Performance Testing of Composite Web-Service with Aspect-Based WS-BPEL Extension

Jong-Phil Kim, Dong-Hyuk Sung and Jang-Eui Hong
Dept. of Computer Science, Chungbuk National University
Cheongju, 361-763, Rep. of Korea
[e-mail: {kimjp, sungdh}@selab.cbnu.ac.kr, jehong@chungbuk.ac.kr]
*Corresponding author: Jang-Eui Hong

Received March 31, 2011; revised July 28, 2011; accepted October 26, 2011; published October 31, 2011

Abstract

The advance in Service-Oriented Architecture (SOA) and web services has led to the development of new types of a system in which heterogeneous service components can connect and compose to solve a complex business problem. In the SOA, even though these service components are valid in their functionality, there is a need to test their behaviors when those services are composited. In recent years, WS-BPEL has received a wide acceptance as a means of integrating distributed service components. To test the composite service, the existing testing techniques have been focused on the functional features based on the WS-BPEL process. However as SOA approach is applying to real-time software development, the performance of composite service becomes one of important issues. This paper proposes a technique to the performance testing of a composite service with WS-BPEL extension which combined with the concept of aspect. Our WS-BPEL extension has been made towards annotating aspect component which is measuring the response time of the composite service. This paper also explains the procedure of performance testing with on-line transaction system. Our technique can apply to choose an adequate component in service composition with considering the performance among several candidate web service components.

Keywords: Composite service, performance testing, WS-BPEL, aspect component

The part of this paper was presented in the ICONI (International Conference on Internet) 2010, December 16-20, 2010, Philippines. This work was supported by Mid-career Researcher Program through NRF grant funded by the MEST (No. R01-2008-000-20485-0)

DOI: 10.3837/tiis.2011.10.010

1. Introduction

Service-Oriented Architecture (SOA) is a special component model which exposed the collaborating scheme among applications (i.e. services) through well-formed contracts and interfaces [1]. A service-oriented software system provides its functionality by the composition of service components scattered on a network, such as the Internet, for satisfying complex business requirements [2]. To specify the service collaboration scheme called service composition, most SOA approaches use the WS-BPEL (Web Service-Business Process Execution Language) which can specify new service with the composition of existing service components [3]. The WS-BPEL defines the calling sequence among services with high-level interface signature to specify a new service. It allows a enterprise to build quickly or adapt new business services by coordinating different web services.

The use of WS-BPEL can expedite the implementation of web service compositions. Although it provides simple and effective specification of the service composition, incorrect specification of WS-BPEL can cause the wrong behaviors of the service. The WS-BPEL specification for a complex business process including a number of services can be difficult to understand and be inherently error-prone [4]. Therefore the service specification written in WS-BPEL must be investigated whether service requirements were met or not, in order to ensure the correctness of composite service. To test the composite service, the existing testing techniques based on the WS-BPEL have been focused on the functional and structural features such as invalid input or output data, composition process validation and service invocation failure. As the SOA approach has been applying to the development of various software domains, the performance testing of composite service including real-time transaction processing becomes one of interesting research issues [5][6][7].

This paper suggests an approach to test the performance of composite service through the extension of WS-BPEL. This extension was made with the application of aspect-oriented programming concept [8]. Using the extension, we are able to add an aspect component to WS-BPEL specification as one of crosscut concerns for service performance testing. To test the performance of composite service, in this paper, we develop the aspect component for measuring the response time of composite service and define a performance testing procedure. We also conduct experiments with on-line transaction system to evaluate the applicability of our approach. By using the aspect component which can provide single-code and multiple-use principle, our proposed technique can reduce the effort for testing readiness and the time for test code development compared with the existing studies.

This paper is organized as follows; Section 2 describes the background and related work. Section 3 explains about what to and how to extend the WS-BPEL, and we propose our performance testing approach in section 4. We also conduct the implementation and experiment of our technique in section 5. Finally, section 6 concludes our paper and briefly discusses the future work.

