Analyzing the Relationships between some Parameters of Web Services Reputation

Babak Khosravifar
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
b_khosr@encs.concordia.ca

Jamal Bentahar
Concordia Institute for Information System Engineering
Concordia University
Montreal, Canada
bentahar@ciise.concordia.ca

Ahmad Moazin
Concordia Institute for Information System Engineering
Concordia University
Montreal, Canada
a_moazi@encs.concordia.ca

***Abstract*—In this paper, we provide an analysis of the impacts of some reputation parameters that an agent-based Web service holds while being active in the environment. To this end, we deploy a reputation model that ranks the Web services with respect to their popularity in the network of users. We model and analyze the arrival of requests and study their impacts on the overall reputation. The Web services may be encouraged to handle the peak loads by gathering to a group. Besides theoretical discussions, we also provide significant results, which elaborate more on the details of the system parameters. We extend the details of these results to empirical results and link the observations of the implemented environment to the results that we theoretically obtain.**

***Keywords*-Agent-based Web service; Reputation; Incentives; Poisson Distribution;**

I. Introduction

The use of Web services is mostly motivated by developing loosely-coupled, cross-enterprize business processes that process users' requests. In environments with multiple Web services, each Web service can be associated with a software agent that acts on its behalf and oversees its performance, commitments, and availability details that are used for service selection [4]. In such environments, Web services may face unavoidable problems such as overload, poor responsiveness, or idle resources that a rational agent-based Web service always tries to avoid. Therefore, a rational agent acting as the intelligent structure of the Web service would need to analyze the cases where such situations would take place in order to predict their occurrences. One clue would be gathering Web services to build a collaborative virtual group called *community* that aims to increase Web services' capabilities [8].

The Web service selection is competitive with respect to the public reputation that Web services hold. This selection process is mostly used in the existing applications where handling users' requests does not necessarily guarantee a high service quality. In fact, users always consider the quality of service (QoS) when they select a Web service. Normally, in the existing applications, the offered QoS is compared with the promised one and a corresponding feedback reflecting users satisfaction is submitted by users.

Challenges. Excellent reputation is a double sword; it can bring high demands from users, which can result in overloading services, and this can cause a drop in overall reputation. Therefore, it is important to seize the stable status for a service to always hold an acceptable reputation since the environment is dynamic. An efficient service aims to 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 balance in which they carry on their stable demands while they increase their popularity in the network. For a single Web service, failing to respond with an acceptable service quality (i.e., being overloaded) would cause negative feedback and thus, reputation's drop. In the literature, there is a lack of systematic analysis and studies of the dynamic relationships between the demand and reputation of Web services. By helping agent-based Web services to make good decisions, such analysis would help in optimizing the performance of Web services in terms of popularity in the network and resource usage.

Contribution. In this paper, we introduce a reputation mechanism that let consumers select the Web services having the best credibility. For simplicity reasons and to achieve a high focus, we restrict the reputation model to two crucial parameters: satisfaction and popularity and analyze the impacts that these parameters have on one another in continuous service selection processes (considering other parameters is our plan for future work). In this mechanism, we use the *non-homogenous Poisson* [12] as the probability distribution that models the arrival of requests for a typical Web service (the motivations and reasons behind using this distribution will be discussed in Section III). We aim to theoretically analyze the impacts that parameters have on one another and deduce cases where the Web service has a clear incentive to change its acting strategy. For example, the Web service might consider to initiate a group gathering a set of collaborative Web services (or even join a pre-built group) to reach some capabilities that it cannot achieve if it acts alone (i.e. a high reputation may not be accessible for some Web services if they act alone). In this analysis, we tackle the general service reputation by considering two factors: satisfaction and popularity (called *inDemand* in this

paper). We represent the reputation as a result of popularity and gained feedback (via users) on efficiency and accuracy of previous provided services. In our implemented system, we empirically elaborate on the inter-relation of the two considered factors with the general reputation and show their evolving values over time in the system using different Web services with different acting strategies. Using these relations, we analyze the efficiency of a group of Web services working together compared to single Web services in different aspects.

Originality. In the literature, the previous proposals address the problems of developing, discovering, composing, deploying, and securing services. To the best of our knowledge, there is no work done regarding the challenges discussed above, which are about analyzing the relationships between reputation factors of Web services in dynamic environments where performance is a crucial factor. In this paper, we address for the first time the theoretical analysis of the relationship between users's requests and reputation. This allows us to analyze the performance of the Web services that choose between two strategies: either, to work alone, or to join a group where members can collaborate to increase their capabilities.

