A Framework for Nested Web Services

Ketan N. Kanere
M.E. (Computer Engineering)
D. J. Sanghvi College of Engineering
Vile Parle, Mumbai-400056

Dr. Abhijit Joshi
Vice Principal (Academic)
D. J. Sanghvi College of Engineering
Vile Parle, Mumbai-400056

***Abstract* — In olden days web applications were created individually to serve the purpose according to the requirements. Today web services are used to develop web based applications. These web services consist of agents for the task operation. The agents running inside a web service contain a nested call to other agents to complete their tasks. Due to which, the processing time of these agents consume the processing time according to external process of other web services. Moreover there will be errors within external web services. Because of all these reasons the flow of the web services must be defined before the implementation of the web service. As a result of all these, the Web Services Composition (WSC) is used to define the flows and descriptions to support the verification process of the web services agents. We are proposing a new architecture which will be a modified version of the existing Execution Mean Time Interval (EMTI), by extending the Web Services Description Language (WSDL) of the nested schema. Here we will also monitor all the elements in the required agents. At the same time the hidden false error and exception can be corrected before the implementation and the execution of the web service using the web agent.**

Index Terms — web services agent, web services composition, web services verification, nested structure, extending WSDL.

I. Introduction

A Web service is a communicating link between two electronic nodes over World Wide Web. A web service is a software program used at execution time provided at a network address over the web; it is a service that is "always running" and never stops executing which is also called as the concept of utility computing.

Nested web services are web services consist of web agents that perform the task of the particular web service or any operation which needs to take place. Web services may or may not call external agents for the completion of their tasks. There are two types of agents in web service basically called as independent agent (IA) and dependent agent (DA).

Independent agent (IA) is the agent of the web service which can perform its task individually, on the other hand Dependent agent (DA) are that type of agent of web service which calls external services for the completion of its task and execution. Hence, nested web services come in execution when agents in web services are dependent agents, i.e., they require external agents or external services for the completion of their tasks.

II. A Testing Environment for Multiple Agents

Web service composition is very large and complex. It is very difficult to test complex web service composition. But, with the help of distributed structure of agents it becomes possible. At the initial stage the test environment was developed for Business Process Execution Language (BPEL) - based Web Service composition. The High Petri Nets (HPN) is used to model the Business Process Execution Language. The High Petri nets can be easily referenced by test case generation and test evaluation, this property of HPN will be used in modeling BPEL - based Web Service composition (WSC). The multi-agent test environment for BPEL - based Web Service Composition can be implemented by analyzing the parameters and ontology of the BPEL - based Web Service Composition. It is defined on the basis of XML due its flexibility and extensible use. The agent communication, the terms for test case generation and test evaluation is based on the parameters and ontology [4].

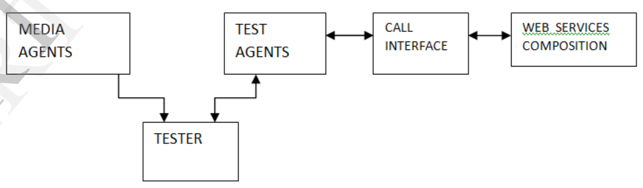

Figure 1: Agents for testing BPEL - based WSC

The figure 1 gives the testing environment for BPEL - based WSC. It consists of media agents, tester, test agents, call interface and BPEL - based web services composition. The media agents consist of Composition Structure (CS) agents which analyze the structure of BPEL - based web services composition. The structure is analyzed to create a HPN presentation, which describe the structure [4].

The tester consists of the Test Case Generator (TCG) agents to generate test cases and test criteria. Similarly, the Test Case Execution (TCE) agents in tester execute test cases and generate results. The Test agents consist of Test Oracles (TO) agents to verify results by TCE whether they match with BPEL-WSDL specification. Test Assistance (TA) agents are a part of test agents which is generally the interface between tester and computer. It guides the tester in the process of testing [4].

The role of providing flexibility to each kind of web service is performed by the Coordinate Interface (CI) components written in different languages and tested in a uniform environment through to CI [4].

III. Trusted Web Service

The Web services developer must ensure that the delivered web services should have the quality such as availability and reliability during their execution time. But, during the execution time of the service agent a critical problem takes place, such as the infinite loop problem in the web service process.

