1-1-2010

A managerial community of web services for management of communities of web services

Abdelghani Benharref
Abu Dhabi University

Mohamed Adel Serhani
United Arab Emirates University

Salah Bouktif
United Arab Emirates University

Jamal Bentahar
Concordia University

A Managerial Community of Web Services for Management of Communities of Web Services

Abdelghani Benharref
Engineering and Computer Science, Abu Dhabi University
Abu Dhabi, United Arab Emirates
abdelghani.benharref@adu.ac.ae

Mohamed Adel Serhani, Salah Bouktif
College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates
{salahb,serhanim}@uaeu.ac.ae
Jamal Bentahar
Concordia Institute of Information Systems Engineering
Concordia University, Montreal, Quebec, Canada
bentahar@ciise.concordia.ca

Abstract — **Nowadays, Web services are considered as new and attracting distributed approach of application/services integration over the Internet. As the number of Web Services is exponentially growing and expected to do so for the next decade, the need for categorizing and/or classifying Web Services is very crucial for their success and the success of the underlying SOA. Categorization aims at systematizing Web Services according to their functionalities and their Quality of Service attributes. Communities of Web Services have been used to gather Web Services based on their functionalities. In fact, Web Services in a community usually offer similar and/or complementary services. In this paper, we augment Web Services communities' classification by adding a new support layer for Quality of Service classification. This is done through Quality of Services specification, monitoring, and certification of different Web Services. A Web Service might be admitted to a community thanks to its high Quality of Service or might be ejected from a community due to its low Quality of Service. We propose a managerial community of Web Services that is able to monitor and certify Quality of Web Services in other communities. This managerial community offers services to other communities, Web Services providers, and Web Services clients by monitoring and certifying Web Services. The focus of this paper is the use of the managerial community to select Web Services.**

Keywords: Web Services, Communities of Web Services, Quality of Services of Web Services Selection of Web Services.

I. INTRODUCTION

The phenomenal growth of Internet technologies, largely impacted by the eXtensible Markup Language (XML) and its related technologies is extending the traditional role (client-to-business) of the World Wide Web to a better support of Business-to-Business interactions. The future perspective of the Internet is being driven by a new concept commonly known as Web Services technologies [1].

A Web Service can be defined as an application that exposes its functionality through an interface description and makes it available for use by other programs. Web Services allow computers and devices to automatically interact with each other using the Internet to exchange and gather data. Moreover, on one hand, a composite Web Service can further be created by aggregating a set of Web Services to produce a more complex Web Service with a wide range of functionalities. On the other hand, a set of Web Services can form and operate inside a community.

In the Revised Webster dictionary, a community is defined as "*a body of people having common rights, privileges, or interests, or living in the same place under the same laws and regulations.* On a similar path, a community of Web Services can be composed by Web Services offering the same functionalities or sharing similar concerns.

Even with a huge number of related works on Web Services and somehow a reasonable amount on communities of Web Services (e.g. [2], [3], [4]), there is a lack of mechanisms and approaches to establish inter-community and intra-community rules and to enforce them.

The aim of this paper is, first, to define the rights of a community and participating Web Services, their duties toward peers and clients, and it proposes a novel certification and monitoring approach to enforce all of these to protect the community, its reputation, its interest, and those of each individual Web Services. Although other aspects are discussed, the focus in this paper is on Web Services Quality of Service (QoWS) mainly in selecting Web Services.

Defining and enforcing terms and regulations of/within communities of Web Services raise a set of questions including:

- How to represent a rule?
- How to make sure a participating Web Service respects rules of the community?
- When a Web Service might be authorized/invited to join a community?
- When a Web Service must be ejected from a community?
- How members of communities should distribute the load to fairly share benefits and to guarantee a certain QoWS?

978-1-4244-7068-6/10/$26.00 ©2010 IEEE

- How to define interactions with other communities?
- When to interact with other communities?
- As a member of a community, how to find and select a community to get services from whenever needed?

Although this paper does not answer all of these questions, we propose a managerial community of Web Services for management of communities of Web Services. This managerial community is composed by Web Services instrumented with adequate functionalities and services to assess the QoWS of other Web Services. Such a Web Service is called Managerial Web Service (MWS). Assessment includes test and certification of a Web Service as a partial-requirement to join a community. Moreover, once Web Services are part of a community, the managerial community can monitor, periodically or on request, their behavior and interactions on the fly to detect any potential violation to the terms of their community, which might result in ejection from the community. Moreover, a participating Web Service can make use of the managerial community to show how much it is useful for the community and get some business credit or consideration. Finally, clients of Web Services can use the managerial community to select a community that suits their needs. In fact, many communities are likely to be competing by offering similar services with different conditions. The managerial community can advise a client which community to do business with according to its requirements and the status of the selected community (as known by the managerial community).