Organization. The remainder of this paper is as follows. In Section II, we define the reputation model using two important metrics. In Section III, we start the discussion about a Web service's performance considering the overall service quality. We base the discussions on the arrival of requests that follow a *non-homogeneous Poisson process*. We elaborate on inter-relation of involved metrics extract their dependencies. In Section IV, we discuss some results obtained from theoretical analysis of reputation parameters. 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

This section briefly introduces the reputation model that we use in our analysis together with its related parameters. Since our contribution in this paper is the theoretical and empirical analysis over the parameters impacts on one another, we do not detail the reputation assessment and only focus on two involved parameters and their combination. Note that the proposed metrics can be computed using users' feedback when accurate data is available. However, in many situations users are not always honest and the feedback can be unprecise. For the purpose of our analysis, we represent these metrics as they are computed (or estimated) by the Web services using only accurate public data: the number of received requests and the Web service's capacity (also called *throughput*, which is the number of requests a Web service can handle per time unit). This approach is useful, as it allows Web services to predict the future values of the reputation parameters based on previous observations, and then decisions can be made based on these predictions.

InDemand Metric: Let i be the Web service that is under consideration. InDemand parameter $inD_i(t)$ (ranges between 0 and 1) is a component that depicts users' interests in this Web service in comparison with other Web services at time unit t (also called *market share*). Let $Req_i(t)$ be the total number of requests sent for Web service i at time unit t ($Req_i(t) \geq 0$). Considering all M active Web services in the network, the inDemand is computed in equation 1.

$$inD_i(t) = \frac{Req_i(t)}{\sum_{k=1}^{M} Req_k(t)} \tag{1}$$

Satisfaction Metric: This is the parameter reflecting the users' rated value regarding their satisfaction of the obtained service from the Web service i at time t ($Sat_i(t)$). Usually, this parameter is measured via users' submitted feedback. As discussed before, we are interested here in its estimation from the Web services' point of view considering its capacity (i.e. throughput) and the number of requests at time unit t. This estimation can be then considered as a prediction of users satisfaction. $Sat_i(t)$ is also set between 0 and 1.

In our analysis, we restrict ourselves to honest Web services in the sense that they provide the best of their QoS once they can handle all the received requests. This restriction is justified by the fact that our objective is not to compare honest and malicious Web services, but to help Web services to make good decisions regarding their acting strategy about how to balance reputation and market share. This means that, if the demand for a Web service is less than its capacity value (Cap_i), then the Web service is capable of offering its actual QoS. To this end, the satisfaction rate that a Web service obtains is actually weighted with a factor that reflects the extent to which its capacity is below the received requests. In this case, the Web service would receive rate of 100% for the users who received the QoS as promised (the number of these users are equal to Cap_i). The Web service would also receive 0% for the users who are rejected and thus, do not receive the QoS as promised. Therefore, the average would be $\frac{Cap_i}{Req_i(t)}$. Equation 2 computes the general reputation assessment for a typical Web service i.

$$Sat_i(t) = \begin{cases} \frac{Cap_i}{Req_i(t)}, & \text{if } Cap_i < Req_i(t); \\ 1 & \text{if } Cap_i \geq Req_i(t). \end{cases} \tag{2}$$

Consider the first case in equation 2, where $Cap_i < Req_i(t)$. In this case, the Web service i is overloaded and thus, the number of received requests is more than its actual capacity. Continuing in this situation would obviously lead to its reputation drop. However, in the second case where $Cap_i > Req_i(t)$, the resources are considered idle, which implicitly expresses the low inDemand and thus, low reputation again. The case where $Cap_i = Req_i(t)$ is very

rare since these values are normally very dynamic. The challenge is then how to achieve a stable situation.

Metric Combination. As discussed earlier, the involved factors have impact on the total reputation that a Web service holds and uses as a mean to absorb users at the next time unit $t+1$. Therefore, the combination would set the $Rep_i(t+1)$ value. In this case, there is a weight that is assigned to each component, which reflects its importance. The reputation value also ranges between 0 and 1. Equation 3 combines the involved factors (satisfaction and inDemand) with the assigned weights (χ and $1-\chi$). The weighted coefficients are set with respect to the environment characteristics and they are application-dependent. However, in reputation mechanism, the metrics are weighed in general and used in discussions to have their corresponding influences.

$$Rep_i(t+1) = \chi \; Sat_i(t) + (1-\chi) inD_i(t) \tag{3}$$

III. Analyzing Web service Reputation Parameters

The main issue we aim to address in this paper is whether, in a specific situation, an active Web service is capable of surviving in highly requested environments (i.e. the requests are relatively higher than the Web service's capacity and thus, the Web service cannot hold and maintain high reputation). On the other hand, there is the activity issue of the Web service when the number of requests is less than its capacity and thus, the Web service does not hold a relatively high inDemand. The objective is to identify the situations within which the Web service can act safely in the sense that it has enough capacity to serve the consumers and the holding resources are effectively utilized.