2. Related Work

2.1 Service Composition using WS-BPEL

WS-BPEL can specify the control logic for orchestrating multiple web services participanting in a process flow by making reference to the web service description language (WSDL)

description of a service. The (a) and (b) in **Fig. 1** show the structures of process definition using WSDL and WS-BPEL, respectively [1][3][9].

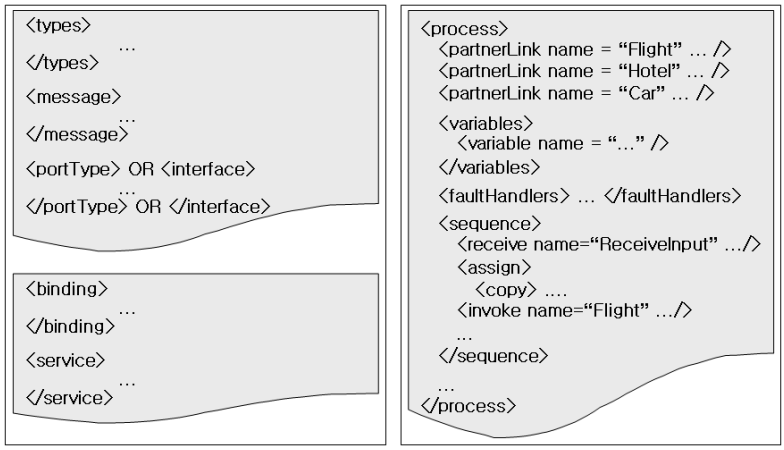

Fig. 1. Structures of WSDL and WS-BPEL

The information of a service is defined in WSDL specification. As shown in **Fig. 1**.(a), the upper part describes the type, transmit or receive message, interface, etc of a service and the lower part describes the name, physical location, and communicating protocol for the service. The <partnerLink> element of WS-BPEL, shown in **Fig. 1**.(b) defines the service participating in the composition by reference to WSDL specification, and the <sequence> element specifies a control logic defining the execution flow of participant services. More detail syntax and their meaning can be referred in WS-BPEL standard [3]. This WS-BPEL specification represents the configuration of service compositions, thus the specification is needed to validate whether it satisfies the requirements of the composite service or not.

2.2 Web Service Testing

As SOA is recently realized in web service area, there is an increasing concern to test web services. Especially, substantial efforts have been consumed on testing a single web service and composite service. Most researches for testing a single web service use the information of WSDL specification, such as interface types, message types and operations, to generate test cases for ensuring the correctness of the service [10][11]. The representative study on the WSDL-based testing technique is performed by Tsai [12]. In order to test a web service, he extended the WSDL features to specify the useful information for testing, such as input-output dependency, invocation sequence, hierarchical functional description and concurrent sequence specification. Other similar studies about web service testing have proposed as a test framework to facilitate web service testing [13][14].

One of the WS-BPEL based testing for composite service is performed by Lertphumpanya [15]. He developed a testing framework which identifies whole workflow paths for composite service, and performed coverage testing for each workflow path. Li [16] proposed a WS-BPEL unit test framework to facilitate the testing and debugging of WS-BPEL processes. As the framework includes a composition model, a test architecture, a lifecycle management schema, a test design outline and a test coverage criterion, it provided the infrastructure for unit testing tools based on WS-BPEL. Yean [17] described a graphical model, called BPEL

Flow Graph (BFG), to represent a WS-BPEL process in a graphical flow. By traversing the BFG, concurrent test paths and test data for each path can be generated using a constraint solving method. Cao [18] proposed a WS-BPEL verification framework. In the framework, WS-BPEL process is modeled using UML activity diagram and the diagram is verified by a model checker. However these researches have been only focused on the testing of service functionalities without considering the performance of composite service.

OASIS TAG (Test Assertions Guidelines) [19] guides to specify a test assertion with the goal of improving the testability for functional conformance. The guidelines provide the design and description methods to develop test assertions via examples. OASIS XTemp (XML Testing and Event-driven Monitoring of Processes) [20] is a specification of XML-based script language which is used to analyze, test, and monitor the service processes or transactions between business partners. These specification stantards focused on how to design and describe test suites or test cases for a formal and processable testing. However our research works address to the technique and the process of performance testing for WS-BPEL-based composite service.

