Reputation of Communities of Web services - Preliminary Investigation

Said Elnaffar[1], Zakaria Maamar[2], Hamdi Yahyaoui[3], Jamal Bentahar[4], and Philippe Thiran[5]
[1]United Arab Emirates University, Al-Ain, U.A.E [2]Zayed University, Dubai, U.A.E
[3]Sharjah University, Sharjah, U.A.E [4]Concordia University, Montreal, Canada [5]University of Namur, Belgium

Abstract

Web services communities can be seen as virtual clusters that agglomerate Web services with the same functionality (e.g., FlightBooking). However, selecting a community to deal with is a challenging task to users and providers. Reputation, besides other selection criteria, has been widely used for evaluating and ranking candidates. Interestingly, the definition of community reputation from the perspective of users differs from the perspective of Web service providers. In this paper, we introduce a reputation-based Web services community architecture and define some of the performance metrics that are needed to assess the reputation of a Web service community as perceived by users and providers.

Keywords. *Community, Reputation, Web Service.*

1. Introduction

The 3W technology has tremendously changed the way users perform their daily activities from checking stock markets to booking airline seats. Round-the-clock users screen Web sites looking for the latest information and the best deals. However, what makes users return and bind to almost the same Web sites? *Reputation* is one of the key answers to this question. Generally speaking, reputation is a nonstop process that is built upon past experiences in terms of satisfaction, reliability, efficiency, etc.

As one of the promising technologies for developing loosely-coupled, cross-enterprise business processes, a plethora of Web services exists on the Web that are ready to process user requests transparently. Different businesses offer almost the same set of Web services (e.g., Weather forecast with The National Digital Forecast Database XML Web Service (www.weather.gov/xml) and The National Weather Service Forecast Office (www.srh.noaa.gov/mfl)) hoping that users will choose and continue to choose their Web services because of the reputation criterion. Bui and Gacher [3] note that although Web services are heterogeneous, their offered functionalities are sufficiently well defined and homogeneous enough to allow for market competition to take place. Web services with similar functionalities constitute what is usually known as *communities* [2, 6, 11]. In agreement with other definitions [1, 8, 9], we define community as a means to gather similar Web services regardless of who developed them, where they are located, and how they function [6].

There is a large body of research on Web services reputation [5, 7, 10, 12, 13]. However, little work exists when it comes to classifying (or ranking) communities of Web services through an appropriate reputation model. Indeed, as several communities come online, competition becomes significant. We argue that a reputation model for communities needs to be looked into from two perspectives: provider and user. In the former perspective, a provider should primarily know in which community it will let its Web services sign up. In the latter perspective, a user should know which community to bind to prior to identifying the Web service (that resides in this community) to invoke later. The purpose of this paper is to discuss our preliminary investigation of developing a reputation model for communities of Web services with focus on how to structure and update the reputation model, how to make the reputation model accessible, and how to maintain the reputation model up-to-date.

Sections 2 and 3 suggest some definitions and summarizes some related works, respectively. Section 4 describes a community of Web services. The reputation model along with its list of metrics are presented in Section 5. Finally, Section 6 concludes the paper.

2. Some Definitions

Reputation. It is *"the opinion (more technically, a social evaluation) of the public toward a person, a group of people, or an organization. It is an important factor in many fields, such as business, online communities or social status"* (Wikipedia Online Dictionary). Nowadays, opinions and user ratings are no longer sufficient for assessing the reputation of computer systems as Elnaffar stresses in [4]. Opinions/Ratings are subjective and can be easily manipulated and tampered with. A reputation system that solely relies on the temporal perspective of humans (i.e., users) can expose the system, intentionally or unintentionally, to dishonest ratings caused by the following types of users: emotional reactors, bad mouthers, and ballot stuffers. For example bad mouthers are users who unfaithfully exaggerate by giving negative ratings to service providers.

