QoS Synchronization of Web Services: A Multi Agent-Based Model

Jaber Kouki[1], Walid Chainbi[2], and Khaled Ghedira[3]

[1] High Institute of Human Sciences of Tunis/SOIE, Tunisia
jaber.kouki@hotmail.com
[2] Sousse National School of Engineers/SOIE, Tunisia
Walid.Chainbi@gmail.com
[3] Higher Management Institute of Tunis/SOIE, Tunisia
Khaled.Ghedira@isg.rnu.tn

Abstract. From the last decade, Web services technology has witnessed a great adoption rate as a new paradigm of communication and interoperability between different software systems. This fact, has led to the emergence of Web services and to their proliferation from outside the boundary of the UDDI business registry to other potential service resources such as public and private service registries, service portals, and so on. The main challenge that arises from this situation is the fact that for the same service implementation, several service descriptions are published in different service registries. Accordingly, if the service implementation is updated all of its descriptions have to be updated too over all of these registries. Otherwise, the service user may not bind to the suitable Web service if its descriptions are inaccurate or outdated. To address the above challenge, we propose in this paper a multi agent-based model that focuses on synchronizing the description of Web services, especially their quality of service, to maintain their consistency and sustainable use.

1 Introduction

To support interoperable machine-to-machine interaction over a network, Web services are emerged as a promising technology that delivers application functionalities as services which are language and platform independent. One of the major building blocks of Web services technology is the UDDI (Universal Description Discovery and Integration) [1] business registry (UBR) where service providers and service consumers (or users) publish and discover Web services respectively.

To use the UBR service consumers can only provide functional descriptions of Web services they need (e.g. keywords, inputs, outputs, etc.), however to select the best suitable Web service nonfunctional descriptions (e.g. QoS) are required but, they are missing in UBR.

Focusing on this limit, other service resources start to be deployed such as public business registries (PBRs), and service portals (e.g. Xmethods.net, webservicelist.com, webservicesx.net, etc). Nevertheless, the main issue with these resources is the fact that they do not adhere with Web service standards (e.g. UDDI) and therefore to be potential service resources like UBR does. Furthermore, as Web services substantially increase in number all over the Web, the number of service

DOI: 10.1007/978-3-319-00551-5_49

ressources is also increased. Accordingly, service users will not be able to browse all of them separately to select Web services of interest.

To cope with these facts, we have proposed in a previous work [2] a local repository-based framework (LRBF) which provides service users with local Web service repositories (LWSRs) to collect service information including binding details and QoS descriptions. This information is used to bind locally to Web services that interest service users without the need to browse heterogeneous service resources independently.

However, since service providers may change the implementation of their published Web services, the service information collected in LWSRs may become inaccurate and outdated. Due to this change, Web services that are locally bound may no more interest service users. To deal with this fact, we extend the LRBF framework with a multi agent-based model to synchronize service information (especially QoS) of the updated Web services over different service resources, and therefore to ensure that the locally bound Web services are always the best suitable ones according to the service users need.

The remainder of this paper is organized as follows. Section 2 outlines the background material of the QoS synchronization issue within the LRBF framework. The extension to this framework with a multi agent-based model for QoS synchronization is discussed in Sect. 3.The implementation issues are presented in Sect. 4. Prior to the conclusion and future work in Sect. 6, related work are reviewed in Sect. 5.

2 QoS Synchronization Background

Considering the need to optimize the binding to relevant Web services from heterogeneous environments, the LRBF framework provides service users with a local access point to bind to Web services of interest without having to browse separately hundreds if not thousands of different service resources. The architecture of this framework involves three levels of service repositories where two levels of binding optimization are performed.

