Integrating Behavioral Trust and Reputation in Web Service Compositions

by

Sharon M. Paradesi

(Under the direction of Prashant Doshi)

Abstract

Algorithms for composing Web services traditionally utilize the functional and quality-of-service parameters of candidate services to decide which services to include in the composition. Users often have differing experiences with a Web Service. While trust in a Web Service is multi-faceted and consists of security and behavioral aspects, the focus in this thesis is on the latter. A formal model for trust in a Web Service is adopted, which meets several intuitions about trustworthy Web Services. In order to make the system robust an additional model to identify the reputation of users that provide feedback is introduced. Predictors of a positive experience with a Web Service and positive reputation of a user are hypothesized. A small pilot study is conducted to explore correlations between subjects' experiences with Web Services in a composition and the predictor values for those Web Services. Furthermore, the method to derive trust for compositions from trust models of individual services is presented. A novel framework, called Wisp, is presented and evaluated that utilizes the trust models and, in combination with any Web Service composition tool, chooses compositions to deploy that are deemed most trustworthy.

Index words: Trust, Reputation, Web Service, Web Service Composition

Integrating Behavioral Trust and Reputation in Web Service Compositions

by

Sharon M. Paradesi

B.Tech., Jawaharlal Nehru Technological University, 2007

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2009

Integrating Behavioral Trust and Reputation in Web Service Compositions

by

Sharon M. Paradesi

Approved:

Major Professor: Prashant Doshi

Committee: John Miller
Khaled Rasheed

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2009

Dedication

Dedicated to my mother, Dr. Mary Prathibha Gollapalli, my father, Mr. Suresh Rao Paradesi and my brother Mr. Martin Samuel Rao Paradesi.

Acknowledgments

I thank my advisor Dr. Prashant Doshi for guiding my research direction. I also thank Dr. John Miller & Dr. Khaled Rasheed for serving on my Master's committee. I am grateful to them for their valuable suggestions and comments regarding my thesis.

I thank my mother, father and brother for their love, support and encouragement. Above all, I thank my God for His grace, mercy, faithfulness and love. Last but not the least, I thank all my friends in the city of Athens, GA.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem

Web Service Composition techniques traditionally utilize the functional and quality-of-service (QoS) parameters of candidate services to decide which services to include in the composition[9]. However, users of services often form an opinion, somewhat subjective, of a service. This opinion may be based on prior interactions with the service, and may include judgments such as whether the perceived behavior of the service conforms to its stated behavior and intangibles such as the overall experience of the user with the service. Web Service Composition techniques that additionally consider this assessment of services by users will form compositions that likely behave in practice as stated, and which are better received by the users.

Trust in a Web Service is multifaceted and includes a security based aspect such as establishing the authenticity and authority of the service, and behavioral (social) aspect such as judging whether the service behaves reliably and as advertised. One focus of related research has been on designing protocols for establishing trust through authentication, as introduced in the Web Service trust standard[5]. In contrast, the focus of this thesis is on studying the behavioral aspect of trust. The notion of trust in a service is mathematically formalized in this work.

A Web Service is considered to be trustworthy if the probability of having a positive experience with the Web Service is high. On the other hand, low belief is assigned to high probabilities of a positive experience with a Web Service and thus makes it untrustworthy. Given that belief is often formalized as a probability distribution[31], trust may be modeled

using a second order probability density function (sometimes called a probability certainty density function (PCDF)[20]). This stochastic model for trust follows the model of Wang & Singh[36] which targets general agents, and which is itself a modication of Jøsang's[20].

It is hypothesized that a positive experience of users with a Web Service correlates with its perceived honesty, reliability and competency. It is possible to objectively measure these variables of a service participating in a Web Service Composition, thereby allowing to test any hidden dependency between users' subjective opinions and objective measurements. In this context, a pilot study is conducted to map the experiences of users (graduate students) interacting with services in example Web Service Compositions to the objective measurements of the three aspects for individual Web Services. Data from this study is used in testing the hypothesis and in formulating different user preference models, which are further utilized in later experimentation.

The belief is that individual Web Services are progressively updated over time based on experiences of multiple users, thereby following a similar approach to Wang & Singh[36]. Although trust could be treated as an additional QoS parameter, this would require that existing Web Service Composition techniques be adapted to account for trust while forming the composition. To avoid this, a separate trust framework is developed, called Wisp, that computes the aggregate trust in the composition, and selects the composition that is most trustworthy. This is additionally useful because the composed service may itself be used as a component service later.

In order to make the system robust, an additional model to identify the reputation of users that provide feedback is developed. Reputation of a user is used to identify whether the system should rely on the feedback of the user in computing the new trust of a Web service composition. If the user is reputable, Wisp considers the feedback as provided. Otherwise, it factors the reputation of the user to minimize the impact of the questionable user. It is hypothesized that the positive reputation of a user in the system correlates with the perceived expertise and honesty of the user.

A method for comparing the different Web Service Compositions based on their derived trustworthiness is presented along with a report on supporting experimentation. Other experiments show how the system mitigates the effect of disreputable feedbacks.

1.2 Contributions

This thesis has a three-fold contribution to the field of trust and reputation systems, especially in the area of Web service. They are:

1. Providing a comprehensive model to represent trust and reputation. This model meets most of the intuitive expectations of trust and reputation in real world.

2. Designing a framework, Wisp, that utilizes the mathematical models. Wisp ensures that only those Web Services that behave as advertised in their service level agreements are chosen for execution. Furthermore, feedbacks from user that are not reputable are discounted by their reputation score in order to shield the system from dishonest & naïve users.

3. Showing the validity of Wisp, with the help of the following two experiments: (i) Ability to select services that perform as advertised, & (ii) Ability to mitigate the effect of disreputable users in the system.

1.3 Structure of Thesis

The document is organized as follows. Chapter 2 outlines related work by different researchers about trust and reputation. Since trust and reputation are broad topics studied in various fields, general definitions and reasons for considering trust are provided, followed by trust and reputation models in Web services, multi-agent systems and Semantic Web. Chapter 3 describes the trust and reputation models and the concepts used to arrive at these models. Chapter 4 describes the components of a framework, called Wisp, for integrating the trust and reputation models in Web service compositions. Chapter 6 demonstrates using three sets of experiments to prove the validity and utility of Wisp. Finally, this document is concluded in chapter 7.

Chapter 2

Background & Related Work

Trust and reputation are studied in various disciplines such as Psychology, Sociology and Computer Science. To remain within the scope of Computer Science, this chapter explains these concepts in the context of Web service compositions, Semantic Web and Multi-agent systems.

2.1 Trust

Trust has been studied in many fields and a vast literature exists to explain trust from different perspectives. However, in order to remain within the scope of this thesis, only the previous characterizations of trust are provided. These characterizations of trust are followed by two trust models in the scope of Web Services and in Web Service Composition respectively.

2.1.1 Social definitions of Trust

Gambetta[12] defines trust as a particular level of the subjective probability with which an agent performs a particular action, before it can monitor the action and in a context in which it affects others actions.

McKnight & Chervany[26] characterize a trusting intention as the extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible.

Olmedilla *et al.*[28] define trust in the field of Web Services as the measurable belief that provider behaves dependably for a specified period within a specified context in relation to a Web Service.

Mui *et al.*[27] refer to past encounters and state that trust is a subjective expectation an agent has about another agent's future behavior based on history of their encounters.

The characterization presented in this thesis is loosely based on all the four definitions. The aforementioned definitions are vague about the characteristics of the trusted entity, which makes the trusting entity trust it. In this work, the experiences with Web Services are grounded along three axes: honesty, reliability and competency. These parameters are orthogonal and sufficient for a comprehensive meaning of the notion of a positive experience in many contexts. Trust is also often classified into direct trust; in which a user has on another user or service, and recommender trust[1] and [6]; in which a user has on another user or service that recommends others. The focus of this thesis is on modeling direct trust. The task of pursuing recommender trust is one of the several avenues for future work.

2.1.2 Dimensions of Research on Trust

While studying trust, there are many aspects to keep in mind. Prominent among them are the following three criteria:

1. Target: The entity being evaluated as trustworthy or not. Usually, agents or services are the entities being evaluated. In case of centralized systems, the central authority (determining trust of entities) is assumed to be trustworthy.

2. Representation: There are many ways that trust can be digitally encoded. In the security aspect of trust, credentials that include digital signatures and tokens can be used. In the social aspect of trust, histories of past interaction with other agents or transfer of trust from associated entities are commonly used to evaluate the trust of an entity. In Semantic Web, detailed ontologies can be used for trust policies, trust negotiations and data provenance.

3. Computation: Computation depends on the type of trust under consideration. There are two broad categories of trust: Policy-based and Reputation-based which are explained below:

(i). Reputation-based trust: This research focuses on using user feedback of past experiences to judge the reputation, and hence, the trust of the entity being evaluated. Usually, a history of the entity's actions/behavior is stored along with recommendations from other entities. In order to implement transitive trust and trust along a chain of network connections, semantic graphs can be used where the nodes represent the entities and the edges represent the relationships among them. The proximity between the entities can be shown using weights. Some of these frameworks implement techniques to maintain a dynamic value of trust (changing with time) and provide for a local computation of trust (the trust of an agent might be different based on which associated agent is being queried from) which leads to decentralized or distributed trust models.

(ii). Policy-based trust: This research focuses on using policies to implement trust by managing and exchanging credentials and enforcing access policies. Work in policy-based trust generally assumes that trust is established simply by obtaining a sufficient amount of credentials pertaining to a specific party, and applying the policies to grant that party certain access rights.

2.1.3 Why Trust?

The need for trust spans all aspects of computer science, and each situation places different requirements on trust. Human users, software agents, and increasingly, the machines that provide services all need to be trusted in various applications or situations. The communication channels between computers and users, and the content exchanged between computers and users also require trust, in both directions, for real world use. Trust can be used to protect data, to find accurate information, to get the best quality service, and even to bootstrap other trust evaluations. Trust may be better seen as a motivating concept underlying

many problems and contexts rather than as a precise idea to be studied under a uniform framework.

The need for trust is different in different fields of computer science. The rest of this section details the significance of trust in the three areas of consideration: Web Services, Multi-Agent Systems and Semantic Web.

1. In Web Services:

Users of services often form an opinion, somewhat subjective, of a service. This opinion may be based on prior interactions with the service, and may include judgments such as whether the perceived behavior of the service conforms to its stated behavior and intangibles such as the overall experience of the user with the service. Also, web services have to automatically determine to which extent they may trust other services to provide the required functionality, before they interact with them. Web Service Composition techniques that additionally consider this assessment of services by users will form compositions that likely behave in practice as stated, and which are better received by the users.

2. In Multi-Agent Systems: In dynamic open systems, many agents interact with each other to achieve their goals. Agents work on the data/services made available by other agents. In such environments, a self-interested agent selects the most trusted and suitable partners to interact with from a pool of agents whose behaviors are not known. However, by itself, an agent cannot say which agent can be trusted and which one cannot (that is, agent collaborations or service compositions do not scale well because it is difficult to build a secure "web of trust"). Ideally, an agent should interact with the agent who most probably fulfills the expectations of the requester agent. Trust model consists of opinions of an agent about other agents; it's formed by using its own experience with the related agent and other agents; opinions about the related agent. Each agent builds its own trust model and uses this model to decide on whom to trust.

3. In Semantic Web: Since the articulation of the Semantic Web vision, it has become the focus of research on building the next web. The philosophy behind the Semantic Web

is the same as that behind the World-Wide Web - anyone can be an information producer or consume anyone's information. However, conflicting information from distinct sources is given, forcing us into a decision about what information to accept on the web. Our decision in these cases is sometimes reduced to picking the more highly trusted information source. Despite many standards for communicating facts, rules and ontologies like XML, RDF(s) and OWL, we still need to address the major issue of how to decide how trustworthy each information source is. One solution would be to require all information on the Semantic Web to be consistent and of high quality. But due to its sheer magnitude and diversity of sources, this will be nearly impossible. Much as in the development of the WWW, in which there was no attempt made to centrally control the quality of information, we believe that it is infeasible to do so on the Semantic Web.

2.1.4 Trust in Web Services

In the area of Web Services, the Web Service Trust Language[5] aims to enable applications to construct trusted message exchanges. It is assumed that the concerned participants are legitimate providers or requesters and no verification on authentication mechanisms are done. Liu *et al.*[24] introduced a dynamic Quality-of-Service(QoS) computation model by means of a central QoS registry. Their broker architecture is human-oriented, i.e., consumer Web Services give feedback to provider Web Services.

Solely relying on user feedback has two disadvantages. The first disadvantage is that different users may have different opinions and a service preferable to one consumer might not appear similar for another. The second disadvantage is that some users might give a deceptively negative or positive feedback on purpose. Thus, a way to correlate the subjective feedback of users with objective measures is beneficial. Adam *et al.*[3] introduce a trust index for a service provider and requester, which is dynamic and propagated throughout the environment. They provide means of determining if a service provider or requestor violates the Web Service usage policies specified.

In this work, trust is associated with a particular service and not with the service provider as mentioned in Adam *et al.*[3]. The intuition behind this approach is based on the fact that users interact with a service and providers may have both low and high quality services.

