A QoS-aware Web Services Selection for Reliable Web Service Composition

Aziz Nasridinov, Byun Jeongyong
Department of Computer Engineering, Dongguk University
aziz_nasridinov@yahoo.com, byunjy@dongguk.ac.kr

Abstract

Web Services have been utilized in a wide variety of applications and have turned into a key technology in developing business operations on the Web. Originally, Web Services can be exploited in an isolated form, however when no single Web Service can satisfy the functionality required by a user, there should be a possibility to compose existing services together in order to fulfill the user requirement. However, since the same service may be offered by different providers with different non-functional Quality of Service (QoS), the task of service selection for Web Service composition is becoming complicated. Also, as Web Services are inherently unreliable, how to deliver reliable Web Services composition over unreliable Web Services should be considered while composing Web Services. In this paper, we propose an approach on a QoS-aware Web Service selection for reliable Web Service composition. In our approach, we select and classify Web Services using Decision Tree based on QoS attributes provided by the client. Service classifier will improve selection of relevant Web Services early in the composition process and also provide flexibility to replace a failed Web Services with a redundant alternative Web Services, resulting in high availability and reliability of Web Service composition. We will provide an implementation of our proposed approach along with efficiency measurements through performance evaluation.

1. INTRODUCTION

Web Services are considered to be a promising technology to add a new level of functionality to the existing World Wide Web and change the way we find and share data in order to fulfill sophisticated business demands. Usually Web Services can be exploited in an isolated form, however when no single Web Service can suit the functionality required by a user, there should be a possibility to compose existing services together in order to fulfill the user requirement [1]. For example, a summer vacation business process requires the collaboration of at least four Web Services: flight reservation, hotel booking, attraction searching, and user notification. It's obvious, in the nonexistence of composition of Web Services, the user spends significant amount of time visiting many sites, determining appropriate service providers, entering his preferences repeatedly, integrating or lining up the different type of results coming from different sites.

The same service may be offered by different providers with different non-functional or QoS, so the task of service selection for Web Service composition is becoming complicated. It is important to provide service consumers with facilities for selecting optimal Web Services according to their non-functional characteristics or QoS. Also, as services are deployed on the unreliable Internet, and as they are often long running, loosely coupled, and cross administrative boundaries, failures are expected to happen frequently during the execution of composite services [2]. Therefore, how to deliver reliable service composition over unreliable services is a challenging problem. In service composition, there is a lack of the flexibility to replace a failed Web Service with a redundant alternative. If one of these services is not available or fails during execution, a dispatcher component needs to be able to dynamically switch to alternative Web Services that provide equivalent functionality in order to fulfill the consumer request.

With the whole consideration of the above two issues, we propose an approach on a QoS-aware Web Service selection for reliable Web Service composition. The ultimate goal of our approach is to dynamically and incrementally select a Web Service for executing each incoming operation on the composite Web Service so as to maximize the likelihood of successful execution. In order to achieve that we propose to select and classify Web Services using Decision Tree algorithm based on QoS attributes provided by the client. Service classifier will improve selection of relevant Web Services early in the composition process and also provide flexibility to replace a failed Web Services with a redundant alternative Web Services, resulting in high availability and reliability of Web Service composition. We will provide an implementation of our proposed approach along with efficiency measurements through performance evaluation.

The rest of the paper is proceeds as follows. In Chapter 2, we present related works. In Chapter 3 we describe our proposed approach. Chapter 4 shows our implementation. Chapter 5 describes performance evaluation and Chapter 6 highlights conclusion and future work.

2. RELATED STUDIES

The Web Service selection problem has been extensively studied in the past few years. Different service selection methods that have been designed from different perspective have been proposed. Reputation based methods are one of them and are used as a measure for narrowing down the service selection. In [3], authors proposed a model of reputation-enhances QoS-based Web Services discovery that combines an augmented UDDI registry to publish the QoS information and a reputation manager to assign reputation scores to the services based on customer feedback of their performance. In [4], authors present a technique to calculate a reputation score per service using centrality measure of Social Networks. They later use this score to produce composition solutions that consists of services provided by reputed providers. However, these reputation mechanisms are simple and not robust against various cheating behaviors, such as collusion among providers trying to boost the quality of their own services and badmouth about the other ones.

