Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Fault Analysis in the Selection of Automatic Web Service Composition

Hima Bindu K* Department of CSE, SVEC
Delhi Babu K Department of CSE, SVEC
Jaya Chandra Reddy B Department of IT, SVEC
Suresh G Department of IT, SVEC

***Abstract*—Now-a-days, web services are a famous way to implement the architecture of the services which exists at the internet. Choreography of more than one web service at same time to complete the workflow of activities given by user is one of the important research issues in the field of SOA. We have many ways to compose different web services according to the activities in the given workflow. Now-a-days, web service choreography will be done automatically and become an important research issues considered under web service choreography. In this way, there is a chance of getting some faults at the time of choreography of web services. This will be reduced when we will trace it at the time of the selection of web services before choreography. This paper express how to analyze the faults occurred in the selection.**

***Keywords*— Web Service Composition; Fault Analysis; Service Selection; Workflows; Automatic Service Composition**.

I. Introduction

Service-Oriented Architecture (SOA) is an upcoming organizational model aiming at simplifying large-scale business operations by consumption of ready-to-use services. Here, services will be considered as a simple calculation or typical business process. The most prominent realization of SOA is currently in the area of web services. Web services are loosely-coupled, platform-independent, self-describing software components that can be published, located and invoked via the web infrastructure using a stack of standards such as SOAP, WSDL and UDDI which indicates binding, discovery and publishing of new web services with existing web services in the web service repository.

Web services are software programs that use XML to exchange information with other software via common Internet protocols. Web services are having some properties such as scalable (e.g. multiplying two numbers together to an entire customer-relationship management system), programmable (encapsulates a task), based on XML - open, text-based standard, self-describing nature (metadata for access and use), and discoverable (search and locate in registries).

The interaction of Web Services is based on the exchange of messages. SOAP is an XML-based protocol for specifying how to format and package the information contained in these messages, and how to transmit them through the network. A SOAP message is specified by an envelope that consists of a SOAP header and a SOAP body. The header contains data that is necessary for intermediate processing of the message or infrastructure protocols (like security or transaction). The application data that the sender wants to transmit to the receiver is placed into the body part of the SOAP envelope. The SOAP standard does not only specify the format of the exchanged messages but also the transport protocol to be used for transmission through the network. This specification of a transport protocol is called binding. A binding defines how messages are to be packaged within the transport protocol and which protocol primitives are to be used.

The Web Services Description Language (WSDL) is an XML-based language for the specification of the interface of a Web Service. Originally it was created by Microsoft, IBM, and Ariba and is now maintained by the Worldwide Web Consortium (W3C). A WSDL document describes the functionality of a Web Service, how to interact with it, and its input and output messages. An XML document specifying a Web Service interface encloses two parts of the WSDL description, an abstract part and a concrete part. The abstract part defines types, messages, operations, and port types of a Web Service without specifying concrete bindings or addresses, whereas the concrete part defines bindings and ports.

UDDI stands for Universal Description Discovery and Integration. This specification was first proposed by Microsoft, IBM, and Ariba and is now supervised by the standards organization OASIS [UDD]. The goal of UDDI is to allow for publishing, discovering, and locating Web Services. The information stored in a UDDI registry can be parted into 3 categories. To understand what kinds of information are contained in a UDDI registry often there is an analogy with a telephone directory used:

- The white pages are listings of organizations, their contact information, and the Web Services they provide. This category allows a client to search for Web Services on the basis of businesses.

- The yellow pages describe a classification of companies and Web Services according to taxonomies. By using this category of UDDI information, a client can search for Web Services based on a wanted category.
- The green pages provide information on the way of invocating Web services.

By using the above standards, we can perform the web service choreography. [1][2][3][4][5][6] specifies some of the techniques to perform web service composition. In general, web service choreography can be done in manual process. It means, human intervention will be more than system performance. This process will leads to the cause of many failures at different situations in different phases of the construction of choreography of composite web services. To reduce the failures in the manual process and get better results of the composite web services, we introduced the methodology of the automation of manual web service choreography.