Abdul-Rahman and Hailes [1] propose a trust model that is grounded in real-world social trust characteristics, and based on a reputation mechanism, or word-of-mouth. They define define two types of Trust, viz. Direct and Recommender Trust. While Direct Trust represents an entity's opinion of another one after interacting with it, Recommender Trust signifies the trust of the first entity on another that recommends additional entities to the first one. They project trust as a discrete variable with five levels - very trustworthy,trustworthy, untrustworthy and very untrustworthy.

Malik and Bouguettaya [25] introduce RATEWeb, a framework for establishing trust in service-oriented environments to facilitate trust-based selection and composition of Web services. RATEWeb supports a cooperative model in which Web services share their experiences of the service providers with their peers through feedback ratings. The different ratings are aggregated to derive a service provider's reputation. This in turn is used to evaluate trust.

Credibility of raters is taken into consideration to weigh the feedback by the raters. A service consumer's credibility determines how much other service consumers may trust its reported ratings regarding the reputation of the Web services it has invoked. These credibility scores are updated based on past rating history of the rater and also based on the majority feedback (that is, the feedback of the community of service consumers). The authors also define an ontology for community creation and service registration at RATEWeb as shown below:

Wang and Singh [36] define trust in a probabilistic way. Their model distinguishes trust from distrust and also a measure of trust and distrust through the concept of certainty. These are achieved by representing the trust space as a three-tuple vector - ¡t,d,u¿ where t stands for trust, d for distrust and u for uncertainty. The procedure to obtain the trust vector is as follows: First, certainty is computed from the present evidence reports and Second, the

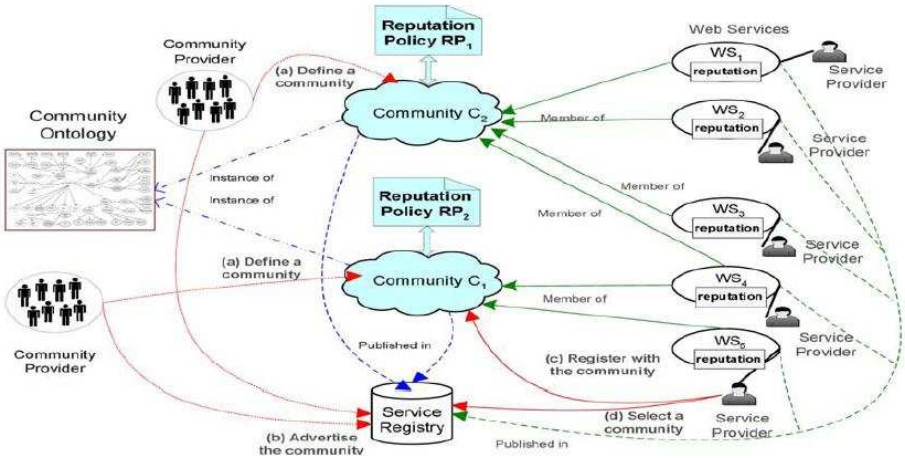


Figure 2.1: RATEWEB Ontology

evidence is converted to trust space. A trust space is defined by a trust vector which is a triple - trust, distrust and uncertainty. With more evidence, certainty increases and an increase in conflict decreases the certainty.

Josang, Hayward and Pope [21] define a subjective logic view of trust networks. They view trust as a transitive phenomena and explain how to aggregate feedback from different sources in order to utilize a parallel trust combination. Their model uses a binary rating system and aggregates the feedback to achieve trust value. The model also considers aging because over time, agents may change their weight to more recent ratings. Trust reasoning is explained using two concepts: discounting and consensus. Discounting is the technique where derived trust is calculated via a chain of referrers. Consensus is used to fuse two (possibly conflicting) beliefs into one.

Mui, Mohtashemi and Halberstadt [27] define a computational model for trust in agents(specifically, e-Businesses) . They define three main concepts: reciprocity, reputation and trust. Reciprocity is defined as the mutual exchange of deeds (such as favor or revenge). Reciprocity norms pervade the human society and refer to social strategies that individuals learn which prompt them to react to the positive actions of others with positive responses

and negative actions of others with negative responses. Reputation is the perception that an agent creates through past actions about its intentions and norms. Trust is a subjective expectation an agent has about another agent's future behavior based on the history of their encounters.

2.1.5 Trust in Web Service Composition

Singh[34] discusses the formation of a trust network among agents who provide services to each other, agents help each other locate trustworthy services to include in a service composition. His approach focuses on settings leading to service compositions. However, he does not provide a concrete model of trust beyond a discussion of its desirable properties. The work presented in this thesis is more focused on identifying trustworthy compositions.

Xu *et al.*[38] propose a framework for reputation-enhanced QoS-based Web Service discovery that extends the UDDI registry to publish the QoS information of Web Services. Their framework uses a reputation manager to assign reputation scores to the Web Services based on customer feedback. The overall score given to a service is a weighted sum of the QoS score, which reflects a dominant QoS attribute, and adjusted reputation score.

A significant difference of the approach used in this thesis when compared with that mentioned in Xu *et al.*[38] is the focus on building trusted Web Service Compositions, instead of the focus on discovering reputable Web Services. Another distinction in this work is to update the trust using a Bayesian approach, instead of storing all previous reputation scores. This approach has two advantages: it saves the storage overhead over time and allows flexibility in the length of the history of reputation scores to be considered.

Paradesi, Doshi and Swaika [29] hypothesize three predictors of a positive experience with a Web Service: competency, reliability and honesty. They conduct a pilot study to explore correlations between subjects experiences with Web Services in a composition and the predictor values for those Web Services. They also show how to derive trust for compositions from trust models of individual services. Their approach is similar to Wang and

Singh's approach [36] by computing the certainty of trust/distrust of a Web Service and then computing the trust vector. They use four constructs (sequence, concurrent, conditional and loop) to demonstrate how to extend the trust model to Web Service Compositions.

2.1.6 Trust in Multi Agent Systems

Hafizoglu and Yolum [16] explain a technique to see how trustworthy a group of service providers are to form a composite service. This is achieved by capturing the relations between the services. The relations are viewed in the form of a service graph where the nodes are services and the edges are interactions between the services. Composite services are also depicted as nodes in the graph. By using this relationship between different composite services, the agent composes a new group of agents for a given service. The authors call this composite trust Group Trust and explain that it is vital to understand how the agent behaves because individual trustworthiness is not gong to be enough to understand the trustworthiness of the team.

Six possible tendencies of agents are examined: Ideal Behavior (agent behaves well both as an individual and in a team), Group Antipathy (agent dislikes being in a team) Group Motivation (agent performs better when in a team), Colleague Effect (agent's behavior depends on the teammates interacting with), Teamwork Effect (agent's performance depends on the task characteristics) and Familiarity Effect (agent's performance increases as the number of interactions with teammates increase).

Hang and Singh [18] propose a distributed trust-aware service selection model based on a Bayesian network for consumers to maintain their knowledge of the environment locally. Through their experiments they show that their model can punish and reward services in terms of Quality of Service properties accurately with incomplete observations so that consumers can prevent themselves from interacting with services with unsatisfying Quality of Service. Given the acyclic service graph of the services to be used in a composition, an associated joint distribution is found over the probability of getting a good service from that

service. In order to compensate for incomplete information available to any agent at a particular instant of time, the authors propose to use an Expectation-Maximization technique to arrive at a more accurate value of trust. The E step deals with estimating latent variables, which are variables that cannot be observed. The now-complete data is used as an observation in the M step to update the parameter estimation by Bayesian inference.

Keung and Griffiths [23] propose a mechanism for agents to build a representation of agents' local environment based on direct interactions, trust and reputation. Their model shows that by building a network of agents an agent interacts with, and with information about interaction details, trustworthiness, recommendation chain and reputation, the agent is in a better position to extract emergent information, such as potential new customers, suppliers, its competitors and potentially collusive groups of agents. Their model consists of three main components: data collection, network building (by constructing three types of graphs: provider graph, witness graph and combined provider-witness graph) and analysis of interaction data. The focus of their analysis is in detecting collusion among agents.

Collusive behavior is characterized by heavy interactions among colluding agents and/or similar responses to queries as witnesses of interactions. The authors' approach to detect collusion is to compare which agents recommend certain agents all the time. Such a scenario would lead us to believe that similar-rating agents are colluding to either increase or decrease the trust of the target agent. The paper specifically detects two types of collusions: collusion between target and witness (that is, between the end node and an intermediate/start node in the service graph) and collusion between witnesses (that is, between two intermediate nodes or between the start and an intermediate node in the service graph). These two types of collusion are illustrated below:

Sen, Malone and Chakraborthy [33] demonstrate learning techniques for balancing exploration and exploitation in a procurement domain (that is, purchasing agents trying to procure required amounts of needed items). The goal of agents in this domain is to trade with a set of service providers to fulfill requirements with minimizing cost. A myopic approach would be

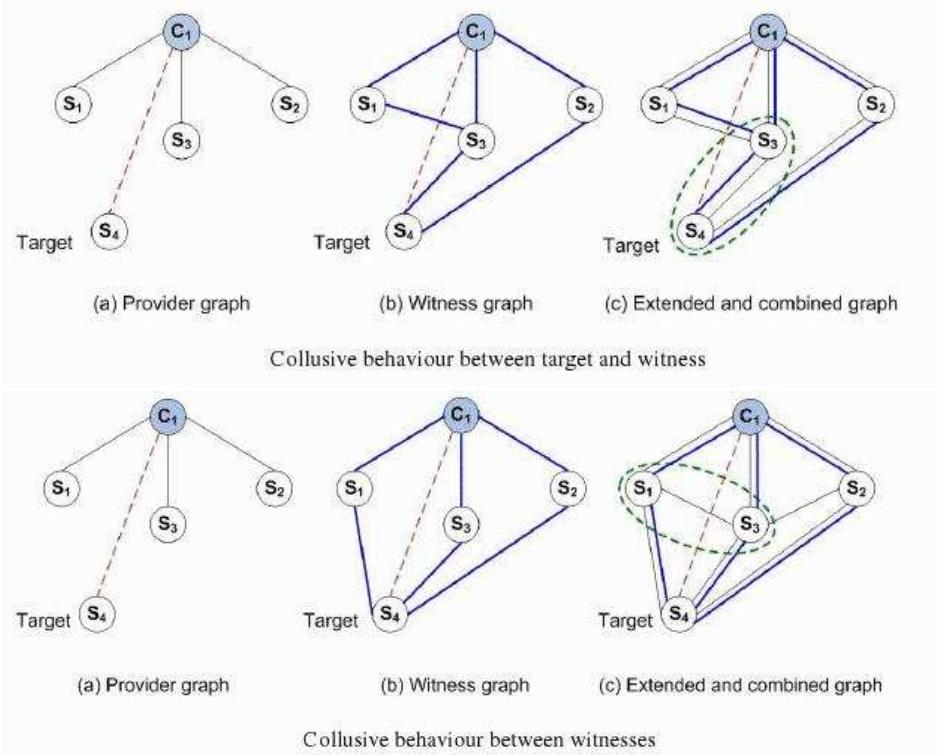


Figure 2.2: Collusions among agents

to develop initial estimates of the existing service providers through random sampling and then contracting with only the most profitable service providers. However, this approach is not useful due to the high turnover in the service provider population. The authors' solution to this dilemma is to use Engagement decision modules. They enable an agent to utilize the expected utilities to minimize cost while still fulfilling the requirement. These modules can be used by varying strategies of agents viz. Risk Averse, Risk Neutral and Risk Seeking. These trust engagement decision functions can be used by agents to create interaction opportunities to better evaluate trustworthiness of possible trading partners.

Fullam and Barber [11] present a technique for dynamically learning the best source of trust information. Since both the trustworthiness characteristics of potential trustees and reputation accuracy may change quickly, an agent must adapt its own trust decisions to

continually achieve maximum payoff. The authors use reinforcement learning to select trust models via Markov Decision Processes. The truster first decides whether to trust or not to trust. Not trusting yields a net payoff of zero to both the trustee and truster, since no fundamental transaction takes place. If the truster chooses to trust, it pays a value (Pr) to the trustee. The trustee then chooses a resource of value (Pe)to return to the truster. The trustee is considered trustworthy in the transaction if Pe-Pr¿0. Otherwise, the trustee is considered to be untrustworthy. The demonstrated learning technique achieves payoffs equal to those achieved by the best single trust information source in nearly every scenario examined.

Conner et al. [8] develop a reputation-based trust management framework that supports the synthesis of trust-related feedback from many different entities while also providing each entity with the flexibility to apply different sorting functions over the same feedback data for customized trust evaluations. They also propose a caching strategy to cache trust values based on recent client activity. Malicious client behavior is addressed by services providing negative feedback rating on previous interactions with those clients.

2.1.7 Trust in Semantic Web

Katz and Golbeck [22] provide a coupling between the method for computing trust values in social networks and the prioritized Reiter defaults.They also compare their approach with specificity-based prioritization and mention how the two approaches can be combined. The authors define an algorithm for inferring trust called TidalTrust. By incorporating path length and trust values, the algorithm can deduce how much an agent should trust another agent with which it had never interacted with previously. The algorithm is a modified breadth-first search where calculations sweep forward from source to sink in the network and then pull back from the sink to return the final value to the source.