In this section, we discuss the aforementioned issues and theoretically identify and analyze the possible situations. In this analysis, we will identify the relations that the involved reputation factors have with one another. For example, we will analyze the relation between the inDemand and reputation, which indicates the fact that if a Web service obtains a high reputation, the mean number of requests for that Web service would increase over the time. Furthermore, the increase of the inDemand would cause a negative impact on the satisfaction rate in the sense that handling a high number of requests would be harder. Although these results are expected, our work is the first attempt in formalizing this issue and confirming it both theoretically and practically. The main objective is to encourage the increase of Web services capabilities by gathering these Web services into a group having the same functionality. To this end, in the following, we discuss in more details the relations that link the reputation model parameters together.

A. Service Request

The metrics discussed in Section II are based on the number of requests sent by the users (service consumers). To have a general analysis of different possible situations and to better predict the future, we need to model the dynamics of the arrival of these requests and compute the probability of having given numbers of these requests. For a typical Web service i, the service requests can be modelled as random variables under the form of discrete events that follow a *Poisson* distribution. *Poisson* distribution is a discrete probability distribution that expresses the probability of a number of events occurring in a fixed period of time. The expected number of occurrences (i.e. the requests in our case) is referred to as the *Poisson* rate λ. As defined in *Poisson* distribution, the probability of having exactly n requests is defined as the function $z(n)$.

$$z(n) = \frac{e^{-\lambda} \times \lambda^n}{n!} = \frac{\lambda^n}{e^{\lambda} \times n!}$$

However, this distribution assumes that the events occur independently of the time since the last event. This assumption means in our setting that requests are independent of the Web service reputation, which changes with time. To better model the dynamics of requests that depend on reputation, we use the *non-homogeneous Poisson process* [12], which is a *Poisson* process with dynamic rate $\lambda_i(x)$ denoting the mean number of requests received by Web service i at time moment x, where x belongs to the time unit t. Here we should distinguish between the time unit t (i.e. the interval $[1,t]$) and time moment x in the sense that the moment is inside the time unit. The rate $\lambda_i(x)$ is a function of time, which means the number of requests can change every time moment x. However, the reputation is supposed to be stable with regard to the time moments x (i.e. stable for a certain period of time), but changes from one time unit to the next one. The arrival of requests for Web service i during the time unit t ($m_i(t)$) can be formulated as a *non-homogeneous Poisson process* as follows.

$$m_i(t) = \int_1^t \lambda_i(x) dx \tag{4}$$

In this case, the probability of having exactly n requests till time unit t is computed in Equation 5.

$$p(m_i(t) = n) = \frac{(m_i(t))^n}{e^{(m_i(t))} \times n!} \tag{5}$$

As described before, the rate $\lambda_i(x)$ is dependent of time x and represents the mean number of requests at time moment x. This rate should be estimated either using real data and statistical methods such as the *maximum likelihood*, or using analytical formulations. In this paper, we use an analytical estimation based on the assumption that in a network of Web services with active users, the mean number of requests would mainly depend on the status that the Web service holds among other services in the network. Regarding this issue, we relate this mean to the portion of reputation that the Web service holds at time unit t considering the other Web

services in the same network since the number of requests for one Web service reflects, in some extent, the market portion of this service. Let $TRep$ and Q_x respectively be the sum of total reputation of all the Web services during the whole time units and the sum of all the received requests by all the Web services at time moment x. Therefore, the rate can be computed as shown in Equation 6. As discussed before, the reputation of a given Web service is stable during the time unit. However, the total number of requests is moment-dependant.

$$\lambda_i(x) = \frac{Rep_i(t)}{TRep} Q_x \tag{6}$$

Using Equations 4 and 6, the arrival of requests during the time unit t ($m_i(t)$) is recomputed in Equation 7. Because the reputation is not a function of the moment x, we take the fraction out of the integral as constant.

