Analyzing Communities vs. Single Agent-based Web Services: Trust Perspectives

Babak Khosravifar[1], Jamal Bentahar[1], Ahmad Moazin[1]
Zakaria Maamar[2], and Philippe Thiran[3]
[1]Concordia University, Canada, [2]Zayed University, UAE, [3]University of Namur, Belgium
$b\_khosr@encs.concordia.ca, bentahar@ciise.concordia.ca, a\_moazi@encs.concordia.ca$
$zakaria.maamar@zu.ac.ae, pthiran@fundp.ac.be$

***Abstract*—Gathering functionally similar agent-based Web services into communities has been proposed and promoted on many occasions. In this paper, we compare the performance of these communities with self-managed, single agent-based Web services from trust perspective. To this end, we deploy a reputation model that ranks communities and Web services with respect to different reputation parameters. By relating the parameters, we extend our discussion to analyze the beneficial cases and incentives for a single Web service to join a community even if this joining could negatively impact other parameters. Besides theoretical discussions of this analysis, we discuss the system implementation along with simulations that depict diverse parameters and system performance.**

Keywords. Community of agent-based Web service, Trust, Reputation, Incentives.

I. Introduction

The use of Web services is mostly motivated by developing loosely-coupled, cross-enterprize business processes that process users' requests. In this paper, each Web service is associated with an agent that acts on its behalf and oversees its performance, commitments, and availability details [7]. Being overloaded, and hence poor responsiveness are unavoidable facts that single Web services along with their associated agents should manage. A solution is to group similarly-functional Web services into communities to improve their overall performance and/or availability [2]. Each Community of Web services (CWS) is led by a master Web service, which is responsible for accepting or inviting (hiring) new Web services to be members of the community and excluding or rejecting (firing) existing Web services. Deploying such a community is more cost-effective and thus, a proper management is required to handle community response. Recently, there have been few attempts to address the formation of communities of Web services that we will discuss later in Section VI. Since handling users' requests does not guarantee a high service quality, users always consider the reputation of the service that could be provided by a single Web service or a CWS in their service selection process. The offered service is compared with the promised quality of service and a corresponding feedback is submitted by the users. Therefore, Web services or CWSs receive continuous feedback on their performances.

Challenges. Excellent reputation is a double sword; it brings high demands from users and results in overloading service providers. The challenge is to identify a tradeoff between one's capacity (maximum number of service offering at a time) and market share (number of users that request service) so that an efficient service can handle the requests in a way that neither it gets overloaded quickly, nor it remains idle. The ultimate objective of all Web services is to tackle such a tradeoff in which the service gains a stable reputation and market share level. For a service (that could be a single Web service or a CWS) failing to respond with an acceptable service quality (i.e., being overloaded) would cause negative feedback and thus, reputation drop.

Contribution. In this paper, we investigate the incentives that would make a single agent-based Web service join a community. The join could be still preferable even under the assumption that the further quality of service could be decreased. We measure and analyze the benefits that a single Web service gains once it joins a community. In this analysis, we measure the general service reputation by considering two factors: satisfaction and inDemand. These factors are chosen to reflect the basic reputation assessment of a Web service. However, more parameters could be involved without changing the main results of this paper, but they are considered here because of space limit. We represent the reputation as a result of gained feedback (via users) on efficiency and accuracy of previous provided services. In our implemented system, we empirically elaborate on the inter-relation between the two considered factors with the general reputation and show their impacts on each other. Using these relations, we analyze the efficiency of CWS compared to a single Web service in different aspects. For comparison purposes, individual Web services are also considered as singleton communities. So the idea is to compare communities having one (or very few) Web services with those having many.

Organization. The remainder of this paper is as follows. In Section II, we define the reputation model using some metrics and propose how they could be combined. In Section III, we start the discussion about a CWS's performance versus a single Web service performance in overall service quality. We elaborate on inter-relation of involved metrics together with general reputation and extract their dependencies. In Section IV, we extend our discussion to theoretical analysis of the incentive that a single Web service has to join a community providing the same service. In Section V, we represent the simulation and outline the properties of our model in the experimental environment. Section VI discusses some relevant related work and finally, Section VII concludes the paper.

II. Reputation Model

In this section, we introduce the reputation model, which is inspired by the one proposed in [6]. We discuss about the opportunities that a CWS offers over single Web services with respect to reputation-model parameters. For simplification reasons, in the remainder of this paper, we only consider 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 II-A, we present the involved metrics that a user may consider in this assessment. These metrics are chosen with respect to their general impact on one's reputation level. In Section II-B, we explain the methodology that the user uses to combine these metrics in order to assess the reputation of a CWS.