There are many methods proposed to protect the unusual errors. Most of these methods focus on the verification and validation part during the development process. Still, the infinite loop problem cannot be resolved completely using these methods as this problem occurs due to unexpected random values obtained from the execution of request and response process. To address this issue, a protection mechanism is suggested that completely detects and protects the unbound loop problem of web services when a dynamic situation occurs in the request services of each requester [2].

The mechanism ensures that users will definitely be protected from a critical lost which takes place from the unusual infinite loop of the web services system. Also, the service agents with dynamic loop control condition can be trustable [2]. The protection mechanism system framework for distributed web services is given in figure 2.

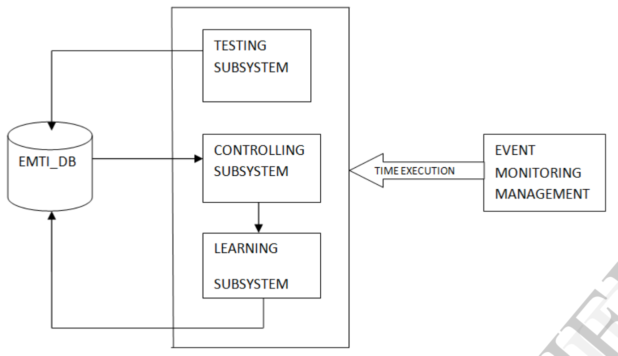

Figure 2: System framework for distributed web services

Here, the system is divided into three subsystems, namely, testing, controlling and learning. And these three subsystems are further divided as follows:

The Testing subsystem consists of two parts, namely, Checking Parameters Module and EMTI Module.
1 Checking Parameters Module (CPM): In order to protect the problem of the infinite loop in service process the CPM tests the services of web services software by choosing right parameters.
2 EMTI Module (EMTIM): The EMTI module records the execution time of the web services when it runs. Under the testing process, the EMTIM will record and store every valid process time of the web services software in the execution mean time interval database, i.e., EMTI_DB.

The second subsystem is controlling which consists of three modules: time checking module (TCM), reporting module (RM), and termination module (TM).
1 Time Checking Module (TCM): TCM checks the time-boundary by comparing the execution time with the EMTI boundary. For example, the TCM will be checking for every 5 seconds; nevertheless, the requesters can set another value based on the critical of the system.
2 Reporting Module (RM): If the TCM found that there is a potential of infinite loop occurs for the running web services software, the process will be transferred to the RM. Now, RM is responsible for creating a warning message to requesters. The message will be sent to requesters to call a termination module, or continue execution.
3 Termination Module (TM): As per the outcome of the TCM and RM, the TM will act if the requesters choose to terminate the computing process. Therefore, there will be no un-expected result from the invalid loop execution.

The last subsystem is learning which consist of the two modules, namely, Time Recording Module (TRM) and EMTI module [2].
1 Time Recording Module (TRM): Referring to the EMTI boundary from the testing process mentioned previously, the every execution time will be recorded when the web services software normally terminates. The number of the normal execution time is depended on the recording period that sets by the administrator or the organizational policy. However, the size of the time period indicates the critical of the system. For example, if the system is not the critical system, the size of time period to record every execution time before calculating the EMTI can be every 1 hour; otherwise, it may be 5 minutes.
2 EMTI Module (EMTIM): This EMTIM is quite similar to the EMTI module in the testing phase except that the recording of the executing time is obtained from the TRM. The EMTIM in the deployment process will select the execution time storing in the EMTI_DB, starting from the last calculation value of the last EMTI to the last execution time of the calculation boundary. For example, if the last execution time of EMTI is 15 seconds, then, in the last 5 seconds of the time recording period, there are 30 values of the execution times before the new EMTI to be calculated. So, these 30 values will be selected to compute the EMTI boundary, meanwhile, the TM still records the incoming execution time of other rounds of the web services software.

The protection mechanism framework also consists of a part called as Event Monitoring Management (EMM). It calculates time execution of service functions, counting from the service is called until it returns the result to the requester. As a result, time of each service process will be obtained. This measured time unit will be sent to the controlling subsystem during the run time process of the web services.