The remaining sections of this paper are organized as follows: next section discusses related works. Section III presents our managerial community and terms and rules to be respected while operating within a community, while section IV discusses the list of services a member of the managerial communities should support. A proof of concept summarizing our experience in using the managerial community for selection of Web Services is presented. We conclude by conclusion and future work in section VI.

II. RELATED WORK

In general, management of Web Services as well as their QoWS (specification, publication, and discovery) are becoming more and more important as the number of similar, though competing, Web Services available in the Internet proliferates and the need for communities and composition of Web Services increases. Management of QoWS, as an integral part of Web Service management, will play an important role for the success of this paradigm. On one hand, providers of Web Services will have to specify and guarantee QoWS to remain competitive and achieve the highest possible returns on investment from their businesses. On the other hand, clients will have the possibility to look for appropriate Web Services according to their QoWS preferences (e.g., highly available, respond to client's requests in reasonable time, etc.).

As discussed before, works on communities of Web Services are mostly on establishing and building communities rather than managing communities and enforcing appropriate rules. However, there are some works on management of Web Services that are relevant to this topic. Hereafter is a short list of some works of interest to this paper.

Managing QoWS of Web services as component of Web Service management has been addressed by several research initiatives. In [5] the work introduces sPAC (Web Services Performance Analysis Centre) and shows how customers can verify timeliness of their Web Services semi automatically from the description of workflow of Web Services to reports analysis and estimation results. In [6], the paper identifies a set of QoWS metrics in the context of Web Services workflows, and proposes a unified probabilistic model for describing QoWS values of a broader spectrum of atomic and composite Web services. In [7], the paper proposes a QoWS-aware binding approach based on Genetic Algorithms. The approach includes a feature for early run-time re-binding whenever the actual QoWS deviates from initial estimates, or when a service is not available. The approach has been implemented in a framework and empirically assessed through two different service compositions.

In [8], the authors survey the key features of Web Services management system (WSMS) and conducts a comparative study on how current research approaches and projects fit in. In [9], the authors propose a Web Service gateway to monitor and control Web Service access according to SLAs and organizational policies. The authors in [10] presents an implementation to derive on-line monitors for Web Services automatically from SLAs using an Eclipse plugin. The efficiency and scalability of this approach using a large-scale case study in a service-oriented computational grid has been evaluated.

Several works proposed broker-based architectures for QoWS management ([11], [12], [13]). In these architectures, a broker mediates between clients and providers of Web Services by providing a set of QoWS management operations such as: QoWS verification, QoWS certification, QoWS-based Web Service selection, QoWS negotiation, and QoWS monitoring. However, this model is not scalable considering the number of clients and the number of Web Services that might need to be supported by the broker. Moreover, QoWS properties might be managed differently due to the nature of each property. For example, managing Web Service availability requires a simple invocation of a Web Service by the broker and a check if it is responding over a period of time. However, management of Web Service's response time requires that the broker implements or reuses measurement and monitoring techniques to measure the time a client's request is sent and the time its response is received.

Interested in Web Service management simplification, Tosic et al. ([14]) have used the 'class of service' term as a discrete variation of the complete service and QoWS. Authors demonstrated that using classes' specification and management is simpler, faster and incurs less run-time overhead than using custom-made service level agreements (SLAs), client's profiles, or separate Web Services. It is then often easier and faster for a consumer to switch to another class of service within the same Web Service than to search for a replacement Web Service or to renegotiate an SLA. For the sake of the formal specification of various types of constraints, Tosic et al.

have developed the Web Service Offerings Language (WSOL) [15]. It references one or more WSDL files and specifies additional information. A corresponding management infrastructure called the Web Service Offerings Infrastructure (WSOI) was developed to manage monitoring and dynamic manipulation of WSOL service offerings.

III. MANAGERIAL COMMUNITY OF WEB SERVICES

A managerial community of Web Services is a community that is composed of QoWS-management-capable Web Services. Each of these Web Services can assess the QoWS of a Web Service and can passively monitor it while the latter is operating inside a community and/or interacting with peers and/or clients.