Richardson, Agarwal and Domingos RAD employ a web of trust, in which each user maintains trusts in a small number of other users. They then compose these trusts into trust

values for all other users. The aim of the research is to let each user maintain a personalized set of trusts, which may vary widely from person to person. The assumption here is that the content on the Semantic Web is (explicitly or implicitly) in the form of logical assertions. If all these assertions are consistent and believed with certainty, a logical calculus can be used to combine them. If not, a probabilistic calculus may be used (e.g., knowledge-based model construction). Trust is constructed from beliefs, where any user may assert her personal belief in the statement, which is taken from [0,1]. A high value means that the statement is accurate, credible, and/or relevant. Using Path Algebra computation, the authors merge beliefs on the paths of trust between the user and any other user with a personal belief in the statement.

2.1.8 Policy-based Trust

WS-Trust [5] is an industrial specification to enable applications to construct trusted message exchanges. This trust is represented through the exchange and brokering of security tokens. As with every security protocol, significant efforts must be applied to ensure that specific profiles and message exchanges constructed using WS-Trust are not vulnerable to attacks (or at least that the attacks are understood). WS-Trust defines a request/response protocol where the client sends RequestSecurityToken and receives RequestSecurityTokenResponse. The common functions of Security Token Service (STS) are: Token Exchange, Token Issuance and Token Validation.

This technique is based on a process in which a Web service can require that an incoming message prove a set of claims (e.g., name, key, permission, capability, etc.). If a message arrives without having the required proof of claims, the service SHOULD ignore or reject the message. A service can indicate its required claims and related information in its policy as described by [WS-Policy] and [WS-PolicyAttachment] specifications. Thus, this framework is geared more towards security aspect of establishing trust among entities, rather than a social aspect.

Salehi-Abari and White [32] demonstrate the need for witness interaction trust to detect naive agents in addition to the need for direct interaction trust to detect malicious agents. The paper demonstrates how learning agents can isolate themselves from naive and malicious agents by a set of policies. The games used to test the model are Iterated Prisoner's Dilemma and Generalized Prisoner's Dilemma and the two actions monitored in those games are cooperation and defection. Two types of trust are observed: Direct Interaction Trust (DIT) and Witness Interaction Trust (WIT).

This research focuses on a policy-based trust. Two kinds of policies are used as explained below: Direct Interaction Policy (information regarding agents with which an agent interacted with previously) and Witness Interaction Policy (information regarding how to provide answers to querying agents and which agents to query from and when). Witness Interaction Policy leads to two sub-policies: Connection Policy (information about when to establish a connection with other agents to query for information) and Disconnection Policy (information about when to terminate a connection with other agents).

Blaze et al [7] address specific syntax and semantics of active trigger mechanisms that allow highly dynamic trust management policies tightly coupled to network health and changing policy. Their model, called Dynamic Policy Evaluation, revisits security policy decisions at any time during the session's lifetime and may recommend actions beyond the typical permit/deny outcome of such security policies. Prominent aspects of their framework are: policies can encode complex rules and risk management strategies appropriate to a particular application or service. Also, "credentials" are digitally signed and written in the same language as policies.

2.2 Reputation

There are several definitions of reputation and few of them which are related to this work are listed below:

Abdul-Rahman & Hailes[1] define reputation as an expectation about an agent's behaviour based on information about or observations of its past behavior.

Grishchenko[15] defines reputation as an expectation that a compliance of some future event will be near to an average compliance level of past events by the same responsible entities.

Golbeck & Hendler[14] state that reputation is a more social notion of trust.

Mui *et al.*[27] define reputation as the perception that an agent creates through past actions about its intentions and norms. It is a social quantity calculated based on actions by a given agent a_i and observations made by others in an "embedded social network" that a_i resides.

The general notion of reputation is commonly implemented in the form of ratings given to an entity or person and aggregated using some metric. Zacharia *et al.*[40] & several E-commerce websites follow this approach by equating reputation to a function of cumulative positive and non-positive ratings received. Yu & Singh[39] follow the approach of using feedback (in terms of testimonies) to calculate reputation of an agent in a multi agent setting. However, their approach differs from that of others because they check the reliability of agents and use those values to weigh the testimonies. Abdul-Rahman & Hailes[1], Grishchenko[15] & Mui et al.[27] view reputation as a perception or expectation that an agent creates about itself based on its past behaviour. Malik & Bouguettaya[25] portray reputation within the context of a community rather than a value that is solely dependant on a user‘s input. If a person‘s feedback is consistent with the community‘s opinion of a Web Service and the previous rating of that user to that Web Service, the reputation of that user is increased, else it is decreased.

Chapter 3

Mathematical Model

3.1 Model of Trust in Web Services

Wang & Singh[36] present a formal model of trust for multiagent systems. The model shares its conceptual underpinnings with the formalization of trust proposed by Jøsang[20], and improves on some of its limitations. The general trust model proposed by Wang & Singh[36] is adopted to the specific context of Web Services.

3.1.1 Belief Density

Trust is inherently uncertain. It is derived from the belief that a subject has in another subject's abilities to perform the actions on which its own welfare depends. The subject must also be cognizant of possible negative consequences that result from the actions performed by another subject. The model of belief is defined more formally as:

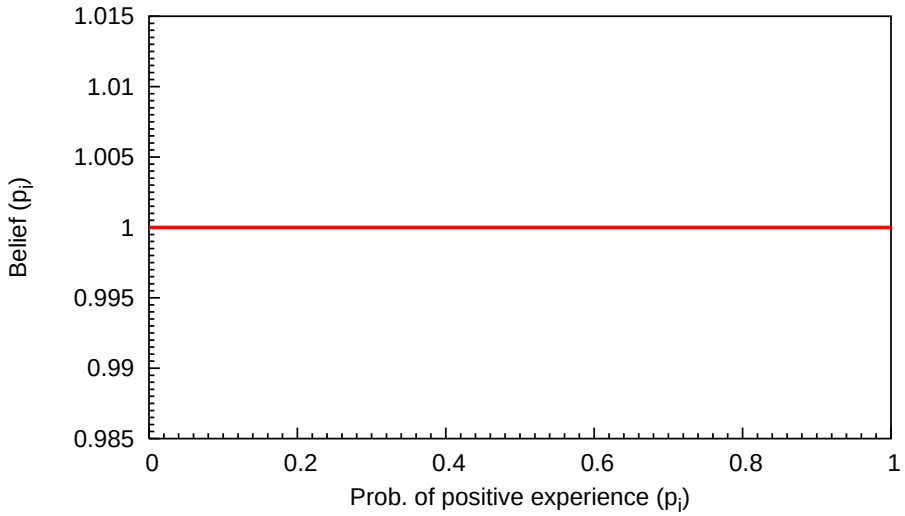
> *Let $p_i \in [0, 1]$ be the probability that users will have a positive experience with web service i. Then, the belief in a positive experience with web service i is defined as a probability density function over p_i, denoted as $B(p_i)$.*

Belief is a density function (and not a discrete distribution) because the space of p_i is continuous. Furthermore, the belief is a second order probability function — probability density over a probability; and Jøsang[20] refers to it as a probability certainty density function (PCDF).

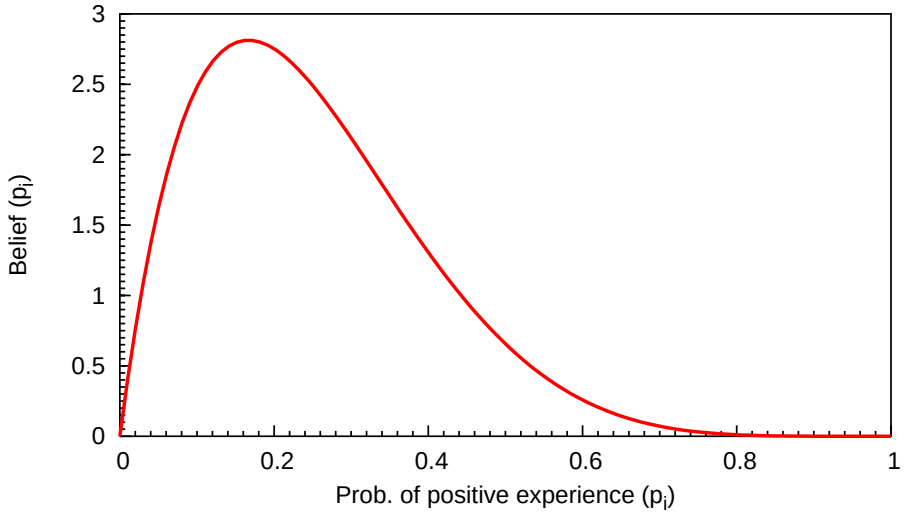
Initially, there may be no opinion about the behavior of a Web Service *i*. At this point, the belief will be a flat line, thereby indicating no information about p_i. As the positive or

negative experiences with the Web Service accumulate over time, the belief function will assume forms that assign large probability masses to high or low p_i values, respectively.

Fig. 3.1 shows belief densities that are represented using the β distribution function parameterized by a and b. The belief density in Fig. 3.1 (a) models a state of complete uncertainty about the probability of a positive experience with Web Service i. The belief density in Fig. 3.1 (b) indicates that the chances of having a positive experience with the Web Service i are likely to be low.



(a) Flat line indicates no information about p_i $(a = b = 1)$



(b) Larger probability mass is assigned to low p_i $(a = 2, b = 6)$

Figure 3.1: Belief densities represented using the beta density function

While the belief densities could take any shape, the form to the family of β probability densities is restricted in this work. This restriction is useful because β densities are well studied and exhibit the desirable property that they represent a conjugate family of distributions in the context of Bayesian updating where a, b parameterize the density and $\beta(a,\ b)$ is the well-known β function.

$$B\left(p_i\right) \stackrel{def}{=} \frac{1}{\beta\left(a,b\right)} p_i^{a-1}\left(1-p_i\right)^{b-1}$$

3.1.2 Belief Update

An important property of trust is that subjects often 'come around' to trusting another person or entity. In other words, trust in a Web Service must be updated using past experiences - both positive and negative. Because trust is derived from the belief, $B(p_i\,;a,\ b)$, a way to update the belief with past experiences of users with the Web Service i must be developed.

After n interactions with Web Service i, let there be r positive experiences and s non-positive (possibly negative) experiences, where $n = r + s$. The prior belief, $B(p_i\,;a,\ b)$ is updated with these experiences resulting in an updated belief, $B'(p_i \mid <r,\ s>)$. It is noticed that the probability of a positive experience is p_i, and that of a non-positive experience is (1 – p_i). Therefore, the probability of r positive and s non-positive experiences is $p_i^r(1-p_i)^s$. If the experiences are provided, then the belief over p_i could be updated in a Bayesian manner as follows:

$$\begin{aligned}
B'\left(p_i \mid <r,s>\right) &= \frac{p_r\left(<r,s> \mid p_i\right)\ B\left(p_i\,;\,a,b\right)}{\int_0^1 p_r\left(<r,s> \mid p_i\right)\ B\left(p_i\,;\,a,b\right)\ \mathrm{d}p_i} \\
&= \frac{p_i^r\left(1-p_i\right)^s\ B\left(p_i\,;\,a,b\right)}{\int_0^1 p_r\left(<r,s> \mid p_i\right)\ B\left(p_i\,;\,a,b\right)\ \mathrm{d}p_i} \\
&= \alpha\ p_i^r\left(1-p_i\right)^s\ B\left(p_i\,;\,a,b\right) \\
&= \alpha\ \frac{1}{\beta\left(a,b\right)}\ p_i^r\left(1-p_i\right)^s\ p_i^{a-1}\left(1-p_i\right)^{b-1} \\
&= \alpha'\ p_i^{(a+r)-1}\left(1-p_i\right)^{(b+s)+1}
\end{aligned}$$

where α is the normalization constant and $\alpha' = \frac{\alpha}{\beta(a,b)}$. Consequently, the updated belief, $B'(p_i \mid <r,\ s>)$ remains a β density function and is parameterized by $a + r$, $b + s$.

3.1.3 Certainty And Trust Vector

Trust in Web Service i represents a level of certainty of the belief in a positive experience with i. A method of measuring the certainty level of a belief is by computing the Shannon entropy[19] of the belief. However, it does not provide a measure of the certainty of the second order belief over p_i[36]. Another method of measuring the certainty level of a belief, $B(p_i\,;a,\ b)$, is to ascertain how much $B(p_i\,;a,\ b)$ is further away from a state of complete uncertainty over p_i.

The latter method is shown in Fig. 3.1 (a), and could be represented as, $B(p_i\,;1,\ 1) = 1$. This special uniform density is denoted as B_u. Based on the approach described by Wang & Singh[36], the L_1 norm[4] is used to measure the distance between the densities. It is important to note that other distance measures such as the KL Divergence may also be used. The certainty of a belief over the probability of positive experience with a Web Service i, $C_i : B \rightarrow \mathbb{R}$, is:

$$\begin{aligned} C_i &\stackrel{def}{=} \frac{1}{2}\, L_1\ (B - B_u) \\ &= \frac{1}{2} \int_0^1 \mid B\,(p_i\,;\, a, b) - 1 \mid \ \mathrm{d}p_i \\ &= \frac{1}{2\,\beta\,(a,b)} \int_0^1 \mid p_i^{a-1}\,(1-p_i)^{b-1} - 1 \mid \ \mathrm{d}p_i \end{aligned}$$

Typically, the L_1 norm accumulates the difference between a peak in $B(p_i\,;a,\ b)$ w.r.t. the flat line and a dip in $B(p_i\,;a,\ b)$ w.r.t. the flat line. To avoid counting the difference twice, the L1 norm of the difference between B and B_u, is scaled by 0.5. This scaling has the beneficial effect of maintaining $0 \leq C_i\,(B) \leq 1$. The certainty of the belief density in Fig. 3.1 (b) is computed as: $C_i\,(B') = \frac{1}{2} \int_0^1 \mid B\,(p_i\,;\,2,6) - 1 \mid \ \mathrm{d}p_i = 0.461$

Both trust and distrust could contribute toward the certainty level of the belief over p_i. Therefore, there is a need to find a way to distribute the certainty among trust and distrust.

