An Approach to Incentive-based Reputation for Communities of Web Services

Babak Khosravifar[1], Jamal Bentahar[1], Philippe Thiran[2], Ahmad Moazin[1], and Adrien guiot[2]
[1]Concordia University, Canada, [2]University of Namur, Belgium
$b\_khosr@encs.concordia.ca, bentahar@ciise.concordia.ca, pthiran@fundp.ac.be$
$a\_moazi@encs.concordia.ca, aguiot@student.fundp.ac.be$

Abstract

Community of web services (CWS) is a society composed by a number of functionally identical web services. The communities always aim to increase their reputation level in order to obtain more requests. In this paper, we propose an effective mechanism dealing with reputation assessment for communities of web services. The proposed mechanism is based on after-service feedbacks provided by the users to a run-time logging system. The proposed method defines the evaluation metrics involved in reputation assessment of a community, and supervises the logging system in order to verify the validity and soundness of the feedbacks provided by the users. In this paper, the proposed framework is described, a theoretical analysis of its assessment and its implementation along with empirical result discussions are provided. We also show how our model is efficient, particularly in very dynamic environments.

Keywords. *Web Service, Community, Reputation.*

1 Introduction

Literature Review. As one of the recent technologies for developing loosely-coupled, cross-enterprize business processes (usually referred to as B2B applications), a plethora of web services exists on the web waiting to receive users' requests for processing. This continuous choice is usually reputation-driven. In literature, the reputation of web services have been intensively stressed [10]. In [1], the authors have developed a framework aiming to select web services based on the trust policy expressed by the users. The framework allows the users to select a web service matching their needs and expectations. In [2], Malik et al. have proposed a model to compute the reputation of a web service according to the personal evaluation of the previous users. The characteristic of this method is that the credibility of the users evaluating the web service is taken into account. If the rater tries to provide a fake rating, then its credibility will be decreased and the rating of this user will have less importance in the reputation of the web service. In [4], the authors have designed a multi-agent framework based on an ontology for QoS. The users' ratings according to the different qualities are used to compute the reputation of the web service. In [6, 10], some web services reputation mechanisms have been proposed, that would lead to an effective service selection, and in [5], service-level agreements are discussed in order to set the penalties over the lack of QoS for the web services. In general, in all the mentioned models, web services are considered to act individually and not in collaboration with other web services.

Recently, there have been few attempts to address the formation and reputation of Communities of Web Services (CWSs) [7, 13]. The main property of a CWS is to facilitate and improve the process of web service selection and effectively regulate the process of user requests. There are underlying reasons for this. First, the community gathers a set of functionally homogeneous web services regardless of who developed them, where they are located, and how they function. Given that some communities offer the same functionality (hotels booking, weather forecasting, etc.), there is a competition between different communities. In this case, reputation is considered as a differentiation driver of the communities. Moreover, reputation helps users to select the most reputable community, which would provide the best QoS, and helps providers to join the best community, which would bring them the most value. In [7], *Elnaffar et al.* propose a reputation-based architecture for CWSs and classify the involved metrics that affect the reputation of a community. They derive the involved metrics by processing some historical performance data recorded in a run-time logging system. The purpose is to be able to analyze the reputation in different points of view, such as users to CWSs, CWSs to web services, and web services to CWSs. The authors discuss the affect of different factors while diverse reputation directions are analyzed. However, they do not derive the overall reputation of a CWS from the proposed metrics. Moreover, they assume that the run-time logging mechanism is an accurate source of information.

Proposed Model. In this paper, we extend the work done in [7] by two contributions. In the first contribution,

