International Journal of Web Services Research, Vol.7, No.3, 2010

Analyzing Communities of Web Services Using Incentives

Babak Khosravifar[1], Jamal Bentahar[1], Ahmad Moazin[1], and Philippe Thiran[2]
[1]Concordia University, Canada
[2]University of Namur, Belgium
b_khosr@encs.concordia.ca, bentahar@ciise.concordia.ca, a_moazi@encs.concordia.ca,
pthiran@fundp.ac.be

ABSTRACT:

This paper aims to propose an effective mechanism dealing with reputation assessment of communities of web services (CWSs) that are known as societies composed by a number of functionally identical web services. The objective is to provide a general incentive for CWSs to act truthfully given that they are allowed to decide about their actions.The considered entities (web services, virtual organizations, etc.) are designed as software autonomous agents equipped with advanced communication and reasoning capabilities. User agents request CWSs for services and accordingly rate their satisfactions about the received quality and community responsiveness. The strategies taken by different parties are private to individual agents. The logging file that collects feedback is investigated by a controller agent. Furthermore, the accurate reputation assessment is achieved by maintaining a sound logging mechanism. To this end, the incentives for CWSs to act truthfully are investigated and analyzed, which allows the controller agent to keep the logging file accurate. The proposed framework defines the evaluation metrics involved in the reputation assessment of a community, and supervises the logging system in order to verify the validity and soundness of the feedback provided by the users. In this paper, the proposed framework is described, a theoretical analysis of its assessment and its implementation along with discussion of empirical results are provided. We also show how our model is efficient, particularly in very dynamic environments.

KEY WORDS:
Web services, communities, incentives, trust, reputation, autonomous agents

INTRODUCTION

As one of the recent technologies for developing loosely-coupled, cross-enterprize business processes (usually referred to as B2B applications), a plethora of web services exists on the web waiting to receive users' requests for processing. Such requests are usually competitive in a a security and reputation-driven environment (Martino and Bertino 2009; Zhang, 2008). To this end, the reputation assessment has been addressed in recent proposals (Jurca and Faltings 2003; Jurca and Faltings 2007; Liu, Ngu et al. 2004). One general solution for such reputation assessment is collection of the after-interaction feedback that users provide with respect to the quality of the received service. However, in feedback-based reputation mechanisms, the precise reputation assessment needs to be verified. Selfish web services might manage to provide feedbacks that support them in the reputation mechanism. In general, online reputation mechanism is always subject to get violated with selfish web services. Another way to address the selection (and management) problem is to gather web services having similar functionalities to a community. Community of web services (CWSs) is a gathering of single and functionally similar web services that are aggregated to perform as one community while offering unique or variety services. The main property of a CWS is to facilitate and improve the process of web service discovery and selection and effectively regulate the process of user requests. There are underlying reasons for

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

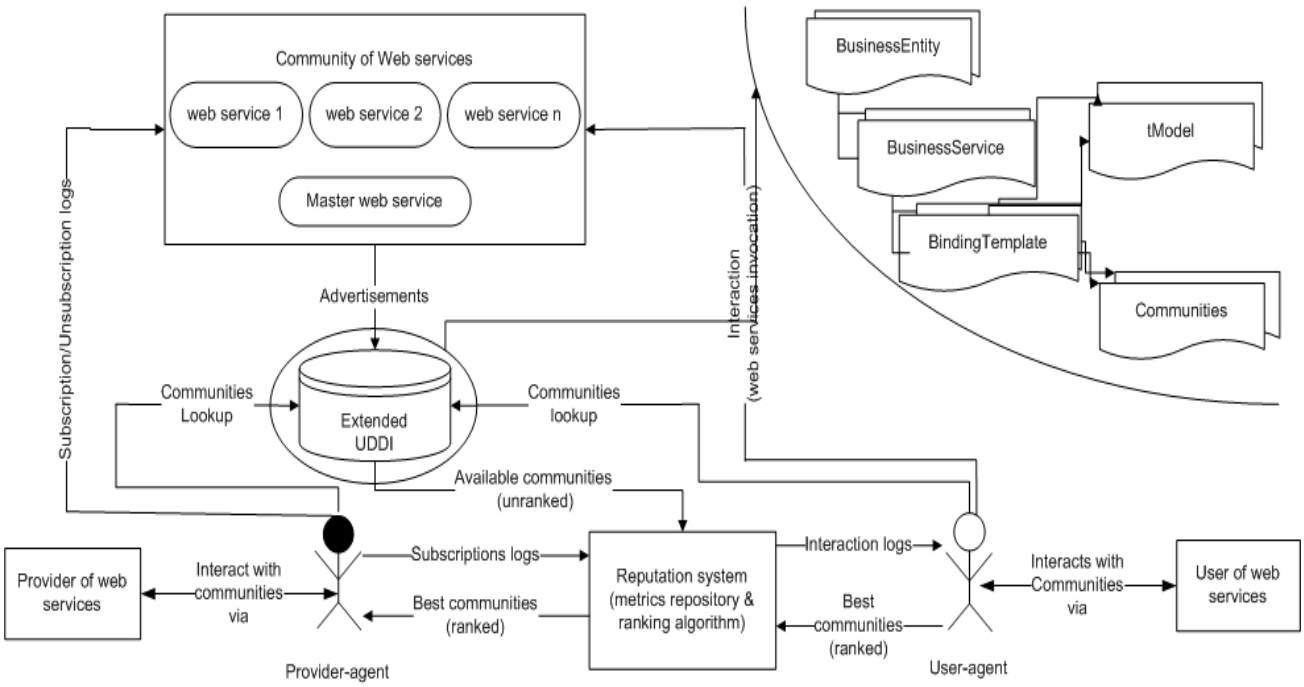


Figure 1: Architecture of reputation-based CWSs

Proposed Model. In this paper, we advance our previous work (Khosravifar, Bentahar et al. 2009) by providing more theoretical and practical results and discussions. Indeed in this paper, we extend the work done in (Elnaffar, Maamar et al. 2008) by two contributions. In the first contribution, we propose a reputation model of a community of web services, which is based on involved metrics (responsiveness, inDemand, satisfaction that has been defined in (Elnaffar, Maamar et al. 2008)). These factors are redefined here in a different way by considering the time factor we call time recency. This model is used by users and providers to estimate the reputation of a community. In the second contribution, we discuss more the feedback logging mechanism and give a reliable mechanism (capable of managing malicious acts of communities). We assume that CWSs may be encouraged to violate such run-time logging mechanism in support of themselves or against other communities. To this end, we try to discover feedback violations using the controller agent Cg (the agent that is assigned to monitor the logging data and

introduced in Section 4) that 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. The idea is to prove that communities gain more if they do not violate the logging system compared to when they violate it. In this analysis, we derive the comparative values of reward and penalties for CWSs in order to obtain such an incentive. The simulation 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 (Weaver and Wu 2006; Jurca and Faltings 2003; Jurca and Faltings 2007; Jurca, Faltings et al. 2007) 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 distracting 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. In this paper, we prove that the best strategy for CWSs is to act truthfully. The advantages of using 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 and experimentally confirmed.