The first level of optimization, which is public, is performed by a public Web service crawler engine (PWSCE) that collects service information from the first level of service repositories that encompasses UBRs, PBRs and service portals into the second level which is represented by a public Web service repository (PWSR). Having collected service information of all Web services in this repository, is very practical for service users to see Web service candidates from different service resources grouped under one roof, and then to select easily the best suitable one. However, what is impractical in this level of optimization is the fact that service users have to access the PWSR each time they need to re-bind to the same Web services which may be a useless work and a waste of time. In fact, as the PWSR is hosted on the administrator server host and as it contains a huge number of service information, to respond to an excessive amount of service requests may take a considerable amount of time.

To deal with these facts, a local Web service crawler engine (LWSCE) performs the second level of binding optimization which is local. In this level of optimization, the LWSCE collects service information of the best suitable Web services from the PWSR into the third level of service repositories which is represented by a local Web service repository (LWSR). In this repository, service users can find locally (within their local hosts) service information required to bind to Web services of interest without using the PWSR.

However, having collected service information in more than one repository within the LRBF framework, means that such information is duplicated over the involved service repositories. According to this fact, service information of Web services and especially their QoS descriptions need to be synchronized for two reasons. The first one is to ensure that the different levels of service repositories, involved within the LRBF framework, see all the changes that have been originated by the service providers, and the second one is to ensure that the locally bound Web services still always fit the service user interest. To meet these two objectives, we introduce in the next section a multi agent-based model for dynamic synchronization of Web service QoS descriptions.

3 QoS Synchronization Proposed Model

The need to agent paradigm in our proposed synchronization model is basically argued by the need of both the autonomy and pro-activeness properties of software agents to synchronize dynamically QoS descriptions of Web services over service repositories with the minimum of human intervention (either by service providers or service users).

As defined by Wooldridge, software agent is “a computer system that is situated in some environment, and that is capable of autonomous actions in this environment in order to meet its delegated objectives” [3]. In our proposed model, the actions to carry out by software agents within LRBF are first to synchronize QoS descriptions of Web services over the involved service repositories once new updates are delivered by service providers and then to ensure that, upon each update in the service information, the locally bound Web services always rank better than their candidates.

From the above considerations and given the infrastructure of LRBF framework which encompasses three levels of service repositories, the architecture of the proposed synchronization model (see Fig. 1) involves three software agents including PISA (Public Information Synchronization Agent), UISA (Universal Information Synchronization Agent), and LISA (Local Information Synchronization Agent). Each class of software agents operates at one level of service repositories.

As depicted in Fig. 1, the processing of QoS synchronization module within LRBF starts when new updates in the service QoS description are delivered by the service provider to the PISA agent that operates at the first level of service repositories which includes UBRs, PBRs and service portals. To interact with this agent, the service provider uses a GUI by which he can provide new updates in the QoS description (e.g. response time) of a particular Web service. Upon receiving these updates, the PISA agent saves them first in the repository (e.g. PBR) of the service provider, and then it sends to the UISA agent a message incorporating the updated QoS description.

As soon as the UISA agent receives this message from the PISA agent, it saves first the included QoS description in the PWSR, the second level of service repositories on which this agent operates, and then it starts listening for new update requests from the LISA agent. When new requests are received, the UISA agent replies the requestor agent with a message incorporating the updated QoS description received from the PISA agent. Upon receiving this message and before saving the included data in the LWSR (the third level of service repositories), the LISA agent checks first whether the locally bound Web service with its new QoS description still match better the predefined requirements of the service user or not. Accordingly, if this Web service always ranks better than its candidates, the LISA agent will save its new QoS description in the LWSR. Otherwise, this agent selects from the list of Web service candidates the service that best fits requirements of the service user and then it collects the service information of the new selected Web service from the PWSR into the LWSR to replace the existing one.