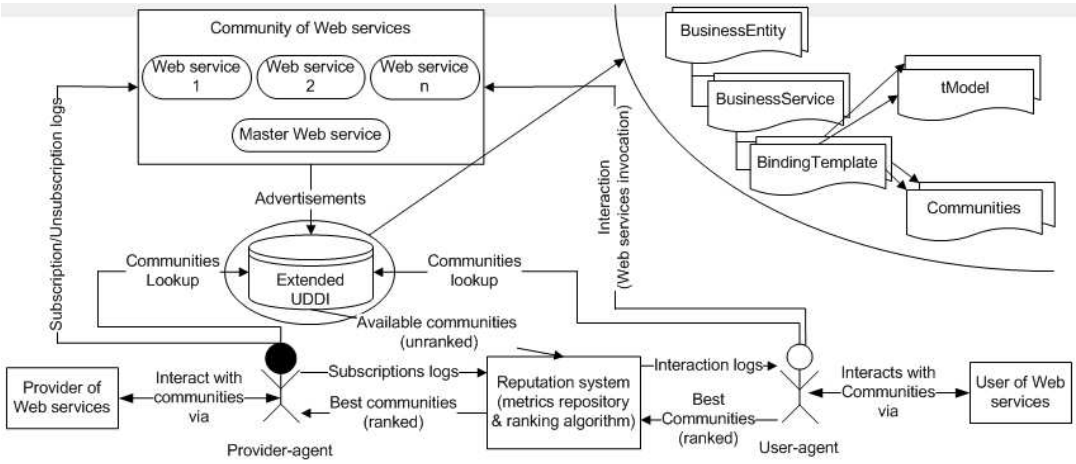

Figure 1. Architecture of reputation-based CWS

we propose a reputation model of a community of web services, which is based on involved metrics (responsiveness, inDemand, satisfaction and time recency). This model is used by users and providers to estimate the reputation of a community. In the second contribution, we discuss more on the feedback logging mechanism and give a reliable mechanism (capable of managing malicious acts of agents). We assume that the CWS may be encouraged to violate such run-time logging mechanism in support of themselves or against other communities. To this end, we discover the points of violation in the sense that the controller agent Cg (the agent, that is assigned to monitor the logging data and introduced in Section 4) to some extent, makes sure that the violation is taken place. Then we propose a method to properly react for such violations. We provide a theoretical analysis based on backward induction to prove that there is an incentive for communities not to violate the logging system. In this analysis, we derive the comparative values of reward and penalties for CWSs in order to obtain such an incentive. The simulations results reveal how, empirically, our trust model yields a system that autonomically adjusts the level of CWS's reputation.

What specifically distinguishes our model from other similar works in the literature is: (1) its sound formation of the reputation assessment for the CWSs; and (2) its incentive-based reputation adjustment in the sense that although the communities are capable of misleading the logging system in support of themselves (or against their opponents), they will not take the risk to do that, given the fact that they are aware of possible consequent penalty that would decrease their current reputation level. The intuitions behind the incentive-based mechanism are: (1) we obtain an accurate information for deriving the involved metrics used for the reputation of a particular community; and (2) we obtain an overall higher reliability and efficiency in the sense that upon violation detection, CWSs are strictly encouraged to show an acceptable performance in their further user request processes. This factor is analytically proved in Section 4.3 and experimentally shown in Section 5.

Organization. The remainder of this paper is organized as follows. In Section 2, we define the architecture of reputation-embedded CWSs, which is composed of extended UDDI, user and provider agents and reputation system. In Section 3, we discuss the reputation model by its involved metrics and we propose a methodology to combine them. In Section 4, we extend the discussion about maintaining a sound logging mechanism used as source of information for the metrics. We discuss the fake positive and negative corrections and provide the incentive to avoid fake attempts. In Section 5, we represent the simulation and outline the properties of our model in the experimental environment. Finally, Section 6 concludes the paper.

2 Architecture of reputation-embedded Web services Communities

In this section, we represent the CWS architecture [7]. This architecture is designed to maintain the reputation of the communities. Here we assume that each web service has is associated with a community and do not function alone. If a web service is not registered in a community, it could not be invoked by a user. However, a web service can be registered in one of many communities. In figure 1, we represent different components of the architecture, with their reputation and interactions. These components together with their detailed performance are explained as follows:

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

Master agent. This agent is considered as the representative of the community in the sense that it manages the community requests in selecting the proper web service. Meanwhile the master agent hires (or fires) some web services to join (or leave) the community.
Provider agent. Like the user agent, it relates the provider with the extended UDDI, CWS and the reputation system.
Extended UDDI. The traditional UDDI XML schema is based on six types of information, allowing people to have information in order to invoke the web services [8]. In the UDDI registry, we restrict the access of the agents in the sense that user and provider agents only consult the list of masters, whereas the masters have access to the list of the web services in the UDDI registry. By adding this new kind of information concerning the CWSs, we would clarify which CWS a web service belongs to.
Reputation system. Considering the fact that the CWSs could offer the same service, thus they always compete in order to obtain more requests. Therefore, evaluating CWSs is unavoidable for users and providers. To be able to compute the reputation of CWSs, the user and provider agents must gather operational data, reflecting different performance metrics, about the interaction between the user, the provider and the CWS. The user agents should intercept some logs like *Submissions log*, *Response Time log*, *Invocation log*, *Successes log*, *Failure log*, *Recoveries log* and so on. It is important that the user and provider agents are independent parties in order to intercept trusted run-time data about each web service interaction.