$$m_i(t) = \int_0^t \frac{Rep_i(t)}{TRep} Q_x dx = \frac{Rep_i(t)}{TRep} \int_0^t Q_x dx \tag{7}$$

Therefore, the probability of having exactly n requests (formulated first in Equation 5) is recomputed in Equation 8.

$$p(m_i(t_1) = n) = \frac{(\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx)^n}{e^{(\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx)}(n!)} \tag{8}$$

The expected number of the received requests for a Web service i at time unit t (i.e the mathematical expectation $[Req_i(t)]$) can be now computed as shown in Equation 9.

$$[Req_i(t)] = \sum_{Rq=1}^{Q_x} p(m_i(t) = Rq) \times Rq$$

$$\Rightarrow [Req_i(t)] = \sum_{Rq=1}^{Q_x} \frac{(\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx)^{Rq}}{e^{(\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx)}(Rq!)} \times Rq \tag{9}$$

B. Expected inDemand, Satisfaction and Reputation Function

There is a general assumption that the reputation of a Web service has impact on the market share. In fact, service consumers normally seek for the best service (or at least better alternative) rather than sticking to the same service even if the QoS is acceptable (i.e. if the user is satisfied with a service s_1, he might prefer to request another similar service s_2 if it provides better QoS). In Equation 1, we showed that the inDemand value of a Web service i is calculated with respect to the portion of the requests to this Web service to the sum of all requests received by all the Web services in the same network ($Q_t = \sum_{k=1}^{M} Req_k(t)$). We also computed the expected requests received by the Web service i during the time unit t. Therefore, we can compute the expected inDemand rate for a Web service i as shown in Equation 10.

$$[inD_i(t)] = \frac{[Req_i(t)]}{Q_t} \tag{10}$$

Similarly, in Equation 2, we defined the obtained satisfaction rate $Sat_i(t)$ by the Web service i. Therefore, the expected rate would be recomputed as shown in Equation 11.

$$[Sat_i(t)] = \begin{cases} \frac{Cap_i}{[Req_i(t)]}, & \text{if } Cap_i < [Req_i(t)]; \\ 1 & \text{if } Cap_i \geq [Req_i(t)]. \end{cases} \tag{11}$$

Therefore, we can recompute the reputation of a Web service i as a function of the expected value of the number of requests. Obviously, the value for the function would be the expected value announced for the next time unit $t+1$ ($Rep_i(t+1)$), which reflects the prediction of future reputation. Let *Case*1 stand for $Cap_i < [Req_i(t)]$ and *Case*2 for the opposite.

$$Rep_i(t{+}1) = \begin{cases} \chi(\frac{Cap_i}{[Req_i(t)]}) + (1-\chi)(\frac{[Req_i(t)]}{Q_t}) & Case1; \\ \chi(1) + (1-\chi)(\frac{[Req_i(t)]}{Q_t}) & Case2. \end{cases} \tag{12}$$

Concerning the reputation value, we need to analyze the cases where the reputation function is increasing and the cases where the function is decreasing. To this end, we compute the derivative of the obtained function with respect to time t in Equations 13 (*Case*1) and 14 (*Case*2).

$$\frac{dRep_i(t+1)}{dt} = \frac{-\chi Cap_i}{[Req_i(t)]^2} \frac{d[Req_i(t)]}{dt} + \frac{(1-\chi)}{Q_t} \frac{d[Req_i(t)]}{dt}$$

$$\Rightarrow \frac{dRep_i(t+1)}{dt} = (\frac{(1-\chi)}{Q_t} - \frac{\chi Cap_i}{[Req_i(t)]^2}) \frac{d[Req_i(t)]}{dt} \tag{13}$$

$$or \quad \Rightarrow \frac{dRep_i(t+1)}{dt} = (\frac{(1-\chi)}{Q_t}) \frac{d[Req_i(t)]}{dt} \tag{14}$$

The expected number of requests is expanded in Equation 15. In deriving the fraction in the right hand side, the quotient rule is used to differentiate the fraction part and thus, the parameter $e^{\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx}$ is factored and simplified. There are two parts on the numerator that we separated to clarify the fraction. The parameter $[R']$ computed in Equation 16 is the chain derivative of the $\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx$. The parameter $[Y]$ is denoted as the factored out values represented in Equation 17.