Organization. The remainder of this paper is organized as follows. First, we define the architecture of reputation-embedded CWSs, which is composed of extended UDDI, user and provider agents and reputation system. Then, we discuss the reputation model by its involved metrics and propose a methodology to combine them. Afterwards, 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 the next section, we present the simulation and outline the properties of our model in the experimental environment. The subsequent section discusses the related work and the last section concludes the paper.

ARCHITECTURE OF REPUTATION-EMBEDDED WEB SERVICE COMMUNITIES

In this section, we represent the CWSs architecture (Elnaffar, Maamar et. al. 2008), which is designed to maintain the reputation of the communities. Here we assume that each web service is associated with a community and do not function alone. If a web service is not registered in a community, it could not be invoked by a user. Indeed, a web service can be registered in one of many communities. In Figure 1, we represent different components of the architecture, with their reputation and interactions. These components together with their detailed performance are explained as follows:

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

Master agent. This agent is considered as the representative of the community in the sense that it manages the community requests in selecting the proper web service. Meanwhile, the master agent hires (or fires) some web services to join (or leave) the community. In general, the master

of the community always tends to increase the community's performance and consequently, its reputation level.

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

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

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

The reputation system is the core component in this architecture. Its first functionality is to register the run-time logs; and the second functionality is to rank the communities based on their reputation by using a ranking algorithm. The ranking algorithm would maintain a restrictive policy, avoiding the ranking violation, which could be done by some malicious CWSs. The violation, which has not been considered in (Elnaffar, Maamar et al. 2008) could be done by providing some fake logging data (by some colluding users) that reflect positive feedback in support of the CWS, or by fake negative data that is registered against a particular community. To deal with this violation, we propose to assign a controller agent Cg . The task of this agent is to update the CWS reputation rankings in order to drop inaccurate registered data and thus enhance accuracy of the reputation system. The detailed discussion of this issue is provided in Section 4.

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

REPUTATION MODEL

For simplification reasons, but without loss of generality, 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.

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

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

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

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

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

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

Satisfaction Metric: Let Sat^{U_j,R^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^t sent at time t to C_i. Equation 3 shows the overall satisfaction of the user U_j to community C_i.

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

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's point of view. In this equation, ν represents the maximum possible response time, so that if a community does not respond, we would have $Res^{U_j,[t_1,t_2]} = \nu$. In the second part, the *inDemand* metric is added. Therefore, the overall reputation of C_i from the users' point of view is obtained in equation 5.

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

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

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

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 correction (recognition and adjustment) and thus effectively maintain a reputation adjustment.

In the proposed architecture for the CWS, the reputation is computed based on the information obtained from the logging system that over the elapsing time, users leave their feedbacks. Thus, it is essential to keep such logging file accurate and discourage malicious actions. It is the responsibility of the controller agent Cg to maintain an accurate attack-resilient logging file. As a part of the UDDI system, Cg has the authority to update information such as overall reputation level of any CWS. In this paper, 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.

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 $[t_1-\varepsilon, t_1]$, 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). Thus, $Rep^{[t_1-\varepsilon,t_1]}$ is the reputation of the community C_i obtained from data measured from $t_1-\varepsilon$ to t_1. Different values of ε will be used in the simulation (see section 5) to observe the effect of the considered period on the overall recognition.

Let $U^{[t_1-\varepsilon,t_1]}$ be the set of users that during this time interval have provided a feedback for the community C_i, and t_b be the beginning time of collecting feedbacks. Cg would consider the positive feedbacks to be suspicious if the reputation improvement ($Rep^{[t_1-\varepsilon,t_1]} - Rep^{[t_b,t_1]}$) divided by the number of users that caused such improvement is greater than the predefined threshold ϑ, i.e:

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

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

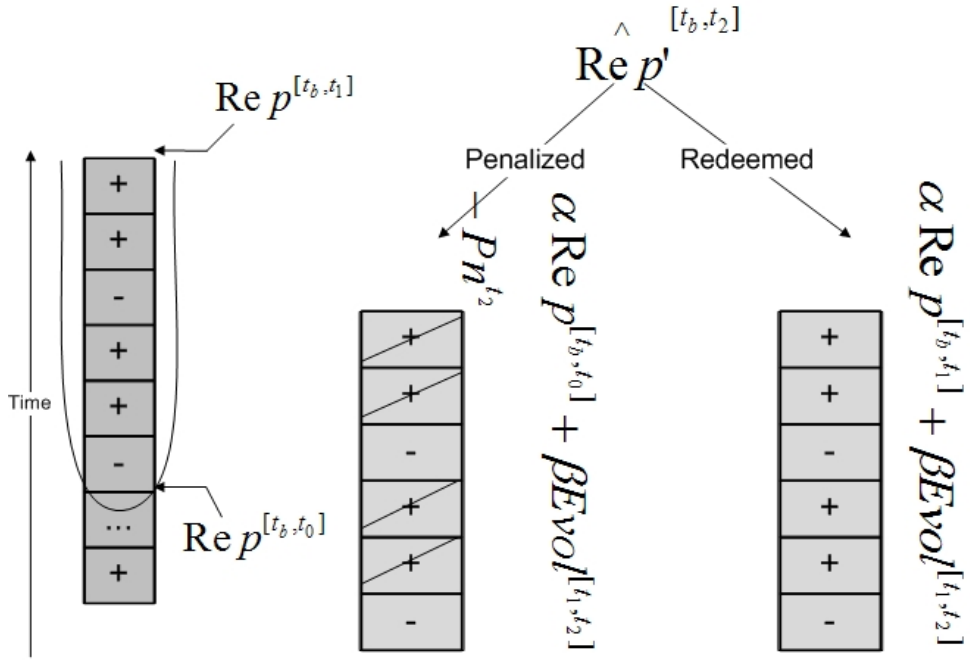


Figure 2: Fake positive correction cases

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. As it is shown in Figure 2, the community's feedback is recognized as suspicious at time t_1. Feedbacks from time t_0 are freezed to investigate the further behavior of the suspicious community C_i. At time t_2 controller agent Cg would decide whether to penalyze community C_i or to redeem the freezed feedbacks. If the community showes the real improved performance, the suspended reputation trust level would be redeemed and considered for his reputation. But if the community failes to do so, the previous reputation level will be decreased by some applied penalties. In this case, the community would be in such a situation that either has to outperform its past in order to improve the enhanced reputation level, or would loose its current reputation, which is not wanted. Therefore, we form an incentive that communities would not risk their current reputation level and thus they do not by any means (colluding with users or providers) provide fake positives in support of themselves. Let $Evol^{[t_1,t_2]}$ be the evolutionary reputation value for the community C_i that is measured by the Cg during specified time interval $[t_1,t_2]$ (investigation period). This value is computed in equation 6, where δ is a small value such that the reputation is measurable within $[t-\delta,t]$.

