A TESTING FRAMEWORK FOR FAULT TOLERANT COMPOSITION OF TRANSACTIONAL WEB SERVICES

Deepali Diwase
Pune Institute of Computer Technology, Pune University
Pune, Maharashtra, India
deepali.diwase@gmail.com

Pujashree Vidap
Pune Institute of Computer Technology, Pune University
Pune, Maharashtra, India
psvidap@pict.edu

Abstract

Software testers have great challenges in testing of web services therefore testing technique must be developed for testing of web services. Web service composition is an active research area over last few years. This paper proposes a framework for testing of fault tolerant composition of web services. It will tolerate faults while composition of web services. Exception handling and transaction techniques are used as fault handling mechanisms. After composition web services are deployed on WS-BPEL engine. Testing Framework will fetch results of composite web service from WS-BPEL engine and check whether composed web service is fault tolerant and it is in the consistent state.

***Keywords*: Web services; Testing ; Fault Tolerance.**

1. INTRODUCTION

Web services are autonomous software identified by URIs which can be advertised, located and accessed through messages using XML based standards Web Service Description Language (WSDL) and transmitted using internet protocols [5]. Web Services can be combined to implement new services which are called composed web services [3]. As services are deployed on the unreliable internet and they are often long running, failures are expected to happen frequently during the execution of composite web services therefore to deliver reliable service over unreliable internet is challenging problem. Reliable system can be achieved by using following four ways fault prevention, fault tolerance, fault removal and fault forecasting. It is impossible for composite web service to avoid all faults during its execution. Fault Tolerance gives correct service even in presence of faults therefore it is used for web service composition [1].

Web Services provides collaboration among distributed applications in an open environment. There is need to generate test cases to ensure the quality of the services that are published, bound, invoked and integrated at runtime and testing should be executed at runtime [10]. There is need to test web services rapidly for this purpose coyote framework can be used [7].

In this paper we propose framework for fault tolerant composition of transactional web services. Atomic transactions are considered for the purpose of composition. While composition of web services faults are tolerated by using exception handling and transactional techniques. Then composed web services are tested by using testing framework.

The rest of the paper is organised as follow Section 2 discusses related work, Section 3 discusses testing framework approach. Section 4 concludes the paper.

2. RELATED WORK

Many web services provide identical functionality with different quality of service so to determine which services are to be participated in a given composite services are very important. Execution price and reputation are Deterministic Quality of Service and Response Time,Throughput , Accuracy Availability are Non Deterministic Quality of Service by using these QoS component web services can be selected for composition [5]. Web services can be tested by using event based approach. Test cases are generated for testing interactions of WSs within a composite Web Service. Dependencies among transferred data and conditions in control flow and real-time constraints are modeled by concurrent event sequence graphs (cESG) by using decision tables (DT) [3].

WSDL can be used to test web services. By using WSDL Web Service test cases are automatically generated. These test cases carries the basic information of a service including its interface operations and the data transmitted. First WSDL file is parsed and transformed into the structured DOM tree. Then, test cases are generated test data generation and test operation generation [10].

The testing of web services are important for both service providers as well as subscribers. To test web services rapidly this paper proposes an XML-based object-oriented testing framework Coyote. It consists of two parts: test master and test engine. The test master allows testers to specify test scenarios and cases as well as perform analysis, and converts WSDL specifications into test scenarios. The test engine interacts with the web services under test and provides information [7].

From the WSDL file, we can not get the information which is useful for testing such as dependence information. To solve this problem, WSDL is extended to support this kind of information description. This paper, discusses four kinds of extension: input-output dependency, invocation sequence, hierarchical functional description and concurrent sequence specifications[8].

New web applications are generated by composing web services which meets the dynamic need of enterprises. Incorrect input or flow of designed process to composed web service encounters various faults during execution. It is difficult to find original failure service. Tools can monitor state of atomic service but can not locate source failure. For service composition execution flow model is used. Fault transmission path is identified for potential faults. In this way the original failure service can be identified.Automated testing tool SCENETester, supports fault diagnosing for web service composition in web service middleware SCENE. For concurrent test execution SCENETester, integrates distributed resources to construct a virtual runtime environment for web service testing and decompose task to distributed nodes[12].

Several web services are combined to create new service called composite web service. Regression testing method can be used for composite web services. This method can instructs to the tester to locate faults in composite web service also makes test data and test behavior independent of each other[2].

3. APPROACH

In this paper we propose a Testing framework for composite web services. First do the composition of web services,which uses exception handling and transaction techniques as fault handling strategies while composition of web services [1]. After composition web services are deployed on WS-BPEL engine. Then testing framework will fetch composed web services from WS-BPEL engine.

The proposed approach is given using set theory is as follow.

