Online Monitoring for Sustainable Communities of Web Services

Abdelghani Benharref
College of 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*—Web Services are considered an 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 connect Web Services based on their functionalities. In this paper, we augment the community approach by defining a new community to monitor Web Services operating in any Web Services community. This will be of prime importance for communities' creators/managers willing to protect and sustain their communities. This paper defines the overall architecture of the monitoring community and the basic services it offers to its various classes of customers.**

Keywords-component; Web Services communities, monitoring of Web Services communities.

I. INTRODUCTION

The last decade has seen a notable growth of Web Services provision and consumption. 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 [1]. Moreover, Web Services offering related services can form and operate inside a "Web Service Community" (WSC).

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 to help sustaining a community in business.

This paper will present a novel framework that would help managers of communities of Web Services keep track of performance of members of their communities as well as enforcing the community rules and terms. This enforcement promotes sustainability of communities by protecting the community, its reputation, its interest, and those of each individual Web Services operating within the community.

This framework is itself a community of Web Services; each of them is a passive monitor. A passive monitor, also known as passive tester ([5], [6]), checks traces exchanged between a Web Service and its client to detect any functional misbehavior or quality violations. A passive monitor can be online or offline: online monitor gets the traces as soon as they happen and analyses them on the fly. However, offline monitoring gets the traces after the interactions are over between a client and a Web Service. In our framework, we put emphasis on online monitoring as it allows real-time detection of misbehaviors and leads client and providers to take necessary actions as quickly as possible.

During creation or when adding a new Web Service, a community manager might decide to monitor that Web Service during a probation period to make sure it is working properly. The manager can also decide to monitor it while operating within the community for a given period.

It might happen that a client of a member of a community complains that the QoWS received is below what he/she agreed on with the community. The manager of the community can initiate a continuous monitoring of the concerned Web Service to investigate the client's allegations.

Members of a community can request services from each other and can balance the load over each other to keep up with the demand during peak periods or when a request cannot be satisfied by a single Web Service. In this case, it has to be forwarded to an appropriate Web Service, in the community, who can satisfy it. If a member feels that (some) other members of the community do not cooperate with it as they are supposed to do, the member can request monitoring of those members.

Remaining sections of this paper are organized as follows: next section discusses related works on monitoring of Web Services, their strengths and weaknesses. Section III presents the overall architecture of our community along with the main actor's interactions in addition to its business model, terms, and rules. Section IV highlights different services offered by the monitoring community and each of its members. A brief discussion of a proof of concept is presented in section IV.

978-1-4244-9221-3/11/$26.00 ©2011 IEEE

Finally, we conclude the paper and we briefly discuss the ongoing and the future works.

II. Related Work

Web Service Communities are collections of Web Services that are supposed to have similar or related functionalities [7]. Community membership is motivated by the ease and the fastness of identifying and discovering Web Services.

Some works have been done towards addressing the problem of how to discover and quantify Web Services community. For instance in [8], Zhang et al. proposed an approach that forms WSC by grouping closely interactive Web Services which construct Web Service interactive network. They also introduced a technique for composite Web Service discovery.

In [7], Subramanian defines high availability of Web Service Community as the ability of a WSC to continue providing services even when master Web Service fails operationally. The author presents a solution to keep the WSC highly available by using a distributed election algorithm that identifies a temporary master Web Service when there is any operational failure in existing master. In their investigation, Elnaffar et al. [9] have stressed the difference between WSC reputations from Web Service providers' perspective and users' perspective. Therefore they proposed WSC architecture based on reputation and defined metrics to evaluate this latter as it is perceived by clients and providers [9]. In order to assess reputation of communities of Web Services, Khosravifar et al. [10] proposed a mechanism based on run-time feedbacks. In [11], the authors have examined WSC management issues such as arrival of new Web Services, departure of existing Web Services, identification of Web Services to join composite Web Services, and sanctions on Web Services in case of misbehavior.