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

Also, let Pn^t be the general penalty value that is assigned by Cg to C_i at a specific time t. Equation 7 computes the adjusted reputation level of C_i ($\mathrm{Re}\, p'^{[t_b,t_2]}$). This equation reflects the incentive we propose, so that CWSs in general would be able to analyze their further reputation adjustments upon fake action.

$$\mathrm{Re}\, p'^{[t_b,t_2]} = \begin{cases} \alpha Rep^{[t_b,t_1]} + \beta Evol^{[t_1,t_2]} & if\ redeemed; \\ \alpha Rep^{[t_b,t_0]} + \beta Evol^{[t_1,t_2]} - Pn^{t_2} & if\ penalyzed. \end{cases} \quad (7)$$

where $\alpha + \beta = 1$.

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

False Alarm Detection

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

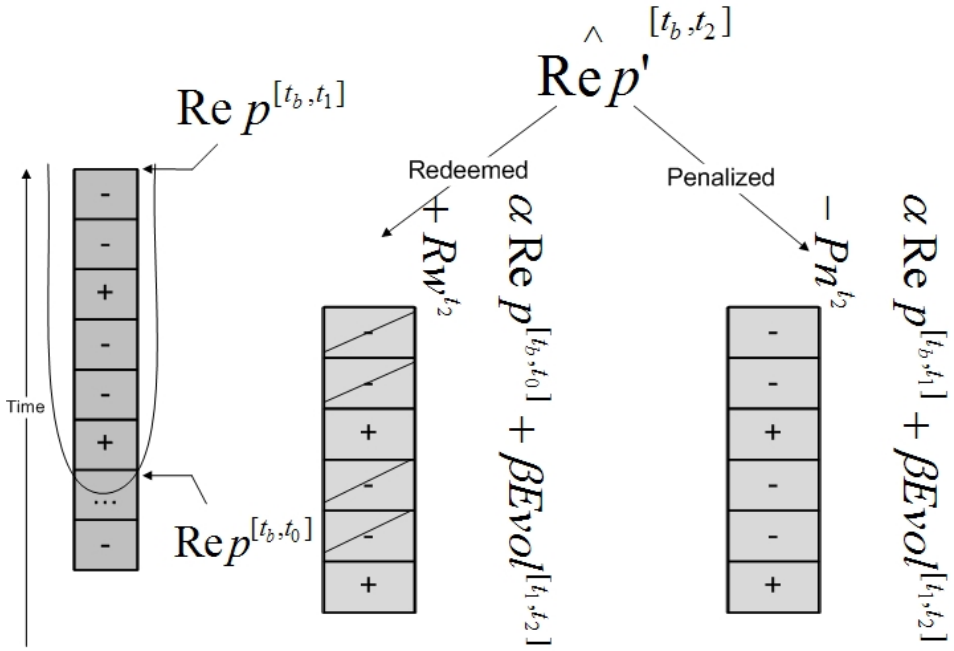


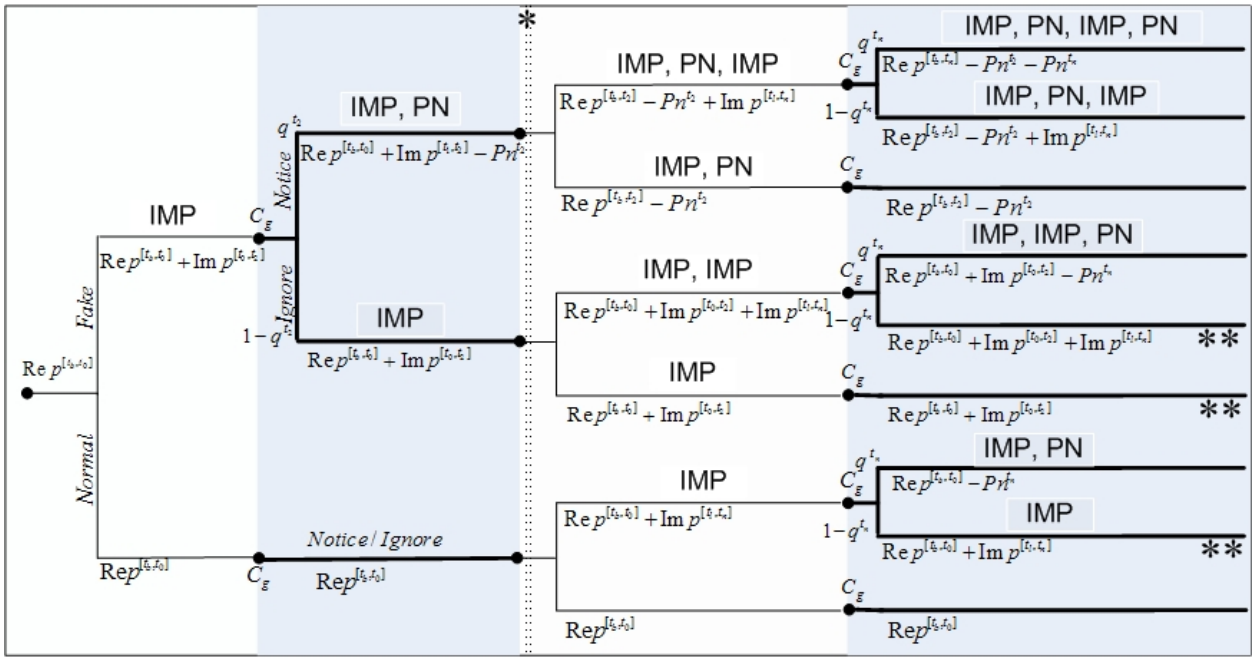
Figure 3: Fake negative correction cases

Fake Negative Correction

Similar to the fake positive case, there might be some fake negatives in order to decrease the reputation level of a particular community (see Figure 3). This could happen when a community or a set of communities would like to weaken a particular community (by dropping its reputation level) hoping not to compete with them. However, one unique case should not be excluded in which, a particular community would mal-behave and after certain number of providing services and obtaining negative 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 the case 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 ($Rep^{[t_0,t_1]}$ is not considered), and a reward Rw^{t_2} is assigned at time t_2. The reason is to discourage the opponent communities not to cause a fake negative 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 (by considering $Rep^{[t_b,t_1]}$), and a penalty Pn^{t_2} is assigned to the community. Equation 8 computes the updated reputation value of the community C_i ($\mathrm{Re}\, p'^{[t_b,t_2]}$).