Previously, trust in a Web Service i was defined as an outcome of the positive experiences with i. Based on this definition, distrust in a Web Service i is defined as an outcome of the non-positive experiences with i. A proportion of certainty equal to the proportion of positive experiences among the total experiences can be assigned to trust. The remaining proportion of the certainty is distrust. The trust vector for Web Service i is defined as:

$$\hat{v}_i = \langle t_i, d_i, u_i \rangle$$

$$\textit{where, } t_i + d_i + u_i = 1, \textit{ and, } t_i = \frac{r}{r+s} C_i(B) \; ; \; d_i = \frac{s}{r+s} C_i(B) \; ; \; u_i = 1 - (t_i + d_i)$$

For simplicity, $\frac{r}{r+s}$ is replaced with γ. Thus, $\frac{s}{r+s}$ becomes $1 - \gamma$[1]. The space of trust vectors, $\hat{v}$, forms a *simplex* (triangle) as shown below in Fig 3.2. Every point on the simplex satisfies the property $t + d + u = 1$. The red dot in Fig 3.2 indicates the position of an example vector, $\langle 0.2625, 0.0875, 0.65 \rangle$, in this simplex.

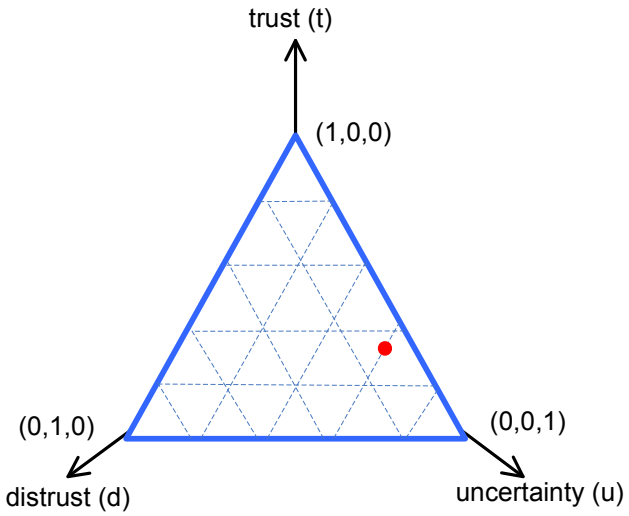


Figure 3.2: Space of trust vectors as a simplex

3.1.4 Reputation Score

User feedbacks are required to compute the trust of a Web Service. However, not all feedbacks genuinely reflect the performance of the Web Service. Therefore, a measure called reputation

[1]If $\langle r = 1, s = 0 \rangle$, then the entire certainty would be construed as trust. To avoid such a premature conclusion from a single experience, Wang & Singh[36] suggest rewriting γ as $\frac{r+1}{r+s+2}$ – a process called Laplace's smoothing.

score is defined that helps us factor out the biases of a user providing feedback. Reputation score shows the divergence between a user's opinion from that of the community. It is similar to the rater credibility of Malik and Bouguettaya[25]. Thus, the reputation score is defined formally as:

> *Let $p_i^c \in [0,1]$ be the probability that the community will have a positive reputation, and $p_i^u \in [0,1]$ be the probability that a user, u, will have a positive reputation. Analogous to the belief definition, the probability density functions over p_i^c and p_i^u are $B^c(p_i^c)$ and $B^u(p_i^u)$ respectively. The reputation score is defined as the mean of the difference between skewness of B^c and B^u.*

There are different ways to measure the divergence between two distributions as mentioned in Aliprantis & Burkinshaw[4]. The concept of skewness is used in this thesis because it shows an overall trend of the aggregate community opinion about a Web Service. Specifically, skewness is a measure of the asymmetry of the probability density function $B(p_i\,;\,a,b)$. The skewness of a beta distribution $B(p_i\,;\,\alpha,\beta)$ is $\frac{2(\beta-\alpha)\sqrt{\alpha+\beta+1}}{(\alpha+\beta+2)\sqrt{\alpha\beta}}$.

A negative-skewed distribution has a long tail to the left and a heavy concentration of its density function lies to the right of the curve. Conversely, a positive-skewed distribution has a long tail to the right and a heavy concentration of its density function lies to the left of the curve. A symmetric distribution has equal concentration of its density function on either sides of its mean. The difference between the skewness of user and aggregate community belief densities about the Web service is used to see whether the user deviates significantly from the community's opinion.

Skewness of the belief densities of the user and community gives an intuition about the reputation of a user. However, it cannot give the probability of user having a high reputation as it does not lie between 0 and 1. Therefore, the reputation of a user is modeled using a beta density function (different from the belief density of having a positive experience with a WS).

Let $B^R(p_u\,;\,a,b)$ be the belief density function over the user u having a high reputation. Given the current belief densities of the web service by community $B^c(p_i^c\,;\,a,b)$ and by the user as a feedback $B^u(p_i^u\,;\,a,b)$, the skewness of B^c and B^u is computed to get s_c and s_u respectively. The new reputation distribution of the user $B'^R(p_u\,;\,a,b)$ is $B^R(p_u\,;\,a+\mid s_c - s_u \mid, b)$ if B^c and B^u have the same orientation of skew. Otherwise, $B'^R(p_u\,;\,a,b)$ is $B^R(p_u\,;\,a, b+\mid s_c - s_u \mid)$. The reputation score from $B'^R(p_u\,;\,a,b)$ is computed as the mean of the beta distribution or $R-Score\left(B'^R(p_u\,;\,a,b)\right)$, which is $\frac{a}{a+b}$.

When there are sufficient users in the community, the aggregated opinion of the community will reflect the correct view about a Web Service. If the user gives a feedback that contradicts the community's belief, the user is either malicious or naïve.

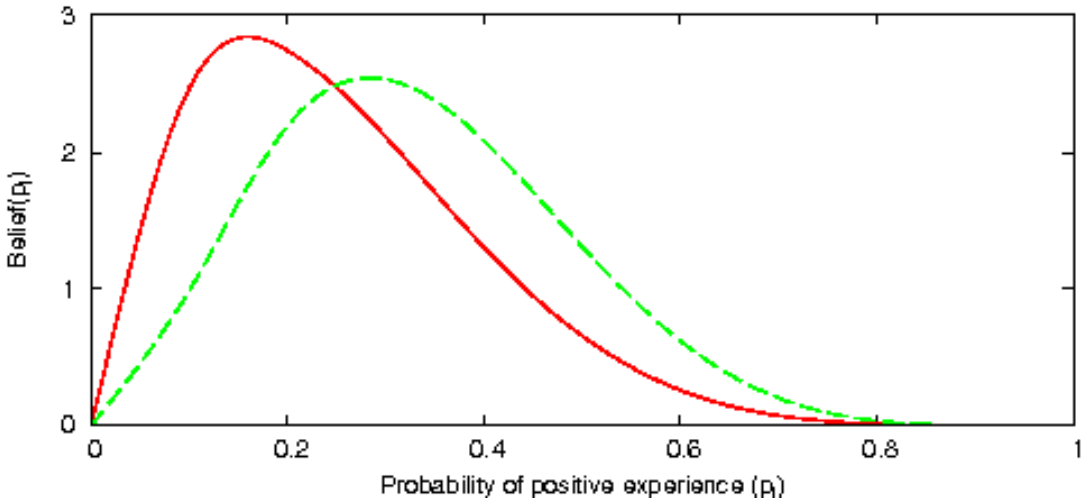
Example beliefs of community and test users are shown in Fig. 3.3. The red dotted line indicates aggregated community feedback and the green solid line indicates the test user's feedback. The belief densities in Fig. 3.3 (a) models a test user with high reputation as the skewness of the user's belief matches the community's belief density. On the other hand, Fig. 3.3 (b) shows a test user with low reputation as the skewness of the user‘s belief is opposite that of the community's belief density.

The belief update can be modified to consider the user‘s reputation while updating the belief in a possible positive experience with Web service i as:

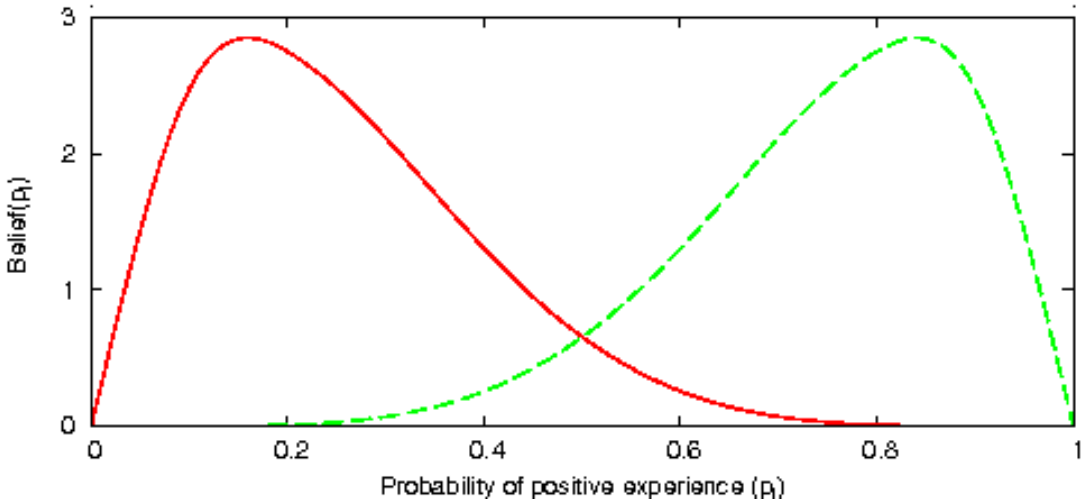
$$B'(p_i \mid \langle r, s\rangle) = \alpha' \, p_i^{(a+(r\cdot R-Score))+1} \; (1-p_i)^{(b+(s\cdot R-Score))+1}$$

3.2 Predictors

After a user interacts with a Web Service, s/he generally forms an opinion about it, denoted by trust of the Web Service. Similarly, when a user provides feedback to a system, the system determines the user‘s standing, denoted by reputation of the user. However, both trust and reputation are subjective terms and it can be difficult to objectively conceptualize them. In this section, the parameters that characterize these trust and reputation are identified.



(a) Test user has high reputation (S_u = 0.41, S_c = 0.69)



(b) Test user has low reputation (S_u = -0.69, S_c = 0.69)

Figure 3.3: Belief densities of the community and test users over the probability of having a positive experience with a WS

3.2.1 Positive Experience with a Web Service

The trust model presented in this thesis is based on the confidence of having a positive experience with a Web Service i. The trust is updated using the number of positive and other types of experiences with the Web Service. Unlike previous efforts described in Jøsang[20]

and Wang & Singh[36], a hypothesis of the definition of an 'experience' with i is presented in this section.

Although experiences tend to be subjective, the variables provided previously are meant to serve as predictors of likely positive experiences. The possible correlations between user experiences and the predictors are explored later in a pilot study. While interacting with a Web Service, a user may consider the competency, reliability and honesty of the service.

Therefore, an experience of a user with Web Service i is defined as the behavior of i in terms of its competency, reliability and honesty. The variables related to the characteristics of the Web Service are defined below in order to provide a precise meaning to the definition of an experience.

- *Competency* is defined as the ability of the service to perform all the necessary actions to achieve a particular goal.
- *Reliability* is defined as the ability of the service to provide the same desired output upon repeated invocations without any discrepancy.
- *Honesty* is defined as the ability of the service to be faithful to terms agreed upon in the service level agreement and other contracts.

The experiences of users with a Web Service are expected to be positive if it is competent, reliable and honest. Therefore, a discussion is provided below about how one may objectively ascertain these characteristics of Web Services.

The honesty of a service is measured as the difference between the advertised or agreed upon values of QoS parameters appearing in the service level agreements and the actual observed values of the QoS parameters. Because the cost of a service is not observed, the parameters, response time $\bar{R}$ and availability A, are only considered. The response times are normalized to make them comparable with availability.

Let A_a and $\bar{R}_a$ be the advertised values and A_m and $\bar{R}_m$ be the observed values of the QoS parameters. Then, an objective measure of honesty, $h \in [0, 1]$ is defined as:

$$h = 1 - \frac{\mid A_m - A_a \mid + \mid \bar{R}_m - \bar{R}_a \mid}{2}$$

$$where\ \bar{R} = \begin{cases} \frac{R - R_{min}}{R_{max} - R_{min}} & if\ R_{max} - R_{min} \neq 0 \\ 0 & otherwise \end{cases}$$

In the definition above, R_{max} and R_{min} are the maximum and minimum values of response times respectively among all available services. The differences could be assumed to default to zero if the observed QoS values improve on the advertised ones. Higher values of h indicate an honest service.