Community. It is *"a group of people living together and/or united by shared interests, religion, nationality, etc."* (Longman Dictionary). When it comes to Web services, Benatallah et al. [1] define community as a collection of Web services with a common functionality although these Web services have distinct non-functional properties such as different QoS attributes and computing capacities. Medjahed and Bouguettaya [9] consider community as a means to provide an ontological organization of Web services that share the same domain of interest. Finally, Maamar et al. [6] define community through the functionality of a representative abstract Web service, i.e., without explicitly referring to any concrete, factual Web service. In this paper, we use the latter definition of communities.

3. Related Work

Our literature review has revealed that a good number of references on Web services and reputation exists [5, 7, 10, 12, 13]. Nonetheless, studies that focus on the reputation of communities are scarce.

In [5], Jurca and Faltings mention that most Web services need to be contracted through service level agreements that specify a certain QoS to guarantee in return for a certain price. Monitoring the veracity and compliance of these agreements at run-time happens through a reputation mechanism. This one permits for example to identify selfish providers that do not put the necessary efforts into maintaining the QoS they announce for their Web services. Reputation relies here on users' feedbacks after discarding non-honest and conflicting ones. In [7], Maximilien and Singh note that a reputation model for Web services could be made available upon request from independent and trusted parties known as agencies (others use reputation managers). The role of an agency is to aggregate the right information about a Web service's quality and present this information in a suitable format to potential users at selection-time.

In addition to support the selection of the best Web services, a reputation model would permit distinguishing good from bad Web services [12]. In a dynamic environment like the Internet, it is unlikely that all Web services will be tagged with a fixed reputation level. Web services appear and disappear, resume operation after suspension, come in a new shape, and get upgraded without any prior notice. Furthermore, they may compromise the good performance levels they used to provide to the end-users in the past.

In [10], Shaikh Ali et al. examine reputation to discover semantic Web services in a grid configuration. Reputation is motivated because of the assumption that in the future a service-rich environment will exist, which requires the selection between similar Web services being offered by independent providers. However, current technologies like UDDI do not help locate Web services based on their capabilities and behaviors. Shaikh Ali et al.'s work enables users among other things to efficiently discover a reputable Web service that is more suitable for their needs and to easily create and share high quality complex workflows. Backing the claim of Shaikh Ali et al. that current technologies are not appropriate for an efficient and accurate selection of Web services, Xu et al. [13] add that the announced Web services' non-functional details are questionable.To this end, they suggest a model of reputation that combines an augmented UDDI registry and a reputation manager that assigns scores to Web services based on customer feedbacks.

4. Community of Web Services

In [6, 11], our work on communities of Web services did not include reputation as a criterion for neither developing nor managing these communities. This section briefly presents the architecture of Web services communities and discusses the operation of such communities. In the next section, we specify the reputation that supports the development and the management of communities.

4.1. Architecture

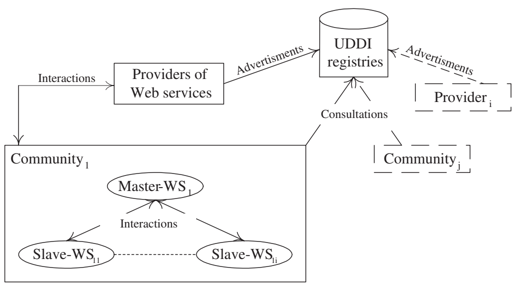

Figure 1. Communities of Web services

Fig. 1 represents the architecture we developed to portray communities of Web services and how they connect to providers of Web services and UDDI registries (or other types of registries such as ebXML). A community is established and dismantled with respect to some protocols (Section 4.2). The community is dynamic by nature: new Web services join, other Web services leave, some Web services become temporarily unavailable, some Web services resume operation after suspension.

A *master* component always leads a community. The other Web services of the community are denoted as *slaves*.

Within the same community, the interactions between the master and the slave Web services are framed according to specific protocols (Section 4.2). Some responsibilities of the master Web service include attracting Web services to its community using rewards (community's reputation not used here as an attracting factor), convincing Web services to stay longer in its community, and nominating the Web services to participate in composition scenarios.

4.2. Functions

A community undertakes many functions that we can classify into three categories: community management functions, Web services attraction and retention functions, and Web services nomination and selection functions for composition. For the needs of this paper the focus is on the first two categories only.