$$\text{Re } p^{[t_b,t_2]} = \begin{cases} \alpha Rep^{[t_b,t_1]} + \beta Evol^{[t_1,t_2]} + Rw^{t_2} & if\ redeemed; \\ \alpha Rep^{[t_b,t_0]} + \beta Evol^{[t_1,t_2]} - Pn^{t_2} & if\ penalized. \end{cases} \quad (8)$$

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



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

Figure 4: The tree of backward induction reasoning

Theoretical Analysis

In this section, we will discuss in details the updates of reputation level when a particular community C_i causes fake feedbacks that is eventually beneficiary for itself. To this end, we follow the steps over this reputation updates and elaborate Cg 's actions on them. For simplicity reasons, we only analyze the case of self-positive 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 decisions the communities could take, we calculate the expected reputation value of a particular community in the case that the community acts maliciously to provide fake positive 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. As discussed earlier, this decision is made based on the probability that C_i estimates to have a successful act. Being malicious, C_i always looks for the cases that could possibly cheat to increase its current reputation. Let q^t be the probability that the controller agent Cg notices the real intention of the community C_i and take actions with penalizing C_i at time t. We compute the expected reputation of C_i as a result of a malicious action in equation 9 and as a result of normal action in equation 10. In these equations, the expected value of the reputation for community C_i is measured under two assumptions. In the case that C_i has faked the feedbacks ($E(\mathrm{Re}\, p'^{[t_b,t_2]} \mid C_i \ faked)$), the community decides to fake at time t_0 (therefore, the reputation till t_0 is considered as normal), the biased feedbacks are recognized by Cg at time t_1, and the investigation is finalized at time t_2. To this end, by penalizing C_i, its previous reputation till t_0 is considered together with the investigation period $[t_1,t_2]$ with its penalty. If the controller agent Cg does not recognize C_i's malicious act, all the feedbacks are taken into account. In this analysis, we consider a very low possibility that Cg warns false negatives, which is the case that Cg falsely recognizes a malicious act. To this end, we assume that if the community C_i acts as normal, the reputation value would be measured as normal.

$$\begin{aligned} &E(\mathrm{Re}\, p'^{[t_b,t_2]} \mid C_i \ faked) = \\ &\quad q^{t_2}(\alpha Rep^{[t_b,t_0]} + \beta Evol^{[t_1,t_2]} - Pn^{t_2}) \\ &\quad + (1-q^{t_2})(\alpha Rep^{[t_b,t_1]} + \beta Evol^{[t_1,t_2]}) \end{aligned} \tag{9}$$

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

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 refers to the fact that the community's reputation is getting improved thanks to fake positives the community has provided. We also refer in this Figure to PN as the state that the community's fake action is detected and thus penalized by Cg. As it is illustrated, the community that provides fake positives, obtains an improvement, which could be followed by a penalty. Here we state that the probability of Cg's detection given the fact that C_i has faked before is high. Therefore, if C_i has been already penalized, it is so hard to retaliate and improve again. There is

a slight chance that C_i fakes and Cg ignores, which comes with a very small probability. Thus, we compute the expected reputation level of both cases and compare them.

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

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

The following proposition gives the condition for the penalty to be used, so that the communities will not act maliciously. If $Pn^{t_2} > \frac{1}{q^{t_2}} Imp^{[t_b,t_2]} - \alpha Rep^{[t_0,t_1]}$, then communities obtain less reputation value if they act maliciously and provide fake feedbacks for themselves.

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

$$E(Rep^{[t_b,t_2]} \mid C_i \ Not \ faked) - E(\text{Re } p'^{[t_b,t_2]} \mid C_i \ faked) =$$
$$Pn^{t_2} - \frac{1}{q^{t_2}} Imp^{[t_b,t_2]} + \alpha Rep^{[t_0,t_1]}$$

The obtained value is positive, so we are done.

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

Assume that C_i has made its malicious act at time t_{l1}. For the performed action, there is a chance ($q^{t_{n1}}$) that the controller agent Cg noticed the act at time t_{n1} and thus, penalized the community by $Pn^{t_{n1}}$. We also consider the chance ($1-q^{t_{n1}}$) that the controller agent ignores the act and thus, the community has obtained the improvement $Imp^{[t_{l1},t_{n1}]}$ through the feedbacks without any

penalty from the controller agent. Considering the probabilities of different strategies that the controller agent may take, as we discussed earlier, there is a small chance that Cg ignores the malicious act. This basically means the probability of notice (for the first time) ($q^{t_{n1}}$) is normally high and that is because the sensitivity of the controller agent in investigating the list of feedbacks for each particular community. However, once recognized, the controller agent becomes more sensitive to the recognized community's further actions. Therefore, the probability of missing the second fake action is less than the first one and so on (($q^{t_{n2}} > q^{t_{n1}}$)). Generally speaking, the community would be more interested to continue its malicious behavior when it has never been recognized via Cg and thus penalized. However, there is always a high possibility for this community to be recognized later (for the first time).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$Pn^{t_n} < Pn^{t_{n'}} \quad n' < n \tag{13}$$

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

EXPERIMENTAL RESULTS

In this section, we describe the implementation of a proof of concept prototype. In the implemented prototype, CWSs are composed of distributed web services (*Java*$^{©TM}$ agents). The agent reasoning capabilities are implemented as Java modules. The 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. 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. As it is shown in table 1, the QoS value is divided into three ranges.

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]

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

responsiveness. The 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.