To conduct a reliable synchronization of QoS information over the different levels of service repositories, software agents have to carry out suitable behaviors. According to the Fipa-compliant agent platform, JADE (Java Agent Development Framework), software agents can carry out One-shot behavior that is executed only once and then completes immediately, Cyclic behavior that is executed forever, and Generic behavior that is executed when a given condition is met [4]. Besides, further behaviors may be executed at given points in time such as Waker behavior that is

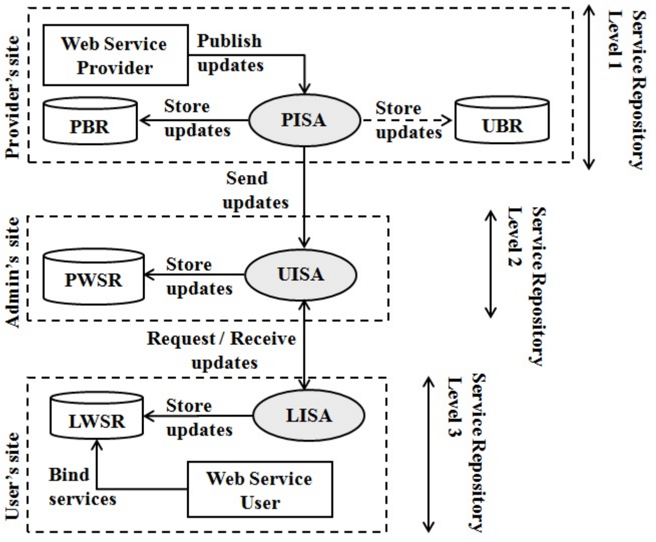


Fig. 1 Architecture of the QoS synchronization model

executed only once just after a given timeout is elapsed and Ticker behavior that is executed periodically waiting a given period after each execution.

According to the tasks to carry out by each agent within the proposed synchronization model and given the types of agent behaviors described above, software agents involved in this model have to carry out the following behaviors:

- PISA behaviors: the PISA agent executes one-shot behavior for each updates reception from the service provider and another one-shot behavior for each updates delivery to the UISA agent.
- UISA behaviors: the UISA agent executes two cyclic behaviors: one dedicated to serve requests for updates reception from the PISA agent and the other dedicated to serve requests for updates delivery to the LISA agent.
- LISA behaviors: the LISA agent carries out a ticker behavior that deals with the update requests to the UISA agent and one-shot behavior that deals with the updates reception from this agent. The last behavior is executed only if the updated QoS descriptions belong to the locally bound Web services. Upon updating these services or their potential candidates the LISA agent carries out another ticker behavior to select periodically the new suitable Web service that fits better the service user needs.

4 Implementation

The QoS synchronization model is implemented using JADE framework to develop software agents involved in this model such as PISA, UISA and LISA. Furthermore, we have integrated these agents in ATAC4WS[1] (Agent Technology and Autonomic Computing for Web Services) which implements the LRBF framework that we have developed to create the three levels of service repositories on which software agents synchronize QoS information of Web services. Upon running ATAC4WS, both PWSCE and LWSCE (the two Web service crawler engines) start collecting service information from UBRs, PBRs and other potential service resources into the PWSR for all Web services and then from PWSR into LWSR for only Web services that interest service users. To launch the processing of the QoS synchronization module within LRBF, service providers use a GUI to deliver to the PISA agent new updates in the QoS description (e.g. cost) of a particular Web service.

Upon receiving these updates, the PISA agent incorporates them first into the repository (e.g. PBR) of the service provider within the first level of service repositories and then it sends them to the UISA agent. To synchronize the updates in QoS description over the second and third levels of service repositories, the UISA and LISA agents exchange a message incorporating the new Qos description to be included in the PWSR and LWSR respectively.

Finally, upon each updates reception in QoS description, the LISA agent computes the QoS ranking of the locally bound Web services compared to their candidates according to the QoS preferences of the service user. The focus of this computation is to select Web services that interest more the service user and therefore to

[1] ATAC4WS provides an environment to create, register, discover and use Web services.

collect their service information in the LWSR where to be bound locally in subsequent use.