The reputation system is the core component in this architecture. Its first functionality is to register the run-time logs; and the second functionality is to rank the communities based on their reputation by using a ranking algorithm. The ranking algorithm would maintain a restrictive policy, avoiding the ranking violation, which could be done by some malicious CWSs. The violation, which has not been considered in [7], is done by providing some fake logging data (by some colluding users) that reflect positive feedback in support of the CWS, or by fake negative data that is registered against a particular community. To deal with this violation, we propose to assign a controller agent Cg. The task of this agent is to update the CWS reputation rankings in order to drop inaccurate registered data and thus enhance accuracy of the reputation system. The detailed discussion of this issue is provided in Section 4.

3 Reputation Model

For simplification reasons, in the remainder of this paper, we only consider the users point of view (rather than users and providers) in reputation assessment. In order to assess the overall reputation of a CWS, the user needs to take some correlated factors into account. In Section 3.1, we present the involved metrics that a user may consider in this assessment. Consequently, in Section 3.2, we explain the methodology that the user uses to combine these metrics in order to assess the reputation of a CWS.

3.1 Metrics

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

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

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

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

$$InD_{C_i}^{[t_1,t_2]} = \frac{Req_{C_i}^{[t_1,t_2]}}{\sum_{k=1}^{M} Req_{C_k}^{[t_1,t_2]}} \quad (2)$$

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

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

$$Sat_{C_i}^{U_j,[t_1,t_2]} = \frac{1}{n}\sum_{t=t_1}^{t_2} Sat_{C_i}^{U_j,R_k^t} \times e^{-\lambda(t_2-t)} \quad (3)$$

3.2 Metrics Combination

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

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

$$Rep_{C_i}^{U_j,[t_1,t_2]} = \eta(1 - \frac{Res_{C_i}^{U_j,[t_1,t_2]}}{\nu}) + \kappa Sat_{C_i}^{U_j,[t_1,t_2]} \quad (4)$$

$$Rep_{C_i}^{[t_1,t_2]} = \chi \frac{1}{m} \sum_{j=1}^{m} \left(Rep_{C_i}^{U_j,[t_1,t_2]} \right) + \phi \, InD_{C_i} \quad (5)$$

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

4 Sound Logging Mechanism

Without loss of generality, in a network composed of CWSs, master agents (as representatives of communities) are selfish and may alter their intentions in order to obtain more benefits (in terms of popularity). This could happen by improving one's reputation level or by degrading other's reputation level. We respectively refer to these cases as fake positive/negative alteration. Violating the logging feedbacks (distracting the reputation levels) could lead to system inconsistency in the sense that low quality CWSs may obtain more users or high quality communities may loose some users. Therefore, it is important to avoid such attacks and keep the logging mechanism accurate. In the rest of this section, we explain how to perform fake positive/negative corrections and thus effectively maintain a reputation adjustment.

In the proposed architecture for the CWS, the reputation is computed based on the information obtained from the logging system that over the elapsing time, users leave their feedbacks. Thus, it is essential to keep such logging file accurate and discourage malicious actions. To this end, a controller agent Cg is assigned that his responsibility is to maintain an accurate attack-resilient logging file. As a part of the UDDI system, Cg has the authority to update information such as overall reputation level of any CWS. Without loss of generality, we assume that this agent is highly secured in order to avoid being compromised. However, if Cg gets compromised with a given community, then inconsistent actions of Cg could be recognized by some other communities, given the fact that they are competing with one another. But this issue is out of the scope of this paper.