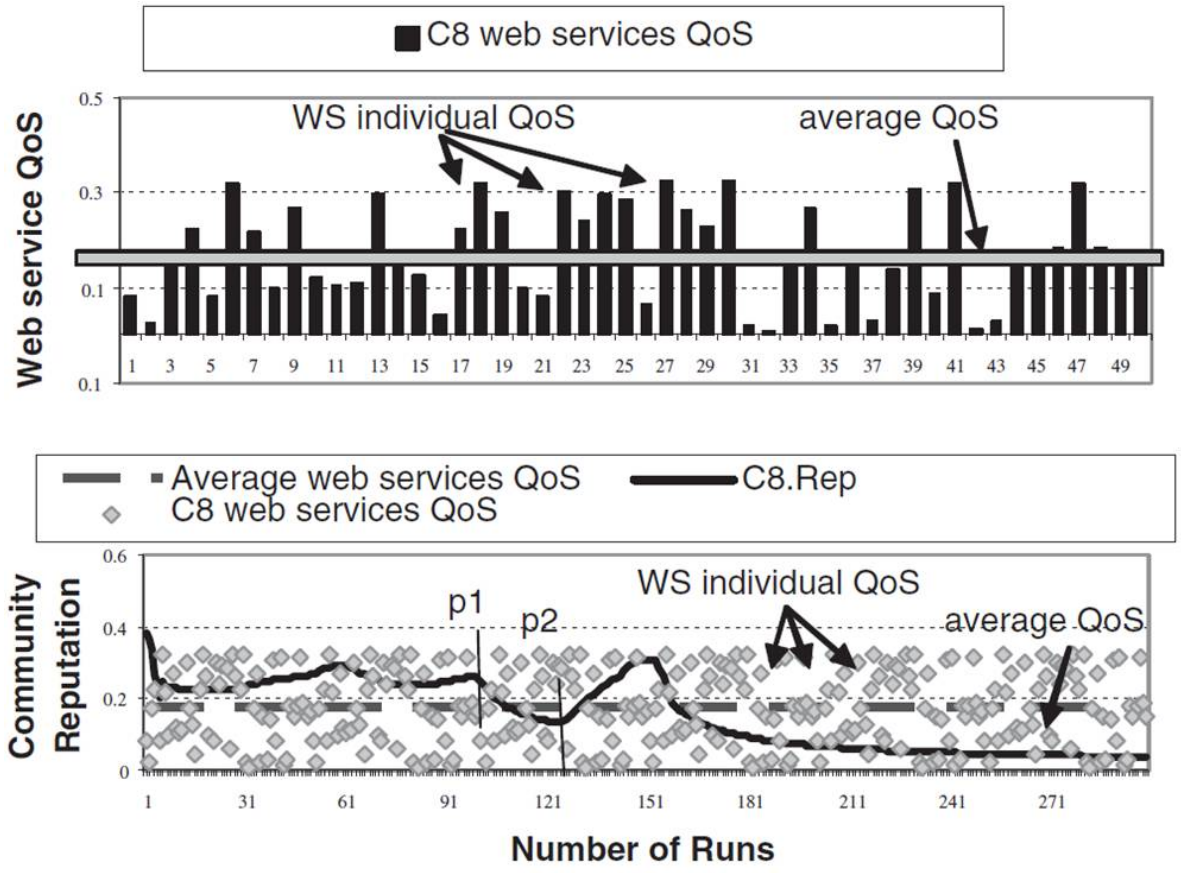


Figure 5: Communities overall quality of service vs. the number of simulation RUNs

Considering the general incentive of CWSs to attract most possible users, communities in general, compete to increase their reputation level. Cheating on reputation level is done by colluding with a user (or a small group of users) to provide consecutive positive feedbacks in support of the malicious (faker) community. In the empirical experiment, we are interested observing the over-RUN reputation level of different types of communities and how fast and efficient the adjustment is performed by Cg. Figure 5 illustrates the plot of reputation level for a faker community $C8$. The upper plot represents the individual QoS for the community's assigned web services. In this plot the gray line defines the average QoS for the web services. The most prominent feature of the plot is the comparison of the reputation level with the average of the community web services QoS. The average value is assumed to be the actual QoS for the community and thus, community's reputation level. In general, there would be convergence to such value if the community is acting in an ordinary manner (for $C8$ is 0.173). The lower plot illustrates the reputation level of this community over the elapsing RUNs. Here we notify that the master of a community is responsible to assign the web services to the user requests. To this end, normally

the high quality web services are assigned first until they become unavailable, which forces the master agent to assign other lower quality web services. Thus starting the RUNs, $C8$ gains reputation value (up to 0.313), which is better than its individual average quality of service. In Figure 4 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 30 RUNs, this community gains higher reputation level up to the point $P4$. The controller agent Cg, periodically verifies the feedback logs, in order to recognize the malicious actions. At $P4$ the controller agent Cg notices the malicious act of $C8$ and freezes the obtained feedbacks for investigation. 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.

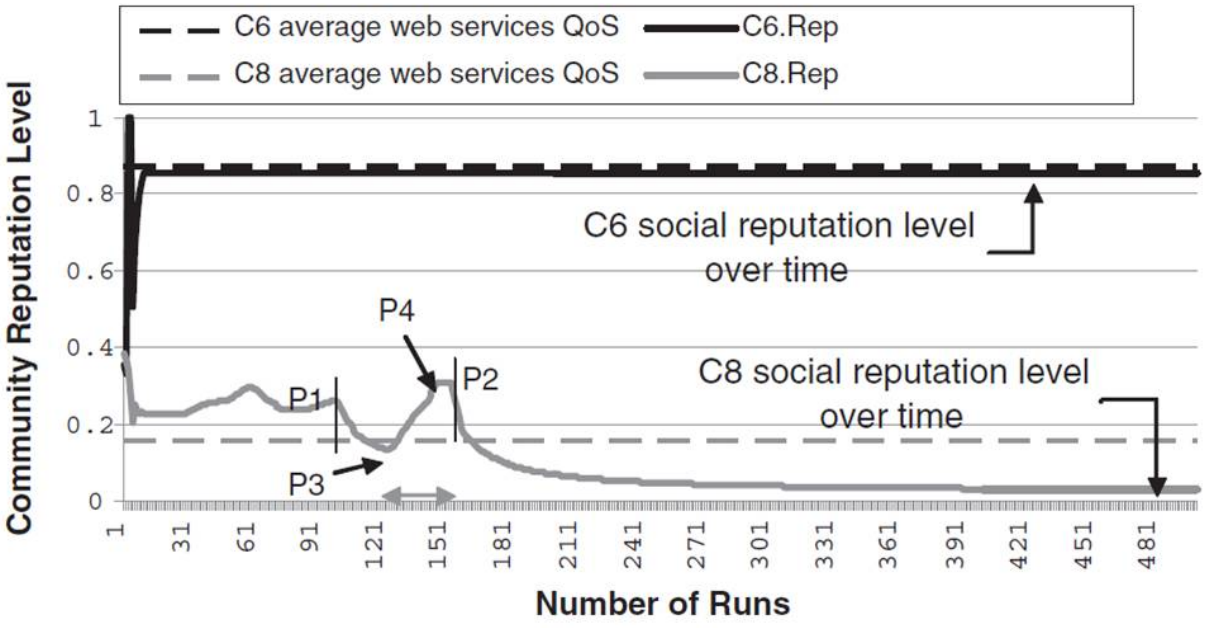


Figure 6: Communities overall quality of service vs. the number of simulation RUNs