Reliability, r, is measured as one minus the fraction of times the service fails or does not behave as per its function measured over as long a sequence of invocations as possible. Finally, competency, c, is formalized as a binary valued concept indicating whether the service is able to satisfy the goals. If it does, then c is assigned 1, otherwise it is assigned 0.

3.2.2 Positive Reputation of a User

The reputation model presented in this thesis is based on the confidence of a positive reputation about the users providing feedback to the system and is similar to that presented in Yu & Singh[39]. Although reputations tend to be subjective, the variables provided are meant to serve as predictors of likely positive reputations of a user.

The reputation of user u is defined as the perception of u in terms of u's expertise and honesty. The variables related to the characteristics of the reputation of a user are defined below in order to provide a precise meaning to the definition of reputation.

- *Expertise* is defined as the ability of a user to use extensive knowledge and judgment to complete an assigned task with exceptional results when compared to others.
- *Honesty* is defined as the ability of a user to accurately depict recent experiences with Web services.

Expertise and honesty are positive values because they convey the intrinsic properties of an individual. This implies that neither expertise nor honesty can have negative values. The lack of honesty and expertise are found at the point closest to the origin and are mapped to zero. The values for expertise and honesty can extend up to positive infinity, but Cantor‘s theorem states that there are as many points along an infinite straight line as there are on a finite segment of it. Therefore, the values for expertise and honesty are limited to a range from zero to one.

The reputation of a user may be expected to be positive if the user is expert and honest. In other words, if the belief density of a positive reputation of the user peaks near the maximum values of honesty and expertise axes, the user is reputable. Based on these two axes, it is possible to classify users into four categories as shown in Fig. 3.4.

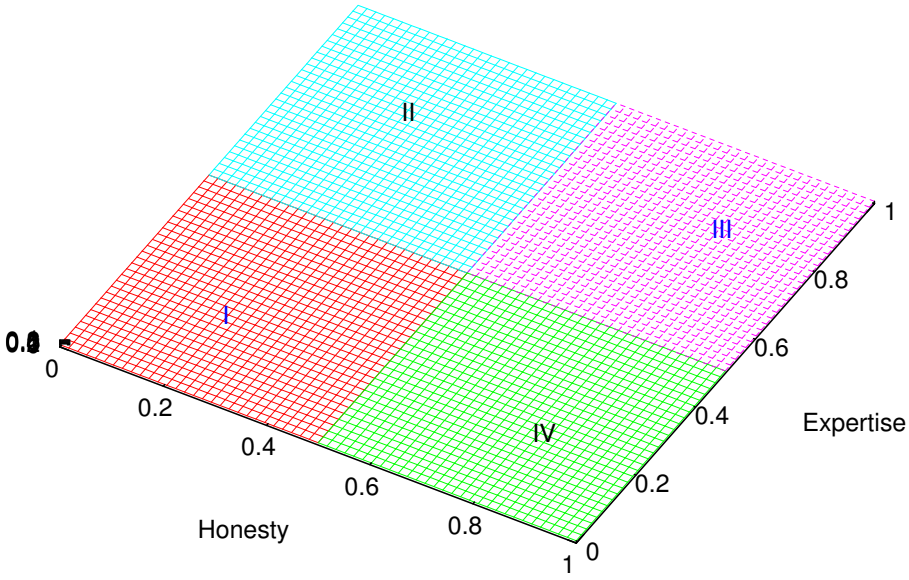


Figure 3.4: Four categories of users based on the expertise and honesty of individual users

In Fig. 3.4, four categories of users based on the characteristics of reputation are shown. Clockwise from the origin: dishonest and naïve users, dishonest and expert users, honest and

expert users, and, honest and naïve users. It is important to note that the presence of user u in a category implies that the peak of the belief density of the probability of a positive reputation of u lies in the region defined by the corresponding category.

Category II users are dangerous because they are both experts about Web Services and dishonest. Therefore, their feedbacks usually do not depict their real experience with a Web Service as they know how it should behave and would give contrary feedbacks to the system. Category IV users generally provide an accurate feedback about their experience with a Web Service but are naïve users of Web Services. Therefore, their judgment might not be representative of the actual execution of the Web Service but instead could be subjective. Category I users are unusual because they neither experts nor honest. Thus, their feedbacks contradict their experiences which are not based on specialized knowledge on Web Services. An ideal user - who is both honest and an expert - lies in category III. The belief density of the probability of a positive reputation of such a user peaks when both honesty and expertise are maximum. This implies that there is a maximum probability that the user u will have a positive reputation in the community when u‘s honesty and expertise are equal to one.

The belief density of the probability of positive reputation of an ideal user peaks in a specific region (category III). The probability distributions of the reputation of normal users of Wisp, however, are not so straightforward and may peak anywhere in categories I, II or IV. This is because Wisp is not directly aware of the honesty and expertise of its users. Without a precise belief density, it is not possible to compute reputation of a user and therefore, an alternate method is required to determine the reputation of a user.

The aggregated opinion of the community of users reflects the actual performance of the Web service because of the users’ personal experience with the WS. Hence, the community of users can be viewed as an ideal user who is both honest and an expert. A test user can be considered to be an expert and honest if the user’s opinion agrees with that of the community. Therefore, the difference between the skewness of the belief densities of both users - test user and ideal - is used to determine the test user‘s reputation as described in section 3.1.4.

3.3 Derived Trust for Compositions

Any existing Web Service Composition tool[10] may be used to generate the compositions of Web Services that meet the functional and nonfunctional requirements. While trust could be treated as an additional QoS parameter, one would need to adapt the composition algorithm to consider trust while composing services.

The aim of this thesis is to allow existing composition tools to be utilized along with the approach mentioned previously. Web Service Composition algorithms often generate multiple compositions that satisfy the requirements. A trust vector for each of these compositions is derived and the composition with the highest trust ratio deemed most trustworthy is selected for execution. Consequently, the selected composition not only satisfies the functional and QoS requirements, it is also the most trustworthy among the candidates, and most likely to perform as expected.

In order to derive the composite trust, four types of basic flows of services that are often encountered in compositions are considered. For simplicity, a composition of two services, w_1 and w_2, are considered in the different configurations, with beliefs over the probabilities of having positive experiences with them, $B(p_1; a_1, b_1)$ and $B(p_2; a_2, b_2)$, respectively. The methods may be generalized to more services in a straightforward way. The approach used in this approach is to compute the belief over the probability of having a positive experience with the Web Service Composition. Given the belief density and a way of deriving the positive experiences for the Web Service Composition, the certainty level and the trust vector can be computed. The trust vector for each construct is derived below:

Sequential flow: Each of the services in a sequential flow is executed as shown in Fig. 3.6 (a). Hence, a positive experience with the Web Service Composition is contingent on positive experiences with each of the two component Web Services. If the executions of the Web Services are independent of each other, then the probability of a positive experience with the Web Service Composition, p_c, is the product of the individual Web Service probabilities $\prod_{i=1}^{2} p_i$.

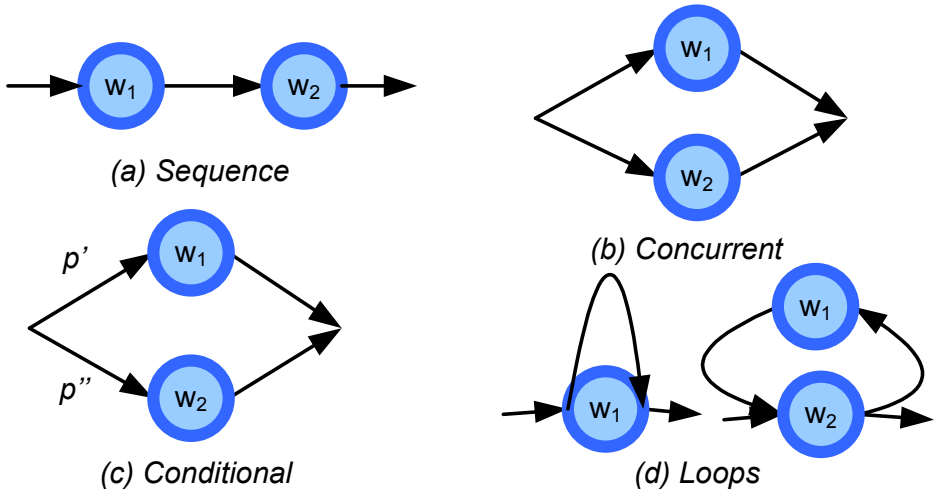


Figure 3.5: Different flow constructs typically appearing in a composition

The range of p_c can be viewed as a product space because p_c is a product of the individual probabilities. Given the individual beliefs over the probabilities p_1 and p_2, the belief density over p_c is derived as:

$$B^s(p_c) = \int_0^1 B(p_1\,;\,a_1,b_1)\;\; B\left(\frac{p_c}{p_1}\,;\,a_2,b_2\right)\;\frac{1}{\mid p_1 \mid}\,\mathrm{d}p_1$$
$$= B(p_1\,;\,a_1,b_1) \otimes B(p_2\,;\,a_2,b_2)$$

The operator $\otimes$ is introduced for simplicity of presentation. The above equation is an application of a well-known way of obtaining the probability density function over a product space of random variables[30]. Although the components are β densities, $B^s(p_c)$ may not be a β density.

Concurrent flow: A concurrent flow is shown in Fig. 3.6 (b). Analogous to the sequential flow, each of the services in a parallel flow must also be executed. Hence, p_c is derived analogously to the one for the sequential flow. Consequently, the belief density is obtained as shown previously: $B^s(p_c) = B(p_1\,;\,a_1,b_1) \otimes B(p_2\,;\,a_2,b_2)$.

Conditional flow: A conditional flow is shown in Fig. 3.6 (c), in which, any one of the branches is executed in a conditional flow. For example, let there be two branches that

are followed with probabilities, p' and p'', such that $p' + p'' = 1$. These probabilities may be obtained from prior interactions. Then, the probability of a positive experience with the Web Service Composition is a weighted sum of the probabilities for the individual Web Services, $p_c = p' \times p_1 + p'' \times p_2$. Given the individual beliefs over the probabilities p_1 and p_2, the belief density over p_c is derived as:

$$B^0(p_c) = \frac{1}{p'p''} \int_0^1 B\left(\frac{p_1}{p'}\,;\, a_1, b_1\right) B\left(\frac{p_c - p_1}{p''}\,;\, a_2, b_2\right) \mathrm{d}p_1$$

The proof of the above equation when p_c is composed as shown previously is available in Rohatgi[30].

Loop: In this thesis, only loops that are iterated a fixed number of times, say n[2] , are analyzed. Analogous to the sequential flow, each Web Service in a loop must be executed n times, as shown in Fig. 3.6 (d). Therefore, for the case where two services participate in a loop, let, $p'_c = p_1 \times p_2$. Then, $B^s(p_c) = B(p_1\,;\, a_1, b_1) \otimes B(p_2\,;\, a_2, b_2)$, where the operator $\otimes$ was defined previously. For the Web Service Composition, $p_c = \prod_{i=1}^{n} p'_c$:

$$B^{l(n)}(p_c) = B^s(p'_c) \otimes B^s(p'_c) \otimes B^s(p'_c) \otimes \cdots n \textit{ times}$$

A composition may consist of one or more of these basic flow constructs. In the above paragraphs, a mathematical way to derive the belief density for each of these constructs was presented. The aggregate belief over the probability of a positive experience with the entire composition can be computed. These methods for deriving the composite probability for a Web Service Composition, p_c, are loosely analogous to calculating the aggregate QoS parameters of a composition as shown by Cardoso *et al.*[9]. A significant difference of the approach used in this thesis and that used by Cardoso *et al.*[9] is in the computations for the loop construct. In this thesis, the probability of entering a loop is not considered.

An algorithm for computing the distribution over the product (i.e. implementing the $\otimes$ operator) is mentioned by [13]. However, exact computation of the distribution often turns

[2]This precludes loops that conditionally terminate (e.g. while loops). The difficulty is in knowing the number of times the loop will run *a priori*.

out to be complex; approximations provide reasonably close distributions in significantly less time. A sampling scheme to generate the density over the product space is used, which converges to the exact as the number of samples approaches infinity. A sampling algorithm for approximating a probability density over a product of two independent random variables is shown below. The individual densities are first sampled and the frequencies of the product of the two independent variables are tabulated. The frequency histogram is converted into a normalized cumulative distribution function (cdf), which is then used to obtain an approximate probability density function.

```
begin
n //number of samples
numBins //number of bins used in histogram, cdf and pdf
frequencyCountBin[1..numBins] //frequency count for histogram
cdf[1..numBins], pdf[1..numBins]
//sample densities and tabulate frequency of product of samples
for i = 1 to n
   sample s1 ~ B(p1 ; a1 , b1 ), s2 ~ B (p2 ; a2 , b2 )
   p <- s1 x s2
   increment frequencyCountBin[p] by 1
//convert to a cdf and then to a pdf
for i from 2 to numBins
   cdf[i] <- cdf[i  1] + frequencyCountBin[i]
for i from 2 to numBins
   use the inverse of trapezoidal rule for numerical
   quadrature with values in cdf[i], cdf[i  1] to find pdf[i]
end
```

As mentioned previously, it is possible to compute the certainty of the belief density. The remaining question is how much of the certainty should be allocated to trust and distrust. This is equal to the proportion of positive, γ_c, and non-positive experiences, $1 \ \gamma_c$, with the composition, respectively. Because users interact with an individual Web Service in the composition, the proportions from the experiences with the individual component Web Service must be computed.