$$\frac{d[Req_i(t)]}{dt} = \sum_{Rq=1}^{Q_x} \frac{Rq}{Rq!} \frac{d[\frac{(\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx)^{Rq}}{e^{\frac{Rep_i(t)}{TRep} \int_t^t Q_x dx}}]}{dt}$$

$$\frac{d[Req_i(t)]}{dt} = \sum_{Rq=1}^{Q_x} \frac{Rq}{Rq!} \frac{[R'][(\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx)^{Rq-1}][Y]}{e^{\frac{Rep_i(t)}{TRep} \int_0^t Q_x dx}} \tag{15}$$

$$[R'] = \frac{\frac{dRep_i(t)}{dt}\int_0^t Q_x dx + Q_t Rep_i(t)}{TRep} \quad (16)$$

$$[Y] = Rq - \frac{Rep_i(t)}{TRep}\int_0^t Q_x dx \quad (17)$$

Before we continue in the next section on the detailed discussions, we can observe that the obtained sign for $\frac{d[Req_i(t)]}{dt}$ depends on two factors: $[R']$ and $[Y]$. The rest of the values in Equation 15 are positive and do not affect the sign. Therefore, we need to investigate the cases where these parameters have different signs.

IV. Discussions

In this section, we discuss the observations we obtain through the corresponding relations between reputation parameters. Regarding this issue, we need to differentiate between two cases: (1) the Web service i is considered as an overloaded service ($Cap_i < [Req_i(t)]$); and (2) the Web service i is considered as an idle service where its resources are not appropriately used ($Cap_i > [Req_i(t)]$).

Case (1). This is the case where the Web service receives more requests than its actual capacity and thus, it is hard to handle the overloaded requests. Therefore, the satisfaction rate that the Web service receives is set as $[Sat_i(t)] = \frac{Cap_i}{[Req_i(t)]}$ (see Equation 11). According to Equation 13, we estimate the rate of change of the reputation with respect to time (through the derivative). There are two parts in this equation. The first part represents a value that is based on different system parameters. We observe that in large networks of different Web services, this part holds a negative value. To show that, we need to expand the following inequality:

$$\frac{(1-\chi)}{Q_t} - \frac{\chi Cap_i}{[Req_i(t)]^2} < 0 \Rightarrow \frac{(1-\chi)}{Q_t} < \frac{\chi Cap_i}{[Req_i(t)]^2}$$

$$\Rightarrow Q_t > \nu \quad where \quad \nu = \frac{(1-\chi)}{\chi}\frac{[Req_i(t)]^2}{Cap_i}$$

This inequality indicates that Q_t has a lower boundary ν given that $Cap_i < [Req_i(t)]$. We can then conclude that when the number of requests is large enough the rate of change (i.e. the slope sign) of the reputation function would follow the opposite direction of the request (the two derivatives have different signs). This means the increase of number of requests would cause the decrease of reputation when the number of requests is greater than the Web Service's capacity.

Proposition 1. *In Case* 1*, where $Q_t > \nu$, a positive rate of change of Web service's reputation function in the current time unit t would cause a negative rate of change in the next time unit $t+1$.*

Proof: The proof is based on Equation 13. In this case, the increase of expected request $[Req_i(t)]$ would cause a decrease in the reputation function. Therefore, we should prove that the rate of change of the expected request computed in Equation 15 is positive. In this equation, there are different parts. All the parts are always positive expect $[Y]$ and $[R']$.

The factor $[Y]$ would be negative when $Rq < m_i(t)$ (see Equation 17). Since Rq ranges between 1 and Q_t (in the sum), it is clear that $[Y]$ is a positive value. In fact, in the sum, the negative values would not have any impact given that Rq values are small in this case. Therefore, we only need to consider the factor $[R']$. Since the $Rep_i(t)$ rate of change is positive in this case, this factor is also positive (See Equation 16). Therefore, we obtain a positive rate of change for $[Req_i(t)]$, which causes a negative rate of change for the next time unit. ■

Proposition 2. *In Case* 1*, if $Q_t > \nu$, a bounded negative rate of change of Web service's reputation function in the current time unit t would still cause a negative rate of change in the next time unit $t+1$.*

Proof: This proof is the extension of the previous proposition regarding to the $[R']$ factor. In this case, a negative rate of change for $Rep_i(t)$ would not necessarily cause a negative value for $[R']$. Since the denominator is always positive, we only need to consider the case in which the numerator is still positive even with a negative value of $\frac{dRep_i(t)}{dt}$. To this end, we obtain a new boundary for this case:

$$|\frac{dRep_i(t)}{dt}| < \mu \quad where \quad \mu = \frac{Q_t Rep_i(t)}{\int_0^t Q_x dx}$$

Empirically, the obtained boundary is close to 1. This denotes the fact that a very small rate of change (even negative) would still cause a reputation drop in the next time unit.