2.3 Performance Analysis of Web Service

There have been several studies that address the performance issue from web service domain. Peng [21] proposed a grid service monitoring framework for performance monitoring on various grid services. He implemented it for monitoring the resource of services and shown how the framework work with a simple example service. Bertolino [22] proposed an approach to assess the performance properties for stand-alone services in the PLASTIC project. In Chandrasekaran [23], a service composition and execution tool for evaluating the performance of a web service is proposed. The tool provides a technique for execution time analysis and execution monitoring that can be used to evaluate the performance of individual web services by the simulation of service execution.

Almost existing techniques for web service-based performance testing simulate the services in test framework which inserts the script code to monitor and estimate the execution time of each service. But these performance testing techniques require lots of efforts to develop the monitoring codes for every service application, and to develop the test framework to control the code execution. Also these techniques are inadepuate to test the real performance because of simulation-based estimation. Therefore it needs an effective and inexpensive testing technique for the performance of composite service in real situation.

3. Extension of WS-BPEL

Our proposed approach tests the performance of composite service with the aspect-oriented concept when a WS-BPEL process is executed. The performance testing is considered to be the common concern of aspect-oriented concept. **Fig. 2** shows the overview of our testing approach using WS-BPEL specification and aspect component.

The aspect component, shown in **Fig. 2**, has the attribute “PointCut” and two operations, “BeforeAdvice” and “AfterAdvice.” The “PointCut” is used to pinpoint the service for the performance testing in a WS-BPEL process. Whenever an interested service invocation occurs during the execution of the WS-BPEL process, the advice of the aspect component will be executed. The “BeforeAdvice” calls the function “StartTimer()” of the timer service before the operation of invoked service is executed for measuring the response time of the service. After

the invoked service operation was executed, the "AfterAdvice" of the aspect component calls "StopTimer()" function. The measured response time is the test result of performance.

To implement our performance testing approach, it is necessary to specify the features of aspect using WS-BPEL grammar. It means that the meta-model of aspect component has to be integrated with the meta-model of WS-BPEL to support the weaving mechanism. Therefore we add features of aspect component such as "Aspect", "PointCut", and "Advice" to the meta-model of WS-BPEL with the purpose of estimating the response time for composite service.

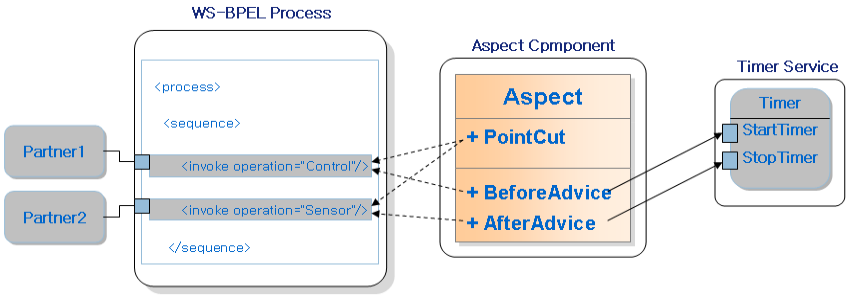

Fig. 2. Our Performance Testing Approach

3.1 Relationship of Aspect and WS-BPEL

To indentify aspect elements required for the extension of WS-BPEL, we first examined relationships between the features of aspect and WS-BPEL meta-model. The meta-model of aspect component is shown in **Fig. 3**. An aspect modularly represents a concern that crosscuts multiple classes or features. As shown in **Fig. 3**, the aspect consists of pointcuts and advice. The pointcut expresses join points, which is a well-defined point to combine the aspect with the execution of a program. The pointcut expression is a kind of expression statement that selects a set of join points and specifies which join points will be selected. The advice performs the specified codes which implement crosscutting concerns whenever a join point selected by the pointcut expression is reached during execution. The advice type can determine the specific time point when the advice is executed.