As with the computation of p_c, the estimation is contingent on the type of flow in the Web Service Composition. For flows where all component Web Services are executed (sequential,

parallel and loop), a positive experience with each of the component Web Service is very likely to translate into a positive experience with the Web Service Composition. Hence, the proportion of positive experiences with the Web Service Composition is at most the minimum of the proportions of positive experiences among all component Web Services. It is assumed that γ_c is the minimum proportion.

Next, the case where the flow involves a condition is considered. If γ_1 and γ_2 are the proportion of positive experiences for Web Services w_1 and w_2, respectively, then $\gamma_c = p'\gamma_1 + p''\gamma_2$. Given γ_c, the trust vector of the composition can be obtained.

Chapter 4

Wisp

A new framework called Wisp for integrating trust considerations into Web Service Compositions and selecting trusted Web Service Compositions for deployment is presented in this section. While traditional composition techniques form compositions that meet the functional and non-functional requirements in theory, the compositions may not behave accordingly in practice. Wisp seeks to reduce this pragmatic gap by associating trust vectors to Web Service Compositions and updating the trust vectors based on user feedback. The design of Wisp is shown below and its design is mentioned in the next subsections. Wisp and the Web Service composition tool are decoupled and may be located on different hosts.

4.1 Filtering Untrustworthy Services

For each Web Service, i, available for composition, Wisp associates and maintains the belief density over the probability of having a positive experience with i. Because there is complete uncertainty about i initially, the belief density is initialized to a flat line as shown in Fig. 3.1.

Wisp measures the certainty level of the belief density and computes the trust vector for i indicating the trust (t_i), distrust (d_i) and any remaining uncertainty (u_i) about having a positive experience with i.

Wisp allows a preliminary filtering step, in which a target user of the Web Service Composition may assume a trust ratio threshold, $tt \in [0, 1]$. Services available for composition whose trust ratios fall below the threshold, $t_i + d_i < tt$, are deemed untrustworthy by the user, and immediately filtered out. The trust threshold may vary with Web Services - critical

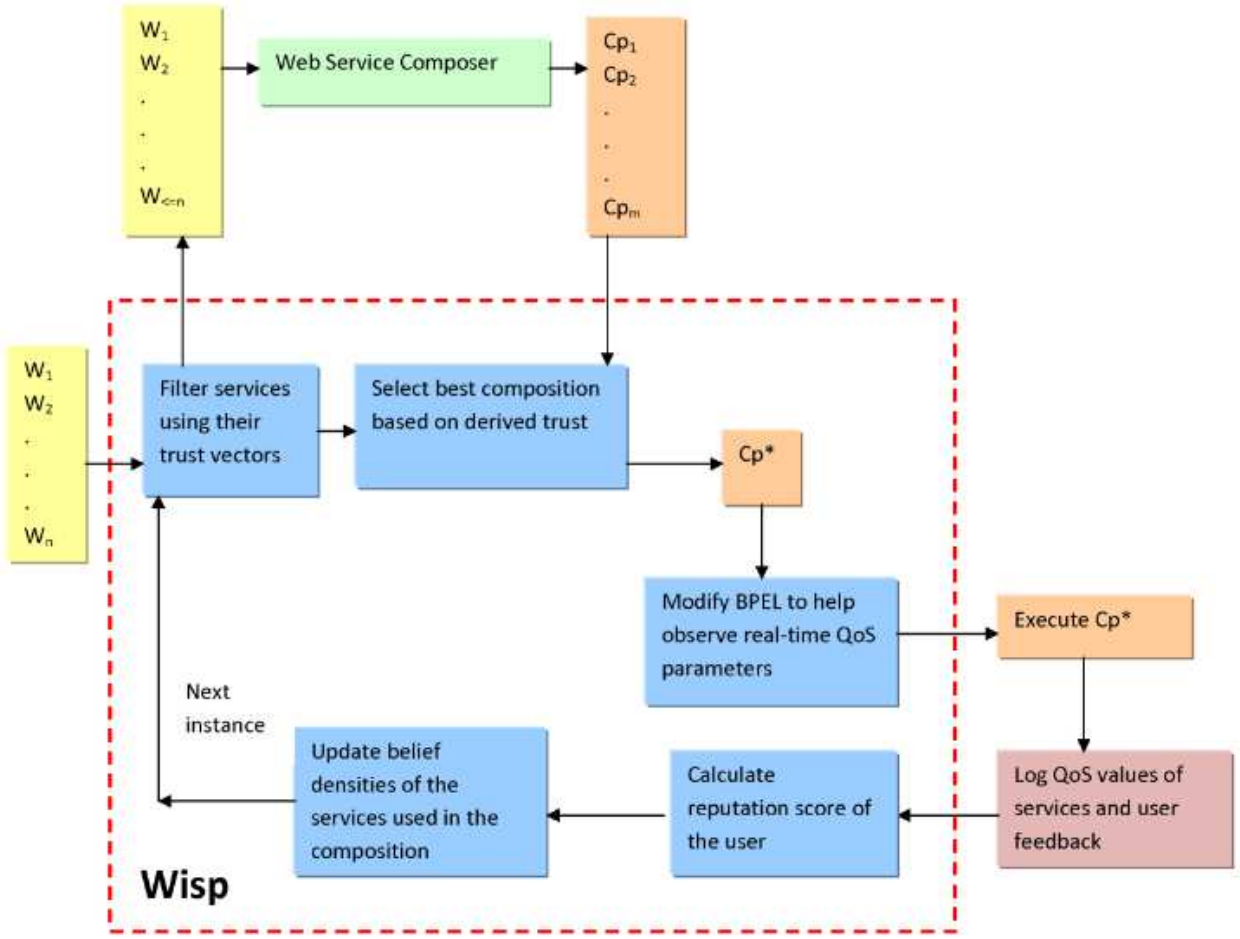


Figure 4.1: Wisp - a framework for deploying trusted Web Service Compositions

Web Services may warrant higher thresholds than non-critical ones. Morever, it need not be fixed and may vary over time according to the judgment of the user.

There are two advantages in allowing a trust threshold. First, it allows the user to decide a minimum quality level below which services cannot participate in the composition based on prior interactions. Second, by reducing the set of candidate services, the composition may occur in lesser time due to a reduced search space.

4.2 Deploying Trusted Compositions

Any Web Service Composition tool could be utilized to generate compositions of the services that theoretically meet requirements. Often, multiple satisfying compositions are produced. Wisp seeks to deploy one of these compositions for execution that is expected to perform well in practice. For each of these compositions, Wisp derives the aggregated belief density over the probability of having a positive experience with the composition. As mentioned in Section 3.3, the derivation is based on the type of flow present in the Web Service Composition. Given the belief density, the certainty and the trust vector for each of the compositions can be computed.

Let the Web Service Composition tool produce m compositions and let each composition be associated with a trust vector, v_j, where $j = 1...m$. Wisp chooses a Web Service Composition, C_p^*, to execute using the following algorithm:

1. Among the m compositions, select C_p^* to be the Web Service Composition j which has the highest proportion of certainty allocated to trust in its trust vector, v_j.

2. If there exists a tie, select the Web Service Composition j among the tied ones to be C_p^* which has the highest level of certainty, $t_j + d_j$, in its trust vector.

3. Compositions that remain tied have identical trust vectors. Finally, break ties randomly to obtain C_p^*.

In order to uncover possible correlations between user feedback and observed QoS parameters of compositions, Wisp supports the logging of response times and availability data of component Web Services in C_p^*. Specifically, Wisp parses the implementation log to obtain the response times. Availability of a service is ascertained by noting whether a valid response is received over repeated invocations. The chosen composition C_p^* is deployed. Wisp provides an intuitive interface to users for collecting feedback on the individual services participating in the composition. In particular, users may indicate whether their experiences with the component services in the deployed Web Service Composition have been positive.

4.3 Measuring Reputation of User

In order for the framework to be robust, the feedbacks obtained as mentioned in the previous section must not be considered blindly. Wisp provides a mechanism to understand the expertise and honesty of the user and factor them in the feedbacks to obtain a realistic input to update the belief densities of the services involved in the composition.

Wisp maintains a history of the feedbacks provided by each user as well as the community feedback for each Web service and composition. The storage is bounded by the number of services and compositions in the system because Wisp only maintains the number of positive and negative feedbacks, trust, distrust and uncertainty of each Web Service and Web Service Composition. The user‘s feedback is added to the community‘s feedback after factoring the user‘s reputation in.

After the user provides feedback, Wisp measures the reputation of the user against the combined feedback of the community. This gives an idea of how closely the user agrees with the general view of the community regarding the Web service. Wisp does not alter the quality or number of services offered to a user based on the reputation of the user. Reputation is only used to weigh the usefulness of the feedbacks received from a user.

4.4 Updating Belief

The weighted positive and non-positive feedbacks for each service in the Web Service Composition, say i, form the tuple, $\langle r, s \rangle$, which is used to update the belief density over p_i, as mentioned in Section 3.1.2. Belief densities for all Web Services in the deployed C_p^* are updated using the feedbacks. Wisp indexes the computed trust vectors of the Web Services in a hash table for quick lookup during the next instance of its usage.

Trust vectors associated with a service are not private to a user and are updated by all users interacting with compositions containing the service. This has the advantage that a service deemed untrustworthy by some user ($\frac{t_i}{t_i+d_i}$ is below threshold tt) could have its trust

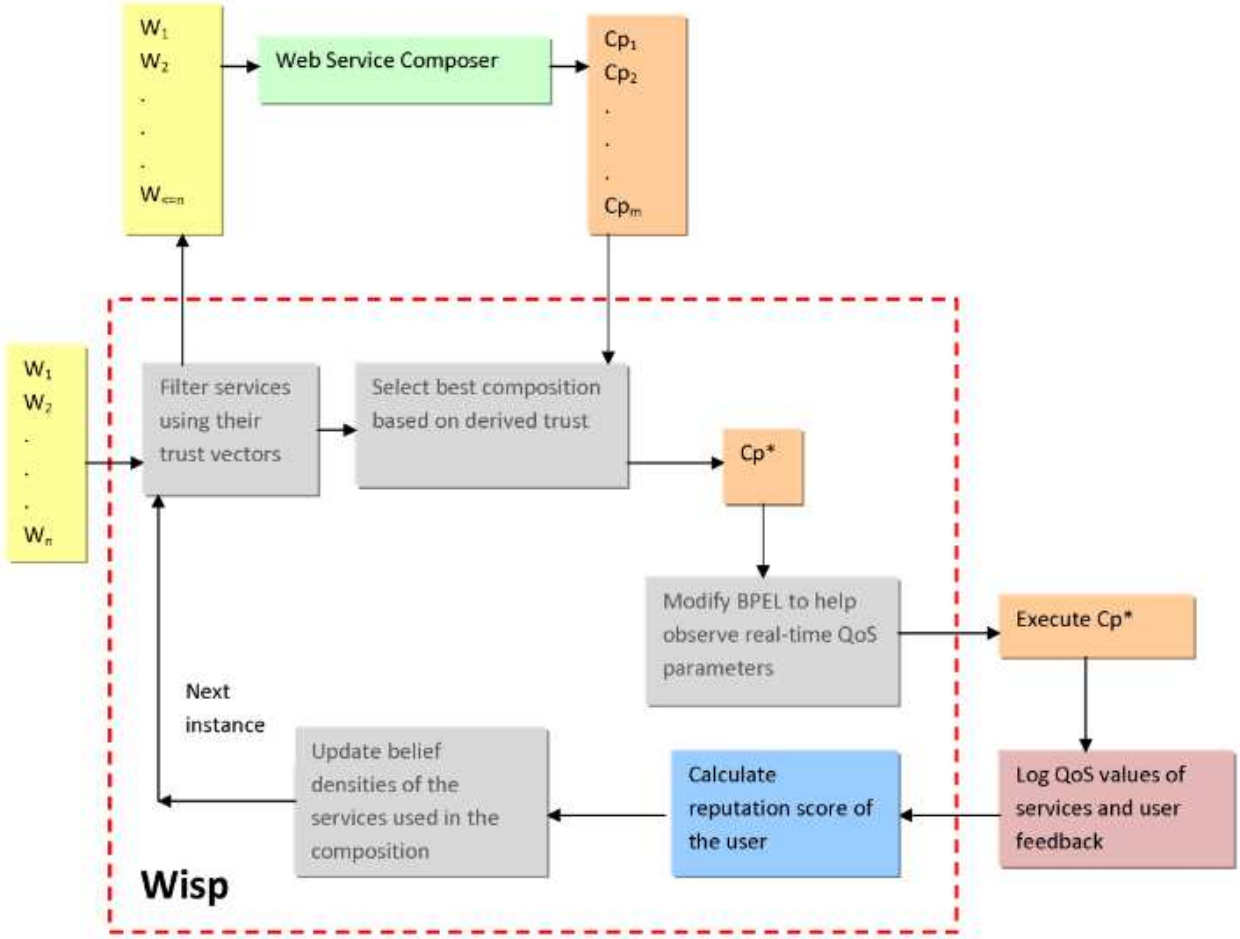


Figure 4.2: Wisp with the reputation module highlighted

ratio improve over time because of interactions with other users, until it meets the threshold of the user. In other words, the update of the trust in a service allows the service to possibly migrate from the filtered list to the unfiltered one, if its performance improves.