A. Metrics

Responsiveness Metric: Let i be the community that is under consideration by user j. Responsiveness metric depicts the time that a CWS spends to answer a request. Let $Res_i^{j,t}$ be the time that community i takes to answer the request received at time t by user j (in the model proposed in [6], this is managed with master agent as a component in the structure of the community). This time includes the time for selecting a Web service from the community and the time taken by that Web service to provide the response back to user j. Equation 1 computes the response time of the community i, computed with user j during the period of time $[t_1, t_2]$ ($Res_i^{j,[t_1,t_2]}$).

$$Res_i^{j,[t_1,t_2]} = \frac{\sum_{t=t_1}^{t_2} Res_i^{j,t} \times e^{-\lambda(t_2-t)}}{\sum_{t=t_1}^{t_2} e^{-\lambda(t_2-t)}} \qquad (1)$$

Here the factor $e^{-\lambda(t_2-t)}$, where $\lambda \in [0,1]$ reflects the time recency of the received requests so that more emphasis on the recent requests is given. If no request is received at a given time t, we suppose $Res_i^{j,t} = 0$.

InDemand Metric: It depicts users' interests in a community i in comparison to other communities. This factor is computed in equation 2.

$$InD_i^{[t_1,t_2]} = \frac{Req_i^{[t_1,t_2]}}{\sum_{k=1}^{M} Req_k^{[t_1,t_2]}} \qquad (2)$$

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

Satisfaction Metric: Let $Sat_i^{j,t}$ be a feedback rating value (which is supposed to be between 0 and 1) representing the satisfaction of user j with the service regarding the request he sent at time t to community i. Equation 3 shows the overall satisfaction of user j about community i. In this equation, V_t represents the importance of the service received at time t. This importance would be represented by the service value or price.

$$Sat_i^{j,[t_1,t_2]} = \frac{\sum_{t=t_1}^{t_2} Sat_i^{j,t} \times e^{-\lambda(t_2-t)\times V_t}}{\sum_{t=t_1}^{t_2} e^{-\lambda(t_2-t)\times V_t}} \qquad (3)$$

B. Metric 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 [8]. 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 community i during interval $[t_1, t_2]$ from user 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 $Res_i^{j,[t_1,t_2]} = \nu$. In the equation 5, the *inDemand* metric is added to consider the reputation of community i from users' point of view, where coefficients $\eta + \kappa = 1$ and $\chi + \phi = 1$ are generic values.

$$Rep_i^{j,[t_1,t_2]} = \eta(1 - \frac{Res_i^{j,[t_1,t_2]}}{\nu}) + \kappa Sat_i^{j,[t_1,t_2]} \qquad (4)$$

$$Rep_i^{[t_1,t_2]} = \chi \, \frac{1}{m} \sum_{j=1}^{m} \left(Rep_i^{j,[t_1,t_2]} \right) + \phi \; InD_i^{[t_1,t_2]} \qquad (5)$$

III. Community of Web services versus single Web services

A general assumption in this paper is that Web services are free to remain self-independent and act individually or to join a community at any time[1]. The question that arises is why a Web service that could individually survive in terms of inDemand (receiving requests from the users) is encouraged to join a community that would possibly degrade its reputation. The answer could be survival in the environment is more critical than carrying on the current reputation level. Also we would like to discuss about the incentive that this single Web service has when it expresses interest in joining a community. To begin our discussions, we first declare three different relations that specify the proportional relevance of discussed metrics for a particular community (or single Web service) i: 1) inDemand as a function of reputation ($InD_i^{[t_1,t_2]} = \Gamma(Rep_i^{[t_0,t_1]})$); 2) satisfaction as a function of inDemand ($Sat_i^{[t_1,t_2]} = \Delta(InD_i^{[t_0,t_1]})$); and 3) reputation as a function of satisfaction ($Rep_i^{[t_1,t_2]} = \Pi(Sat_i^{[t_0,t_1]})$).

[1]In this paper, only the join issue is being discussed. However, the issue of when join event can take place is another important challenge that is out of scope of the paper.

Functions Γ and Π are monotonically increasing, whereas the function Δ is monotonically decreasing. Considering the time steps $t_0 < t_1 < t_2$, the *inDemand* during period $[t_1, t_2]$ is proportionally relevant to the obtained reputation at time $[t_0, t_1]$ (i.e., high (or low) reputation during $[t_0, t_1]$ will increase (or decrease) inDemand during $[t_1, t_2]$ and so on for other functions). The reason behind elaborating these links between the parameters is the fact that a single Web service can predict the expected parameter value in the future and thus, may change its decision for the current moment. These relations enhance the performance of communities or Web services that analyze their current status in the environment.

A. inDemand and Reputation

We adopt the assumption that the propagated reputation of a community among users highly influence its selection. Normally users do not tend to always select the same communities upon their need of a service and prefer to seek the communities that fulfill their requirements at best. In fact, the users select the best option for the first trial and if being rejected (due to overloaded reasons) go to the second best choice and so on, which would guide many users towards the most reputable communities. Therefore, we define the relation between the inDemand of a community i and show that over time the inDemand value approaches the portion of community reputation to the sum of all other communities reputation. This means the more reputable a community is, the more inDemand it gets. Therefore, we define the relation of the inDemand of i during the period $[t_1, t_2]$ as the ratio of the reputation value of the community i to the sum of all other communities reputation.