Community management. A community groups Web services with the same functionality. Its establishment is designer-driven and starts as follows. Initially, the designer defines the functionality like "FlightBooking" of a community. Afterwards, he deploys the master Web service that will lead this community, which means among other things inviting Web services to sign up in this community and checking their credentials, e.g., QoS, prior to final acceptance.

Dismantling a community is designer-driven as well and happens upon request from the master Web service. The master monitors all events happening in a community such as the arrival of new Web services, departure of existing ones, identification of Web services to be part of composite Web services, and imposing sanctions on Web services in case of misbehavior. If the master Web service notices that (i) the number of Web services in the community is less than a certain threshold and (ii) the number of participation requests in composite Web services that arrive from users over a certain period of time is less than another threshold, then the community will be dismantled. Both thresholds are predetermined by the designer. Web services to eject out of a community are invited to join other communities.

Web services attraction and retention. Attracting new Web services to and retaining existing Web services in a community fall under the responsibilities of the master Web service. A community could vanish if the number of Web services that reside in it drops below a certain threshold. Attracting Web services makes the master Web service regularly consult UDDI registries looking for new Web services. When a candidate Web service is identified according to its functionality, the master Web service engages the provider of this candidate in interactions. Some arguments that are used during interactions include high participation-rate of the existing Web services in compositions, and Web services efficiency in handling users' requests.

5. Reputation and Web Services Communities

5.1. Reputation Model

Prior to explaining our reputation model for communities of Web services, two comments are to be made: 1) research works on reputation have focused on Web services only (Section 3), and 2) current practice in Web services field is that there is a one-to-one relationship between a given community and its associated functionality, that is, there is only one community that represents a functionality such as FlightBooking (Fig. 1). However, since reputation is sustained with the existence of several options out of which one (sometimes several) is selected, the development of a reputation model for communities of Web services requires reviewing comment 2). As a result, we relax the relationship between functionalities and communities by making it one-to-many; a common functionality is allowed to be represented by several independent communities.

Henceforth, C_i denotes the candidate community i that is under consideration by either provider or user, and $|C_i|$ denotes the number of Web services members that are subscribed to community C_i. Next, we examine the role of a community reputation model as viewed by Web services users and providers.

Reputation from the user's perspective

As perceived by users, a community reputation model would help them select communities that can offer to them Web services that meet their quality expectations. The reputation model for communities would drive the discovery process that is based on aggregating and operating on historical performance metrics. In general, most of these metrics are monitored over an observation window, w, which is measured in days or weeks, etc., and recorded in run-time logs. Next, we outline some of the performance metrics that can play a role in assessing the reputation of a community as seen by users.

Responsiveness metric: is one of the important qualities that users care about when selecting a community. It is the metric that determines how fast a community master is at nominating a Web service slave from this community that can handle the user's request. The responsiveness of community C_i, denoted by $responsiveness_{C_i}$, is measured by computing the average of the n response times to the requests that took place throughout the observation window (w). To simplify the formulas, the window w is omitted. So,

$$responsiveness_{C_i} = \frac{1}{n}\sum_{k=1}^{n} rt_{C_i}^{k} \qquad (1)$$

Where $rt_{C_i}^{k}$ is the response time taken to select a Web service that will fulfill user request number k in a community C_i, and n is the number of requests observed over w.

InDemand: assesses how much users are interested in a specific community more than in any other one. This metric is measured by the percentage of requests that a community receives to the total requests submitted to all communities throughout w. That is,

$$inDemand_{C_i} = \frac{receivedRequests_{C_i}}{\sum_{k=1}^{M} receivedRequests_{C_k}} \quad (2)$$

Where M denotes the number of communities that are under consideration.

Satisfaction metric: represents the subjective opinion of clients who dealt with a community recently, i.e., over the observation window w:

$$satisfaction_{C_i} = \frac{pos_{C_i}}{pos_{C_i} + neg_{C_i}} \quad (3)$$

where pos_{C_i} and neg_{C_i} respectively represent the aggregation of positive and negative votes that a community C_i got from all clients.