4.1 Fake Positive Correction

Fake positive recognition. One of the main responsibilities of the controller agent Cg is to perform fake positive correction. To this end, initially Cg should recognize a malicious behavior from one or a set of user agents (that could possibly collude with a particular community). This recognition is done based on the recent observable change in the reputation of a community. To this end, Cg would always check the recent feedbacks of the communities. So Cg would consider the reputation that is computed for a specific period of time $[t_1 - \epsilon, t_1]$, where t_1 is the current time. The value ϵ is set by the controller agent regarding to the system inconsistency in the sense that if the network is inconsistent, so Cg would need to check most recent feedbacks (ϵ as relatively small amount). Otherwise, Cg would take even older feedbacks into account (ϵ as relatively large amount). So, $Rep_{C_i}^{[t_1-\epsilon,t_1]}$ is the reputation of the community C_i obtained from data measured from $t_1 - \epsilon$ to t_1. Let $U_{C_i}^{[t_1-\epsilon,t_1]}$ be the set of users that during this time interval provided a feedback for this community, and t_b be the beginning time of collecting feedbacks. Cg would consider the positive feedbacks to be suspicious if the reputation improvement ($Rep_{C_i}^{[t_1-\epsilon,t_1]} - Rep_{C_i}^{[t_b,t_1]}$) divided by the number of users that caused such improvement is greater than the predefined threshold ϑ, i.e:

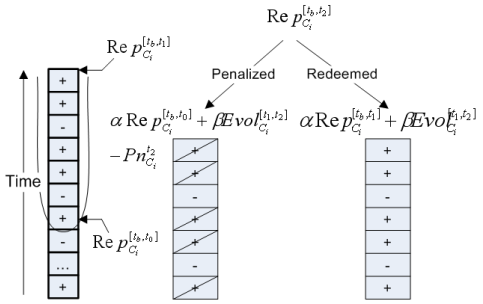

Figure 2. Fake positive correction cases.

$$\frac{Rep_{C_i}^{[t_1-\epsilon,t_1]} - Rep_{C_i}^{[t_b,t_1]}}{|U_{C_i}^{[t_1-\epsilon,t_1]}|} > \vartheta$$

In that case, it is assumed that community C_i had a drastic reputation increase in the recent ϵ time.

Fake positive Adjustment. Exceeding the threshold ϑ, Cg would figure out that a particular community is receiving consequent positives. Then Cg, in order to reload the previous and actual reputation level, would freeze the recent positive logs and notifies the corresponding community of such suspending. So, Cg would observe the upcoming behavior (in terms of satisfaction and responsiveness) of the community in order to match the actual efficiency with the suspended enhanced reputation level. During this period, the community is encouraged to behave in such a way that reflects the suspended enhanced reputation level (see Figure 2). If the community showed the real improved performance, the suspended reputation trust level would be redeemed and considered for his reputation. But if the community failed to do so, the previous reputation level will

be decreased by some applied penalties. In this case, the community would be in such a situation that either has to outperform its past in order to improve the enhanced reputation level, or would loose its current reputation, which is not wanted. Therefore, we form an incentive that communities would not risk their current reputation level and thus they do not by any means (colluding with users or providers) provide fake positives in support of themselves. We assume that t_0 is the start time of leaving fake positives in the logging file, t_1 is the time that Cg recognizes such fake actions and consequently starts investigating the upcoming efficiency of such community, and t_2 is the time that Cg would update the particular community's reputation value. Let $Evol_{C_i}^{[t_1,t_2]}$ be the evolutionary reputation value for the community C_i that is measured by the Cg during specified time interval $[t_1, t_2]$ (investigation period). This value is computed in equation 6, where δ is a small value that the reputation is measurable within $[t-\delta, t]$.

$$Evol_{C_i}^{[t_1,t_2]} = \frac{\sum_{t=t_1+\delta}^{t_2} Rep_{C_i}^{[t-\delta,t]}}{t_2 - t_1} \tag{6}$$

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

$$Rep_{C_i}^{*[t_b,t_2]} = \begin{cases} \alpha Rep_{C_i}^{[t_b,t_1]} + \beta Evol_{C_i}^{[t_1,t_2]}, & \text{if redeemed;} \\ \alpha Rep_{C_i}^{[t_b,t_0]} + \beta Evol_{C_i}^{[t_1,t_2]} - Pn_{C_i}^{t_2} & \text{if penalyzed.} \end{cases} \tag{7}$$

where $\alpha + \beta = 1$.