A. Overall architecture

The core idea in our approach is the managerial community of Web Services. Figure 1 illustrates an environment with one managerial community, two normal competing communities, a client, and a provider. Two communities are said to be competing if they are offering same functionalities in the same marketplace.

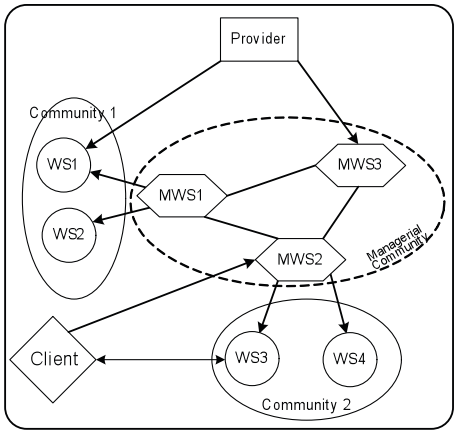

Figure 1. Managerial community

Client: in our approach, a client might invoke the managerial community to help in finding and selecting an appropriate Web Service and/or an appropriate community of Web Services. The managerial community has information on Web Services and communities that has been previously involved in their management. Using this information, managerial community can guide the client during Web Services selection. Moreover, a client might request monitoring of a Web Service with which it is interacting to assess the quality of the interaction.

Provider: a provider is the organization that owns and offers a Web Service. A provider of a Web Service invokes the managerial community to assess and certify the quality of the Web Service it is providing. This might be for internal Quality Assurance (QA) auditing assessment or as partially required by a community the Web Service is willing to join.

Communities 1 and 2: these are normal communities where: Web Service 1 and Web Service 2 offer complimentary services while Web Service 3 and Web Service 4 are competing.

Managerial community: this is the community taking care of management of Web Services in communities 1 and 2 (and probably others).

A managerial community can have expertise in one application domain (e.g. weather information) or various application domains (e.g. banking services, hotels booking, car rental, weather information etc). The number of domains of expertise depends on the orthogonal categorization of involved Web Services. In the latter case, the community will present a hierarchy of sub-communities (or sub-sub-communities) each with expertise in one application domain.

A client or a provider has first to locate a member of the managerial community, then, solicits its support for Web Services selection, certification, or monitoring. Second, Web Services in the managerial community communicate and cooperate in order to efficiently guide the process of QoWS operations. Each MWS has enough expertise in a single domain to be able to manage Web Services within that domain. Moreover, it might interact with other peers in the same community whenever needed.

A managerial community might receive two types of requests: 1) single-domain requests (SDR) and 2) multi-domain requests (MDR). A request is said to be single-domain if it can be satisfied in a single domain of expertise, that is, within one (sub) community. However, multi-domain requests require cooperation between two or more different domains of expertise (e.g. car rentals and travel package), hence, requiring cooperation of many (sub) communities. Each multi-domain request, received by a managerial community is decomposed into single-domain requests, falling each within a single domain. For processing, each single-domain request is assigned to a MWS in the corresponding community.

To guarantee the scalability of the managerial community, and upon reception of a request, a MWS processes it if it is not overload. If it is overloaded, it forwards the request to appropriate peers in order to find out a suitable MWS that can process this request.

B. Communities terms and rules

The decision to create a managerial community might be implied by many factors but most probably by a business decision. In this case, the business model of the community defines who initiates the process to build the community. Initially, the managerial community is empty and MWSs are added following a predefined procedure. The same procedure defines how these Web Services leave the community.

1) Join: a MWS joins a managerial community following an invitation from that community (join-out) or by issuing a join request to the community (join-in). In both cases, the Web Service should agree to the community terms and rules and the community should honour the conditions of the Web Service, if any.

2) Leave: unless otherwise specified by the community terms and rules, a Web Service can decide to leave the

community at any time. However, it should follow the leave procedure as specified by the community. Moreover, if a Web Service violates the terms and roles of the managerial community, it might be requested to momentally leave or its membership completely terminated.