Other approaches mainly focused on QoS-based Web Service selection for composition. In [5], to solve QoS-aware service selection problem in Web Services composition, authors proposed a global optimization selection mechanism based on prediction of local services' QoS values. In [6], authors propose a QoS-aware service selection model based on fuzzy linear programming techniques, in order to identify their dissimilarity on service alternatives, assist service consumers in selecting most suitable services with consideration of their expectations and preferences. However, as a composition is composed by different Web Services invocations, when one component service fails, the execution of whole process will greatly influence. Therefore, how to deliver reliable service composition over unreliable services should be considered.

Several works have been proposed in the literature to derive the reliability of a service composition. In [7], authors proposed a mechanism that allows programmers to easily develop fault-tolerant compositions using diverse Web Services. The mechanism allows programmers to specify alternative Web Services for each operation and offers a set of artifacts that simplify the coding process. In [2], authors present FACTS, a framework that can address the aforementioned problems for fault-tolerant composition of transactional Web service. The ultimate objective of FACTS is an integrated environment for specification, verification, and execution of fault-tolerant composite services.

3. SYSTEM DESIGN

In the last chapter we saw different service selection methods for Web Service composition that have been designed from different perspective. Taking the shortcomings of all the discussed solutions into consideration, in this section, we will describe our approach on a QoS-aware Web Service selection for reliable Web Service composition.

The ultimate goal of our approach is to dynamically and incrementally select a Web Service for executing each incoming operation on the composite Web Service so as to maximize the likelihood of successful execution. To achieve this, our approach is divided into three main modules. These modules and the relationship between them are shown in Figure 1. Each part is designed and implemented as a separate module. Detailed descriptions of each module are given in the subsequent sections. These modules and their functions are as follows:

- *The Initialization Module:* This module performs selection of Web Services as well as replication management.
- *The Diagnostics Module:* The module is responsible for fault detection and fault notification.
- *Recovery Module:* In this module, the invocation activities are logged to a reliable storage for the future recovery process.

3.1 The Initialization Module

As the first step to compose Web Services is to choose the appropriate atom services, the FT-ADMIN retrieves candidate Web Services interfaces from UDDI, and then performs service selection for Web Service composition. In our approach, we used Decision Tree for selection and classifying QoS attributes. We can summarize the general approach, as follows:

1. We can choose an attribute that best differentiates the output attribute values.
2. Create a separate tree branch for each value of the chosen attribute.
3. Divide the instances into subgroups so as to reflect the attribute values of the chosen node. For each subgroup, terminate the attribute selection process if:
   A. All members of a subgroup have the same value for the output attribute, terminate the attribute selection process for the current path and label the branch on the current path with the specified value.
   B. The subgroup contains a single node or no further distinguishing attributes can be determined.
4. For each subgroup created in third step that has not been labeled as terminal, repeat the above process.

The algorithm is applied to the training data. We observe that learning algorithms, including our new tree classifier, generally improve accuracy after the attribute selection. Thus, we believe our service classifier will improve selection of relevant Web Services early in the composition process.

Replication management is also performed in initialization module. When Fault Notifier informs a failure to the replication management, according to our ranking made by Decision tree, it selects one of the backup services instead of failed service. Thus, flexibility is provided to replace a failed Web Services with a redundant alternative.