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

False alarm detection. It is worth to discuss more about alternatives of Cg's fake positives recognition. Consider the two cases that Cg falsely, and truly recognizes the fake positives. In the former case, the positives are real, therefore, they reflect the actual performance of the community. Then even being suspended, the community can easily prove the quality level as it continues as before and basically would not loose anything. In the later case, the positives are fake, so the community needs to improve its actual quality level to prove suspended enhanced reputation level. If the community failed to fulfill such reputation, Cg would decrease its previous reputation level.

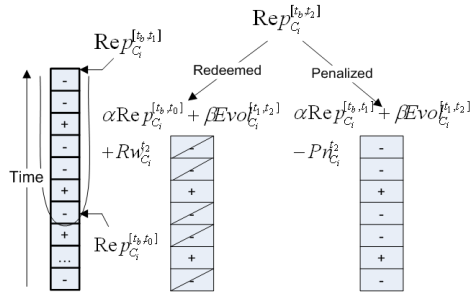

Figure 3. Fake negative correction cases.

4.2 Fake Negative Correction

Similar to fake positive case, there might be some fake negatives in order to decrease the reputation level of a particular community. This could happen when a community or a set of communities would like to weaken a particular community (by dropping its reputation level) hoping not to compete with them. However, one unique case should not be excluded in which, a particular community would mal-behave and after certain number of providing services and obtaining negative feedbacks, claims that the feedbacks were fake and do not reflect its actual reputation level. To avoid such a situation, each community is responsible to recognize a change in its reputation level and consequently report to Cg. Upon received report, Cg would decide whether the negative feedbacks were really as a result of the mal-behavior of the community or as a result of some other parties fake negatives. If Cg initiates the investigation at time t_1, after a period of evolutionary time, Cg would decide for the reputation adjustment at time t_2. In case of redeeming the community C_i that was suspected to have fake negative feedbacks, the negatives are discarded ($Rep_{C_i}^{[t_0,t_1]}$ is not considered), and a reward $Rw_{C_i}^{t_2}$ is assigned at time t_2. The reason is to discourage the opponent communities not to cause a fake negative feedbacks for C_i and hope to degrade its reputation level. However, if after evolutionary investigation, Cg decides to penalize C_i, then the negative feedbacks are also considered (reputation is computed by $Rep_{C_i}^{[t_b,t_1]}$), and a penalty $Pn_{C_i}^{t_2}$ is assigned to the community. Equation 8 computes the updated reputation value of the community C_i ($Rep_{C_i}^{*[t_b,t_2]}$).

$$Rep_{C_i}^{*[t_b,t_2]} = \begin{cases} \alpha Rep_{C_i}^{[t_b,t_0]} + \beta Evol_{C_i}^{[t_1,t_2]} + Rw_{C_i}^{t_2}, & \text{if redeemed;} \\ \alpha Rep_{C_i}^{[t_b,t_1]} + \beta Evol_{C_i}^{[t_1,t_2]} - Pn_{C_i}^{t_2} & \text{if penalyzed.} \end{cases} \tag{8}$$

4.3 Theoretical Analysis

In this section, we would like to discuss in details the updates of reputation level when a particular community C_i