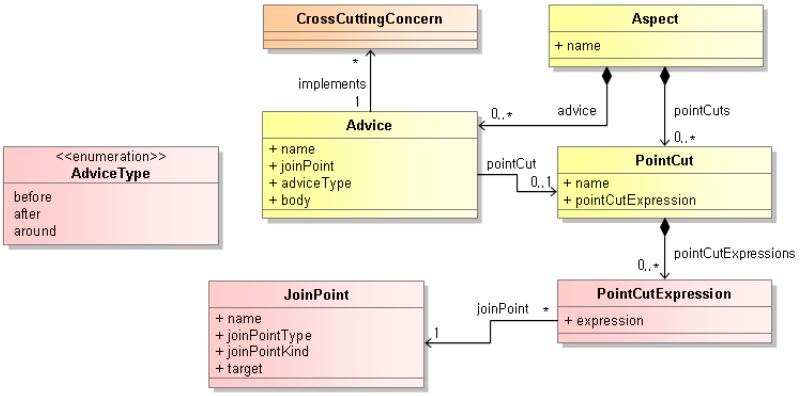

Fig. 3. Meta-model of Aspect

The informaion of WS-BPEL meta-model is referred in WS-BPEL standard [3]. The

relationships of aspect and WS-BPEL in meta-model level are summarized in **Table 1**.

Table 1. Relationship between Aspect and WS-BPEL Meta-model

Aspect meta-model	WS-BPEL meta-model	Relationship
Aspect	ActivityContainer	Packaging behavioral logic or execution flow
PointCut	-	-
Advice	ExtensionActivity	Representing additional behavior
AdviceType	-	-
JoinPoint	Invoke	Invoke is regarded as join point in process
PointCutExpression	Expression	Expression can specify pointcuts

In **Table 1**, the ActivityContainer of WS-BPEL meta-model can represent packaging behavioral logics or execution flows like the aspect, and the use of the ExtensionActivity is similar to the role of advice. The Invoke element of WS-BPEL can be regarded as the JoinPoint during a process execution. So the position of the Invoke can be specified using the PoinCutExpression, which is similar to the Expression element of WS-BPEL. The Expression can be used to combine the PointCuts that select the Invoke because it provides the several operators of XPath [24] such as union operator and intersect operator. In this manner, we can identify the elements of aspect required to WS-BPEL extension. For these identified elements, we perform their extensions as the following section.

3.2 Extending with the Aspect

The class "ActivityContainer" in WS-BPEL meta-model is a basic element to package the activities within the WS-BPEL process for web service composition, and our aspect component can also package the activities to perform the performance testing. Thus we define the aspect component as a subclass of the class "ActivityContainer". This definition can be formalized as the following First Order Logic (FOL) [25] predicate:

$$\forall x.Aspect(x) \supset ActivityContainer(x) \tag{1}$$

where *x* is an instance of a class, $Aspect$ and $ActivityContainer$ of the predicate (1) are the class of "Aspect" and "ActivityContainer" from their meta-model, respectively. Note that all properties of $Aspect$ class must include all one of $ActivityContainer$ class.

The class "Aspect" has aggregation relationships with the class "PartnerLink" to represent the access information of participant service at the interface level, and with the class "Variable" to define the input or output data of the service. The aggregation relationship of the class "PartnerLink" can be expressed in FOL as follows:

$$\forall x, y.partnerLinks(x, y) \supset Aspect(x) \wedge PartnerLink(y) \tag{2}$$

where the first argument $Aspect(x)$ of the predicate (2) is the containing class and $PartnerLink(y)$ is contained class.

$$\forall x.PartnerLink(x) \supset (0 \leq \#\{y \mid partnerLinks(x, y)\} \leq 1) \tag{3}$$

$$\forall y.Aspect(y) \supset (1 \leq \#\{x \mid partnerLinks(x, y)\}) \tag{4}$$

where these predicates express cardinality restrictions in the aggregation relationship. The predicate (3) and (4) Formalize the multiplicity [0..*] and [1..1], respectively.