Reputation from the provider's perspective

A reputation model would here support providers of Web services identify the communities where their Web services will sign up. Presently, we assume that a Web service is a member of at most one community[1]. Below we list some performance metrics that we conceive important to the final assessment of the community's reputation.

Selectivity metric: any provider wishes to join the community in which the provider has the maximum likelihood of being selected for fulfilling user's request compared to all other communities. Assuming that the community master adopts fair (uniform) selection policy among Web services, the selectivity of community C_i is the average number of requests that get delegated to a Web service member inside a community and is computed as follows:

$$selectivity_{C_i} = \frac{receivedRequests_{C_i}}{|C_i|} \quad (4)$$

where $receivedRequests_{C_i}$ is the number of requests that community C_i gets over the observation window w.

Market Share metric: is the ratio of the Web services subscribed to a community to the total number of Web services subscribed to all communities:

$$marketShare_{C_i} = \frac{|C_i|}{\sum_{k=1}^{M} |C_k|} \quad (5)$$

[1]"At-most" constraint could be relaxed by allowing Web services sign up in several communities at a time.

where $\sum_{k=1}^{M} |C_k|$ represents the total number of Web services that signed up in all M communities.

From the provider's perspective, $marketShare_{C_i}$ factor may be a double-edged sword. On the positive side, the larger $marketShare_{C_i}$ is the more powerful the candidate community is, which means a better chance of having it more often selected by clients. On the negative side, the bigger $marketShare_{C_i}$ is means a lower selectivity rate and a tougher competition for the provider.

Fairness metric: providers look for communities to sign up where they are treated equitably and fairly. In this context, the simplest definition of fairness is giving each provider almost the same opportunity of participation like its peer members in the community. This can be assessed by computing the frequency of participation of each provider followed by computing the dispersion among these frequencies. Higher spread means that some providers are favored over others; lower spread suggests that the community master adopts a fair internal selection policy.

A prospective provider will surely favor a community with a lower dispersion. A typical statistical tool for measuring the dispersion is the standard deviation, which we compute as follows:

$$\sigma_{C_i} = \sqrt{\frac{1}{|C_i|} \sum_{j=1}^{|C_i|} (rf_{C_i}^{WS_j} - \mu_{C_i})^2} \quad (6)$$

Where $rf_{C_i}^{WS_j}$ is the relative frequency of using Web service WS_j inside community C_i, i.e.,

$$rf_{C_i}^{WS_j} = \frac{receivedRequests_{C_i}^{WS_j}}{receivedRequests_{C_i}} \quad (7)$$

$receivedRequests_{C_i}$ denotes all requests received by community C_i, $receivedRequests_{C_i}^{WS_j}$ denotes requests received by community C_i then specifically delegated to Web service WS_j to process, and finally μ_{C_i} is the mean of all relative frequencies:

$$\mu_{C_i} = \frac{\sum_{j=1}^{|C_i|} rf_{C_i}^{WS_j}}{|C_i|} \quad (8)$$

Therefore, σ_{C_i} can tell the concerned provider how consistent the community master is when it comes to uniformly selecting Web service members for handling incoming requests. A larger σ_{C_i} means that the community master favors some Web services over others. Thus, fairness of community C_i can be defined as the inverse of σ_{C_i}:

$$fairness_{C_i} = \frac{1}{\sigma_{C_i}},\ where\ \sigma_{C_i} \neq 0 \quad (9)$$