Chapter 5

Experiments

Three experiments were conducted to test the hypotheses and framework in this thesis. The first experiment tests the validity of our hypotheses connecting the predictor variables to trust of a Web service. The second and third experiments validate the trust and reputation models of Wisp respectively.

5.1 Pilot Study

A small study was conducted to explore possible correlations between users' assessment of a positive or non-positive experience with Web Services in a composition and our hypothesized variables - honesty, reliability and competency - as mentioned in Section 3.2.1. The subjects were volunteer graduate students in the Department of Computer Science at the University of Georgia whose prior experience interacting with Web Services ranged from significant to none. Fifteen subjects responded to our call and participated in the study.

Each subject was presented with ten randomly selected sequential compositions of three Web Services each. To maintain realism, the compositions were deployed using the ActiveBPEL engine[2], the WSDL Web Services using the Apache server and inputs were provided to the services using soapUI[35]. The subject was informed about the advertised response times, ranging from 1,000ms to 1,500ms, and availability rates, ranging from 0.75 to 1.0, of the Web Services participating in the compositions. Subjects witnessed four executions of each of the compositions and were shown the average observed response times and availability rates of the Web Services. Some of the Web Services in the compositions were programmed to deviate from their advertised parameters by varying levels. Subjects

were then asked to rate their opinion of the Web Services in the composition as 'positive' or 'non-positive' in a computerized questionnaire.

Given the advertised and observed values of the QoS parameters, the honesty and reliability of each Web Service in the ten compositions was calculated. The subjects were not made aware of the variables and their computations. For the sake of simplicity, it was assumed that each Web Service is competent. These variables were then combined to produce a single predictor variable, whose value is $\frac{h+r+c}{3}$.

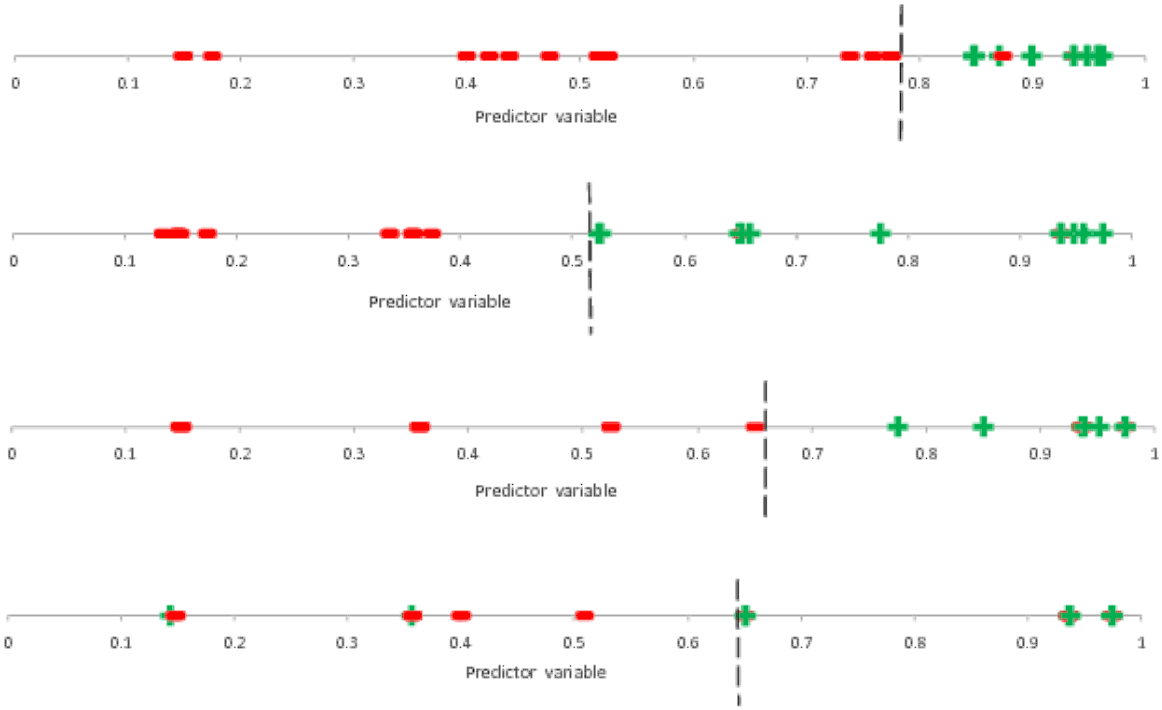


Figure 5.1: Preference models of four subjects out of fifteen that participated in the pilot study

The predictor variable ranges from zero to one, and higher values of the variable signify Web Services that are honest, reliable and competent. In Fig. 5.1 (a)-(d), the preference models of four of the subjects who participated in the study are shown. The - and + symbols indicate the subject's non-positive or positive opinion about a Web Service and their positions on the horizontal axis indicate the value of the predictor variable for that Web Service. Each model contains thirty such data points; many of them too close to distinguish them clearly.

A decision tree classifier, J48, available in the Weka machine learning package[17], is used to find a threshold value of the predictor variable that best separates positive from non-positive experiences. For most subjects, finding the partition was possible as their feedbacks were consistent. However, Fig. 5.1 (a) shows an exception where some Web Services predicted to offer a good experience were disliked by the subject. The threshold in each model is indicated by the vertical dashed line.

Several preliminary conclusions are drawn from this pilot study:

(i) Because Web Services having high predictor variable values were consistently rated by subjects to offer a 'positive' experience, a statistical correlation between our hypothesized predictors and actual experiences exists.

(ii) Further exploration of the exceptions in Fig. 5.1 (a) revealed that the Web Services rated 'non-positive' met their advertised response times but fell slightly short of the advertised availability rates. This suggests that users could be attaching unequal importance to the different QoS parameters - a conclusion that requires further study.

(iii) Finally, as it may be expected, users exhibited differing bars below which they judged their experience with Web Services to be not positive. However, these thresholds were strictly higher than 0.5. Interestingly, subjects that did not have prior experience with Web Services displayed lower thresholds.

5.2 Evaluation of Trust Model

The objective of this study is to validate the utility of our trust framework. It is empirically demonstrated that a consideration of trust in the selection of compositions results in compositions that progressively exhibit less deviations from their advertised or agreed upon QoS values.

A five-step sequential, concurrent and looping compositions is used with a choice of two to three Web Services identical to those used in the study, at each step. Thirty-two distinct compositions of the services were utilized. Web Services are described using WSDL and

the compositions using WS-BPEL. ActiveBPEL[2] is used to deploy the compositions and soapUI[35] is used to execute them. Wisp is deployed as a Web Service and it retrieves details of execution of services from the process log of the Web server. For the experimental environment, one-third of the Web Services were programmed to significantly deviate from their advertised QoS parameters. This is done by adding redundant CPU cycles in the services to slow down the response times and a special message to indicate unavailability. As compositions were executed, positive or non-positive feedback was assigned to services by following the user preference models that were elicited previously. In this way, actual users of the compositions were simulated.

After executing a composition, the updated trust vector was computed as mentioned in Section 4.3. Once the belief densities of the services are updated, the composite trust vectors of each of the thirty-two compositions were computed as described in Section 3. The composition that was deployed was selected using the algorithm mentioned in Section 4.2.

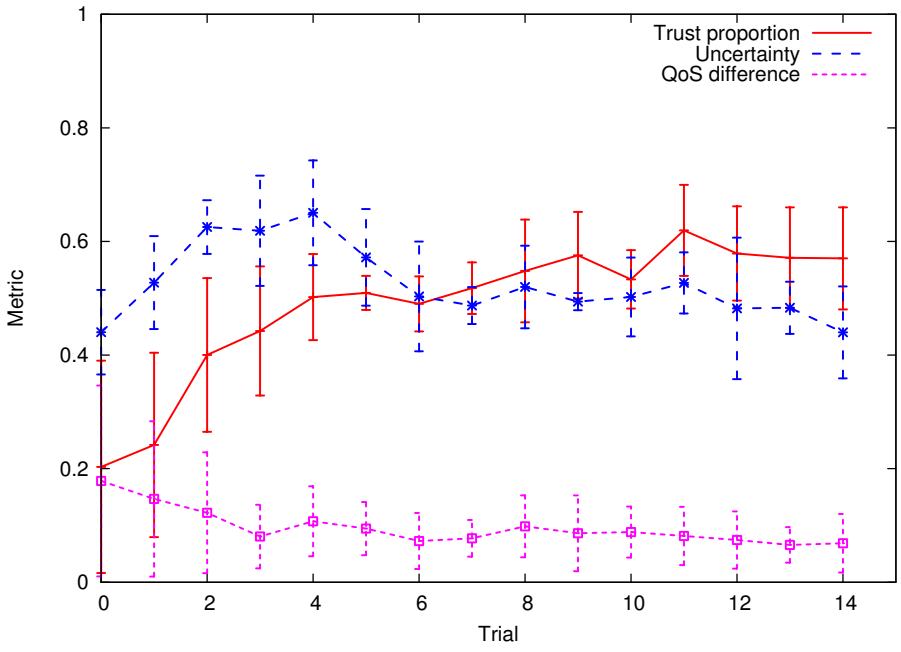


Figure 5.2: Trust proportion, uncertainty and composite QoS difference across trials

In Fig. 5.2, the trust proportions, uncertainties and difference between the composite empirically measured QoS of the deployed compositions and the composite advertised QoS

of the composition are shown. The composite QoS values were computed as per the rules described in the book by Jaynes[19]. Fifteen trials, averaged over five iterations, were executed during which each composition was executed four times. The empirically observed QoS parameters were averaged over these four runs. In summary, the experiments in this study validate the value of considering trust in Web Service Compositions. The experiments also demonstrate that Wisp operating in conjunction with a service composition method could deliver compositions of high quality in practice.

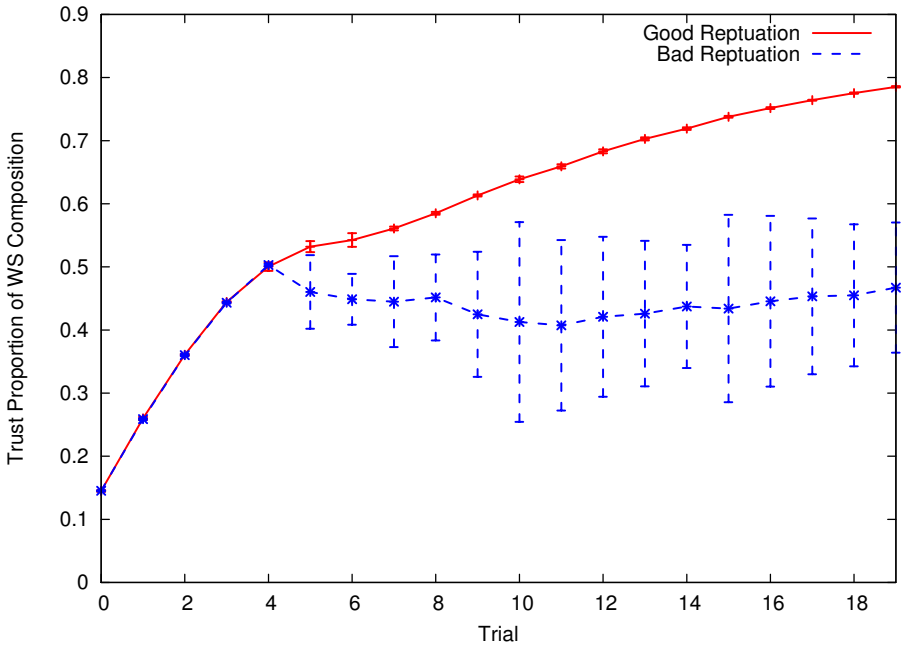
5.3 Evaluation of Reputation Model

The objective of this study is to validate the utility of the reputation model. It is emperically demonstrated that by considering the reputation of the users, one can obtain a more accurate feedback about the Web Services and thus diminish the effect of malicious and naïve users in the system.

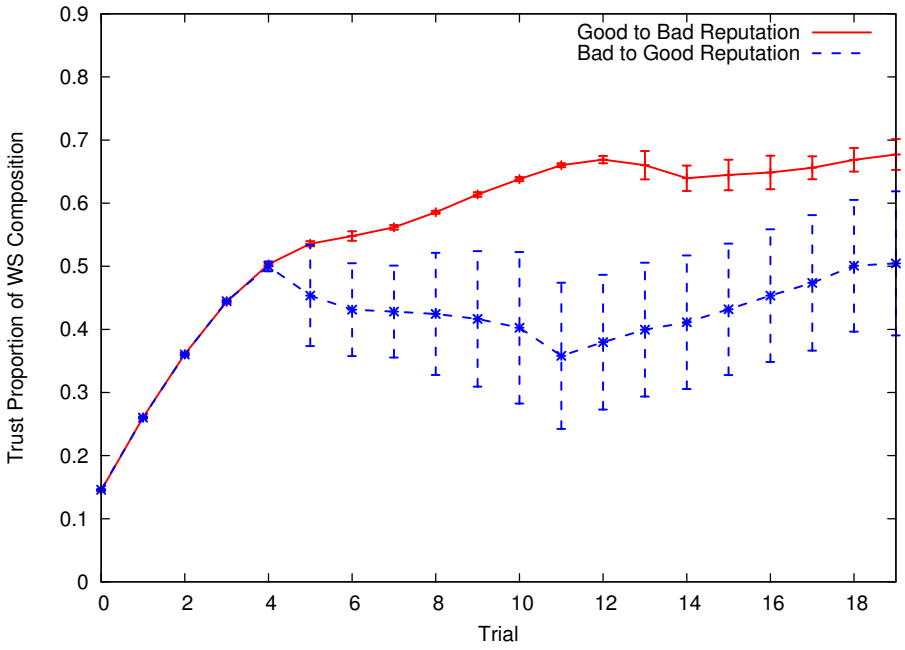
The setup for the experiment consists of a single Web service composition with a choice of two to three Web Services identical to those used in the study, at each step. Web Services are described using WSDL and the compositions using WS-BPEL. ActiveBPEL[2] is used to deploy the compositions and soapUI[35] is used to execute them. Wisp is deployed as a Web Service and it retrieves details of execution of services from the process log of the Web server.