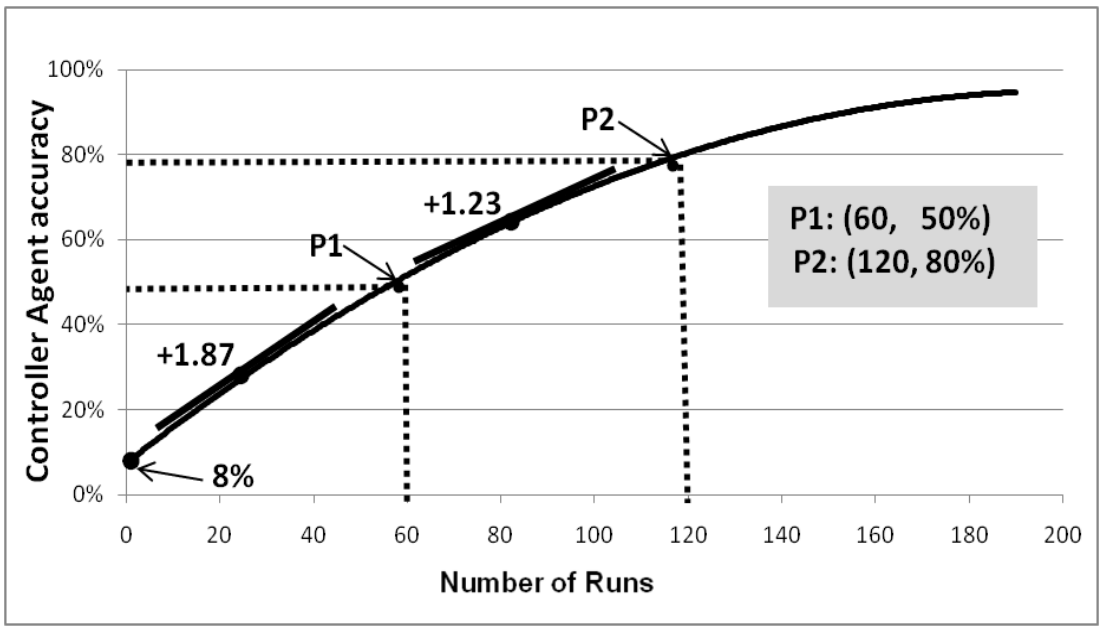


Figure 7: Controller agent *Cg* 's accuracy in detection vs. the number of simulation RUNs

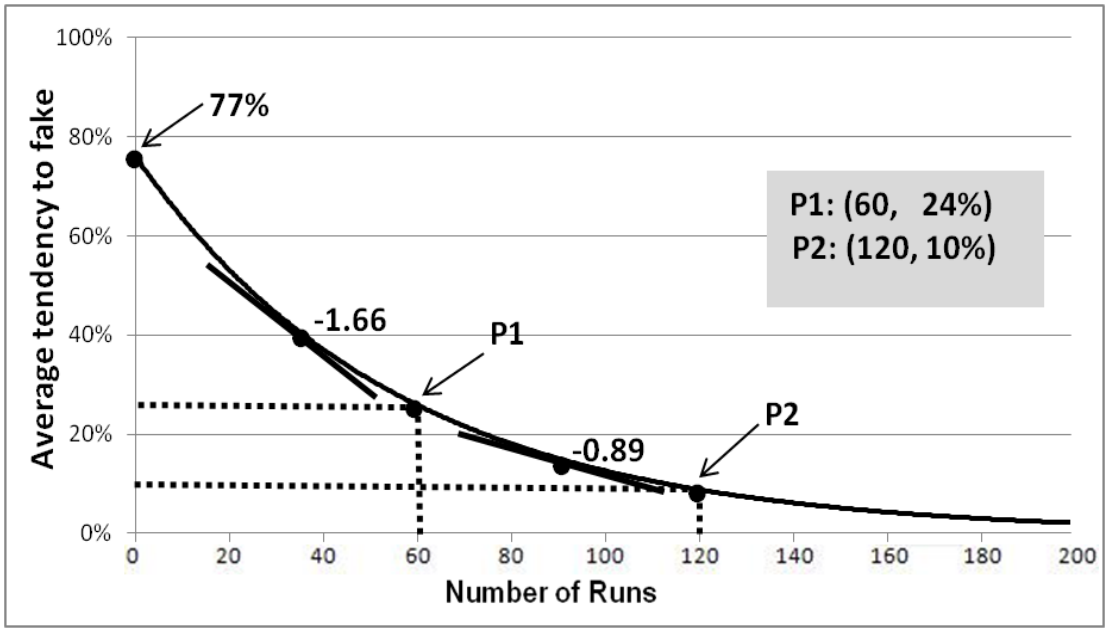
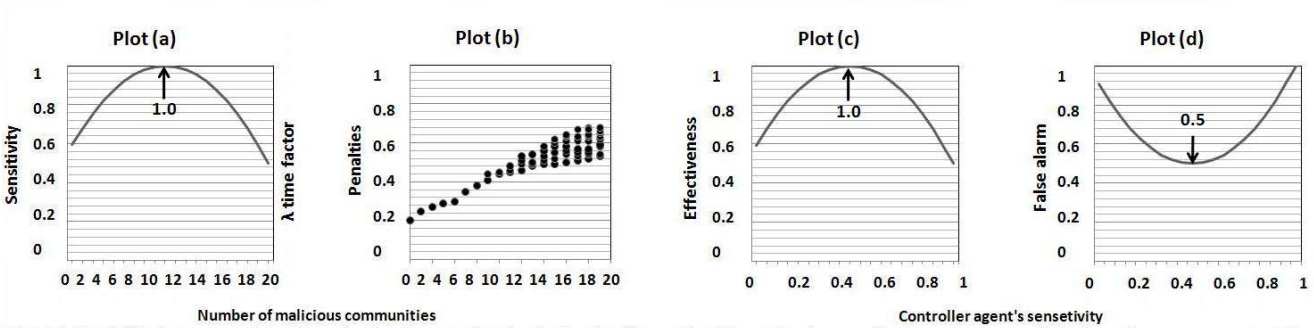


Figure 8: Communities' tendency to fake vs. the number of simulation RUNs

We continue our discussion in more details by analyzing some parameters related to the controller agent's performance and accuracy. One of the main factors in such a system is the accuracy of the controller agent in fake detection. The controller agent is supposed to investigate the feedbacks and recognize the malicious acts while the requesting users provide their rates as feedbacks to

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



Plot (a). Sensitivity is computed for the ratio that Cg goes deep in the logging file. λ is the parameter to set the sensitivity.

Plot (b). Penalty is set with respect to the environment accuracy. The smooth slope for the penalty increases as +0.03.

Plot (c). Effectiveness is the extent to which detections are correct. False positive and false negative influence the total effectiveness.

Plot (d). False alarm is the sum of false positive and false negative.