In order to prove such approach, we need to define a new parameter in equation 6, which represents the adjusted percentage over the requests for a particular community i in period $[t_1, t_2]$ (β_i can be positive/negative and is corresponding to community i). If the value is positive, this means that the community was more reputable than many others in the previous period, so the number of requests towards this community would be most likely more than the average of received requests and therefore, more than its capacity. Similarly, the number of requests towards the less reputable communities are most likely less than their capacities. This is fair in the sense that users by nature look for the first best choice and upon rejection they move to other possible choices. In equation 6, M represents the number of existing communities and $\overline{Rep}^{[t_0,t_1]}$ denotes the average reputation among communities.

$$\beta_i = \frac{Rep_i^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M} \tag{6}$$

$$\overline{Rep}^{[t_0,t_1]} = \frac{\sum_{j=1}^{M} Rep_j^{[t_0,t_1]}}{M}$$

The value β_i acknowledges that highly reputable communities would obtain more requests, which overflow their capacities. If the first best community already reached the capacity of its handling requests, there would be no room for new requests. Therefore, the best choice for the new users would not be the first best community, whereas it would be a community that might accept the request. In equation 7, we estimate the expected number of requests for the community i during the period $[t_1, t_2]$ related to the capacity of the community (Cap_i) and how much its reputation is greater or less than the average reputation in period $[t_0, t_1]$. In this equation, the expected number of requests are computed with respect to the fact that rejecting requests (that could lead to overloading state) are cascaded through users.

$$[Req_i^{[t_1,t_2]}] = (1 + \beta_i) \times Cap_i \tag{7}$$

Assume in general the capacity of the communities are the same, so we can consider a general value for the capacities. We can rewrite the inDemand value obtained in equation 2 by factorizing the Cap_i and replacing the expected requests in equation 8. The obtained inDemand is the expected number of requests that are sent to the community given that community i currently holds a particular reputation and thus, a particular β_i value from previous interactions with users.

$$InD_i^{[t_1,t_2]} = \frac{1 + \frac{Rep_i^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M}}{\sum_{k=1}^{M} 1 + \frac{Rep_k^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M}} \tag{8}$$

By multiplying the numerator and denominator of the equation by M and taking out the constant 1, we would obtain the following equation.

$$InD_i^{[t_1,t_2]} = \frac{M + Rep_i^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M^2 + \sum_{k=1}^{M} Rep_k^{[t_0,t_1]} - \sum_{k=1}^{M} \overline{Rep}^{[t_0,t_1]}}$$

In the denominator we can also simplify $\sum_{k=1}^{M} Rep_k^{[t_0,t_1]}$ by $\sum_{k=1}^{M} \overline{Rep}^{[t_0,t_1]}$ as they both declare the total reputation value for all communities. Therefore, the most simplified equation would be obtained in equation 9, where $\Gamma(Rep_i^{[t_0,t_1]})$ is monotonically increasing.

$$InD_i^{[t_1,t_2]} = \frac{1}{M}\left(1 + \frac{Rep_i^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M}\right)$$

$$InD_i^{[t_1,t_2]} = \Gamma(Rep_i^{[t_0,t_1]}) \tag{9}$$

We can multiply the two sides of the equation by the total reputation value for the communities ($\sum_{k=1}^{M} Rep_k^{[t_0,t_1]}$). Therefore, we can just rewrite $\overline{Rep}^{[t_0,t_1]}$ when the total value is multiplied by $\frac{1}{M}$. To this end, we would obtain:

$$InD_i^{[t_1,t_2]} \sum_{k=1}^{M} Rep_k^{[t_0,t_1]} = \overline{Rep}^{[t_0,t_1]}\left(1 + \frac{Rep_i^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M}\right)$$

In equation 6 we calculate the adjusted ratio (β_i) of the requests for a particular community i. We also consider the reputation of a community adjusted to the average reputation of communities, so that we substitute $\overline{Rep}^{[t_0,t_1]}(1 + \frac{Rep_i^{[t_0,t_1]} - \overline{Rep}^{[t_0,t_1]}}{M})$ by $\tau Rep_i^{[t_0,t_1]}$ (where $\tau > 0$), which would result in equation 10, and define the relation between the inDemand value of a community by its reputation among all other communities. Basically this equation is obtained over the assumptions that we made in computing the value β_i as adjustment percentage and this is possible when the users network and communities are established and the reputations are already propagated so that the parameters can be expected to be rationally set.

$$[InD_i^{[t_1,t_2]}] = \frac{\tau Rep_i^{[t_0,t_1]}}{\sum_{k=1}^{M} Rep_k^{[t_0,t_1]}} \quad (10)$$