3.2 The Diagnostics Module

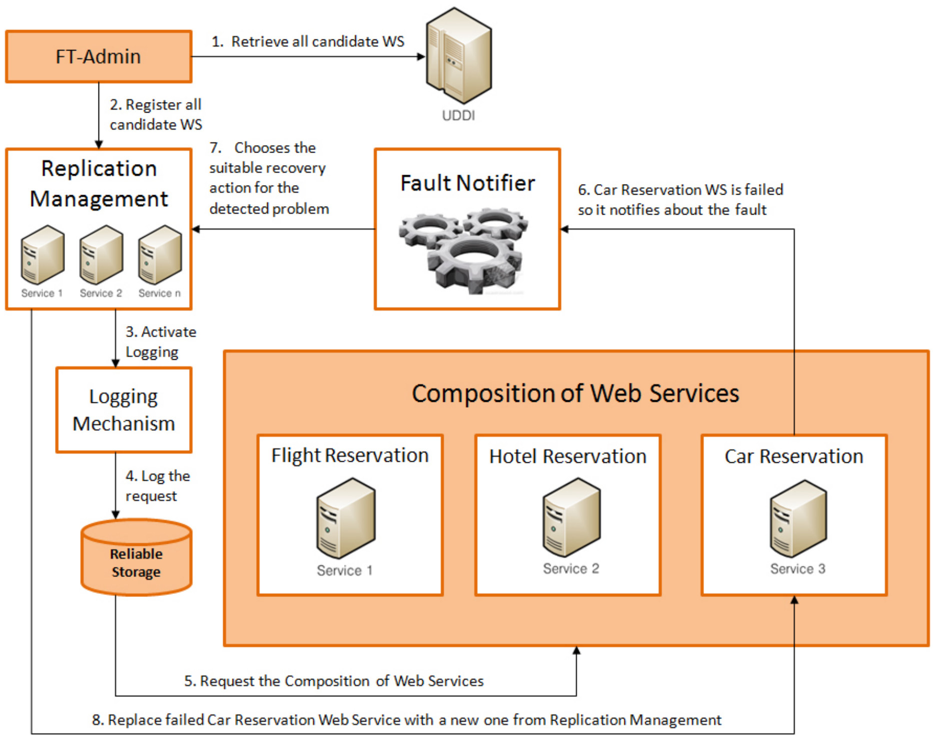
The module is responsible for fault detection and fault notification. Based on [2], we identified five types of Web Services faults in the service composition process.

- *Logical faults* are deliberately thrown by external Web Services as they cannot complete successfully due to various reasons.
- *System faults* are raised by the supporting execution environment.
- *Content faults* refer to the corrupt service results.
- *Service Level Agreement (SLA)* faults are raised when Web Services complete the functional requirements but violate the predefined SLA.
- *Security faults* can happen when integrity or confidentiality of some sensitive data, such as credit card number or identification number, is attacked by malicious hosts [8].

The Fault Detector diagnoses the services and sends fault reports to the Fault Notifier when it detects a service failure on the primarily selected services. The Fault Notifier then notifies Replication Management to start the recovery process.

3.3 The Logging/Recovery Module

The logging mechanism logs the invocation activities to a centralized reliable logging file system for the future recovery process. The logging mechanism has two main


FT-Admin
1. Retrieve all candidate WS
UDDI
2. Register all candidate WS
Replication Management
Service 1
Service 2
Service n
7. Chooses the suitable recovery action for the detected problem
Fault Notifier
6. Car Reservation WS is failed so it notifies about the fault
3. Activate Logging
Logging Mechanism
4. Log the request
Reliable Storage
Composition of Web Services
Flight Reservation
Service 1
Hotel Reservation
Service 2
Car Reservation
Service 3
5. Request the Composition of Web Services
8. Replace failed Car Reservation Web Service with a new one from Replication Management

(Figure 1) Proposed approach

functions. One function is to intercept the message and log it in a reliable storage for the future recover process. The other function is to checkpoint critical states periodically to backups. When the recovery mechanism on new primary member is activated, the recover mechanism retrieves the invocation logs and replays the invocations if necessary.

4. SYSTEM IMPLEMENTATION

The prototype has been developed using the NetBeans 6.5 IDE environment using Java. Web Service Composition is created in the Netbeans BPEL Designer as a BPEL module. In order to present the most suitable service to the service requester, we used the Quality of Service attributes. QoS is a combination of several qualities or properties of a service, such as: Availability, Reliability, Price, Throughput, Response Time, Latency, Performance, Security, Regulatory, Accessibility, Robustness/Flexibility, Accuracy, Servability, Integrity and Reputation. QoS parameters determine the performances of the Web Services and find out which Web Services are best and meet user's requirements.