Figure 9: Controller agent's characteristic analysis

To this end, one of the main objectives is to enhance the efficiency of the controller agent to decrease the false alarm ratio and strengthen the logging feedback crawling algorithm. Figure 7 shows the controller agent's accuracy over the elapsing RUNs while the recognized communities are penalized and thus, discouraged to redo the fake actions. As shown in this Figure, the controller agent is relatively less accurate in detection during the initial RUNs. Basically, detection weakness would highly encourage the faker and intermittent communities to do fake actions. Mostly as a result of the reward that they obtain without the penalty. Basically, the accuracy of Cg is increased while Cg acts successfully in detecting and thus, penalizing faker communities. Cg would act better over the Runs since previously detected communities are investigated more carefully and thus, the chance of failing to detect is decreasing.

In Figure 8 we discuss this issue as we observe the tendency of the communities to provide fake feedbacks in support of themselves. In this Figure, the vertical axes plots the average percentage of the intermittent communities that might be encouraged to fake and the horizontal axes plots the RUNs, which reflects the elapse of time. In this Figure, the average tendency to fake is decreasing as the number of intermittent agents that are penalized are increasing.

We take a narrower analysis on the characteristics of the controller agent Cg and their impact that eventually influence the incentive of different communities to act maliciously. To this end, we study the aforementioned issues towards the network condensity and the extent to which the

controller agent is crawling the feedbacks. In the former study, the idea is to observe how dealing with different malicious communities make the controller agent sensitive to get suspicious while crawling the feedbacks. Basically, the controller agent sets the threshold ϑ in section 4.1 by observing the number of malicious communities in the environment. This means that the controller agent tries to get more though when the number of malicious communities are increasing (see $Plot\,(a)$ in figure 9). However, this harsh manner could not be kept on since Cg cannot keep track of all communities at the same time. On the other hand, by getting suspicious for any community, the false positive ratio would be going up, which reflects the low efficiency of Cg in terms of detection performance. Following the idea that Cg tries to avoid the increase of the malicious communities, we observe that this agent increases the average penalty value assigned to malicious communities while their number is increasing. $Plot\,(b)$ assigns a dot point to each community that gets penalized. The dot points are getting more condense, which shows their high number.

In the second part of the Figure 9, we study the efficiency of the controller agent versus its sensitivity. Since we analyzed the threshold that is set to declare Cg 's sensitivity, here we study how well Cg can act with different thresholds. $Plot\,(c)$ sketches a graph that shows a parabola for the effectiveness of Cg . In this graph there is a tradeoff between the false positive and false negative errors. At a low sensitivity period, there are high number of false negatives. This basically encourages the malicious communities to highly redo their malicious acts as they distract in the logging file and increase their reputation and do not get penalized afterwards. To this end, the observed slope for the effectiveness is relatively small. There is a maximum point for the effectiveness, but this is not always true and may change depending on the environment and surrounded communities. Therefore, we cannot finalize the controller agent's efficiency to a specific value. $Plot\,(d)$ is depicting the same problem from another point of view. Indeed, in this plot we study the false alarm in spite of effectiveness. The false alarm is computed as the sum of false positive and false negative ratios. In this plot, the total false detections is minimized once the controller agent reaches its maximum efficiency. Likewise the decreasing slope is so slow.

RELATED WORK

In the literature, the reputation of web services has been intensively stressed (Kalepu, Krishnaswamy et al. 2003; Maximilien, 2002; Jurca and Faltings 2003; Jurca and Faltings 2007; Liu, Ngu et al. 2004) aiming to facilitate and automate the good service selection. In (Ali, Ludwig et al. 2005), the authors have developed a framework aiming to select web services based on the trust policies expressed by the users. The framework allows the users to select a web service matching their needs and expectations. In (Weaver and Wu 2006), the authors propose an indirect trust mechanism aimed at establishing trust relationships from extant trust relationships with privacy protection. In (Malik and Bouguettaya 2007), the authors proposed to compute the reputation of a web service according to the personal evaluation of the previous users. In general, the common characteristic of these methods is that the reputation of the web service is measured by a combination of data collected from users. To this end, the credibility of the user that provides this data should be taken into account. There should be a mechanism that recognizes the biased rates provided from the users and accordingly updates the credibility of the users. If the user tries to provide a fake rating, then its credibility will be decreased and the rating of this user will have less importance in the reputation of the web service. In (Maximilien, 2005), the author 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 (Jurca, Faltings et al. 2007; Jurca and Faltings 2007), service-level agreements are discussed in order to set the penalties over the lack of QoS for the web services. In general, in all the mentioned models, web services are considered to act individually and not in collaboration with other web services. In such systems, the service selection process is very complicated due to their relatively high number in the network. In addition, web services can easily rig the system by leaving and joining the network when they better off to do so (i.e. when a their reputation is fall off for some reason). This is a rational incentive for such web services that manage to start as new once they have shown a low efficiency. Meanwhile it is hard to manage the huge number of data in web services settings. Considering these inefficiencies, we focused more on the concept of gathering web services together so that we could address the problem of facing web services individually. Communities are in general aimed to get stronger and more publicized in the system, so they do not resign and register as new. In such methodology, users interconnect with the community as the service provider and there would be a web service assigned through the community.

Regarding the aforementioned issue, there have been some proposals that try to gather web services and propose the concept of community-based multi-agent systems (Elnaffar, Mammar et al. 2008; Kastidou, Cohen et al. 2009; Fourguet, Larson et al. 2006). In (Elnaffar, Maamar et al. 2008), the authors propose a reputation-based architecture for CWSs and classify the involved metrics that affect the reputation of a community. They derive the involved metrics by processing some historical performance data recorded in a run-time logging system. The purpose is to be able to analyze the reputation in different points of view, such as users to CWSs, CWSs to web services, and web services to CWSs. The authors discuss the effect of different factors while diverse reputation directions are analyzed. However, they do not derive the overall reputation of a CWS from the proposed metrics. Failing to assess the general reputation for the community leads to failure in efficient service selection. Moreover, authors assume that the run-time logging mechanism is an accurate source of information. In general, in open reputation-feedback mechanisms, always the feedback file is subject to be the target by selfish entities. To this end, the feedback mechanism should be supervised and its precise assessment should be guaranteed. In (Kastidou, Cohen et. al 2009), the authors proposed a framework that explores the possibilities that the active communities act truthfully and provide their actual information upon request. This method is related to the ideas proposed in this paper in the sense that the communities are provided of the incentives that push them to act truthfully. However, in (Kastidou, Cohen et al. 2009), the concept of anonymity is not resolved and the registered communities are to be known in the system to manage a stable framework. In (Fourguet, Larson et al. 2006), a layered reputation assessment system is proposed mainly addressing the issue of anonymity. In this work, the focus is on the layered policies that are applied to measure the reputation of different types of agents, specially the new comers. Although, the proposed work is nice in terms of anonymous reputation assessment, but the layered structure does not optimally organize a community-based environment that gathers web services and also the computational expenses seems to be relatively high.