Similarly, the aggregation relationship of the class "Variable" has to satisfy the following predicates:

$$\forall x, y. \text{var}\, iables(x, y) \supset Aspect(x) \wedge Variable(y) \quad (5)$$

$$\forall x. Variable(x) \supset (0 \leq \#\{y \mid \text{var}\, iables(x, y)\} \leq 1) \quad (6)$$

$$\forall y. Aspect(y) \supset (1 \leq \#\{x \mid \text{var}\, iables(x, y)\}) \quad (7)$$

The extension of aspect feature in WS-BPEL meta-model by these predicates is shown in **Fig. 4**. The class "Aspect" has the attributes of "type" defining a kind of services, and "PointCutExpressionLanguage" defining pointcuts for measuring the response time of services. The specification of this extension in a WS-BPEL process is shown in the right side of **Fig. 4**.

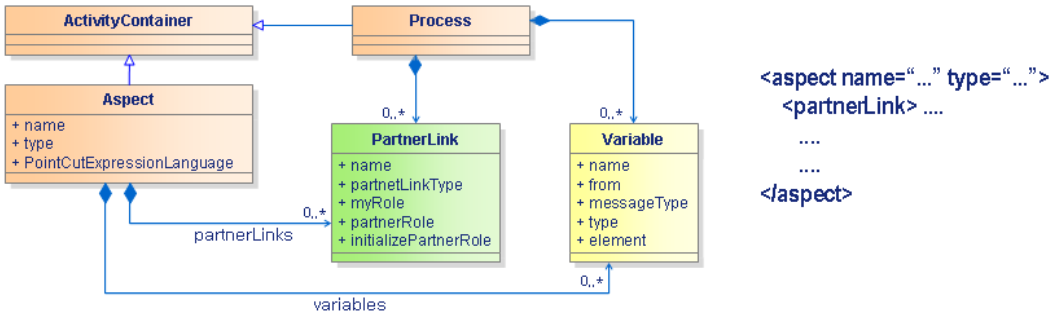

Fig. 4. Extension of WS-BPEL with Aspect

3.3 Extending with the PointCut

The "PointCut" element uses to select the target services for performance testing. Aspect component can have several "PointCut" elements and their designators. Thus these elements must be related with the class "Aspect" in the meta-model. The definition for extending with the "PointCut" element is the following predicates:

$$\forall x, y. po\,\text{int}\, Cuts(x, y) \supset Aspect(x) \wedge Po\,\text{int}\, Cut(y) \quad (8)$$

$$\forall x. Po\,\text{int}\, Cut(x) \supset (1 \leq \#\{y \mid po\,\text{int}\, Cuts(x, y)\} \leq 1) \quad (9)$$

$$\forall y. Aspect(y) \supset (1 \leq \#\{x \mid po\,\text{int}\, Cuts(x, y)\}) \quad (10)$$

where a binary relation between class "Aspect" and class "PointCut", denoting a part-whole relationship, is expressed in (8) and the multiplicity of the relation is predicated in (9) and (10).

$$\forall x. Po\,\text{int}\, CutExpression(x) \supset Expression(x) \quad (11)$$

where each attributes or operations of the class "Expression" are inherited by the class "PointCutExpression" of the aspect meta-model.

Fig. 5 shows the extension for the "PointCut" element. The attribute "PointCutExpressionLanguage" of the class "PointCut" defines the expression to describe the pointcut designator in aspect-oriented approach.