A Web Service within the managerial community has to respect the rules and terms that govern that community. Although we present these rules for managerial community, they can be adapted to normal communities. Rules and terms might include, but not limited to:

- Interfaces: a MWS should implement and publish interfaces to interact with its peers (other MWSs), with providers of other Web Services, and with clients (details in section IV).
- Service provision: unless the MWS is loaded beyond a certain threshold, it should process requests to the best of its knowledge within a reasonable QoWS.
- Protect community privacy: information related to each member of the community and MWS should not be shared with any other entity outside the community unless a written informed consent is provided.
- Proper leave: a MWS should follow appropriate steps to leave the community. Therefore, it needs to inform other peers and the manager of the community, if any.
- Non-abuse of peers: a MWS should not intentionally overload peers with useless requests or with legitimate requests if peers are overloaded.
- Load balancing: a MWS should respect the community load balancing policies in requesting services from peers (e.g. round robin, no request after threshold until notified, etc).
- Accurate broadcast: MWS should engage in advertising accurate information about their status to the best of their knowledge.
- Ban reprimand: a MWS might be banned from the communities if it doesn't respect the above rules.

The manager is responsible of determining whether a Web Service respects (or not) the community terms and conditions. In case it violates (some of) these terms and conditions, the manager enforces a set of penalties that can range from simple warning to Web Service dismissal from the community.

IV. MWS SERVICES

A MWS offers services to its peers in the managerial community, to Web Services providers, to other Web Services communities, and to clients of Web Services. Although, the main service offered to all of these partners is the verification and monitoring of QoWS of a Web Service, a MWS offers services to its peers as part of its duties while operating within a community.

A. Services to peers

Cooperation between MWSs is conducted through the MWS-MWS interface. It concerns three categories of interactions: negotiation of mutual services, validation/retrieval of information about a given Web Service, and exchange of summary reports and status information.

Negotiation of mutual services: MWS negotiate the terms and conditions of the services they deliver or they receive using SLA. An agreement specifies the kind of services a MWS is willing to provide to other MWSs and the cost of each of these services (if any).

MWS requests delegation: whether serving a community, a Web Service provider, or a client, when a MWS cannot process a request due to lack of expertise or high load, it requests cooperation of other unloaded MWS with appropriate expertise. MWS provides support for Web Services selection based on client's QoWS requirements, verification of QoWS claimed by Web Service providers, and QoWS monitoring. In a managerial community, a MWS may not have enough knowledge about a specific Web Service when making decisions (e.g. selection of potential Web Services). This is eventually the case for a composite Web Service offering different services and requiring different expertise domains. In this case, a MWS may ask other MWS within its community in order to get information about that Web Service, such as whether its QoWS has been verified before, and if any, what was the verdict of that process.

Sharing of Web Services' rating information. MWS within the same community may share rating information of Web Services, in very restrained situations, by sending reports to each other periodically or on demand (e.g., list of top qualified Web Services, list of worst qualified Web Services). These reports are dated and updated by all MWS and made available to other MWS belonging to that community.

Sharing load and summary reports. MWS can get help from each other when they receive a large number of requests from clients. Thus, they need to inform each other about their loads.

Figure 2 depicts a partial WSDL description of the MWS-MWS interface illustrating three MWS operations: requesting Web Service reputation, QoWS information, and MWS load status. Interfaces with clients, providers, and other communities are similar to Figure 2 and hence, will not be illustrated.

```
<wsdl:definitions …..
<wsdl:message name="getQoWSInfo">
<wsdl:part name="return" type="SOAP-ENC:string"/>
<wsdl:part name="QoS parameters" type="wsx:QoWS
Information"/>
<wsdl:part name="Web Service Name" type="wsx:QoWS of
Web service"/>….
<wsdl:portType name="MWSService">
<wsdl:operation name=" getRate" parameterOrder="symbol">
<wsdl:input message="intf:getRateRequest"/>
<wsdl:output message="intf:getRateResponse"/>
</wsdl:operation>
</wsdl:portType> ….
</wsdl:binding> </wsdl:definitions>
```

Figure 2. Partial WSDL of MWS-MWS interface

B. *Services to providers of Web Services and other communities*

As part of their responsibilities and as stated by their business models, managers of communities of Web Services should (would like to) protect their communities, Web Services members of these communities, and their clients. Before adding a Web Service to a community, the manager should make sure the stature of this potential member is at an acceptable level and will improve the reputation of the community or, at the worst case, will not downgrade this reputation. In addition to strict respect of the terms and rules stated in section III.B, the stature of a Web Service is impacted by the QoWS properties presented in section D.

The managerial community can fully verify the QoWS of a Web Service. This verification consists of checking if the QoWS claimed by a Web Service is valid. This requires generation and application of tests cases and/or passive monitoring interactions of that Web Service with clients. Verification of QoWS might be required in two scenarios: 1) when adding a Web Service to a community and 2) when a provider would like to certify the QoWS its Web Service can offer.