Let U is a system

$U = \{ \quad I , O, S_u , F \}$

Input:

$I = \{W, R, E_r\}$

- W is set of atomic Web Services, $W = \{ w_1, w_2, w_3, \ldots\ldots, w_n \}$
- R is Request From User
- E_r is set of Event Condition Action (ECA) Rules , $E_r = \{ E_{r1}, E_{r2}, E_{r3,\ldots\ldots} E_{rn} \}$

$E_{ri} = \{E_v, C_n, A\}$

Where

E_v is set of Events, $E_v = \{ E_{v1}, E_{v2}, E_{v3, \ldots\ldots}, E_{vn}\}$

C_n is set of Conditions, $C_n = \{True, False\}$

A is set of Actions, $A = \{A_1, A_2, A_3, \ldots., A_n\}$

Example:

EVENT	Buy Product: Physical Fault
CONDITION	True
ACTION	retryUntil (Buy Product, 5,1)

Above example shows that buying product is event if physical fault occurs then retry Buy Product event every 1minute if it is unavailable then at most try 5 times.

- Output:

$O = \{C, E, P, T, q_{pr}, n_1, q_{rp}, n_2, q_{rt}, N, n_3, D_t, U_t, N_t, N_f, , , q_{rt}, T_h, A_c, A_v\}$

Testing Framework with report which shows that composite web services are without any fault.

Where

C is set of Composite Web Services $C = \{C_1, C_2, C_3, \ldots\ldots, C_n\}$

- Success State:

Su = Composite web service in a consistent state.

$C \notin E$

Where

Su = Success State

E = Exception Handling Strategies in composite web service**.**

E = {Ignore, Notify, Skip, Retry, Retry Until, Alternate, Replicate, wait}

- Failure State:

F is failure state

F = Composite web service in an inconsistent state

$C \in E$

- Supported Quality Of Service (QoS) Attributes**:**

1. Execution Price
2. Reputation
3. Response Time
4. Throughput
5. Availability
6. Accuracy

Suppose QoS of the composite service, when executed using plan

$P = \{<w_1,t_1>,<w_2,t_2>,<w_3,t_3>\ldots\ldots\ldots\ldots<w_n,t_n>\}$

Where

f (P->(W, T))

T = Set of task, in web services

- Execution Price:

$$q_{pr}(P) = \sum_{i=1}^{n1} q_{pr}(w_i, op(t_i))$$

Where

q_{pr} = Execution Price

$q_{pr}(P)$ = Execution price of plan P is a sum of the execution prices of the operations invoked over n_1 number of services that participate in composite web service.

$op(t_i)$ = operation invoked by task t_i.

- **Reputation:**

The reputation $q_{rp}(w)$ of a service w is a measure of its trustworthiness.

$$q_{rp} = \sum_{i=1}^{n2} q_{rp}(w)$$

Where

q_{rp} = Reputation Of Service

R_p = The end user's ranking on a service's reputation,

n_2 = The number of times the service has been graded.

$$q_{rp}(P) = \frac{1}{n2} \sum_{i=1}^{n2} q_{rp}(w_i)$$

The reputation $q_{rp}(P)$ of an execution plan P is the average of the reputations of the n_2 number of services that participate in P.

3. Response Time:

$$q_{rt}(P) = \frac{1}{N} \sum_{i=1}^{n3} rt_i$$

Where

r_t = Response Time
N = Number of requests
n_3 = Number of observations of response times

4. Throughput:

T_h = Number Of requests per Second.

5. Availability:

$$A_v = 1 - \frac{Dt}{Ut}$$

A_v = Availability of web service
D_t = Downtime of web service
U_t = Uptime of web service

6. Accuracy:

$$A_c = 1 -$$

A_c = Accuracy
N_f = Number of failed requests
N_t = Number of total request

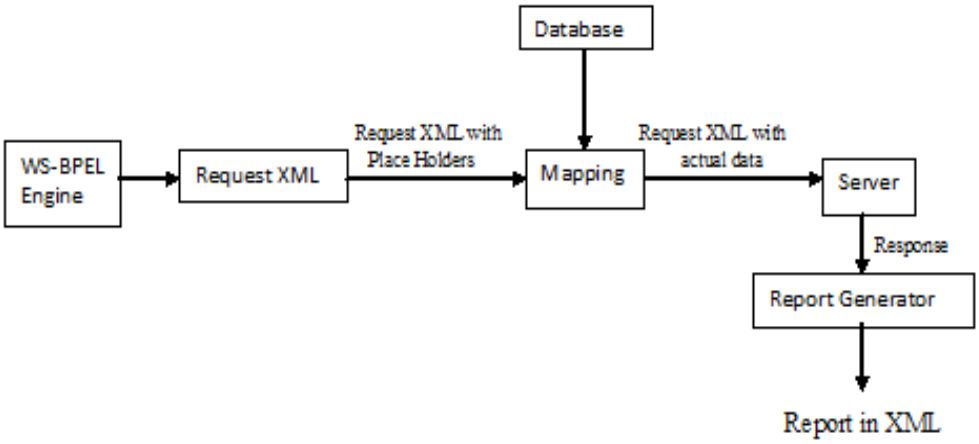

Figure 1: Proposed architecture for testing framework of web services composition.