B. Satisfaction and inDemand

In this section, we discuss the relation between satisfaction and inDemand values for a typical community i. In general, i is able to handle the requests submitted unless it gets busy with an overloaded inDemand and thus, cannot offer a good QoS to the number of users that exceeds the community capacity. This chaotic situation would cause inefficiency in offering service quality and consequently lead to drop in the satisfaction value of the community that is assigned by users. We refer to this drop by the parameter μ, so that $1 - \mu$ represents the community drop factor. Obviously different communities have different drop factors. For instance, the old communities are the ones that are more familiar with the network of users and thus, the chaotic situation would less affect them (small drop factor, i.e., high μ value). This is due to the fact that a typical community i with a capacity Cap_i serves the first group of users (with quantity of Cap_i) with its actual quality of service (QoS_i) and the second group of Cap_i users with $QoS_i \times \mu$, the third group of users with $QoS_i \times \mu^2$, and so on. Therefore, in the community in which the drop factor is smaller, the value μ is closer to 1. The settled community's QoS is the average of its composed Web services (equation 11). In this equation, the value QoS_{ij} indicates the quality of represented service by the Web service j that belongs to community i and m is the number of Web services belonging to community i.

$$QoS_i = \frac{\sum_{j=1}^{m} QoS_{ij}}{m} \quad (11)$$

We compute the age of a community relative to the age of all other communities. Equation 12 computes the age and drop factor of community i as the non-zero difference between the current time CT and the community initialization time IT_i. In equation 12, α represents the best handling parameter, and depends on the service time and loading time (the assigning time that the master of CWS would take to assign a proper Web service to a given user's request $\alpha = \frac{loadingTime}{serviceTime}$). The value φ is generic and depends on the loading time and environment stability, which reflects how crowded the service request line is and how easy CWSs handle user requests.

$$Age_i = |CT - IT_i| \qquad \mu = \alpha + \varphi ln(Age_i) \quad (12)$$

Considering the factor μ, we can now compute the provided service quality and consequently rate the obtained satisfaction from the served users. Equation 13 represents the provided quality of service $PQoS_i$ that community i provides for a fixed period of time serving n segments.

$$PQoS_i^{[t_0,t_1]} = \frac{\sum_{j=0}^{n-1} Seg_i^{[t_0,t_1]} \times QoS_i \times \mu^j}{\sum_{j=0}^{n-1} Seg_i^{[t_0,t_1]}} \quad (13)$$

In equation 13, parameter $Seg_i^{[t_0,t_1]}$ counts the number of users who received the same QoS from community i. For simplicity reasons we assume the case that the segment sizes are the same (same as community capacity Cap_i). To this end, segment size Seg_i could be factored out and therefore simplified. QoS_i is also fixed and could be factored out and therefore, we can rewrite equation 13 in equation 14, where n represent the number of segments.

$$PQoS_i^{[t_0,t_1]} = \frac{QoS_i \sum_{j=0}^{n-1} \mu^j}{n} \quad (14)$$

We assume that being aware of the community's public quality of service (QoS_i is a public value and represents the average QoS of its Web services), a user that is requesting a service would be satisfied 100% subject to obtaining the promised QoS (what is obtained is the same as what is proposed). Similarly, the user would be less satisfied if the QoS is decreased by μ. Here we calculate the fair feedback of such user to μ as the provided quality of service is $QoS_i \times \mu$. The obtained satisfaction for each group of users with the same served quality is considered the same. Thus in equation 15, the obtained average satisfaction value in period $[t_1, t_2]$ is computed as the sum of all the provided feedback in a similar way and written as a monotonically decreasing function of inDemand ($\Delta(InD_i^{[t_0,t_1]})$) in period $[t_0, t_1]$.

$$Sat_i^{[t_1,t_2]} = \frac{\sum_{j=0}^{n-1} \mu^j}{n} = \frac{1 - \mu^{n-1}}{n(1-\mu)} \quad (15)$$

$$\text{where} \quad n = \frac{InD_i^{[t_0,t_1]}}{Cap_i} \Rightarrow Sat_i^{[t_1,t_2]} = \Delta(InD_i^{[t_0,t_1]})$$

C. Reputation and Satisfaction

In this section, we discuss the relation between Rep and Sat parameters. In equation 4, we define how reputation is formed regarding the responsiveness Res and the obtained satisfaction of a community Sat. In equation 15, we compute the obtained satisfaction with respect to the fact that the busy community starts decreasing its quality and thus becomes less reputable for users. Now we discuss how the responsiveness (Res) value is computed. In equation 7, we estimate the expected requests (that are all accepted with the community) with respect to the value β_i as adjustment percentage over the average reputation of a community. In general, the responsiveness of a community is computed as a percentage of the accepted requests to the total filed requests from users in a specific time period ($TReq$). Equation 16 computes the aforementioned portion for a typical community i. In equation 16, the parameter θ is the percentage of the total requests that are attracted to community i. Basically we declare that θ is the portion of users that know community i (either they are in his network $\frac{1}{|i|}$ or indirectly connected δ).