C. *Services to clients of Web Services*

Web services clients can make use of the managerial community in two situations: 1) to select a Web Service, and 2) to monitor interactions with a Web Service.

Selecting suitable Web Services with regards to QoWS provision is a determinant factor to ensure customer satisfaction. Different users may have different requirements and preferences regarding QoWS. For example, a client looking for a Web Service may require minimal reputation while satisfying certain constraints in terms of price and availability; while another client may put more emphasis on the price rather than the reputation; others consider more the availability of a Web Service rather than both previous properties. As the managerial community collects sufficient information about Web Services (with their consent), it can be invoked during selection of Web Services based on QoWS.

As for monitoring, each MWS, member of the managerial community, is capable of passive monitoring of Web Services using passive testers. This monitoring is of prime importance to assess the QoWS of a Web Service when serving clients and/or operating within a community.

The following subsection discusses a minimal set of QoWS properties that our managerial community supports up to now.

D. *QoWS properties*

The set of QoWS properties (response time, cost, reputation…) can be very large and depends widely on Web Services and their clients. To keep the paper concise, we only consider four main properties: availability, reputation, response time, and cost.

- Response time: this represents the time needed between issuing a request and getting its response.
- Cost: this is the cost charged for using a Web Service. The Web Service cost may be estimated by operation, by volume of exchanged data, and/or a flat rate plan.
- Availability: it represents the probability that a Web Service is accessible (available for use) or the percentage of time that the Web Service is operating.
- Reputation: this is a measure of Web Service trustworthiness. It depends on clients' experiences in using the Web Service.

V. IMPLEMENTATION

To demonstrate the usefulness of the managerial community, we have developed a partial proof of concept. To cope with space issue in this paper, we just report some of our experimentations and results in using the managerial community (and its MWSs) for Web Services selection. As stated above, this is useful for clients of Web Services and managers of communities.

Using this prototype and the selection algorithm of Figure 4, we have conducted a series of experiments in order to evaluate QoWS-aware Web Service selection schemes supported by certification and monitoring within the context of the managerial community. Scenarios in which all components of the managerial community are involved have been considered. To keep up with the scalability of the community, two selection policies of MWS have been applied: 1) round robin and 2) threshold-based admission control. In addition, a clients' generator application has been developed so that a large number of requests can be sent to the community. Each request specifies the QoWS requirement of the client including the desired weight for each QoWS parameter. Each weight represents the importance of the QoWS property to the client. During the experiments, the number of requests has been gradually increased and the type of requests diversified (single domain and multiple domain). The impact of these two factors on Web Services selection has been evaluated.

For the sake of this prototype, we have chosen to specify the QoWS information in a separate document as in Figure 3 instead of extending the WSDL of each Web Service with corresponding QoWS information. The reason behind this is that specifying QoWS information in WSDL documents is not flexible because of frequent changes of QoWS (compared to changes of operations definitions in WSDL). Using a separate format is more appropriate and allows flexible modification of QoWS information from QoWS description document.

The development platform we have used is NetBeans Enterprise pack, which includes the development environment (IDE 6.0 Preview (M9)), Sun Java System Application Server 9, and MySQL Database. Also, an extended QoWS-aware Web Services registry called UDDIe [16] have been used. Moreover, we have used an application that has been implemented in [11] to support the provider and the client in the publication and the discovery of QoWS-aware Web Services.

<**QoWS** *name*= "Name of Web Service">
<**Profile** name="GOLD">
<**operation** *name*= "op1" *Reputation = NULL RTmin = 8ms RTmax=10ms Cost= "$0.1" Min Availability= 90%*</**operation**>

<**operation** *name*= "op2" *Reputation = 4 RTmin = 5ms RTmax=8ms Cost= "$1" Min Availability= 80%* </**operation**>
</**Profile**>
<**Profile** name="SILVER">
<**operation** *name*= "op1" *Reputation = 3 RTmin = 10ms RTmax=20ms Cost= "$0.1" Min Availability= 50%*
</**operation**>
<**operation** *name*= "op2" **….** </**operation**>
</**Profile**>
</**QoWS**>

Figure 3. Example of QoWS Specification

A. QoWS-driven Web Services' selection

In the following, we formalize the Web Service selection process using a mathematical model (Figure 4), which is based on utility functions. These functions consider the QoWS properties presented above (response time, cost, availability, and reputation). Each incoming request belongs to a specific type. Figure 4 describes our algorithm for QoWS-driven Web Services' selection using a community of MWS.