IV. EXTENDING WSDL SCHEMA

The Web Service Description Language (WSDL) is used to define the interface of a web service in XML format. The interface is used to define the functional and non functional properties of the web service [3].

The focus will be on the non functional property of a web service, especially the criteria which is non functional property of a web service. The Web Service Definition Language (WSDL) schema will be extended by adding criteria information as a new element, i.e., ‘criteria service’, which is available in the new namespace. It is also possible to specify the criteria with a service in an X-WSDL document using the ‘criteria service’ element [9].

In Service Oriented Architecture (SOA), the functional properties are published and invoked. That is, the pattern of register, search and invoke service is maintained [5]. However, WSDL and Universal Description, Discovery and Integration (UDDI) are modified to accommodate the specification of criteria. Following steps are used to add criteria in the web service:

1. Specify criteria by using X-WSDL: The provider of the service specifies the criteria in the description of the web service using X-WSDL. The criteria have to be specified along with the service in the WSDL document. In order to be able to do so the elements required for the specification of the criteria must be defined in the WSDL schema. We therefore first extend the WSDL schema and then show the extension to the WSDL document.
2. Publish the criteria associated service: Publish the X-WSDL description in the X-UDDI registry.
3. Invoke the criteria associated service: Client searches in the X-UDDI to find the appropriate web service according to the desired criteria.

The above mentioned steps results in new software architecture with X-WSDL and it is given in figure 3.

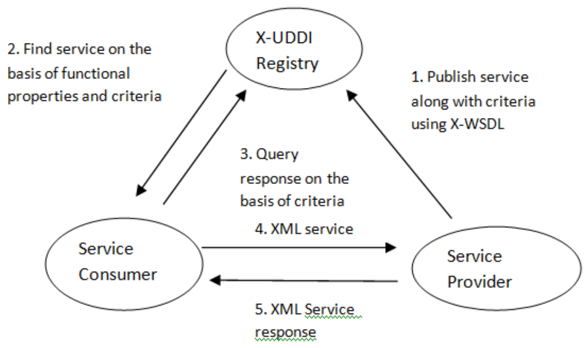
X-UDDI Registry
2. Find service on the basis of functional properties and criteria
1. Publish service along with criteria using X-WSDL
3. Query response on the basis of criteria
4. XML service
Service Consumer
Service Provider
5. XML Service response

Figure 3: Modified SOA architecture with X-WSDL

The X-WSDL schema is shown as follows:

```
<xs:element name="definitions">
<xs:key name="criteriaservice">
<xs:selector xpath="cr:service"/>
<xs:field xpath="@name"/> </xs:key
</xs:element>
```

Definition - it is a root element of the WSDL which contains the name of web service and namespace.
Types – a container for data type definitions using some type system (such as XSD).
Message – an abstract typed definition of the data being communicated.
Port Type – an abstract set of operations supported by one or more endpoints.
Binding – a concrete protocol and data format specification for a particular port type.
Port – a single endpoint defined as a combination of a binding and a network address.
Service – a collection of related endpoints

The WSDL document is extended at the service level. This is sufficient as the search is based on the criteria associated service [3]. Therefore, the service element of the standard WSDL is extended to support the additional feature of criteria and description. Service is a collection of ports where port is the endpoint which is the collection of binding and service access address. The service element is extended to include the criteria information along with the ports and bindings. The criteria name attribute provides a unique name to every criterion defined within the service element. Service element may be containing more than one criteria name. Criteria name is the name of the non-functional property. This helps the user to find the more appropriate service by specifying the criteria. The description attribute provides the detailed description of the criteria attribute. This is used to specify the complete requirement of the user in the documented form. The WSDL schema shown below is extended using the element criteria name and criteria service name.

```
<definition name=" "
targetNamespace=http://localhost:8080/X-UDDI/wsdl1
xmlns=http://schemas.xmlsoap.org/wsdl/
xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
xmlns:cr=" http://localhost:8080/X-UDDI/wsdl1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<message name=" ">
<part name=" " type=" "/>
</message>
<portType name=" ">
<operation name=" ">
<input message=" "/>
<output message=" "/>
</operation>
</portType>
<binding name=" " type=" ">
<operation name=" ">
<soap:operation soapAction=" "/>
<input>
<soap:body encodingStyle=" "/>
</input>
<output>
<soap:body encodingStyle=" "/>
</output>
</operation>
</binding>
<cr:service name=" ">
<criteria name=” ”
<description=” ”
<port binding=" " name=" ">
<soap:address location="">
</port></service></definition>
```

Consider a pizza ordering example where customer wants to order a pizza from a pizza shop on the basis of two criteria that are farmhouse pizza and margarita pizza.