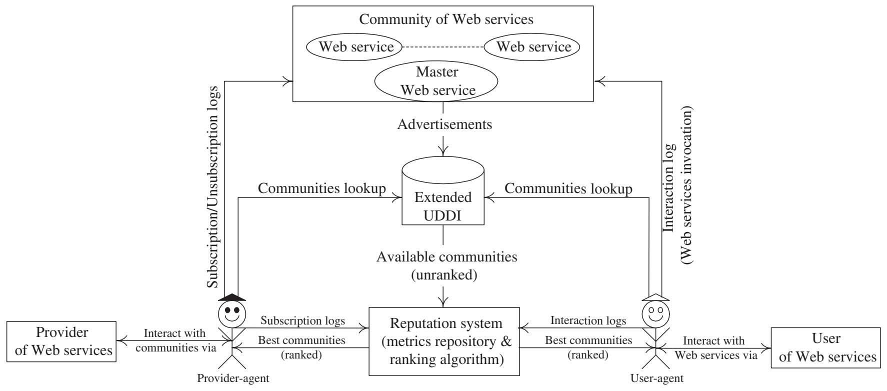
Community of Web services
Web service
Web service
Master Web service
Advertisements
Extended UDDI
Communities lookup
Communities lookup
Available communities (unranked)
Subscription/Unsubscription logs
Interaction log (Web services invocation)
Provider of Web services
Interact with communities via
Subscription logs
Best communities (ranked)
Reputation system (metrics repository & ranking algorithm)
Interaction logs
Best communities (ranked)
Interact with Web services via
User of Web services
Provider-agent
User-agent

Figure 2. Architecture of reputation-based Web services communities

5.2. Reputation-based architecture

Fig. 2 depicts an augmented Web services community architecture that takes into consideration the notion of community reputation. As shown, the architecture consists of the following main components:
Extended UDDI. The traditional UDDI registry of the service-oriented architecture contains entries that describe individual Web services. We suggest to extend the UDDI by enrolling additional entries for communities of Web services.
User Agent. It is a proxy between the user and other parties, namely, the extended UDDI registry, Web services communities, and the reputation system.
Provider Agent. Akin to the user agent, a provider agent is a proxy between the provider and other parties, namely, the extended UDDI registry, Web services communities, and the reputation system.
Reputation System. It is the core component in this architecture as it has two functions: (i) maintaining a repository of run-time operational data, called *logs* (explained below), that are needed to compute the performance metrics of a community, and (ii) ranking communities by their reputation using a proposed ranking algorithm, which is not the focus of this paper. Roughly speaking, the ranking algorithm computes reputation by analyzing a set of performance metrics such as those we list above. Performance metrics are the results of aggregating run-time logs intercepted by the independent user and provider agents as they handle interactions with the community.

To compute reputation from the perspective of users or providers, user agents should intercept the following time-stamped logs for each event that happens during the interaction with the community[2]:

[2]Due to space limitation, we list here only the logs that are needed to compute the example performance metrics we list in this paper.

Received Requests log counts all requests directed to the community by the user agents. This log is needed to compute *inDemand* and *selectivity* metrics.

1. *Response Time* log tracks how long it took a community's Web service master to find and nominate a Web service slave to handle a user's request. This log is required to compute the *responsiveness* metric.
2. *Invocations* log tracks all received requests that have been accepted by the community master and invoked by the designated Web service slave. This log is required when computing the *fairness* metric.

In addition to these logs, and to compute the provider-perceived performance metrics, provider agents should intercept the following additional time-stamped logs:

1. *Subscriptions/Unsubscriptions* logs track providers signing up/off to a community. These logs are needed to compute the *market share* metric as they can tell how many Web service members are presently there in the community.
2. *Suspensions/Reactivations* logs track Web service being suspended/reactivated by the community master. These logs are needed to compute the *market share* metric as they can tell how many Web service members are there in the community that are ready to serve (i.e., not in a dormant state).

The different logs permit to compute performance metrics in order to assess a community reputation. Data stored in these logs are monitored and collected by agents independent of the three key players in the computing platform (users, providers and community masters). This is aimed at fostering the trust between them.

As shown in Fig. 2, a Web services community is primarily represented by its master, which plays the role of the community gate to the external world. Specifically, the master is responsible for advertising the existence of its community at multiple Extended UDDI registries. In addition, it is in charge of administering the (un)subscriptions sought by providers' agents. And most importantly, the master is in charge of handling Web service requests relayed by users' agents. From user's perspective, they wish to be served by WS communities that provision high and reliable quality of service. When a user (or application) needs a Web service, it delegates its request to an agent, which is in charge of searching for a reputable community that provides the desired functionality (e.g., `GetStockQuote`) and expected QoS criteria (e.g., `maximum response time`). The agent looks up the Extended UDDI for a list of all feasible, yet unranked Web services communities that satisfy the minimum requirements of the user. In order to select the top communities, the unranked list is dispatched to the reputation system in order to rank communities based on their reputation using our under-development ranking algorithm. The top list of reputable communities is eventually returned to the user through its agent.