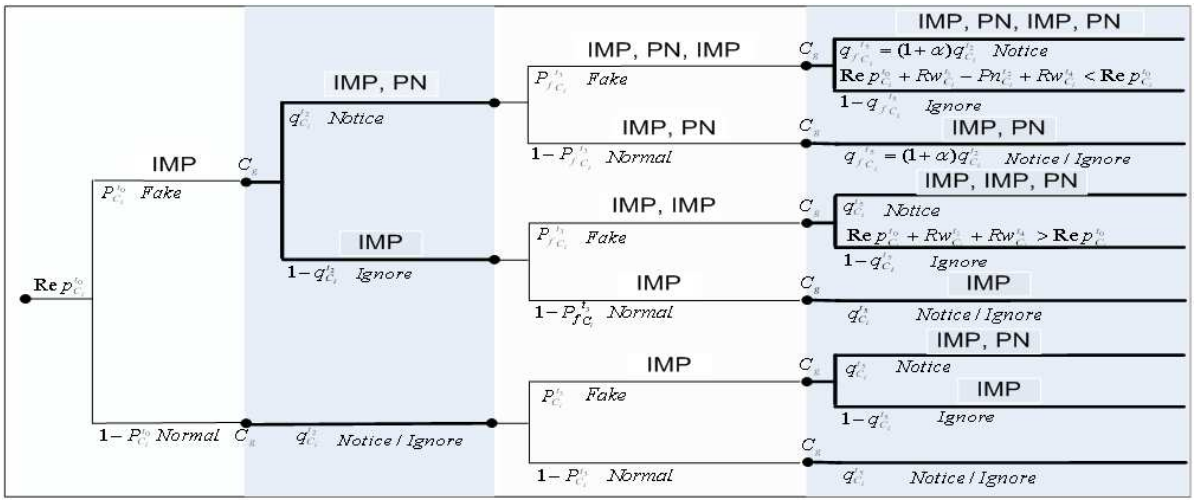

Figure 4. The tree of backward induction reasoning.

causes fake feedbacks that eventually is beneficiary for itself. To this end, we follow the steps over this reputation updates and elaborate Cg's actions on them. For simplicity reasons, here we only analyze the case of self-positive feedbacks and generalize our discussion to fake negative feedbacks. We objectively assume that penalizing a community is relative to the reputation improvement that community had obtained. In this section, we use backward induction reasoning technique to show that CWSs loose interest in doing malicious acts that cause extra (fake) positives for themselves or extra (fake) negatives for some others.

To better analyze the intentions the communities could follow, we calculate the expected reputation value of a particular community in the case that the community acts maliciously to provide fake positive feedbacks for itself and the case that the community acts as normal and performs its actual capabilities. By comparing the two expected values, the typical community C_i will decide either to act maliciously or as normal. Let q^t_{Ci} be the probability that the controller agent Cg notices the real intention of the community C_i and take actions with penalizing C_i at time t. We compute the expected reputation of C_i as a result of a malicious action in equation 9 and as a result of normal action in equation 10.

$$E(Rep^{*[t_b,t_2]}_{C_i}|C_i\ faked) = q^{t_2}_{C_i}(\alpha Rep^{[t_b,t_0]}_{C_i} + \beta Evol^{[t_1,t_2]}_{C_i} - Pn^{t_2}_{C_i}) + (1-q^{t_2}_{C_i})(\alpha Rep^{[t_b,t_1]}_{C_i} + \beta Evol^{[t_1,t_2]}_{C_i}) \quad (9)$$

$$E(Rep^{[t_b,t_2]}_{C_i}|C_i\ not faked) = Rep^{[t_b,t_2]}_{C_i} \quad (10)$$

Figure 4 is the tree representing the backward induction reasoning through actions of the community C_i and corresponding reactions made by the controller agent Cg in two steps. In this figure, IMP is considered as the state that the community has provided some fake positives ad thus improved its reputation level. We also refer in this figure to PN as the state that the community's fake action is detected and thus penalized by Cg. As it is illustrated, the community that provides fake positives, obtains an improvement, which could be followed by a penalty. Here we state that the probability of Cg's detection given the fact that the C_i has faked before is high. Therefore, if C_i has been already penalized, it is so hard to retaliate and improve again. There is a slight chance that C_i fakes and Cg ignores, which comes with a very small probability. Thus, we compute the expected reputation level of both cases and compare them.

Definition 1. *Let $Imp^{[t_b,t_2]}_{C_i}$ be the difference between the adjusted reputation (in the case where the community is under investigation) and normal reputation (in the opposite case) within $[t_b, t_2]$, i.e:*

$$Imp^{[t_b,t_2]}_{C_i} = Rep^{*[t_b,t_2]}_{C_i} - Rep^{[t_b,t_2]}_{C_i}$$

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

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

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

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

The obtained value is positive, so we are done. □

Table 1. Simulation summarization over the obtained measurements.