For monitoring of WSC, which is the main focus of the present paper, the authors in [12] proposed monitoring capabilities within the Manageable and Adaptive Service Composition middleware. In this middleware, they propose to detect business exceptions and runtime faults. They also provide synchronous and asynchronous monitoring both at the SOAP messaging layer and the process orchestration layer. As another aspect of monitoring, Sun et al. [13] stressed the dynamic aspects of web services and their attributes and proposed a run-time monitoring that analyzes the conformance of a Web Service to the original. A similar aspect of Web Service monitoring was proposed by Wang et al. in [14]. The proposed approach uses a pattern of service constraints that correspond to service requirements and a monitoring model that covers five kinds of system events relevant to client request, service response, application, resource, and management.

The main drawback of works presented above is that they suppose that a WSC will implement its own required management operations. That is, each WSC offers its basic functionalities (hotel, flight, weather…) in addition to management functionalities such as monitoring. Requiring a WSC to offer management functionalities that will be used intermittently is a burden and unwise investment with an eventual low return.

Our framework tries to relief managers of WSC from all monitoring-related issues in such a way that those managers can focus on their primary and core business functionalities rather than management aspects. In fact, our monitoring community offers monitoring services to anyone willing to monitor the behavior of a Web Service.

III. Monitoring Community

The core idea in our approach is the observer-based monitoring community of Web Services. Figure 1 illustrates an environment with one monitoring community, a client, and two normal competing communities; inside each of them are competing and complementing Web Services. Two communities/Web Services are said to be competing if they are offering same functionalities in the same market space. Similarly, two communities/Web Services are said to be complementary if they offer complementary non-competitive services.

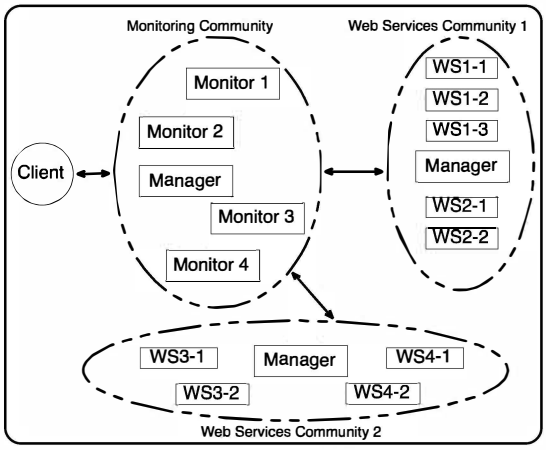

Figure 1. A monitoring Community and Two Normal Communities

All communities, including the monitoring community, are created and maintained by a manager. In Figure 1, community 1 offers two different services (i.e. functionalities): 1) WS1-1, WS1-2, and WS1-3 compete for the first service and 2) WS2-1 and WS2-2 compete for the other service. Community 2, competing with community 1 offers exactly the same services as community 1: WS3-1 and WS3-2 offer the first service, thus, directly competing with WS1-1, WS1-2, and WS1-3. The second service is delivered by WS4-1 and WS4-2, which are direct competitors of WS2-1 and WS2-2. The monitoring community, however, is offering one service, that is monitoring of Web Services. All Web Services in this monitoring community compete in the same market space while cooperating whenever needed and authorized.

When a request is made to the monitoring community, it is received and inspected by the manager. Based on its knowledge on capabilities and resources of each MWS, the manager decides which MWS is likely to handle this request. When a MWS receives a request from the manager or from a peer, it has four options: 1) accept the request and fully take care of it if it can process it, 2) accept the request and

collaborate with a peer if it can not serve it alone, 3) entirely forward the request to a peer if it can not process it neither cooperate due to a high load, or 4) reject the request if it can not process it neither finds a peer to which the request can be delegated.

Figure 2 illustrates a step-by-step scenario using the monitoring community. A client makes a request to the manager (step 1). The manager then selects monitor 2 to take care of the request (step 2). Being little bit overloaded, monitor 2 decides to ask for collaboration from monitor 3 (step 3). Monitor 3 is handling other requests and cannot collaborate; so, it delegates the request to monitor 1 (step 4). In step 5, monitor 5 confirms its will to collaborate with monitor 2 for the sake of this request. Only at this stage, monitor 2 can officially inform the manager that it will satisfy this request (step 6). The manager notifies the client (step 7) so that the later can forward all exchanged traces with the Web Service to be monitored (i.e. the WSUO) to monitor 2 (step 8). All these traces are forwarded to monitor 1 (step 9). The monitoring process culminates at the end of the monitoring process or when a fault is detected. Monitor 1 reports to monitor 2 (step 10), which notifies the manager (step 11). The manager then sends a monitoring report to the client (step 12).