To address the aforementioned problems, we elaborate in this paper on the reputation mechanism that is supervised by the controller agent and based on the incentives provided to encourage more truthful actions. What mainly distinguishes our proposed model from the related work in the literature is its detailed focus on the logging mechanism accuracy and reputation assessment. The reputation system is observed by the controller agent but still communities are allowed to take any policy that they get the most benefit from. The incentive-based system provides a mechanism that guarantees the least fake actions since communities that gain benefit from malicious acts are eventually penalized such that their further decisions are altered. In this work, the concept of anonymity is also barely observed since the infrastructure is based on communities. This means

that the users face communities for their requested service and the concept of join or leave does not involve users. This mechanism maintains a better quality reputation management and control.

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 of the users being serviced by the communities. 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, having a sound logging mechanism, the communities would obtain the incentive not to act maliciously. 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 using a comprehensive approach we developed in (Khosravifar, Gomrokchi et al. 2009), which considers the trust issue as an optimization problem. In the logging system, we need to optimize detection process, trying to formulate it in order to be adaptable to diverse situations. Finally, we plan to extend the empirical analysis to capture more results reflecting the proposed model capabilities.

ACKNOWLEGMENT

We would like to thank the anonymous reviewers for their valuable comments and suggestions. The second author is partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC), Fonds québécois de la recherche sur la nature et les technologies (NATEQ), and Fonds québécois de la recherche sur la société et la culture (FQRSC).

REFERENCES

Ali, A.S., Ludwig, S.A., Rana, O.F. (2005). A Cognitive Trust-based Approach for Web Services Discovery and Selection, *Proceeding of the 3rd European Conference on Web Services (ECOWS)*, Växjö-Sweden, November 14-16, 38-40.

Elnaffar, S., Maamar, Z., Yahyaoui, H., Bentehar, J., Thiran, P. (2008). Reputation of Communities of Web Services - Preliminary Investigations, *Proceeding of the 22nd IEEE international Conference on Advanced information networking and application (AINA)*, Okinawa-Japan, March 25-28, 1603-1608.

Fourguet, E., Larson, K., Cowan, W. (2006). A Reputation Mechanism for Layered Communities, *SIGecom Exchanges*, 6(1), 11-22.

Jurca, R., Faltings, B. (2003). An Incentive Compatible Reputation Mechanism, *Proceeding of the IEEE Conference on E-Commerce Technology (CEC)*, Newport Beach (California)-USA, June 24-27, 1026-1027.

Jurca, R., Faltings, B. (2007). Obtaining Reliable Feedbacks for Sanctioning Reputation Mechanisms, *Journal of Artificial Intelligence Research*, 29(1), 391-419.

Jurca, R., Faltings, B., Binder, W. (2007). Reliable QoS Monitoring Based on Client Feedback, *Proceeding of the 16th International World Wide Web Conference (WWW)*, Banff (Alberta)-Canada, May 8-12, 1003-1011.

Kalepu, S., Krishnaswamy, S., Loke, S.W. (2003). Verity: A QoS Metric for Selecting Web Services and Providers. *Proceeding of the 4th international Conference on Web Information Systems Engineering Workshops*, Roma-Italy, December 10-12, 131-139.

Kastidou, G., Cohen, R., Larson K. (2009). A Graph-based Approach for Promoting Honesty in Community-based Multiagent Systems, *In 8th International Workshop for Coordination, Organization, Institutions, and Norms in Agent Systems (COIN@IJCAI)*, Pasadena (California)-USA, July 11.

Khosravifar, B., Bentahar, J., Thiran, P., Moazin, A., Guiot, A. (2009). An Approach to Incentive-based Reputation for Communities of Web Services, *Proceeding of the 7th International Conference on Web Services (ICWS)*, Los Angeles (California)-USA, July 6-10, 303-310.

Khosravifar, B., Gomrokchi, M., Bentahar, J., Thiran, P. (2009). Maintenance-based Trust for Multi-Agent Systems, *Proceeding of the 8th International joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Budapest-Hungary, May 10-15, 1017-1024.

Liu, Y., Ngu, A.H., Zeng, L.Z. (2004). QoS Computation and Policing in Dynamic Web Service Selection. *Proceeding of the 13th International World Wise Conference on Alternate Track Papers and Posters*, 66-73.

Malik, Z., Bouguettaya, A. (2007). Evaluating Rater Credibility for Reputation Assessment of Web Services, *Proceeding of the 8th International Conference on Web Information Systems Engineering (WISE)*, Nancy-France, December 3-6, 38-49.

Martino, L.D., Bertino, E. (2009). Security for Web Services: Standards and Research Issues, *International Journal of Web Services Research*, 6(4), 48-74.

Maximilien, E. (2005). Multiagent System for Dynamic Web Services Selection, *The 1st Workshop on Service-Oriented Computing and agent-based Engineering (SOCABE)*, Utrecht-The Netherlands, July 25-29.

Maximilien, E.M., Singh, M. (2002). Conceptual Model of Web Service Reputation, *SIGMOD Record*, 31(4), 36-41.

Organization for the advanced of structured information standards. Introduction to UDDI: Important Features and Functional Concepts (2004). Retrieved October 12, 2009, from http://www.oasis-open.org.

Weaver, A., Wu, Z. (2006). Using Web Service Enhancement to Establish Trust Relationships with Privacy Protection, *International Journal of Web Services Research*, 6(1), 49-68.

Zhang, L-J. (2008). Web Services Security, Composition, and Discovery, *International Journal of Web Services Research*, 5(1), 1-23.

ABOUT THE AUTHORS

Babak Khosravifar is a Ph.D. candidate in the Department of Electrical and Computer Engineering, Concordia University in Montreal, Canada. He is a research assistant in Multi-Agent and Web Services Laboratory at Concordia University under the direction of Dr. Jamal Bentahar. His research interests include multi-agent systems, trust frameworks, reputation mechanism, Web services, and game theory. He received his master of Computer Engineering from Eastern Mediterranean University, Cyprus.

Jamal Bentahar is an assistant professor of computer science and software engineering at the Concordia Institute for Information Systems Engineering at Concordia University, Montreal. His research interests include multi-agent systems, Web services, argumentation theory, logic and formal methods, and grid computing. He received his Ph.D. in computer science and software engineering from Laval University, Canada. He is a member of the IEEE, ACM and Professional Engineering Ontario.

Ahmad Moazin is a research assistant in Multi-Agent and Web Services Laboratory at Concordia University under the direction of Dr. Jamal Bentahar. His research interests include multi-agent systems, trust and reputation, Web services, and argumentation theory. He received his master of Information Systems Security from Concordia University.

Philippe Thiran is an associate professor in Web and Science Engineering at the Faculty of Computer Science of the University of Namur (Belgium). His research interests include Web services, databases, and distributed information systems. He received his Ph.D. in computer science from the University of Namur. He is a member of the PReCISE Research Center.