The first few executions of a trial, called a burn-in phase, are run solely by the community. This phase helps Wisp set a base level of trust for the Web services used. Test users are added to the community after the burn-in phase. Their reputations are calculated based on the divergence between their and the community‘s belief densities of positive experience with the Web service. The number of available compositions in this experiment is limited to one otherwise there will be a need to have burn-in phases for each available composition.

After executing a composition, the updated trust vector was computed as mentioned in Section 4.3 and 4.4. Once the belief densities of the services are updated, the composite trust vector of the composition in the system was computed as shown in Section 3.



(a) User has a static reputation through the experiment



(b) User has a dynamic reputation through the experiment

Figure 5.3: Trust proportions of Web services across trials with different types of users

In Fig. 5.3 (a) and (b), the trust proportions of the deployed compositions with test users in the system is shown. Fig. 5.3 (a) shows the scenario where the test users' behaviors are static (either good or bad during all the trials). Fig. 5.3 (b) shows the scenario where the test users behaviors are dynamic (change from being good to bad and vice versa). Twenty trials, averaged over five iterations, were executed during which each composition was executed four times. The empirically observed QoS parameters were averaged over these four runs.

As shown in Fig. 5.3 (a), the trust proportion of a Web Service Composition in a system of ideal users with good reputations steadily rises and reaches a maximum peak that far exceeds the peak of all other experiments (of Figs. 5.3 (a) and 5.3 (b)). This is because the user gives accurate feedback. Therefore, compositions that perform better are selected with progress of time. The trust proportion of a Web Service Composition in a system of a dishonest or naïve user falls immediately upon the arrival of the user. The trust proportion soon starts rising with a few dips along the way. The dips further along in the curve are smaller than the initial dip (that occurs when the test user enters the system). This set of experiments show that Wisp was able to gauge the reputation of the test user and was reducing the impact of the feedbacks of a user with a bad reputation, while considering the feedbacks of a good user in entirety. Furthermore, the peak of trust proportion curve of Web Service Compositions in system where the test user is dishonest or naïve is lesser than the case where the test user is both honest and expert.

As shown in Fig. 5.3 (b), the trust proportion of a Web Service Composition in a system where the test user goes from being honest and expert to being otherwise grows initially and then falls when the user‘s behavior changes. However, this dip is not as large as the dip of the trust proportion of Web Service Compositions in the system with a user of low reputation throughout the experiment. This is due to the reason that Wisp started viewing that user favorably during the phase of the user exhibiting a good reputation. Once the user starts defaulting, his/her reputation drops slowly. The trust proportion curve, however, starts rising and peaks because Wisp measures the reputation score of a user after each execution of a

WSC. Therefore, a user who is dishonest or naïve gets a lower reputation score as soon as the user‘s behavior changes. Conversely, the trust proportion of Web Service Compositions falls initially but later rises when the user‘s behavior changes from being bad to good.

Chapter 6

Conclusion

Trust in Web Services is conceptualized using a mathematical model that meets much of our intuition:

(i) It permits modeling our uncertainty about trusting or distrusting Web Services initially when no prior experience with them exist.

(ii) As the number of experiences with a Web Service increases though the proportion of positive experiences, γ may remain fixed, the certainty level increases. The certainty is higher if there are forty two positive experiences out of fifty in comparison to eight positive ones out of ten.

(iii) Furthermore, the certainty is lowest when an equal number of positive and non-positive experiences are obtained. A mechanism to make Wisp robust has been implemented by shielding it from malicious or naïve feedbacks. Reputation model is useful because as the malicious users enter the system, the trust proportion initially decreases but later on increases as Wisp starts to counter the effect of dishonest and naive users.

While the approach in this thesis for updating trust considers past experiences, some potential refinements are discussed below:

(i) One may wish to vary the importance of the past experiences in updating beliefs. One could assign a higher weight to the more recent experiences because they reflect the current trustworthiness of the service better than the historical experiences with the service.

(ii) Bootstrapping the reputation of a user (similar to Skopik *et al.*, 2009) would be helpful in situations where running a burn-in phase is not possible.

(iii) Another potential modification would be to handle collusions among test users of Wisp (similar to Keung & Griffiths[23]).

(iv) Finally, handling dependencies among services in a composition or in scenarios where the services cannot be independently identified would be useful.

Bibliography

[1] Abdul-Rahman, A., & Hailes S. (2000) *Supporting trust in virtual communities.* In Proc. of Hawaii International Conference on System Science, vol. 6, pp. 6007.

[2] Active BPEL. Web: http://www.activevos.com/community-open-source.php

[3] Adam, N. R., Kozanoglu, A., Paliwal, A. V., & Youssef, M. (2006) *Mutual trust in open environment for cascaded web services.* In Proc. of Third ACM Workshop on Secure Web Services, pp. 107–108.

[4] Aliprantis, C. D., & Burkinshaw, O. (1998) *Principles of Real Analysis.* Third Edition. Academic Press.

[5] Anderson, S., Bohren, J., Boubez, T., Chanliau, M., Della-Libera, G., Dixon, B., Garg, P., Gudgin, M., Hallam-Baker, P., Hondo, M., Kaler, C., Lockhart, H., Martherus, R., Maruyama, H., Nadalin, A., Nagaratnam, N., Nash, A., Philpott, R., Platt, D., Prafullchandra, H., Sahu, M., Shewchuk, J., Simon, D., Srinivas, D., Waingold, E., Waite, D., Walter, D., & Zolfonoon, R. (2005) *Web Services Trust Language (WS-Trust).* Web: http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf

[6] Beth, T., Borcherding, M., & Klein, B. (1994) *Valuation of trust in open networks.* In Proc. of European Symposium on Research in Computer Security, pp. 3–18.

[7] Blaze, M., Kannan, S., Lee, I., Sokolsky, O., Smith, J., Keromytis, A., & Lee, W. (2009) *Dynamic Trust Management.* Computer, vol. 42, no. 2, pp. 44–52.

[8] Conner, W., Iyengar, A., Mikalsen, T., Rouvellou, I, & Nahrstedt, K. (2009) *A Trust Management Framework for Service-Oriented Environments.* In Proc. of Eighteenth International Conference on World Wide Web, pp. 891–900.

[9] Cardoso, J., Sheth, A., Miller, J., Arnold, J., & Kochut, K. (2004) *Quality of service for workflows and WS processes.* Web Semantics: Science, Services and Agents on the World Wide Web, vol. 1, pp. 281–308.

[10] Dustdar, S., & Schreiner, W. (2005) *A Survey of Web Services Composition.* Intl. J. of Web and Grid Services, vol. 1, no. 1, pp. 1–30.

[11] Fullam, K., & Barber, S. (2007) *Dynamically learning sources of trust information: experience vs. reputation.* In Proc. of Sixth International Joint Conference on Autonomous Agents & Multiagent Systems, pp. 1055–1062.

[12] Gambetta, D. (1998) *Can we trust trust?* In D. Gambetta, editor, Trust: Making and Breaking Cooperative Relations, pp. 213–237. Basil Blackwell.

[13] Glen, A., Leemis, L., & Drew, J. (2004) *Computing the distribution of the product of two continuous random variables.* Computational Statistics and Data Analysis, vol. 44, no. 3, pp. 451–464.

[14] Golbeck, J., & Hendler, J. (2004) *Reputation Network Analysis for Email Filtering.* In Proc. of Conference on Email and Anti-Spam.

[15] Grishchenko, V. (2001) *Redefining Web-of-Trust: reputation, recommendations, responsibility and trust among peers.* Web: http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/redefining_web_of_trust/

[16] Hafizoglu, F., & Yolum, P. (2009) *Composing Trust: An Effective Trust Model for Multiagent Teamwork.* In Proc. of Workshop on Trust in Agent Societies, Seventh International Conference on Autonomous Agents & Multi-Agent Systems, pp. 27–42.

[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I.H. (2009) *The WEKA Data Mining Software: An Update.* SIGKDD Explorations, vol. 11, no. 1.

[18] Hang, C., & Singh, M. (2009) *Selecting Trustworthy Service in Service-Oriented Environments.* In Proc. of Workshop on Trust in Agent Societies, Seventh International Conference on Autonomous Agents & Multi-Agent Systems, pp. 43–54.

[19] Jaynes, E. T. (1979) *Where do we stand on maximum entropy.* In Levin and Tribus, editors, The Maximum Entropy Formalism, pp. 15–118. MIT Press.

[20] Jøsang, A. (1998) *A subjective metric of authentication.* In Proc. of European Symposium on Research in Computer Security, pp. 329–344.

[21] Jøsang, A., Hayward, R., & Pope, S. (2006) *Trust Network Analysis with Subjective Logic.* In Proc. of Twenty Ninth Australasian Computer Science Conference, pp. 85–94.

[22] Katz, Y., & Golbeck, J. (2006) *Social Network-based Trust in Prioritized Default Logic.* In Proc. of Twenty First AAAI Conference on Artificial Intelligence, pp. 1345–1350.

[23] Keung, S., & Griffiths, N. (2009) *Building a Trust-based Social Agent Network.* In Proc. of Workshop on Trust in Agent Societies, Seventh International Conference on Autonomous Agents & Multi-Agent Systems, pp. 68–79.

[24] Liu Y., Ngu A. H., & Zeng L. (2004) *Qos computation and policing in dynamic web service selection.* In Proc. of Thirteenth International World Wide Web Conference, pp. 66–73.

[25] Malik, Z. & Bouguettaya, A. (2007) *Evaluating rater credibility for reputation assessment of web services.* In Proc. of Eighth International Conference on Web Information Systems Engineering, pp. 38–49.

[26] McKnight, D. H., & Chervany, N. L. (1996) *The meanings of trust.* Technical report, Management Information Systems Research Center, University of Minnesota.

[27] Mui, L., Mohtashemi, M., & Halberstadt, A. (2002) *A Computational Model of Trust and Reputation.* In Proc. of Thirty Fifth Hawaii International Conference on System Science, vol. 7, pp. 188.

[28] Olmedilla, D., Rana, O. F., Matthews, B., & Nejdl, W. (2005) *Security and trust issues in semantic grids.* In Proc. of Dagstuhl Seminar on Semantic Grid: The Convergence of Technologies, pp. 05271.

[29] Paradesi, S., Doshi, P., & Swaika, S. (2009) *Integrating Behavioral Trust in Web Service Compositions.* In Proc. of IEEE International Conference on Web Services, pp. 453–460.

[30] Rohatgi, V. K. (1976) *Introduction to Probability Theory and Mathematical Statistics.* John Wiley & Sons.

[31] Russell, S. & Norvig, P. (2002) *Articial Intelligence: A Modern Approach.* Second Edition. Prentice Hall.

[32] Salehi-Abari, A., & White, T. (2009) *Detecting and Dealing with Naive Agents in Trust-aware Societies.* In Proc. of Workshop on Trust in Agent Societies, Seventh International Conference on Autonomous Agents & Multi-Agent Systems, pp. 117–128.

[33] Sen, S., Malone, N., & Chakraborty, K. (2009) *Comprehensive Trust Management.* In Proc. of Workshop on Trust in Agent Societies, Seventh International Conference on Autonomous Agents & Multi-Agent Systems, pp. 129–146.

[34] Singh, M. P. (2002) *Trustworthy service composition: Challenges and research questions.* In Proc. of Autonomous Agents and Multi-Agent Systems Workshop on Deception, Fraud and Trust in Agent Societies, Springer-Verlag.

[35] soapUI. Web: http://www.soapui.org

[36] Wang, Y. & Singh, M. P. (2007) *Formal trust model for multiagent systems.* In Proc. of Twentieth International Joint Conference on Artificial Intelligence, pp. 1551–1556.

[37] Wikipedia - Dirichlet Distribution.
Web: http://en.wikipedia.org/wiki/File:Dirichlet_distributions.png

[38] Xu, Z., Martin, P., Powley, W., & Zulkernine, F. (2007) *Reputation-enhanced qos-based web services discovery.* In Proc. of IEEE International Conference on Web Services, pp. 249–256.

[39] Yu, B., Singh, M. P. (2000) *A Social Mechanism of Reputation Management in Electronic Communities.* In Proc. of Fourth International Workshop on Cooperative Information Agents, pp. 154–165.

[40] Zacharia, G., Moukas, A., & Maes, P. (1999) *Collaborative Reputation Mechanisms in Electronic Marketplaces.* In Proc. of Hawaii International Conference on System Science, vol. 8, pp. 8026.