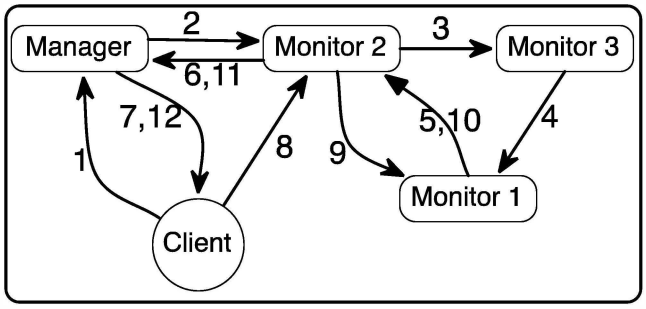

Figure 2. Monitoring Community Utilization Procedure

The manager of the community and all participating MWS provide a set of services to their customers. Each MWS offers two basic services to its peers in the same monitoring community namely: collaboration and delegation. The first one is required so peers can request help from a MWS during the monitoring while the second is required to forward a request to another MWS.

All interactions of the monitoring community to/from clients go through the manager except forwarding traces, which goes directly from a client of the monitoring community to the appropriate MWS. The manager accepts requests for monitoring and returns back (later) the monitoring result/verdict.

Providers of Web Services can use the monitoring community to certify the QoWS of their Web Services. Such certification gives them more credibility and/or a competitive advantage in front of their competitors as some clients might request their service providers to be certified. Certification can also be requested after a client submits a complaint that the received QoWS from a Web Service is below what has been agreed upon with the provider.

Web Service can request services from the monitoring community to check interactions with a Web Service. This request can serve two complimentary purposes: 1) check if the paid for QoWS is delivered and 2) have a valid and supporting document of QoWS violations, to be used as part of an eventual complaint.

The monitoring community can be of great help to communities' managers willing to build solid communities of high quality Web Services. In fact, a manager can ask the monitoring community to provide a comprehensive report on the sustainability of a new community composed of a list of Web

For a community already in business, its manager can make use of the monitoring community during probation periods, after complaints about a member Web Service, certification of a Web Service, or to check and/or certify an upgrade of the QoWS of the community.

For a monitor to be able to detect functional and non-functional misbehaviors, the behavior and QoWS should be unambiguously described. Our framework uses an XML-based description of behavior as illustrated in Figure 3. The behavior is described by an Extended Finite State Machine (EFSM) augmented with QoWS annotations. Each transition, i.e. operation, might have one or more profiles, i.e. classes of QoWS.

```
<efsm name="Name of EFSM/Web Service">
 <state name="State1" initial="YES">
  <transition ID="t1" input="Input1" predicate="true"
    assignments="x:=0;y:=0;z:=0" output="Output1"
    next="State2"/>
    <Profile name="GOLD"> MinRT = NULL MaxRT = 10ms
       availability= "95%"
    </Profile>
  </transition>
  <transition ID="t2" input="Input2" predicate="X<3"
    assignments="x:=2;y:=7" output="Output2" next="State3"/>
    <Profile name="SILVER"> MinRT = 10ms MaxRT = 30ms
      Availability= "75%"
    </Profile>
  </transition>
 </state>
</efsm>
```

Figure 3. EFSM specification as an XML representation

IV. PROOF OF CONCEPT

For the purpose of proving the feasibility of our proposed approach, we have developed a case study in which we validate monitoring features of both functional and non-functional requirements of Web Services belonging to communities. As shown in Figure 1, we have created two communities of Web services. The first one is a community of flight reservation Web Services and the second is a community of Web Services for hotel reservation. Each community is managed by a manager and consists of a number of Web Services providing related functionalities. Hotel reservation and flight reservation use almost the same parameters. They both rely on dates, and provide different types of services. For instance, we find standard room, double

room, and suite for hotel reservation and economic, business, and first class for flight reservation.

Using this case study, we have conducted a series of experiments in order to evaluate QoWS-aware Web Service monitoring schemes supported by our architecture. Scenarios in which all components of the architecture are involved have been considered.