$$Res_i = \frac{(1+\beta_i) \times Cap_i}{\theta \times TReq} \quad where \quad \theta = \frac{1}{|i|} + \delta \qquad (16)$$

Consider equation 4, the community would get more reputation if the Res value is less. This is generic in terms of time. However, here we address this issue in terms of the percentage of accepting the filed requests. To this end, if the community is accepting more than its capacity, the value Res would be high and therefore, the total Rep value would decrease. Therefore, in equation 17, we present the obtained reputation of community i during $[t_1, t_2]$ as a monotonically increasing function of satisfaction during $[t_0, t_1]$.

$$Rep_i^{[t_1,t_2]} = \Pi(Sat_i^{[t_0,t_1]}) \qquad (17)$$

IV. Theoretical analysis of joining benefit

In this section, we summarize the predefined relations and conclude with a decision making strategy, which enables the communities of single (or even few Web services) to estimate their benefits in terms of joining another larger community. In order to estimate the aforementioned benefit and thus, enable communities to make decisions based on the benefits they gain, we need to define a performance function that measures the extent to which a community acts successfully. Then maximizing one's performance, the community would decide what course of actions to choose (join a larger community or act self-independent). Consider i as a typical community with a current performance P_i. We call "join" a beneficial act once community i joins another community and as a result its performance gets increased. Likewise, we have the notion of degradation in performance and that happens when a community could not stop decreasing its performance (either by carrying on the same strategy or after choosing an action like "join"). In addition, we consider a performance loose when a community cannot optimally use its allocated resources (Web services). In this case, a number of Web services are not engaged upon user requests and thus, the master of the community should lay off some Web services to avoid decreasing performance. Of course the extent to which a community is committed to satisfy users' requests depends on how successful the community was in terms of quality of service and quick response to the users.

So far we mentioned that performance would be affected by the use of allocated Web services and a simultaneous obtained feedback. As mentioned before, a community with overloaded inDemand cannot obtain an acceptable feedback and adversely, an idle community cannot manage to optimally use its resources. Therefore, both cases are not preferred and in general, a community that can optimally balance its obtained inDemand and capacity would manage high performance. In equation 18, we formulate the aforementioned performance function considering $InD_i^{[t_0,t_1]}$, Cap_i, and $Rep_i^{[t_0,t_1]}$ parameters. In this equation, the parameter $\varpi Rep_i^{[t_0,t_1]}$ is the weighted reputation value of the community. Coefficient ϖ reflects the importance of reputation factor in the performance. In this equation, a community that holds a high reputed community and also manages to provide a balance between its InD and Cap, can range an acceptable performance.

$$P_i = \begin{cases} \frac{\varpi Rep_i^{[t_0,t_1]}}{|InD_i^{[t_0,t_1]} - Cap_i|}, & \text{if } InD_i^{[t_0,t_1]} \neq Cap_i; \\ \varpi Rep_i^{[t_0,t_1]} & \text{if } InD_i^{[t_0,t_1]} = Cap_i. \end{cases} \qquad (18)$$

P_i is a heuristic that we use to reflect communities' successes since they have been initialized. Since the master oversees the community progress, a good decision making mechanism is needed to enhance the efficiency of active communities. To this end, masters of communities would always compare their performance values with their previously rated values in order to figure out the extent to which they are progressing. The comparison of these values are also happening when a community that is not progressing seeks to join another community hoping to increase or keep its current value. We measure the benefit as a difference between the values of future and current state. To this end, the communities are encouraged to join other communities when they observe a continuous decrease in their current performance values. The decrease reflects the failure of the community in its action among other communities, and thus the community is encouraged to seek to join other communities so that the performance function changes to get a positive slope.

Consider the community i (with cardinality $|i|$ that could be one) that seeks to join another community j (with cardinality $|j|$). Here we analyze the reasons and cases that would make this join possible. Also, we elaborate the cases that would make the join inappropriate. Obviously a community i would consider the choice of join only when the performance gets increased once the join is done. Since the decision of join might end up with join denial, we address the performance after the join as the estimated performance value and denote it by $\widehat{P}_{i\to j}$ ($P_i < \widehat{P}_{i\to j}$ if join is to be done). To this end, community i considers to join j once the following inequality holds. Obviously, community j as a bigger community needs to consider acceptance or rejection of i's request for join, but because of lack of space, more details about these issues are omitted.