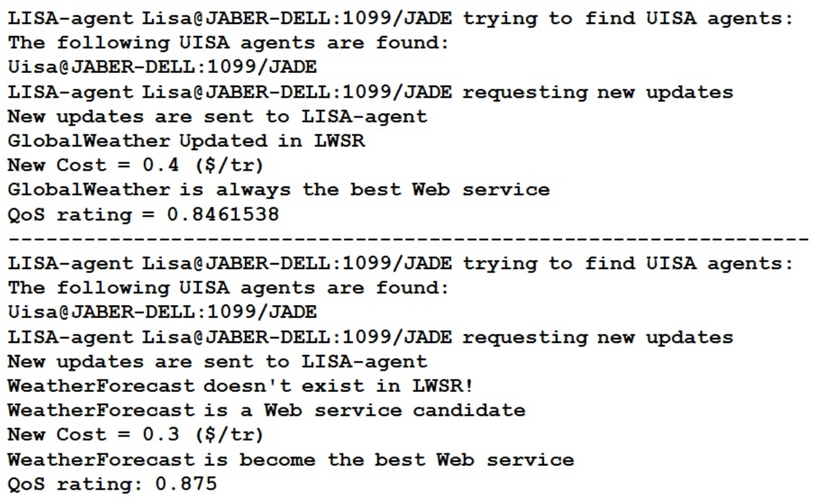
Details about the running of LISA agent in response to the update of the cost attribute of two Web services (e.g. GlobalWeather and WeatherForecast) are shown in the NetBeans IDE outputs of Fig. 2. These Web services are used to get up-to-date weather conditions for all major cities around the world.

In the first output (see the top of Fig. 2), although the cost of GlobalWeather (a locally bound web service) is raised to 0.4 $, this service still always the best suitable one assuming that the service user is only interested in the cheapest Web service. However, in the second output (see the buttom of Fig. 2), upon decreasing the cost of WeatherForecast (a Web service candidate) to 0.3 $, this service becomes cheaper than GlobalWeather and in this case the LISA agent will collect its service information from the PWSR into the LWSR to replace the existing service (GlobalWeather).

Note that, before each request of new QoS updates, the LISA agent tries to find new UISA agents since they may dynamically appear and disappear in the system.

5 Related Work

As Web services start to expand across the Internet, most of them are still inaccessible or have duplicated descriptions that do not match the new implementation of the original Web services. Therefore, several works focused on addressing QoS issues to guarantee the reliability of published Web services while others are interested in addressing synchronization concern of Web services in different aspects.



```
LISA-agent Lisa@JABER-DELL:1099/JADE trying to find UISA agents:
The following UISA agents are found:
Uisa@JABER-DELL:1099/JADE
LISA-agent Lisa@JABER-DELL:1099/JADE requesting new updates
New updates are sent to LISA-agent
GlobalWeather Updated in LWSR
New Cost = 0.4 ($/tr)
GlobalWeather is always the best Web service
QoS rating = 0.8461538
-----------------------------------------------------------------
LISA-agent Lisa@JABER-DELL:1099/JADE trying to find UISA agents:
The following UISA agents are found:
Uisa@JABER-DELL:1099/JADE
LISA-agent Lisa@JABER-DELL:1099/JADE requesting new updates
New updates are sent to LISA-agent
WeatherForecast doesn't exist in LWSR!
WeatherForecast is a Web service candidate
New Cost = 0.3 ($/tr)
WeatherForecast is become the best Web service
QoS rating: 0.875
```

Fig. 2 LISA running outputs

To address the QoS issue of Web services, many researchers have extended the service oriented architecture (SOA) [5] to incorporate nonfunctional properties of Web services including QoS. For example, in [6] the author proposed a regulated UDDI model by adding a new data structure type where to represent QoS information of Web services such as availability, reliability, etc.

The authors in [7] did not extend the UDDI model but they used the existing data structure type tModels. In tModels, QoS are represented as a KeyedReference element where the KeyName attribute contains the QoS name and the KeyValue contains the QoS value. In the same effort, Hollunder in [8] and the authors in [9] used WS-Policy [10] and OWL-S [11] respectively instead of tModels to advertise QoS information of Web services.