The objective of the algorithm is to optimize/maximize the global utility function (GU_{ki}) based on the QoWS requirements of the client as shown in the algorithm above. A utility function is used to measure the degree at which a MWS fulfills a client's request with respect to its required QoWS. Request types are known in advance from the WSDL document of the Web Services. The global utility function (equation 7 of Figure 4) is computed based on the aggregation of four utility functions (equations 3, 4, 5, and 6). Each utility function is associated with a specific QoWS property. For a requested service of type i, U^1_{ki} maximizes the availability, U^2_{ki} minimizes the cost, U^3_{ki} maximizes the reputation, and U^4_{ki} maximizes the response time. We normalize equations 3, 4, 5, and 6 to a scale between 0 and 1 to have a linear representation of the global utility function. Equation 7 includes the weight assigned by the client to each quality property depending on his/her preferences. The weight is a decimal number, which is between 0 and 1 (equation 2).

The managerial community optimizes the global utility function GU of each MWS and returns back the best match Web Service that serves the QoWS preferences of the client request by considering the weight given to each QoWS property value specified in the client request as described earlier is Figure 4 as α, β, δ, and φ.

B. Test-bed Configuration

In this experimentation, we are considering a case study that involves selecting the Weather Forecasting, Stock Information, and Integrated Payment Web Services with scenarios including Web Services providers, clients, and a managerial community with three MWS. The Test-bed used in the experiments consists of:

- A managerial community with three MWS that manage the same set of QoWS-aware Web Services (WF, SI, and IP).
- QoWS-aware Web Services support different QoWS. Weather Forecasting (WF), Stock Information (SI), and Integrated Payment (IP) Web Services. Table 1 presents a description of each Web Service and the main operations it provides. MWS and Web Services have been deployed on different hosts.

Let $B = \{B_1,\ldots,B_n\}$ the set of MWS in a given managerial community
Let $T = \{T_1,\ldots,T_n\}$ the set of types of requests that may be addressed to that community
Let T_i a request type from T
Let B_k a MWS from B
Let $W^k = \{WS_{1k}, WS_{2k},\ldots,WS_{pk}\}$ the WS_s managed by the MWS k.
Let AV_{WSi} the availability of WS_m
Let C_{WSm} the cost of WS_m
Let P_{WSm} the reputation of WS_m
Let R_{WSm} the response time of WS_m
Let r_{ij} a request of type T_i from client j sent to all MWS of the community.
The QoS requirements of client j may be expressed by:

(1) $QoWS_j = \alpha AV + \beta C + \delta P + \varphi R$

Where $\alpha, \beta, \delta, and \varphi$ are the weights given by the client to each quality property.

(2) $\alpha + \beta + \delta + \varphi = 1$

B_k utility functions for each QoS property are:

(3) $U^1_{ki} = Max(AV_{ws_m})$ // WS_m in W^k

(4) $U^2_{ki} = Min(C_{ws_{mi}})$

(5) $U^3_{ki} = Max(P_{ws_{mi}})$

(6) $U^4_{ki} = Min(R_{ws_{mi}})$

Our global utility function is defined by:

(7) $GU_{ki} = \alpha \underset{\rightarrow}{U^1_{ki}} + \beta \underset{\rightarrow}{U^2_{ki}} + \delta \underset{\rightarrow}{U^3_{ki}} + \varphi \underset{\rightarrow}{U^4_{ki}}$

All the above utility functions (3, 4, 5 and 6) are normalized to a scale between 0 and 1 to have a linear representation of the global utility function.
A MWS selects the Web Service that maximizes the global utility function (GUki).

Figure 4. QoWS based Web Service selection in a mnagerial community

- A multi-threaded Java client that generates requests to the managerial community. The client application allows the specification of the following:
  - Test setup: number of requests, period of test, requests distributions, etc.

- Locations of MWS (and their manager) that will receive the requests generated by the client application.