The user’s two criteria are incorporated in the WSDL document using the newly added elements, criteria name and description [9]. This modified WSDL schema is given below.

```
    <wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:cr=" http://localhost:8080/EUDDI/wsdl1"
    targetNamespace="http://localhost:8080/EUDDI/wsdl1"
    <cr:service name="PizzaOrderService">
    <cr:criteria name= "Farmhousepizza"/>
    <cr:description name ="Farmhouse"
    </criteriadescription">
    <cr:criteria name= "margerita" element=
    "cr:bookpizza"/>
    <cr:description name = " grated cheese should be added as
topping" element= "cr:Bookpizza"
    </criteriadescription">
    <cr:port name="Bookpizza" binding=" "> </cr:port>
    </cr:service>
    </definition>
```

Next step is to publish the X-WSDL of the service in XUDDI. The definition of the service which is associated with the criteria is published in X-UDDI. Suppose the Pizza_Order_Service is providing two types of pizza, farmhouse and margerita, which are to be associated as criteria with the service. The code for publishing pizza order with criteria list is given as follows:

```
    <save_dService generic="2.0" xmlns=" ">
    <businessService businessKey="*****" serviceKey="">
    <name>Pizza_Order_Service</name>
    <criteria Bag>
    < criteria >
    < criteria Name>Farmhouse</ criteria Name>
    < criteriaDescription> "Pizza is available with
    grated cheese topping" </ criteriaDescription>
    </criteria >
    < criteria >
    <criteria Name>Margerita </ criteria Name>
    < criteriaDescription> "with extra cheese and chicken
sizzlings"</criteriaDescription>
    </ criteria >
    </ criteriaBag>
    </BusinessService>
    </save_dService>
```

V. OUR PROPOSED APPROACH

We have proposed a framework based on the requirements for checking the externally created nested web services and finding out the error rate. The changes will be made to the EEMTI framework as shown in figure 4.

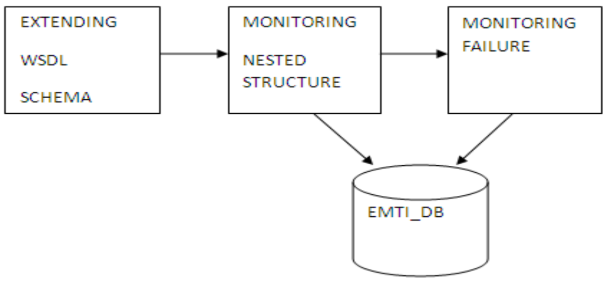

Figure 4: Extended EMTI working [10]

The proposed changes which need to be made in Extending WSDL Schema (EWS) and Monitoring Failure (MF) block of EEMTI framework. The EWS will be designed using the XML schema for extending in WSDL. This method adds description in the message type of the WSDL definition which is described for explaining nested structure on the EMTI architecture. Here we will be adding additional information to the WSDL schema as shown in figure 5. The added information will be criteria name and criteria description. This will extend more information of the WSDL schema. This extended information will be given to the Monitoring Nested Structure (MNS) and the Monitoring Failure (MF). The nesting in the structure and the faults will be checked hence forth.

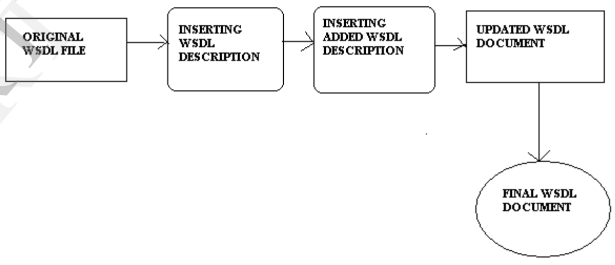

Figure 5: Extending WSDL Schema

Hence, the schema will be extended by adding parameters like criteria name and criteria description. The output will be an updated extended WSDL document. The MNS is an engine for tracking the nested structure remains same as mention in [10].