$$\frac{\varpi Rep_i^{[t_0,t_1]}}{|InD_i^{[t_0,t_1]} - Cap_i|} < \frac{\varpi \widehat{Rep}_{i\to j}^{[t_1,t_2]}}{|\widehat{InD}_{i\to j}^{[t_1,t_2]} - \widehat{Cap}_{i\to j}|}$$

where

$$\widehat{Rep}_{i\to j}^{[t_1,t_2]} = \frac{|j| \times Rep_j^{[t_0,t_1]} + Rep_i^{[t_0,t_1]}}{|j| + |i|}$$

$$\widehat{InD}_{i\to j}^{[t_1,t_2]} = InD_i^{[t_0,t_1]} + InD_j^{[t_0,t_1]} \qquad \widehat{Cap}_{i\to j} = Cap_i + Cap_j$$

$\widehat{Rep}_{i\rightarrow j}^{[t_1,t_2]}$ represents the merged current reputation value with the average reputation value of community j. Therefore, the obtained value would be the new average reputation of community j. $\widehat{InD}_{i\rightarrow j}^{[t_1,t_2]}$ and $\widehat{Cap}_{i\rightarrow j}$ respectively represent the new inDemand and capacity after join. The join is encouraged either when the community i is overloaded with many users or when the community i cannot attract enough users that satisfy its Web services. In the first case, the master of community i compares its current performance level P_i with the expected of that ($\widehat{P}_{i\rightarrow j}$) upon join to community j. In this case the overloaded users would be handled once the capacity of the community would be increased. This means that the community j can share some of its own resources if needed and thus, the total requests for community i would be handled. To this end, the value $|InD_i^{[t_0,t_1]} - Cap_i|$ is smaller than $|\widehat{InD}_{i\rightarrow j}^{[t_1,t_2]} - \widehat{Cap}_{i\rightarrow j}|$ in the sense that the total performance level is higher. This may even happen when the community i faces a decrease in its present reputation value ($Rep_i^{[t_0,t_1]} > \widehat{Rep}_{i\rightarrow j}^{[t_1,t_2]}$). We simplify the presented inequality in order to obtain the following inequality, which defines the threshold by which the join is preferred.

$$|InD_i^{[t_0,t_1]} - Cap_i| \times \Theta - \nu(1-\Theta) > |\widehat{InD}_{i\rightarrow j}^{[t_1,t_2]} - \widehat{Cap}_{i\rightarrow j}|$$

where $\Theta = \frac{\widehat{Rep}_{i\rightarrow j}^{[t_1,t_2]}}{Rep_i^{[t_0,t_1]}}$

In the simplified inequality, the parameter Θ represents the comparison of the community i's reputations before and after join to community j. Therefore, the community i would be encouraged to join community j even though its reputation is decreased ($\Theta < 1$). In this case, the join is beneficial as long as the total performance is higher. Here we can consider the generic case that the reputation remains unchanged ($\Theta = 1$). Therefore, the join is beneficial if we get the following.

$$|InD_i^{[t_0,t_1]} - Cap_i| > |\widehat{InD}_{i\rightarrow j}^{[t_1,t_2]} - \widehat{Cap}_{i\rightarrow j}|$$

In the second case of failure in which the community i cannot attract enough users with respect to its present capacity, the value $|\widehat{InD}_{i\rightarrow j}^{[t_1,t_2]} - \widehat{Cap}_{i\rightarrow j}|$ is still such high that causes a low performance level for community i. Of course this community would be happy to join another community j as long as it can increase its inDemand (smaller $|\widehat{InD}_{i\rightarrow j}^{[t_1,t_2]} - \widehat{Cap}_{i\rightarrow j}|$) and reputation values. In this case the condition that is shown is the previous inequalities is obtained easily.

V. Empirical Observations and Analysis

In this section, we provide an empirical analysis over the observed results regarding characteristics of a typical community C and a single Web service S. We motivate the join option by depicting the challenges that a single Web service S faces when it cannot handle further requests because of capacity limitation ($InD_S^{[t_0,t_1]} > Cap_S$). In the implemented prototype, CWSs are composed of distributed Web services and community components ($Java^{©TM}$ agents). The agent reasoning capabilities are implemented as Java modules. Agents are equipped with reasoning functionalities that enable them to decide about course of actions to take. Besides communities, the testbed environment is also populated with numerous users ($Java^{©TM}$ agents) that are programmed to look for services being offered by service providers. Users in general provide feedback regarding the quality of the offered service than rank community reputation. The simulation consists of a series of empirical experiments tailored to show different parameters of system components in diverse aspects. Table I summarizes the simulated environment which is populated with users and providers. Users are multiple and scattered over the environment, but providers are divided into communities (CWS) and Single Web Services (SWS). CWS cover 40.0% of providers while SWS are more (60.0%). In the simulated environment, we deployed relatively lower quality of service for CWSs to motivate their higher performance from their request handling rather than solely their service qualities. In general, CWS host at least 10 Web services, which covers at least 100 requests at a time, while a single Web service handles 10 at a time. The drop factor μ is also relatively small in a CWS due to its higher capabilities compared to SWS.