■

Obtaining the two thresholds, we can discuss about this category of Web services that receive exceeded requests. The rate of change of the reputation function in current time unit would affect the next time unit in the sense that its positive or negative value (if $|\frac{dRep_i(t)}{dt}| > \mu$) would cause a negative rate of change. The negative rate of change with a larger absolute value would cause a negative rate of change of the expected request and thus, a positive rate of change of the reputation for the next time interval. However, that would be temporary in the sense that the next time unit would still face reputation drop. Therefore, we observe an unstable reputation value for the Web services having high expected requests and belonging to networks that are generally active with high request rate $Q_t > \nu$.

Case (2). This is the case where the Web service receives fewer requests than its actual capacity ($[Req_i(t)] < Cap_i$). In this case, the Web service sets its maximum expected satisfaction rate ($[Sat_i(t)] = 1$). However, the Web service is idle in this case. In general, to handle an idle situation, a Web service can try to increase its participation in serving

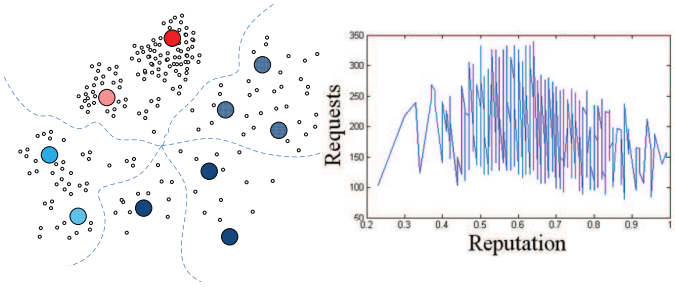

Figure 1. Network representation (Left): Network of Web services and users. Characteristics of Web service i (right): The plot of reputation versus the received request of Web service i.

consumers. In this case, the rate of change of the reputation would follow the direction of the rate of change for the expected requests (see Equation 14).

Proposition 3. *In Case* 2*, a positive rate of change of the Web service reputation function in the current time unit* t *will cause a positive rate of change in the next time unit* $t+1$ *(regardless of* Q_t *value).*

Proof: As it is clear from Equation 14, the rate of change of the expected request in the current time unit t *has the same direction as the rate of change of the reputation function in the next time unit* $t+1$*. In this case, if* $|\frac{dRep_i(t)}{dt}| > \mu$*, the* $[R']$ *factor will be still positive, which causes a positive rate of change for the expected request function.* ■

V. Empirical Observations and Analysis

In this section, we elaborate the observations that we have in the empirical results. We aim to point out the theoretical results that we obtained in theoretical discussions. Due to space limit, we skip the extension of the results to analyze the Web service's behaviors in diverse systems, and limit the empirical analysis to the system observations. These observations to some extend reflect the theoretical analysis discussed in Section IV. In the implemented environment, we have many users that are randomly distributed in the network and by default seek for the best service provider. These users are implemented as Java agents and their implemented prototype let them to analyze the received QoS and submit the satisfaction feedback accordingly. Since there is a public logging file for these, we restrict our discussions to honest feedback submission and thus, consider the reputation assessment accurate. In the implemented prototype, we are interested in analyzing the affect that system parameters impose on one another and investigate their evaluation over time.

Similar to users, the implemented Web services are activated with intelligent agents that are capable of computing the Web services' public reputation parameters. These Java agents are also capable of reasoning in order to select their strategy of acting in the environment with diverse parameters. The reasoning technique let agents to analyze the overall performance of the Web services they belong to. This could help changing a strategy if needed. The strategies and their details are out of scope if this paper. However, to illustrate the obtained results and their impact over time, we need to implement them (although for the space limits we skipped their detailed explanations).

Table I
Environment summarization over the obtained measurements.

Users	WS	In/Out	QoS	Cap	Req
N(2000,20)	N(200,10)	N(100,3)	N(0.5,0.5)	N(10,5)	Pois(10)

Table I summarizes the simulated environment which is populated with users and Web services scattered in the system. User distribution follow a normal distribution $N(2000, 20)$. We have Web services that are activated following normal distribution $N(200, 10)$. There is an input/output rate for Web services reflecting their active/deactive rate for the Web services. This would keep the environment more dynamic. The assigned QoS for the Web services also follow a normal distribution that guarantees any kind of QoS. With a normally distributed capacity for Web services we have arrival of requests following Poisson distribution.