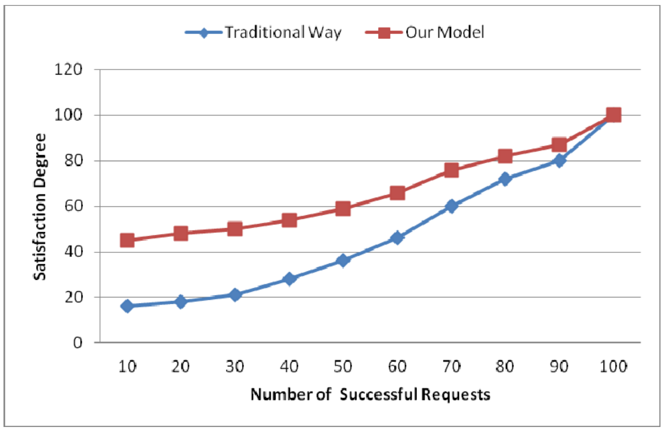
We classify Web Services using Decision Tree into three classes, such as High, Low and Average. Using this method we obtained a classifier tree shown in Figure 3. In this tree, nodes belong to the QoS attributes. Tracing these nodes, we could reach leave nodes, which represent the classification of the Web Service. This classifier consists of a committee of cascading decision trees. Each tree is constructed by using one of the top-ranked features as its root node.

5. PERFORMANCE EVALUATION

Performance evaluation was done by comparing two cases: when using our selection model and when not using it. Conventionally, provider Web Services are selected among candidate provider Web Services based on different criteria such as reputation and so on. In our model, provider Web Services is selected using Decision Tree based on QoS attributes provided by clients. We have applied both conventional way and our model for the case study and identified satisfaction of client Web Services by following satisfaction degree equation.

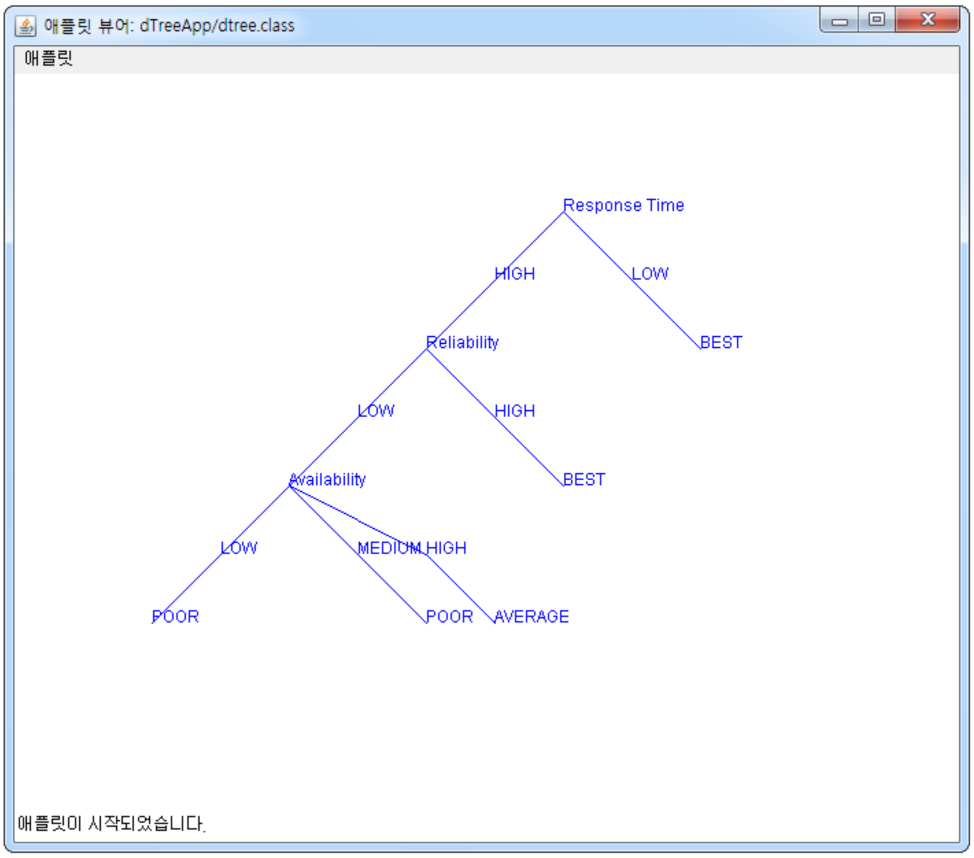
$$Satisfaction\ Degree\ (SD(n)) = n - \sum_{i=1}^{n} \frac{\left|QoS_{required}^{i} - QoS_{provided}^{i}\right|}{QoS_{provided}^{i}}$$