Table I
Environment summarization over the obtained measurements.

Type	Density	QoS	Capacity	μ
CWS	40.0%	[20.0%, 80.0%]	[100,300]	10.0%
SWS	60.0%	[20.0%, 95.0%]	[10,20]	15.0%

One of the main reasons that distracts service provider's overall performance (P_S or P_C) is its reputation update range. Since they are associated with a reputation level as a result of provided feedback by the users, if the number of interacting users is relatively low, the update over the reputation rank would be more visible than the case when the number of interacting users is relatively high. Figure 1 shows characteristics of CWS (C) in the left part and SWS S in the right part. Plots reflect average values that have been measured as a result of analysis over all communities of the same type. $Plots(a)$ and (c) respectively represent the reputation increase for the same number of RUNs while both C and S are gaining high reputation. Since number of interacting users with S is lower than that of C, the effect that positive feedback make on the reputation value of S is relatively high compared to the effect on C. Similarly reputation degradation ($plot(b)$ for C and $plot(d)$ for S) shows dramatic change in single Web service compared to CWS. Overall, such high range of change reflects lower feedback density, which also reflects lower market share

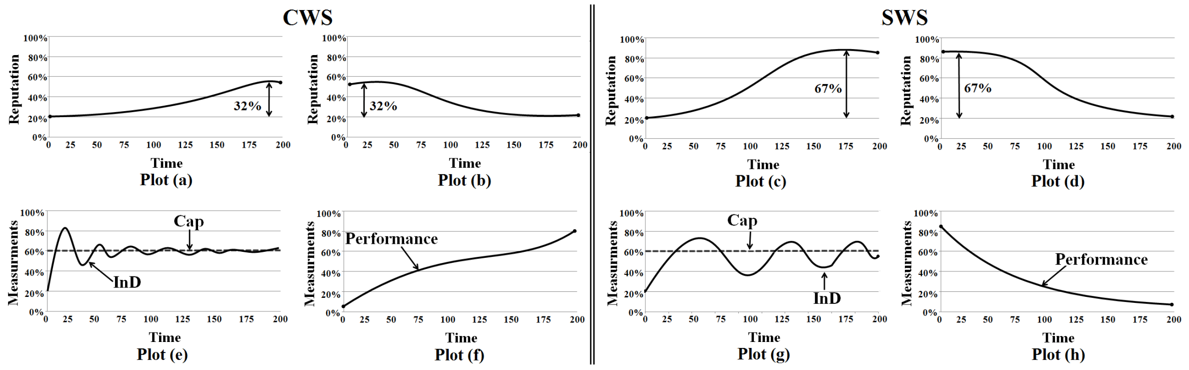

Figure 1. Characteristics of CWSs vs. SWSs.

($InD_S^{[t_0,t_1]} < InD_C^{[t_0,t_1]}$). This is normal since single Web service would share a small portion of the market. We elaborate on the effect of such a range more in $plots(e)$ and (g). In these plots, the horizontal line represents a community's capacity. What is interesting in these plots is the way that inDemand approaches capacity. In $plot(e)$, we observe closer oscillating curve compared to that in $plot(g)$. Community C handles its user increase until it gets overloaded ($InD_C^{[t_0,t_1]} > Cap_C$) and starts to offer lower quality services. Therefore, being overloaded, C gets some negative feedback that cause a drop in inDemand. Once handling the requests again ($InD_C^{[t_0,t_1]} < Cap_C$), C obtains good feedback that increase its inDemand again ($InD_C^{[t_1,t_2]}$). But not necessarily C would obtain the same inDemand as before ($InD_C^{[t_1,t_2]} < InD_C^{[t_0,t_1]}$), and that is simply as a result of users' evaluations that might not show interest requesting C again. In general, inDemand value would be relaxed by capacity and that is the point where C handles user requests at best thanks to collaboration between Web services that share users. Such relaxation would take much longer time with S and that is due to its lower number of interacting users. As shown in $plot(g)$, the inDemand curve oscillates in a higher range and takes more time to merge with S's capacity level. We also show this fact with performance parameter in $plots(f)$ and (h). As discussed in Section IV, performance is affected by community's efficiency in minimizing the difference between its capacity and inDemand values. As shown in $plot(f)$, C as a good community obtains an interesting performance ($\Delta P_C > 0$) over time while S (see $plot(h)$) gets decreased ($\Delta P_S < 0$). Performance parameter can be considered as obtained utility (or payoff) as a result of acting either alone or with other Web services in a community.