Table 1. Web Services desoperations

Web Services	Description	Main operations
Weather Forecasting (WF) Web Service	Provides information about the weather in any city in the world min, max…)	*getAverageTemperature(), getSunrise(), getSunset(), getRelativeHumidity(), getWindForecast(),…*
Stock Information (SI) Web Service	Implemented a set of operations to get information about the stock market.	*getMarketCompanies(), getCompanyQuote(), getCompanyCurrentClosing(), getCompanyPreviousClosing(),*
Integrated Payment (IP) Web Service	Provides information about payment of bills.	*getTotalAmountDue(), payAmount(), viewPaymentHistory(), viewBillingHistory(),…*

C. Experiments

We have conducted experiments with the above configuration by considering different scenarios for a number of generated requests addressed to the managerial community. We have two main scenarios: 1) QoWS-aware selection of Web Services using one MWS and 2) QoWS-aware selection of Web Services using the managerial community (i.e. many MWSs).

1) Scenario #1: one MWS

We have generated a series of requests to select Web Services using the single MWS model where a MWS works outside a community and does not cooperate with other MWSs. For each request, we calculate both theoretical global utility function (TGU) and experimental global utility function (EGU). The TGU is calculated using the formula 6 described in the algorithm of Figure 4 and by using the requests requirements as parameters of the formula. EGU is calculated based on the results of conducted experiments. We also calculate the time the MWS takes to process the request: MWS processing time. The aim of this experiment is to evaluate the single-MWS-based Web Service selection with main focus on: 1) Scalability of the single MWS model, 2) the degree of satisfaction of each client's preferences, and 3) the capacity of a single MWS to manage heavy load.

The result of this experiment is shown in Figure 5.

2) Scenario #2: a community of MWSs

We have generated a series of requests to select Web Services using the managerial community. As we did in the previous scenario, we calculated both the theoretical and the experimental global utility functions (TGU and EGU). The aim of this experiment is to evaluate the community configuration with MWS implemented as basic Web Services that cooperate together to achieve Web Services selection. In this scenario, we consider the load of each MWS in handling clients' requests as an important variable for selecting Web Services. The result of this experiment is shown in Figure 6.

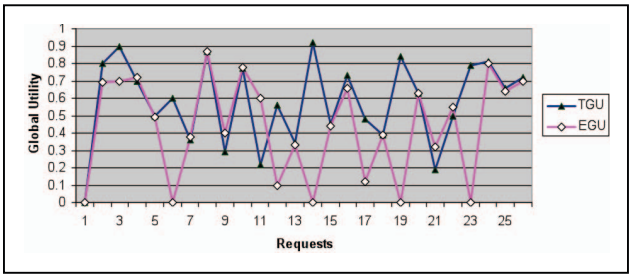

Figure 5. Web Service selection using a Single MWS

3) Discussion

Figure 5 and Figure 6 illustrate the distribution of TGU and EGU values calculated in both scenarios respectively: single MWS model and community of MWSs. By analyzing the curves, we can clearly deduce that the EGU curve is closer to the TGU curve while moving from the single MWS model to well established managerial community. The closer the two curves are, the better that clients' requests are fulfilled with regards to their required QoWS. This confirms that by using a community of MWSs, most of requests are fulfilled with respect to the QoWS they require. We also observe that in Figure 5, which corresponds to the scenarios of a single MWS, a considerable number of requests (represent 20% of total requests) haven't been processed, thus their EGU is zero or very close. This can be explained by the fact that a single MWS cannot process these requests either because the QoWS these requests require cannot be guaranteed by this MWS or because the MWS reaches its capacity and can not process further requests.

In summary, we can deduce from the comparison of Figure 5 and Figure 6 that the managerial community is a better model in serving clients requests because it provides the best match service that maximizes the value of the global utility function. Below are some key interpretations we can conclude from the above experiments:

- For all requests generated to the community, all of them have been processed because the aggregation of MWSs together served all the requests of different domains. The community is able to satisfy a wide range of clients' requests with different QoWS requirements. Moreover, this model can distribute the load among MWSs members of the community during periods of heavy load. In addition, the community can scale to include other MWSs covering different domains.

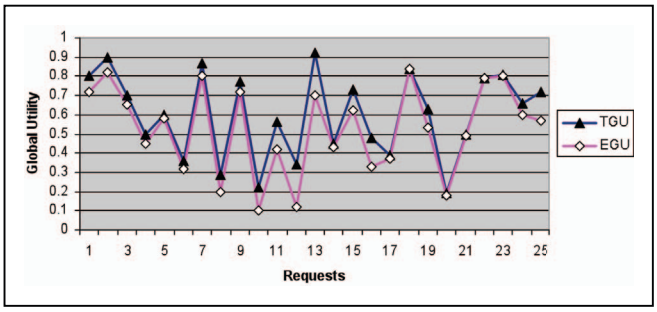

Figure 6. Web Services selection using the managerial community

- For all requests generated to the single MWS model, only 80% of requests have been processed. This can be explained from 3 perspectives: (1) the single MWS is overloaded, or (2) the MWS is not able to process a given request because it is out of its expertise, (3) the MWS can not guarantee the QoWS value specified in the request.
- With a managerial community, a client request is always redirected to an appropriate MWS. This reduces the time, the effort, and the overload that can be induced in case a wrong MWS is selected.
- Once a MWS reaches its capacity and cannot process additional requests, an alternate MWS from the community can take over and process subsequent requests.