Figure1 shows Testing framework for composed web services. First Web Services are composed while composition faults are tolerated by using exception handling strategy and taxonomy of transaction [1].

Exception handling strategies allow catching faults and continuing composition of web services also taxonomy for transactional web services is used which ensures that web services are in the consistent state. Web services and fault handling logic are created. fault handling logic is created by using Event Condition Action (ECA) rules [1]. Three types of fault are identified in this paper: Physical faults, Logical Faults and Interaction Faults.

Physical Faults :

Failures in the network medium or failures on the server side are observed as physical faults. most of the times failure occurs in composition of web services because of unavailability of component services.

The availability of a Web service is affected by the server and by the networking media. Service unavailability can occur because either the service is down or the network connection to the service is down. When there is no response then such fault is

unavailability fault.

Logical Faults:

These are Faults thrown by external web services. When web service is unable to execute successfully due to various reason then Logical Faults are generated.

Interaction Fault :

Interaction faults occur at execution phase of composite service. These faults can degrade the performance of system or total failure to deliver service. Content and SLA these faults are interaction faults.

These faults are handled and composed web service requests are created and deployed on WS-BPEL engine. Testing Framework will fetch the request XMLs from WS-BPEL engine and these XMLs are stored. In these request XMLs place holders are created.

Eg. Suppose Request XML have statements

<add: Name> ? </add : Name>

<add: Address> ? </add : Address>

then instead of symbol "?"placeholder for this statement will be created as :

<add: Name>#Name#</add: Name>

<add: Address>#Address#</add: Address>

In this way request XMLs with place holders are created. This request XML format will not change . Only data in these XMLs will be changed. Mapping module is used to map data and placeholders. Data required for mapping is taken from database, properties files or xls sheets. Then placeholders in request XMLs are replaced by actual data then it becomes Request XML with actual data. Expected results of XML are stored in xls sheet.

When request XML is sent towards server and it will get response from server. Testing Framework will test these responses with the expected results which are stored in xls sheets. Report generator module is used to generate reports in XML format it will store the request and response XMLs.

After getting the composed service request it will estimate Quality of Service of that service. These QoS are Execution Price , Reputation,Response Time ,Throughput , Availability, Accuracy.

4. CONCLUSIONS

This paper discusses testing framework for composed web services. While composition it uses fault handling strategies so it allows to handle fault and continue composition so that number of web services in composition can be increased. Testing framework is automated so manual testing is not required. While using web service QoS is also important so QoS of web services are calculated.

REFERENCES

[1] A Liu,Qing Li,(2010)"FACTS : A Framework For Fault Tolerant Composition of Transactional Web Services",IEEE Transactions on Services Computing, Vol 3, No.1

[2] B Yang, J Wu, C Liu, L. Xu (2010): "A Regression Testing Method for Composite Web Service" 978-1-4244-5316-0/10/.2010 IEEE.

[3] F.Belli,M.Linschulte,(2009)"Testing Composite Web Services- An Event Based Approach" IEEE International Conference on Software Testing Verification and Validation Workshops .

[4] H.Zhu, Y.Zhang, (2012)"Collaborative Testing of Web Services", IEEE Transactions Web Services Computing, Vol 5, No.1 Jan 2012.

[5] L.Zeng, B.Benatallah, (2004)"QoS-Aware Middleware for Web Services Composition", IEEE Transactions on Software Engineering, Vol 30, NO5.

[6] S.Bhiri , O.Perrin , C.Godart,(2005) "Ensuring Required Failure Atomicity of Composite WebServices"ACM conference.

[7] W.T.Tsai, Ray Paul, W.Song, Z. Cao(2002) "Coyote: An XML-Based Framework for Web Services Testing" 7th IEEE International Symposium on High Assurance Systems Engineering.

[8] W.Tsai, R Paul,Y Wang, C. Fan, and D.Wang ,(2002)" Extending WSDL to Facilitate Web Services Testing" 7th IEEE International Symposium on High Assurance Systems Engineering .

[9] W.Tsai , Y.Chen, R.Paul, N. Liao and H. Huang (2004)"Cooperative and Group Testing in Verification of Dynamic Composite Web Services" 28th Annual International Computer Software and Applications Conference .

[10] X.Bai,W.Dong,W.Tsai,Y.Chen,(2005)"WSDL Based Automatic Test Case Generation for Web Services Testing" IEEE International Workshop on Service-Oriented System Engineering (SOSE'05).

[11] Y. Zheng, J.Zhou, P. Krause(2007) "An Automatic Test Case Generation Framework for Web Services" Journal of software, VOL. 2, NO. 3.

[12] Z Zhu, J Li, Y Zhao, Z li (2011):"SCENETester: A Testing Framework to Support Fault Diagnosis for Web Service Composition." 2011. 11th IEEE conference.