The MF checks the failure in the system. Currently, MF handles SOAP exceptions and EMTI faults. This is not enough. So we proposed here to modify MF block. The modified MF block will check for bound errors and exceptions along with the errors handled by MF block in current system as shown in figure 6.

The information from the SOAP Exception includes the cause of errors and the error nodes; these errors will be recorded to the EMTI_DB.

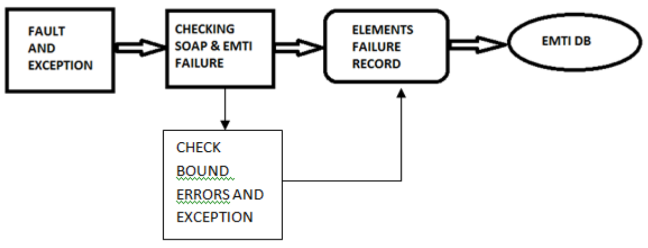
FAULT AND EXCEPTION
CHECKING SOAP & EMTI FAILURE
ELEMENTS FAILURE RECORD
EMTI DB
CHECK BOUND ERRORS AND EXCEPTION

Figure 6: Monitoring failure module

VI. CONCLUSIONS

The proposed framework consists of modified Extending WSDL Schema (EWS) and Monitoring Failure (MF). The extended WSDL schema criteria will be added as the definition to the existing WSDL file which changes the description of the document.

The additional description in the WSDL file which are criteria type and information will give an extended schema with more information. Hence, this will help in capturing the information which is to be given to the Monitoring Failure block.

Further, the modified Monitoring failure(MF) will check for bound errors and exception in addition to SOAP-Exception and EMTI-fault which will improve the performance of the system.

REFERENCES

[1] OASIS. (2007, April.), "Web Services Business Activity(WSBusinessActivity) Version 1.1," Available: http://docs.oasisopen. org/ws-tx/wstx-wsba-1.1-spec-os.pdf [Aug. 10, 2012].

[2] N. Srirajun, P. Bhattarakosol, P. Tantasanawong, and S. Han, "Trustable Web Services with Dynamic Confidence Time Interval," in Proc. of the 4th Int. Conf. on New Trends in Information Science and Service Science (NISS), 2010, pp.11-13.

[3] WSDL 2.0: A Pragmatic Analysis and an Interoperation Framework, 2008 SYS-CONMedia,

[4] W. Dong, "Multi-agent Test Environment for BPEL-based Web Service Composition", the IEEE Int. Conf. on Cybernetics and Intelligent Systems, 2008, pp.855-860.

[5] C. -H. Liu, S. -L. Chen, and X. -Y. Li, "A WS-BPEL Based Structural Testing Approach for Web Service Compositions," the IEEE International Symposium on Service-Oriented System Engineering, 2008, pp.135-141.

[6] N. Srirajun, P. Bhattarakosol, P. Tantasanawong, and S. Han, "A Trustable Software with A Dynamic Loop Control Mechanism," in Proc. of the 5th Int. Conf. on Future Information Technology, 2010, pp. 20-24.

[7] W3C. (2007, June), "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," Available: http://www.w3.org/TR/wsdl20/ [Aug. 10, 2012].

[8] U. Dayal, M. Hsu, and R. Ladin, "A transactional Model for Long- Running Activities," in Proc. of the 17th Int. Conf. on Very Large Data Bases, 1991, pp. 113-122.

[9] N. Parimala, and A. Saini, "Web Service with Criteria: Extending WSDL," in Proc. of the 6th Int. Conf. on Digital Information Management, 2011, pp. 205-210.

[10] Nalinrat Srirajun, Pattarasinee Bhattarakosol, Panjai Tantasanawong, Sunyoung Han, "EEMTI: An Extending Framework for Nested Web Service Verification" in Computing and Convergence Technology (ICCCT), 2012 7th International Conference.