VI. CONCLUSION

Nowadays, Web Services providers are trying to maximize their revenues by creating and/or joining appropriate communities. In a community, a Web Service has better visibility and benefits from cooperation of other members when it is overloaded or lacking expertise in a requested domain. Furthermore, owners or members of communities of Web Services would like to protect their communities and its benefits.

In this paper, we proposed a managerial community of Web Services to help in assessing and monitoring the quality of service provided by a Web Service. The managerial community is useful before adding a Web Service to a community or to monitor a Web Service operating within a community. Such monitoring gives communities' managers very important and sensitive information about behaviors of different Web Services in their respective communities.

As a proof of concept, we presented an example for selecting Web Services to join a community or to serve a client's request. In our ongoing and future work, we are working on the implementation and tuning of other functionalities such as the passive monitoring of different Web Services within a community.

REFERENCES

[1] W3C, "Web Services Architecture," 2006; http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[2] B. Benatallah, et al., "Facilitating the rapid development and scalable orchestration of composite web services," Distributed and Parallel Databases, vol. 17, no. 1, 2005, pp. 5-37.

[3] Z. Maamar, et al., "Web Services Communities-Concepts & Operations," Proc. The 3rd international conference on Web information systems and technologies, 2007.

[4] B. Medjahed and Y. Atif, "Context-based matching for Web service composition," Distributed and Parallel Databases, vol. 21, no. 1, 2007, pp. 5-37.

[5] H. Song and K. Lee, "sPAC (Web Services Performance Analysis Center): Performance analysis and estimation tool of web services," Lecture notes in computer science, vol. 3649, 2005, pp. 109.

[6] S. Hwang, et al., "A probabilistic approach to modeling and estimating the QoS of web-services-based workflows," Information Sciences, vol. 177, no. 23, 2007, pp. 5484-5503.

[7] G. Canfora, et al., "A framework for QoS-aware binding and re-binding of composite web services," The Journal of Systems & Software, vol. 81, no. 10, 2008, pp. 1754-1769.

[8] Q. Yu, et al., "Deploying and managing Web services: issues, solutions, and directions," The VLDB Journal, vol. 17, no. 3, 2008, pp. 537-572.

[9] B. Overeinder, et al., "Web service access management for integration with agent systems," ACM New York, NY, USA, 2008, pp. 1854-1860.

[10] F. Raimondi and W. Emmerich, "Efficient online monitoring of web-service SLAs," ACM New York, NY, USA, 2008, pp. 170-180.

[11] M.A. Serhani, et al., "VAQoS: architecture for end-to-end QoS management of value added Web services," International Journal of Intelligent Information Technologies, IGI-Global, vol. 2, no. 4, 2006, pp. 37-56.

[12] M.A. Serhani, et al., "CompQoS: Towards an Architecture for QoS composition and monitoring (validation) of composite Web Services," International Conference on Web Technologies, Application, and Services "WTAS", 2006, pp. 78-83.

[13] S. Kalepu, et al., "Verity: a QoS metric for selecting Web services and providers," Fourth International Conference on Web Information Systems Engineering Workshops, IEEE Computer Society, 2004, pp. 131-139.

[14] V. Tosic, et al., "Web Service Offerings Infrastructure (WSOI) - A management infrastructure for XML Web Services," IEEE Symposium Record on Network Operations and Management Symposium, Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ 08855-1331, United States, 2004, pp. 817-830.

[15] V. Tosic, et al., "Web Service Offerings Infrastructure (WSOI)-a management infrastructure for XML Web services," Network Operations and Management Symposium1, IEEE/IFIP, 2004, pp. 817-830.

[16] A. Shaikh Ali, et al., "UDDIe: an extended registry for Web services," Symposium on Applications and the Internet Workshops (SAINT), IEEE Computer Society, 2003, pp. 85-89.