In automatic web service choreography, we have four phases to perform automatic web service choreography. The clear details about the phases are specified in [2]. In this paper, we concentrate only on the analysis of the faults which occur at the time of selection of composite web services in the automatic web service choreography.

The rest of the paper is organized as follows. Section II explains about the background of the web service choreography. Section III presents motivation for fault analysis. Fault analysis is presented in section IV. Section V concludes this work.

II. Background

There are two ways to perform web service composition. Web services choreography is one way and other is orchestration. Orchestration is typically a business process involving one role (i.e. a single execution point) for the process. Business does not always exist in such simple terms especially in enterprise environments. In this situation, we use choreography which considered as extension of orchestration. In simple way, choreography means a coordination of a set of individual web services being executed by one group of web services with the steps in the web services accessing the other groups or institutions. By using the way of choreography of web services, we perform the automatic web service composition by using both Orchestration and choreography. It means selection of individual web services to form composite web services is done by Orchestration and compiling and execution of composite web services will be done by Choreography.

III. Motivation

Web services are becoming progressively popular in the building of both inter- and intra-enterprise business processes. These processes are composed from existing Web services based on defined requirements. In collecting together the services for such a composition, developers can employ languages and standards for the Web that facilitates the automation of Web service discovery, execution, composition and interoperation. However, there is no guarantee that a composition of even very good services will always work. There is a chance of occurrence of any kind of faults at the time of selection of web service composition which in turn leads to the failure of software components. If any software component will fail before execution of composite web services, this will result in the total failure of the automatic web service composition. Before the software component become fail, failure analysis will help to reduce the occurrence of failure in the automatic web service composition. To overcome those failures, we know about what is the fault of the software component which leads to failure. This reason will give motivation to the analysis of faults which occur at the time of selection of composite web services.

IV. Fault Analysis

Before going to the analysis of the faults which occurs in the selection of automatic web service composition, we know about the definition of faults which is specified as follows.

“A fault is an abnormal condition or defect at the component, equipment, or sub-system level which may lead to a failure”.

There are many techniques to analyze the occurrence of faults. One technique is to classify the faults according to the situations in the selection of the composite web services. They are considered in the different views such as physical view, Development view and interaction view. These views are considered at the time of the selection of the automatic web service composition. The clear explanation of the above views is listed as follows.

A. Physical View

Physical View is a situation of the conversation of web services with the system in automatic way. That means the occurrence of faults in the system architecture or software components of the web services which was supposed to execute it successfully. There are two choices of faults in the physical view. One choice is failures in the system architecture which was supposed to execute the given list of web services. This will be done when system will got damaged or failures in the power supply to the system which leads to the major failures in the automatic web service composition. Second choice of expecting failures in the concept of physical view is a chance of occurrence of failure in the software which supports the extraction of the web services from repository. From this explanation, we clear about what faults will be considered as physical view.

B. Development View

Development View is a situation where the chance of occurrence of faults in the selection of web services to perform web service composition in automatic way. There are different ways to get a chance of occurrence of the faults in the development view of the fault analysis. Some of the critical issues related to this view are listed as follows.

1) Input/output is not correct /complete,
2) Preconditions/ post conditions are not satisfied,
3) SLA is violated, etc.

The above chance of occurrence of faults will occur at the time of the development of the web service composition. This will effect in the development of the composite web services which leads to yield wrong outputs.

C. Interaction View

This view specifies the chance of the occurrence of the faults in the interaction between system and the web services. This view will also be considered as communication view. In this view, we know about how faults will occur in two different ways.

One is the occurrence of faults in the communication between system and web services in repository. Second one is the occurrence of failure in the communication of web services in the composite web services which will participate in the automatic web service composition.

Some issues which is considered under this view is stated as follows

1) Message is lost

This kind of faults will be raised when a communication message is sent from one web service to other or from system to Repository. It means a message which was sent by one side was not received by other side. This will leads to the effect of “a service which can interact with other will not coordinate with other service which will be useful to achieve user requirements”.

