial Intelligence (AAAI), USA. pp. 719-725.

[67] M. Raya, P. Papadimitratos, V. D. Gligor, and J-P. Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In Proceedings of the 27'th Conference on Computer Communications, pp. 1238-1246, INFOCOM 2007.

[68] K. Regan, T. Tran, and R. Cohen. Sharing models of sellers amongst buying agents in electronic marketplaces. In Proceedings of the 10'th International Conference on User Modeling Workshop on Decentralized, Agent Based and Social Approaches to User Modelling, UM 2005.

[69] S. Rosario, A. Benveniste, S. Haar, and C. Jard. Probabilistic QoS and soft contracts for transaction based Web services. IEEE International Conference on Web Services, pp. 126-133, ICWS 2007.

[70] M. Ruth and T. Shengru. Concurrency issues in automating RTS for web services. IEEE International Conference on Web Services, pp. 1142-1143, ICWS 2007.

[71] J. Sabater, M. Paolucci, and R. Conte, Repage: REPutation and ImAGE among limited autonomous partners. Journal of Artificial Societies and Social Simulation, 9(2), 2006.

[72] J. Sabatar. Trust and reputation for agent societies. Ph.D. thesis, Universitat autonoma de Barcelona, 2003.

[73] M. Sensoy and P. Yolum. Ontology-based service representation and selection. Journal of IEEE Transactions on Knowledge and Data Engineering, 19(8):843-857, 2007.

[74] M. Sensoy and P. Yolum. Experimental evaluation of deceptive information filtering in context-aware service selection. In Proceedings of the 11'th International Workshop on Trust in Agent Societies, pp. 153-165, 2008.

[75] M. Sensoy, J. Zhang, P. Yolum, and R. Cohen. Poyraz: context-aware service selection under deception. Computational Intelligence, 25(4):335-366, 2009.

[76] G. Shafer. A Mathematical Theory of Evidence. Princeton University Press, 1976.

[77] E. Shakshuki, L. Zhonghai, and G. Jing. An agent-based approach to security service. International Journal of Network and Computer Applications, 28(3):183-208, 2005.

[78] C. Shen, T. Li, and C. H. Ding. Integrating clustering and multi-document summarization by bi-mixture probabilistic latent semantic analysis (PLSA) with sentence bases. Proceeding of the 25'th AAAI Conference on Artificial Intelligence (AAAI), pp. 914-921, 2011.

[79] J. Shi, G.V. Bochmann and C. Adams. A trust model with statistical foundation. IFIP International Federation for Information Processing, vol 173, pp. 145-158, 2005.

[80] Y. Shoham. Game theory pragmatics: a challenge for AI. Proceeding of the 22'nd AAAI Conference on Artificial Intelligence (AAAI), pp. 1606-1608, 2008.

[81] M. Tambe and B. An. Game Theory for Security: A real-world challenge problem for multiagent systems and beyond, AAAI Spring Symposium on Game Theory for Security, Sustainability and Health, 2012.

[82] P. Tang and F. Lin. Discovering theorems in game theory: two-person games with unique pure nash equilibrium payoffs. Proceedings of the 18'th international joint conference on Artificial intelligence IJCAI09; pp. 312-318; 2009.

[83] W.T. Teacy, J. Patel, N.R. Jennings, and M. Luck. Travos: trust and aeputation in the context of inaccurate information sources. Journal of Autonomous Agents and Multi-Agent Systems, 12(2):183-198, 2006.

[84] T. Tran and R. Cohen. Improving user satisfaction in agent-based electronic marketplaces by reputation modeling and adjustable product quality. In Proceedings of the 3'rd International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 828-835, AAMAS 2004.

[85] B.Y. Qu, P.N. Suganthan. Multi-objective evolutionary programming without non-domination sorting is up to twenty times faster. In Proceeding of IEEE Congress on Evolutionary Computation, pp. 2934-2939, CEC 2009.

[86] P.B. Velloso, R.P. Laufer, M.B. Duarte and G. Pujolle. HIT: A human-inspired trust model. IFIP International Federation for Information Processing, vol 211, pp. 35-46, 2006.

[87] M. Vijsel , J. Anderson. Coalition formation in multi-agent systems under real-world conditions. Proceeding of the 18'th AAAI Conference on Artificial Intelligence (AAAI), 2004.

[88] Y. Wang, and M.P. Singh. Formal trust model for multiagent systems. Proceeding of the International Joint Conference on Artificial Intelligence, pp. 1551-1556, 2007.

[89] Y. Wang and J. Vassileva. Bayesian network-based trust model. In Proceedings of the 6'th International Workshop on Trust, Privacy, Deception and Fraud in Agent Systems, pp. 372-380, 2003.

[90] Y. Wang and J. Vassileva. Trust-Based Community Formation in Peer-to-Peer File Sharing Networks, Proc. of IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004), pp. 341-348, China.

[91] L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering, 16(7):843-857, 2004.

[92] W. Yao, J.Vassileva. A Review on trust and reputation for web service selection. 1'st International Workshop on Trust and Reputation Management in Massively Dis. Comp. Sys., pp. 22-29, TRAM 2007.

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

[94] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 3(2):82-109, 1999.

[95] Y. Ye and Y. Tu. Dynamics of coalition formation in combinatorial trading. Proceedings of the 18'th International Joint Conference on Artificial intelligence IJCAI09; pp. 625-631; 2003.

[96] P. Yolum and M.P. Singh. Engineering self-organizing referral networks for trustworthy service selection, IEEE Transaction on Systems, Man, and Cybernetics, 35(3):396-407, 2005.

[97] B. Yu and M.P. Singh. An evidential model of distributed reputation management. Proceeding of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 294-301, 2002.

[98] B. Yu and M.P. Singh. Detecting deception in reputation management. Proceeding of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 73-80, 2003.

[99] B. Yu and M.P. Singh. Searching social networks. Proceeding of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 65-72, 2003.

[100] B. Yu and M. P. Singh. A social mechanism of reputation management in electronic communities. In Proceedings of the 4'th International Workshop on Cooperative Information Agents, pp. 154-165, 2000.

[101] G. Zacharia and P. Maes. Trust management through reputation mechanisms. Applied Artifitial Intelligence, 14(9):881-908, 2000.

[102] J. Zhang, R. Cohen. An incentive mechanism for eliciting fair ratings of sellers in e-marketplaces. In Proceeding of the 6'th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 108, 2007.

[103] J. Zhang, R. Cohen, and K. Larson. A Trust-based incentive mechanism for E-Marketplaces. Lecture Notes in Artificial Intelligence Special Issue on Trust in Agent Societies, pp. 135-161, 2008.

[104] J. Zhang and C. Zhang. Multitask bregman clustering. Proceeding of the 24'th AAAI Conference on Artificial Intelligence (AAAI), pp. 655-661, 2010.

[105] W. Zhang. Cpmputational trust model in online auctions. Wireless Communications, Networking and Mobile Computing, WiCom 2007, pp. 3762-3765, 2007.