In Figure 1, we have some generalities in the network of active Web services. In the left part, we have a network of the Web services that are surrounded with the users that request services from them (or connected to them for services). This is a snapshot from a part of the network that shows the distribution of users and their interactions with different Web services of diverse capabilities. In this network, the high quality Web services can handle larger number of users compared to the ones of lower capacities. In this network, we also have a part that Web services act in a joint manner such that they share their users to increase their capabilities. As mentioned before, the details of this framework is skipped. In the right part, we have a plot in which, the reputation is shown in X-axes versus the received request value shown in Y-Axes. In this graph, the received request of all the Web services are counted. Web services hold diverse reputation values. Each one also receive a number of requests accordingly. For example, if there are 10 Web services that hold the system reputation of the same value, the Y-value would reflect the received request value for all of them. We observe the increase of reputation and request such that the increase of reputation would not necessarily increase the request and that depends of some other parameters in the system. As it is deductable from the graph, the reputation between values 0.4 and 0.75 would receive relatively high requests. The reason behind this is discussed in propositions in Section IV. The high

reputation value is not stable and the Web services with temporary high reputation values would loose a portion of their users.

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 Web service i is depicted in $plot(k)$ while the horizontal line represents i's capacity (Cap_i). In this plot, characteristic of Web service i is measured to observe its cooperative parameters impacts over time units. There are four different time intervals regarding the reputation change of rate. In time interval I_1, InD_i is less than Cap_i, therefore, the Web service belongs to Case 2. In this case, the positive rate of change for reputation brings positive rate of change for its requests (in plots, we have shown the InD_i rate, which represents the percentage of total requests triggered for the Web service). The interval I_2, is the most preferable time interval for the Web service as a service provider. That is the time where, the reputation rate of change is still positive, and so as its demand. Therefore, the Web service absorbs lots of users as they are shown in dot points around reputation graph, surrounding the Web service in the network. In this interval, the Web service belongs to case 1. Therefore, the positive rate of change will not last long. The maximum difference (in terms of percentage) that the Web service obtains betweeb its capacity and inDemand value is denoted as d at the end of interval I_2. This is out of scope of the discussions, but it is worth saying that the malicious Web services who aim to maximize their selfish goals, this value would be maximized. As it is obvious from the graph, in Interval I_3, the reputation (of the next time unit) is head down with negative rate of change. This is the affect discussed in proposition 1. This is the interval that an intelligent Web service always avoids in the sense that the users are dispersed (as shown by dots), and the reputation is decreasing. This would impose a negative affect in InD_i value in the sense that it undergoes the Web service Cap_i value. This situation needs some time that the Web service could absorb users again, and it is shown in the graph, the reputation rate of change approaches a small number (negative though) after certain number of system RUNs. The corresponding graphs in $plot(l)$ shows the evolution of the satisfaction together with the demand rate for the Web service. In this case, the total demand for the system exceeds the threshold defined in case 1 ($Q_t > \nu$). Therefore, the affect it makes on the Web service reflects a large scale network that the Web service as a service provider would loose its users if it cannot handle them and it takes quite number of system RUNs that the Web service could re-absorb the users for serving its service.

In Figure 3, we have the similar scenario but for the network where the total requests does not exceed the threshold ($Q_t \leq \nu$). In this case, the affect the reputation parameters make on one another are different in the sense that the reputation rate of change is relatively slower than the previous case (comparing $plot(m)$ with $plot(k)$). The absorbtion of users is also more stable and looks the Web service has more time to handle the users. In this case, the Web service in interval I_1 is in case 2 and thus, the positive rate of change for the reputation would bring more users. The request also has a positive rate of change. But, in interval I_2, the Web service is in case 1. Therefore, the increase of reputation is not stable and as the satisfaction starts to decrease, the requests are also heading down (users start to disperse), and therefore, the reputation gets a negative rate of change (small though).

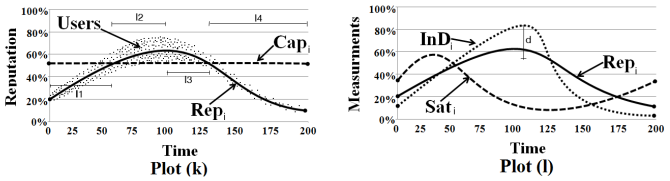

Figure 2. Evolution of cooperative parameters for Web service i over time when $Q_t > \nu$.