Providers, on the other hand, are concerned with signing up in reputable communities hoping to increase their user outreach and visibility in the computing marketplace. To that extent, providers' agents search for communities which match the providers's functionality (e.g., `FlightBooking`). The agent looks up the Extended UDDI and submits the matching list to the reputation system, which ranks communities based on their reputation. Finally, the ranked list of communities is returned to the provider through its agent for selection.

6. Conclusion

In this paper, we discussed a reputation model for communities of Web services. Both users and providers seek reputable communities where the former wish to get high quality of service while the latter wish to increase its visibility and its outreach in order to reap lucrative income. The different reputation perspectives essentially entail different methods for computing community reputation, each of them requires different sets of historical performance metrics to analyze. In this preliminary investigation, we propose a reputation-based architecture for Web services communities in which the Extended UDDI, user and provider agents, and the reputation system are the key components in it. The extended UDDI is supposed to host entries for communities in addition to the conventional entries of Web services. The agents play an important role as independent parties that intercept run-time operational logs needed for computing the performance metrics. The reputation system is the core component of the architecture that retains the data (logs) needed for calculating the reputation of a community, and implements the reputation ranking algorithm. Presently, we are working on designing and implementing an effective ranking algorithm that transforms the performance metrics of a community into a single value that represents its reputation.

References

[1] B. Benatallah, Q. Z. Sheng, and M. Dumas. The Self-Serv Environment for Web Services Composition. *IEEE Internet Computing*, 7(1), January/February 2003.

[2] J. Bentahar, Z. Maamar, D. Benslimane, and Ph. Thiran. Using Argumentative Agents to Manage Communities of Web Services. In *Proceedings of WAMIS'2007*, Niagara Falls, Ontario, Canada, 2007.

[3] T. Bui and A. Gacher. Web Services for Negotiation and Bargaining in Electronic Markets: Design Requirements and Implementation Framework. In *Proceedings of HICSS'2005*, Big Island, Hawaii, USA, 2005.

[4] S. Elnaffar. Beyond User Ranking: Expanding the Definition of Reputation in Grid Computing. In *Proceedings of CIS^{2}E'2006*, 2006.

[5] R. Jurca and B. Faltings. Reputation-Based Service Level Agreements for Web Services. In *Proceedings of ICSOC'2005*, Amsterdam, The Netherlands, 2005.

[6] Z. Maamar, M. Lahkim, D. Benslimane, P. Thiran, and S. Sattanathan. Web Services Communities - Concepts & Operations -. In *Proceedings of WEBIST'2007*, Barcelona, Spain, 2007.

[7] M. Maximilien and M. Singh. An Ontology for Web service Ratings and Reputations. In *Proceedings of OAS'2003*, Melbourne, Australia, 2003.

[8] B. Medjahed and Y. Atif. Context-based Matching for Web Service Composition. *Distributed and Parallel Databases, Springer*, 21(1), January 2007.

[9] B. Medjahed and A. Bouguettaya. A Dynamic Foundational Architecture for Semantic Web Services. *Distributed and Parallel Databases*, 17(2), March 2005.

[10] A. Shaikh Ali, S. Majithia, O. Rana, and D. Walker. Reputation-based Semantic Service Discovery. *Concurrency and Computation: Practice and Experience, John Wiley & Sons*, 18(8), 2006.

[11] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar. Towards an Approach for Web Services Substitution. In *Proceedings of IDEAS'2006*, Delhi, India, 2006.

[12] Y. Wang and J. Vassileva. A Review on Trust and Reputation for Web Service Selection. In *Proceedings of ICDCSW'07*, Toronto, Ontario, Canada, 2007.

[13] Z. Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-Enhanced QoS-based Web Services Discovery. In *Proceedings of ICWS'2007*, Salt Lake City, Utah, USA, 2007.