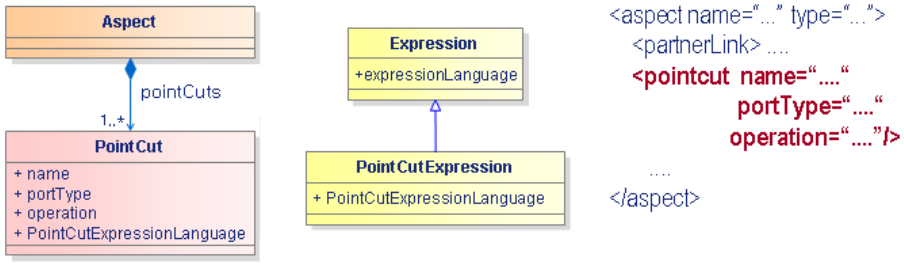

Fig. 5. Extension of WS-BPEL with PointCut

3.4 Extending with the Advice

The "Advice" element uses to call the service defined in the class "PatnerLink". In our testing approach, the advice calls the timer service. In the WS-BPEL specification, the class "ExtensionActivity" can be used to include the additional activities in WS-BPEL process. Thus the class "Advice" is defined as a subclass which is inheriting its properties from the class "ExtensionActivity". And several "Advice" classes can be aggregated with the class "Aspect". The extension of the advice is captured by means of the following FOL predicates:

$$\forall x, y.advice(x, y) \supset Aspect(x) \wedge Advice(y) \quad (12)$$

$$\forall x.Advice(x) \supset (1 \leq \#\{y \mid advice(x, y)\} \leq 1) \quad (13)$$

$$\forall y.Aspect(y) \supset (1 \leq \#\{x \mid advice(x, y)\}) \quad (14)$$

$$\forall x.Advice(x) \supset ExtensionActivity(x) \quad (15)$$

The extension of WS-BPEL meta-model for the above predicates is represented in **Fig. 6**.

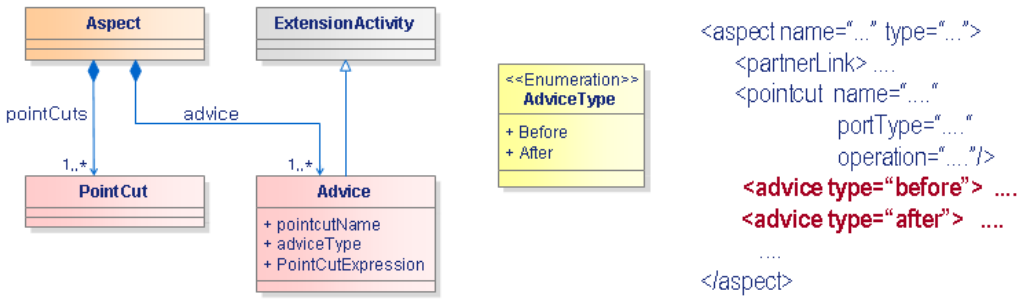

Fig. 6. Extension of WS-BPEL with Advice

Each advice executes the service according to the "Advice type" – either "Before" or "After." It means that the function "StartTimer()" or "StopTimer()" of timer service begins its operation before or after the execution of composite service. The extended meta-model of WS-BPEL which consolidate the all features of the aspect component, is depicted in **Fig. 7**.