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

5 Experimental Results

In this section, we describe the implementation of a proof of concept prototype. In the implemented prototype, CWSs are composed of distributed web services, that are implemented as $Java^{©TM}$ agents. The agent reasoning capabilities are implemented as Java modules. The testbed environment is populated with two agent types: (1) service provider agents that are known as web services and gathered in a community (we assume only one type of service is provided and therefore consumed); and (2) user agents that are seeking for the best service provided by a web service. In general, the simulation consists of a series of empirical experiments tailored to show the adjustment of the CWS's reputation level. During the elapsing RUNs, web services, masters and users (that are initially activated) build their private knowledge. Table 1 represents three types of CWSs we consider in our simulation: ordinary, faker and intermittent. Ordinary community acts normal and reveals what it has, the faker community is the one that provides fake feedbacks in support of itself, and the intermittent community is the one that alternatively changes its strategies over the time. CWSs contain a number of web services, that offers a quality of Service (QoS). As it is shown in table 1, the QoS value is divided into three ranges.

In each RUN, a number of users are selected to search for the best service. Strictly speaking, users are only directed to ask CWSs for a service and thus user would not find out about the web service that is assigned by the master of the community. In order to find the best community, the requesting user would evaluate the CWSs regarding to their reputation level. Some times, the users are in contact with some communities that are very good for the user, so the users re-select them. If the user is rejected from the best selected community, he would ask the second best (and so on) community in terms of reputation level. After getting a response from a community, the user agent would provide a feedback relative to the quality of the obtained service and the community responsiveness. The feedbacks are logged in the logging mechanism that is supervised by Cg. The accumulated feedbacks would affect the reputation level of communities. In other words, the communities would loose their users if they receive negative feedbacks, by which their reputation level is dropped.

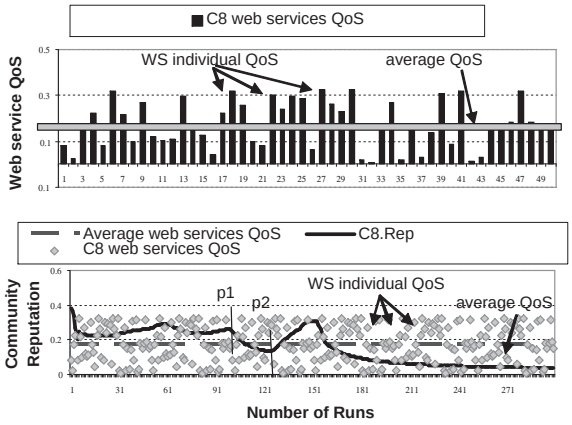

Figure 5. Communities overall quality of service vs. the number of simulation RUNs.

Taking into account the general incentive for the CWSs to process most possible users, communities in general, compete to increase their reputation level. This is done by colluding with a user (or a small group of users) to provide consecutive positive feedbacks. In the empirical experiment, we are interested to observe the over-RUN reputation level of different types of communities and how fast and efficient the adjustment is performed by the Cg. Figure 5 illustrates the plot of reputation level for a faker community $C8$. The upper plot represents the individual QoS for the community's assigned web services. In this plot the gray line defines the average QoS for the web services. The most prominent feature of the plot is the comparison of the reputation level with the average of the community web services QoS. The average value is assumed to be the actual QoS for the community. In general, there would be convergence to such value if the community is acting in an ordinary manner (for $C8$ is 0.173). The lower plot illustrates the reputation level of this community over the elapsing RUNs. Here we notify that the master of a community is responsible to assign the web services to the user requests. To this end, normally the high quality web services are assigned first until they become unavailable, which forces the master agent to assign other lower quality web services. Thus starting the RUNs, $C8$ gains reputation value (up to 0.313), which is better than its individual average quality of service. In figure 5 the peek $P1$ defines the RUN in which the community $C8$ is out of high quality web services. After passing this point, the reputation level of this community is decreased.

Figure 6 illustrates community $C8$ reputation level in comparison with an ordinary community $C6$. $C8$ at point $P3$ decides to provide fake positive feedbacks for himself to increase self reputation level. For the interval of 19 RUNs, this community gains higher reputation level up to the point $P4$. The controller agent Cg, periodically verifies the feed-

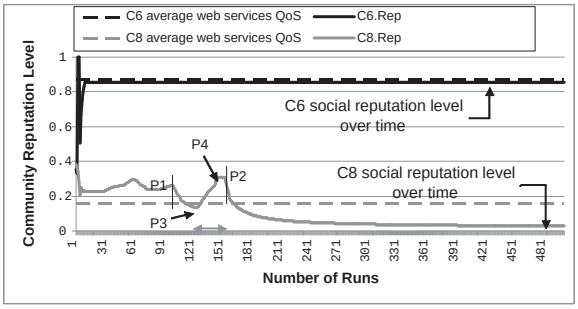

Figure 6. Communities overall quality of service vs. the number of simulation RUNs.