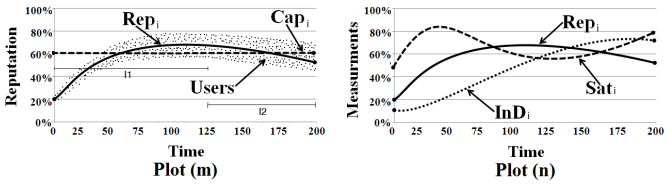

Figure 3. Evolution of cooperative parameters for Web service i over time when $Q_t \leq \nu$.

The affects in this graph reflects the facts that in small scale networks, the Web services have more time to recover their incapabilities, but in large scale networks, an intelligent Web service should be aware of the fact that increase of reputation would be not stable and may cause loss of many users. Therefore, in such networks, the Web services might consider join to other Web services to increase their capabilities.

VI. Related Work

In the literature, the reputation of Web services has been intensively stressed [11]. In [2], 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 [9], some Web services reputation mechanisms have been proposed, that would lead to an effective service selection. Regarding to this service selection, there are also some algorithms proposed to manage a high performance service delivery platforms [3]. 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 proposals have been done regarding formation (and reputation) of communities of Web services [5], [6]. In [6], Elnaffar et al. propose a reputation-based architecture for communities and classify the involved metrics that effect the reputation of a community. However, they do not investigate the cases where the community is initiated. In [7], [8], 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.

Regarding Web service performance, there have been some frameworks that are mainly characterized with their combination into a group or a cluster. In [1], authors propose an architecture for filtering and clustering Web services. This architecture is aimed to save time execution and increase the performance in using data. In [10], authors propose a statistical clustering Web service structure that enhances the capability to retrieve services that match users' requirements in large-scale environments. In general, these approaches address the service selection and assignment using clustering. However, they do not consider the performance of handling users and the relation between this performance and the Web services reputation.

To the best of our knowledge, there is no theoretical analysis on the performance of Web services in diverse network situations like the one proposed in this paper. The ultimate objective is to equip Web services with reasoning capabilities allowing them to choose among different acting strategies. The reasoning technique would help the Web services to increase their overall performance in dynamic networks.

VII. Conclusion

The contribution of this paper is on the analysis that we provide for the affect of reputation parameters for Web service as service providers in dynamic environments. We are interested in inter-relation of the reputation parameters and how the intelligent agent can be aware of the impacts that are imposed from the environment that encourage him to follow a course of action. We also elaborate more on the results that we obtain theoretically and observe empirically. The analysis performed in this paper is the first theoretical and empirical work that takes into account the system parameters and provokes intelligent movements for Web services.

Our plan for future work is to advance the discussion to analyze the concept of join in a more systematic way. We need to extend the framework that considers the join from two perspectives: Web service to the group, and acceptance of the group for the join. Regarding this, we could investigate the benefits for each party with defining a game between the joining Web service and the group. In the performance analysis, we need to elaborate more on the reputation and request values of the Web services active in different time slots. We also need to compare the results with other frameworks or with the Web services of the same framework but with different capabilities. Similarly, we need to discuss more about the group 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] W. Abramowicz, K. Haniewicz, M. Kaczmarek, and D. Zyskowski. Architecture for Web services filtering and clustering. In proceeding of the International Conference on Internet and Web Applications and Services (ICIW 2007).

[2] 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.

[3] M.K. Agarwal. Performance Management for Large Scale Service Delivery Platforms. In Proc. of IEEE International Conference on Services Computing, pp. 120-127, SCC 2009.

[4] E. Al-Masri, and Q.H. Mahmoud. Identifying Client Goals for Web Service Discovery. In Proc. of IEEE International Conference on Services Computing, pp. 202-209, SCC 2009.

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

[6] 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.

[7] B. Khosravifar, J. Bentahar, A.Moazin, and P. Thiran. Analyzing Communities of Web Services Using Incentives. In International Journal of Web Services Research (IJWSR), 7(3), 2010.

[8] 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.

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

[10] Ch. Platzer, F. Rosenberg, and Sh. Dustdar. Web service clustering using multidimensional angles as proximity measures. ACM Transactions on Internet Technology (TOIT), 9(3) Article 11, 2009.

[11] 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.

[12] F. Ruggeri and S. Sivaganesan. On Modeling Change Points in Non-Homogeneous Poisson Processes. Statistical Inference for Stochastic Processes 8(3):311-329, 2005.