2) Connection error

This kind of failure will arise when two systems i.e. client and server which one extracts web services from repository and send to other which will perform selection of individual web services to form composite web services are not communicate to each other. This will leads to the failure not only for selection of composite web services but also for automatic web service composition.

3) Timeout is expired

This is a fault which occurs at the interaction of the client and server communication. Communication is done based on the time slot for a particular task. Based on that time slot, messages are exchanged between client and server to access the web services to perform the task given by the user. If there is any problem in the system like the speed on transmission of the messages will be slow, slow response by the system for received request, then the response or acknowledgement messages will not receive in correct time according to the given time slot. Then the system considers as “message will be lost but in actual process, that message was sent by other system”

From the above views, we have a popular terms is messages which communicates one system with other, one web service with other or web services to the other system or vice versa. Messages are implemented by a standard language named SOAP (Simple Object Access Protocol).

Some of the faults in the automatic web service selection will be discussed in [7] [8]. These papers will also specify the detection, detection and diagnosis of the faults in the web service selection. In some situations, we will call the faults in the web service composition as failure risks. This can be clearly discussed in [9].

Existing service composition selection models do not consider unpredictable service faults take into account. The problem cannot be fully resolved by discarding the failed service. The time for finding a new solution from scratch increases latency for the requests arrived in the system. We have to propose to choose several configurations with good qualities at the selection stage. If a failure occurs in the chosen configuration, selection can be switched into another configuration. Furthermore, we have an idea about the raising of faults and how to diagnose it by an example is described in [9] [10]

V. CONCLUSIONS

In general, we have a chance of occurrence of faults in the web service composition in the time of its development. We have several methods to resolve these faults but we know about what kind of fault will appear and how it can be resolve by using the existing methodologies. This will be understood by analyzing the raised fault in the corresponding automatic web service composition. In this paper, we discuss about what are the faults raised at the time of the selection of the automatic web service composition. We have to study about the other phases (i.e. execution and design) in the automatic web service composition as future work. Extension to this work is to study about how it can be traced automatically by the system. It means designing the corresponding tool to trace the faults which will occur at not only the time of selection but also for total web service composition.

ACKNOWLEDGMENT

I want to say thankful to my institution named Sree vidyanikethan Engineering College (SVEC), Tirupati and also to my project guide to encourage me to prepare this paper by giving his moral support to improve my skills.

REFERENCES

[1] Sleiman Rabah, Dan Ni, Payam Jahanshahi, Luis Felipe Guzman,"Current State and Challenges of Automatic Planning in Web Service Composition",
[2] Elias Ioup, John SampleAnya Kim, Myong Kang, Catherine Meadows"A Framework for Automatic Web Service Composition"
[3] Marco Pistore, "Synthesys and Composition of Web Services.",2009
[4] Srivastava, B., Koehler, J.,”Web Service Composition - Current Solutions and Open Problems”, Proceedings of ICAPS Workshop on Planning for Web Services, 2003.
[5] Yujie Yao, Yujie Yao, "A Rule-based Web Service Composition Approach", Sixth International Conference on Autonomic and Autonomous Systems, 2010.
[6] Chafl, G., Chandra, S., Kankar, P., Mann, V.: ”Handling Faults in Decentralized Orchestration of Composite Web Services”, International Conference on Service-Oriented Computing, 2005, pp. 410–423.

[7] S.Poonguzhali1, R.Sunitha2, Dr.G.Aghila3, "Self-Healing in Dynamic Web Service Composition", S.Poonguzhali et al. / International Journal on Computer Science and Engineering (IJCSE), pp-2054-2060.
[8] Mohammad-Reza Motallebi, "Failure Recovery in Web Service Composition", LIP6-NII Workshop -Paris, June 2010
[9] Adina Mosincat, Walter Binder, “Automated Performance Maintenance for Service Compositions"
[10] Natallia Kokash,"A Service Selection Model to Improve Composition Reliability"