Where $QoS_{required}^{i}$ is i^{th} required QoS factors and $QoS_{provided}^{i}$ is provided QoS value for i^{th} required QoS.



(Figure 2) Comparison by Customer's Satisfaction Degree

We had obtained average satisfaction degree for conventional way is 0.61 and the average satisfaction rate when using our model is 0.701. Although, there are pros and cons in our evaluation, the initial result is good enough to



(Figure 3) Screenshot of Decision Tree

motivate us to move next step. We consider 0.701 is reasonable result for initial step. The comparison result is shown in Figure 2.

6. CONCLUSION

In this paper, we proposed to select and classify Web Services using Decision Tree algorithm based on QoS attributes provided by the client. Service classifier improved selection of relevant Web Services early in the composition process and also provide flexibility to replace a failed Web Services with a redundant alternative Web Services, resulting in high availability and reliability of Web Service composition. We also provided an implementation of our proposed approach along with efficiency measurements through performance evaluation. Results of our implementation and performance evaluation are good enough to motivate us to move next step.

This work is just a first step of a wider project we are currently working on towards to reliable Web Service Composition. Due to the shortage of the space, we could not formulate how to detect faults and recover from them in details. Therefore in the future, we are planning to perform more detailed research on reliable Web Services Composition by describing fault detection, fault-tolerant strategies and recovery procedures.

REFERENCE

[1] C. Lin, R. Sheu, Y. Chang, S. Yuan, “A relaxable service selection algorithm for QoS-based web service composition,” *International Journal on Information and Software Technology*, Vol.53, No. 12, December 2011.

[2] A. Liu, Q. Li, L. Huang, M. Xiao, “FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services,” *IEEE Transactions on Services Computing*, Vol.3, No.1, 2010.

[3] Z. Xu, P. Martin, W. Powley, F.Zulkernine, “Reputation-Enhanced QoS-based Web Service Discovery,” In the *Proceeding of IEEE International Conference on Web Services (ICWS 2007)*, USA, 2007.

[4] S. K. Bansal, A. Bansal, “Reputation-based Web Services Selection for Composition,” In the *Proceeding of IEEE World Congress on Services*, USA, 2011.

[5] M. Li, T. Deng, H. Sun, H. Guo, X. Liu, “GOS: A Global Optimization Selection Approach for QoS-Aware Web Service Composition,” In *the Proceeding of IEEE International Symposium on Service Oriented System Engineering*, pp.7- 14, 2010.

[6] P. Wing, K.M. Chao, C.C. Lo, “On optimal decision for QoS-aware composite service selection,” *Expert Systems with Applications*, Vol.37, No.1, pp.440-449, 2010.

[7] N. Laranjeiro, M. Vieira, “Towards Fault-Tolerance in Web Service Composition,” In the *Proceeding of the Workshop on Engineering Fault-Tolerant Systems (EFT 2007)*, Croatia, 2007.

[8] A. Nasridinov, P.P. Hung, L. Qing, J.Y. Byun, “XaP-SOAP: XML-based Attacks Tolerant SOAP Messages,” *Journal of KIISE: Computing Practices and Letters*, 2012.