However, since service providers may change the QoS description of their published Web services, it is important to synchronize the new description over different services ressources. Although it is hard to find works addressing the synchronization issue of QoS description, some other works focused on the validity of service information in service registries in general.

For example, the authors in [12] extended WS-Policy as WS-TemporalPolicy to describe service properties with time constraints. That is, to define the validity of a policy or its included service properties a set of time attributes are used such as startTime, endTime and expires.

The authors in [13] extended UDDI as UDDIe where Web services hold a lease which defines how long the service information should remain registered in a registry. If a lease expires, the service provider should renew it otherwise the registered information will be deregistered.

Finally, the authors in [14] proposed an agent-based model for dynamic synchronization of Web services. However the synchronization aspects of this model focus only on service binding details.

6 Conclusion

In this work, we have proposed a multi agent-based model for QoS synchronization of Web services which is integrated within LRBF framework to synchronize the QoS description of updated Web services over different levels of service repositories (PBRs, PWSR, and LWSR respectively). This model has been implemented using the JADE platform.

In Future work, we envision to deal with the trustworthy of the delivered QoS information since the service provider may greatly influence how QoS metrics are generated to obtain the most suitable results, and therefore may provide inaccurate information about QoS.

References

1. Clement, L., Hately, A., Riegen, V.C., Rogers, T.: Universal description, discovery, and integration (UDDI 3.0.2). Technical committee draft, OASIS (2004), `http://uddi.org/pubs/uddi_v3.htm/`

2. Kouki, J., Chainbi, W., Ghedira, K.: Binding optimization of web services: a quantitative study of local repository-based approach. In: ICWS 2009: Proceedings of the IEEE International Conference on Web Services, pp. 646–647. IEEE computer society (2012)
3. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons Ltd., UK (2009)
4. Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: Jade programmers guide. Technical report, TILab (2010), `http://jade.tilab.com/doc/programmersguide.pdf/`
5. Krafzig, D., Banke, K., Slam, D.: Enterprise soa: service-oriented architecture best practices. Prentice-Hall Inc., New Jersey (2005)
6. Ran, S.: A model for web services discovery with qos. ACM SIGecom Exchanges 4(1), 1–10 (2003)
7. Rajendran, T., Balasubramanie, P.: An optimal agent-based architecture for dynamic web service discovery with qos. In: ICCCNT 2010: International Conference on Computing Communication and Networking Technologies, pp. 1–7 (2010)
8. Hollunder, B.: Ws-policy: on conditional and custom assertions. In: ICWS 2009: Proceedings of the IEEE International Conference on Web Services, pp. 936–943. IEEE Computer Society (2009)
9. Lakhal, R.B., Chainbi, W.: A multi-criteria approach for web service discovery. In: MobiWIS 2012: Proceedings of the 9th International Conference on Mobile Web Information Systems, vol. 10, pp. 609–616. Elsevier PCS (2012)
10. Vedamuthu, A., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T., Yalcinalp, U.: Web services policy (1.5). W3C recommendation, W3C (2007), `http://www.w3.org/TR/ws-policy/`
11. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, C.: Owl-s: semantic markup for web services. W3C member submission, W3C (2004), `http://www.w3.org/Submission/OWL-S/`
12. Mathes, M., Heinzl, S., Freisleben, B.: Ws-temporalpolicy: a ws-policy extension for describing service properties with time constraints. In: COMPSAC 2008: Proceedings of the 32nd Annual IEEE International Computer Software and Applications, pp. 1180–1186 (2008)
13. ShaikhAli, A., Rana, O.F., Al-Ali, R., Walker, D.W.: UDDIe: an extended registry for web services. In: SAINT-W 2003: Proceedings of the 2003 Symposium on Applications and the Internet Workshops, pp. 85–89. IEEE Computer Society (2003)
14. Kouki, J., Chainbi, W., Ghedira, K.: An agent-based approach for binding synchronization of web services. In: AWS 2012: Proceedings of the 1st International Workshop on the Adaptation of Web Services, pp. 921–926. Elsevier PCS (2012)