back logs, in order to recognize the malicious actions. At $P4$ the controller agent Cg notices the malicious act of $C8$ and freezes the obtained feedbacks in order to decide to penalize. Peek $P2$ is the point in which the community $C8$ is penalized in his reputation level. After $P2$ a drastic decrease in reputation value is seen which goes underneath $C8$'s average quality of service (up to 0.112). There is also a continuing but slower increase for the reputation of the faker community $C8$ that persists long after the first fake action recognition. Thus, there appear to be strong restriction effects, in which eventually the faker communities loose their users. However, there is also an ongoing effect of social influence, in which users doubt in communities that have drastic decrease in their reputation level.

6 Conclusion

The contribution of this paper is the proposition of a new incentive-based reputation model for community of web services gathered to facilitate dynamic users requests. The reputation of the communities are independently accumulated in binary feedbacks reflecting the satisfaction or dissatisfaction of the users being serviced by a web service assigned via the master of the community. The model represents a sound logging mechanism in order to maintain effective reputation assessment for the communities. The controller agent investigates the logging feedbacks released by the users to detect the fake feedbacks as a result of collusion between a community and a user (or a group of users), which are provided in support of the community. Upon detection, the controller agent maintains an adjustment in the logging system, so that the malicious community would be penalized in its reputation level.

Our model has the advantage of providing a suitable metrics used to assess the reputation of a community. Moreover, the assessed metrics are valid having a sound logging mechanism in the sense that communities would obtain the incentive not to fake and act as their ordinary capabilities. The proposed mechanism efficiency is analyzed through a defined testbed. Our objective for future work is to advance the assessment model to enhance the model efficiency. In the logging system, we need to optimize detection process, trying to formulate it in the sense to be adaptable to diverse situations. Finally, we plan to extend the empirical analysis in order to capture more results reflecting the proposed model capabilities.

References

[1] A.S. Ali, S.A. Ludwig, and O.F. Rana. A cognitive trust-based approach for web service discovery and selection. Proc. of the 3'rd European Conf. on WS, pp. 38-40, USA, ECOWS 2005.

[2] Z. Malik and A. Bouguettaya. Evaluating rater credibility for reputation assessment of web services. 8'th Int. Conf. on Web Information Systems Engineering (WISE 2007), 14(2-3), Nancy, France, Decr 2007.

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

[4] E. M. Maximilien. Multiagent system for dynamic web services selection. Proc. of 1'st Workshop on Service-Oriented Computing and Agent-Based Eng., pp 2529, SOCABE2005.

[5] R. Jurca, B. Faltings, and W. Binder. Reliable QoS monitoring based on client feedback. Proceedings of the 16th International World Wide Web Conference (WWW07), pp. 1003-1011, Banff, Canada, May 8-12 2007.

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

[7] S. Elnaffar, Z. Maamar, H. Yahyaoui, J. Bentahar, Ph. Thiran. Reputation of communities of web services - preliminary investigation. Proc. of the 22'nd IEEE Int. Conf. on Advanced Inf. Networking and App., pp. 1603-1608, AINA2008.

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

[9] W. Yao, J.Vassileva. A Review on trust and reputation for web service selection. 1'st Int. Workshop on Trust and Reputation Management in Massively Dis. Comp. Sys. TRAM2007.

[10] S. Kalepu, S. Krishnaswamy, S. W. Loke. A QoS metric for selecting Web services and providers. Proceedings. Fourth International Conference on Web Information Systems Engineering Workshops, pp. 131-139, 2003.

[11] E. M. Maximilien, M. P. Singh. Toward automatic web services trust and selection. Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC), New York, November 2004.

[12] M. Ruth, and T. Shengru. Concurrency Issues in Automating RTS for Web Services. IEEE International Conference on Web Services, pp. 1142-1143, ICWS 2007.

[13] J. Bentahar, Z. Maamar, D. Benslimane, and Ph. Thiran. An Argumentation Framework for Communities of Web Services. In IEEE Intelligent Systems, 22(6): 75-83, 2007.