We continue our discussions in more details by comparing how the aforementioned parameters evolve over time. In Figure 2, the reputation level of a typical single Web service S is depicted in $plot(k)$ while the horizontal line represents S's capacity (Cap_S). In these plots, characteristic of a typical Web service is measured to observe its cooperative parameters impacts over time. In this case, once the reputation increases, the users requesting the service increase, and thus the dot points (assigned for each user) look more crowded around the reputation curve. Once reputation exceeds such Cap_S value, users are dispersed and inDemand undergoes a faster decrease. $Plot(l)$ illustrates more information as inDemand, reputation, and satisfactions parameters are represented. In this plot, inDemand gets the highest value since a high reputation is followed by a large number of requests while satisfactions look more steady. In contrast, after a low quality service, all curves head down among which, inDemand decreases more since users stop requesting until such a single Web service manages to handle user requests again.

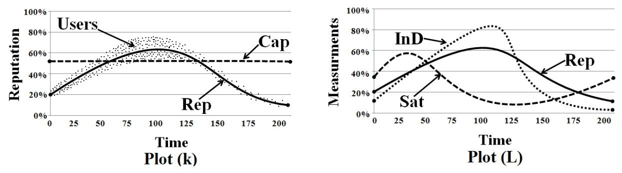

Figure 2. Evolution of cooperative parameters for a single Web service over time.

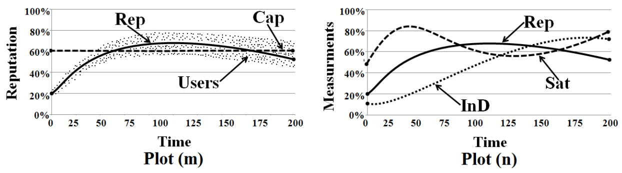

Figure 3. Evolution of cooperative parameters for a community of Web services over time.

Figure 3 illustrates the same structure analyzing the parameters regarding typical community C hosting some Web services. In this Figure, $plot(m)$ depicts a more normal reputation adjustment over its Cap_C value and the crowd over such reputation value show a more stable inDemand, which reflects C's stable market share. This is more elaborated in $plot(n)$ once the curves tend to approach each other at the end. This extends to more details about parameters of a stable community that managed to maintain a tradeoff between its capacity and inDemand.

VI. Related Work

In the literature, the reputation of Web services has been intensively stressed [9], [10]. In [1], the authors have developed a framework aiming to select Web services based on trust policy that users express. The framework allows the users to select a Web service matching their needs and expectations. In [8], the authors have designed a multi-agent framework based on an ontology for QoS. Users' ratings according to the different qualities are used to compute the reputation of the Web service. In [5], [7], some Web services reputation mechanisms have been proposed, that would lead to an effective service selection, and in [4], service-level agreements are discussed in order to set the penalties over the lack of QoS for the Web services. All these models address the reputation in environments where Web services function alone. In these models, Web service performance is not discussed in details and in general, handling is not considered as an issue for Web service besides its reputation.

Recently, some works have been done regarding formation (and reputation) of communities of Web services [2]. 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 [3]. In [2], Elnaffar et al. propose a reputation-based architecture for CWSs and classify the involved metrics that affect the reputation of a community. 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. In [6], the authors mainly address the overall assessed reputation that is used as a main reason for service selection. The authors do not consider handling as a parameter that impacts service selection in future. A sound logging mechanism is proposed that penalizes malicious acts and maintains accurate reputation rankings. In general, the recent aforementioned works motivate the existence of communities rather than single functional Web services, but fail to systematically provide potential benefits and technically compare CWS and individual Web services.

VII. Conclusion

The contribution of this paper is the analysis over the existence of communities of Web services and their overall comparison in different aspects with single Web services. The analysis covers the inter-relation between inDemand, satisfaction, and reputation parameters. Such analysis is concluded with performance measurement by which a single Web service is encouraged to join a community within which a better handling ability over the users' requests is guaranteed. The analysis performed in this paper is the first theoretical and empirical work that takes into account the system parameters and motivates higher performance even under lower reputation level. In this analysis, single Web services are allowed to predict their further reputation level (and thus, performance) that let them make the best decision.

Our plan for future work is to advance the discussion to analyze the concept of join in a more systematic way. To this end, a repeated game can be defined between a community and a single Web service with assigned payoffs as a result of their selected strategies. In the performance analysis, we need to elaborate more on the expected performance after join and consider possible cases that might discourage a single Web service to join a larger community. Similarly, we need to discuss more about the community that could refuse to accept the join of a single Web service.

Acknowledgments. Jamal Bentahar is supported by NSERC (Canada) and FQRSC (Quebec).

References

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

[2] S. Elnaffar, Z. Maamar, H. Yahyaoui, J. Bentahar, and Ph. Thiran. Reputation of communities of Web services -preliminary investigation. In Proc. of the 22'nd IEEE Int. Conf. on Advanced Inf. Networking and App., pp. 1603-1608, AINA 2008.

[3] M. Jacyno, S. Bullock, M. Luck, T.R. Payne. Emergent Service Provisioning and Demand Estimation through Self-Organizing Agent Communities. 8'th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 481-488, AAMAS 2009.

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

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

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

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

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

[9] 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.

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