Initial experimentations show very promising results. In fact, few scenarios are executed to handle monitoring of Web service client, providers, and other monitors (peers) within these communities. Three monitoring schemes of the monitor were evaluated and applied. Hence, three scenarios have been conducted to evaluate both functional and non-functional behavior of Web Services within communities. In these scenarios, the non-functional properties include mainly Web Service Response Time (RT) and Availability. The monitoring community was able to detect all misbehaviors.

V. CONCLUSION

High quality communities of Web Services improve visibility and reputations of participating Web Services, and high quality Web Services make their communities of high reputation. Such sustainable communities would like to make sure their high-reputation is being protected and none of their members is damaging it. To make sure all members are adhering to various rules and terms while operating inside the community, community managers can monitor members in specific circumstances.

This paper proposed a new framework that levitates all monitoring-related activities off the shoulders of communities' managers. In fact, the monitoring framework offers a community of Web Services, which are passive observers capable of observing any Web Service operating in any community. We presented and discussed different components of the framework and developed a proof of concepts and used it in observing Web Services members of communities during probation, after complaints, and periodical check out.

As future work, we are extending the framework to cover more QoWS properties and looking into methods for traces collection to reduce the monitoring overhead.

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," 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] M. Diaz, et al., "Observer-a concept for formal on-line validation of distributed systems," IEEE Transactions on Software Engineering, vol. 20, no. 12, 1994, pp. 900-913.

[6] B.T. Ladani, et al., "Passive testing - a constrained invariant checking approach," 17th International Conference on Testing of communicating systems (TestCom), Lecture Notes in Computer Science (LNCS), Springer Verlag, 2005, pp. 9-22.

[7] S. Subramanian, "Highly-available web service community," 6th International Conference on Information Technology: New Generations, ITNG 2009, April 27, 2009 - April 29, 2009, ITNG 2009 - 6th International Conference on Information Technology: New Generations, IEEE Computer Society, 2009, pp. 296-301.

[8] Z. Xizhe, et al., "A composite web services discovery technique based on community mining," 2009 IEEE Asia-Pacific Services Computing Conference (APSCC 2009), 7-11 Dec. 2009, 2009 IEEE Asia-Pacific Services Computing Conference (APSCC 2009), IEEE, 2009, pp. 445-450.

[9] S. Elnaffar, et al., "Reputation of communities of Web services - Preliminary investigation," 22nd International Conference on Advanced Information Networking and Applications Workshops/Symposia, AINA 2008, March 25, 2008 - March 28, 2008, Proceedings - International Conference on Advanced Information Networking and Applications, AINA, Institute of Electrical and Electronics Engineers Inc., 2008, pp. 1603-1608.

[10] B. Khosravifar, et al., "An approach to incentive-based reputation for communities of Web services," 2009 IEEE International Conference on Web Services (ICWS), 6-10 July 2009, 2009 IEEE International Conference on Web Services (ICWS), IEEE, 2009, pp. 303-310.

[11] Z. Maamar, et al., "Sustaining Web services high-availability using communities," 2008 3rd International Conference on Availability, Reliability and Security (ARES '08), 4-7 March 2008, 2008 3rd International Conference on Availability, Reliability and Security (ARES '08), IEEE, 2008, pp. 834-841.

[12] A. Chourmouziadis and G. Pavlou, "Web services monitoring: An initial case study on the tools perspective," NOMS 2008 - IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services, April 7, 2008 - April 11, 2008, NOMS 2008 - IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services, Inst. of Elec. and Elec. Eng. Computer Society, 2008, pp. 827-830.

[13] S. Mingjie, et al., "Monitoring BPEL-based Web service composition using AOP," 2009 8th IEEE/ACIS International Conference on Computer and Information Science (ICIS), 1-3 June 2009, 2009 8th IEEE/ACIS International Conference on Computer and Information Science (ICIS), IEEE, 2009, pp. 1172-1177.

[14] W. Qianxiang, et al., "An online monitoring approach for Web service requirements," IEEE Transactions on Services Computing, vol. 2, no. Copyright 2010, The Institution of Engineering and Technology, 2009, pp. 338-351.