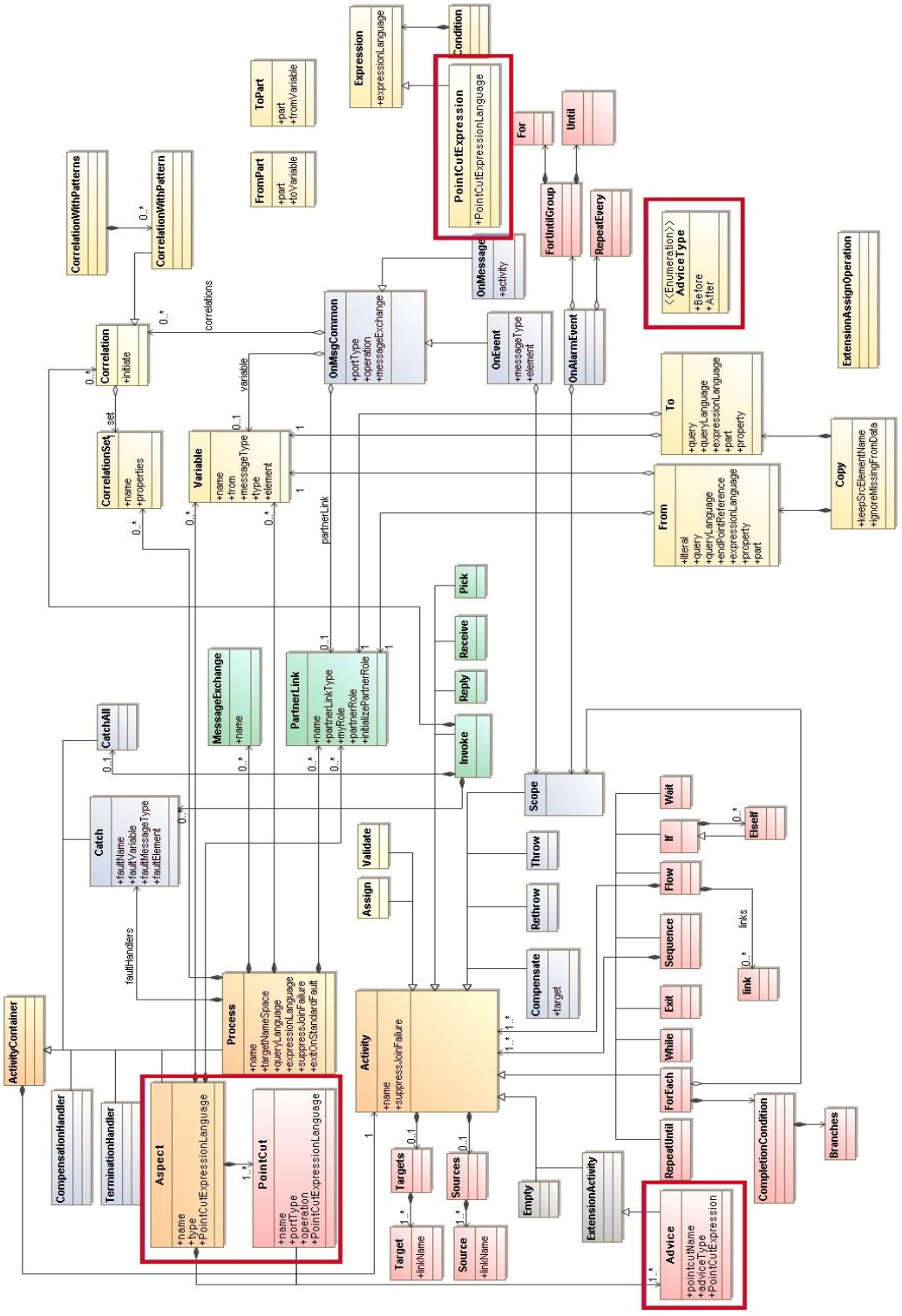

Fig. 7. Extended WS-BPEL Meta-model

4. Performance Testing

When a WS-BPEL specification for the testing is given, our proposed technique for performance testing is performed in following steps.

(1) Develop timer service and its WSDL specification. As the participant service of WS-BPEL process, the timer service measures the response time of composite service.
(2) Specify an aspect component for the invocation of the timer service. The aspect component describes XML-based statements similar to existing WS-BPEL to invoke the operation of the timer service.
(3) Weave the aspect component into WS-BPEL specification. According to the weaving mechanism of aspect-oriented concept, the aspect and WS-BPEL specification can be integrated by the join point of the aspect component. The integrated specification is a testable WS-BPEL specification that can test the performance of composite service.
(4) Test the performance of composite service. The results of performance testing are aquired through the execution of the testable WS-BPEL specification.

4.1 Develop Timer Service and WSDL Spec.

The component "Timer Service" measures the elapsed time for service execution. To record the elapsed time, a variable of list construct must be defined. The following is the skeleton code of the timer controller developed for the timer service.

```
Component TimerController(int Svc_id, int xy) {
    Private int xy;
    Public List time;
        Public int SetTimer();
        Public void StartTimer();
        Public void StopTimer();
        Public void SaveTime();
        Public void Timeout();
}
```

The WSDL specification of the timer controller is depicted in **Fig. 8**.

```
<definitions targetNamespace="http://TS/"
name="TimerService" … >
  <types> …</types>
  <message name="SetTimer">
      <part name="SetTimerVar" element="tns:StartTimer"/>
  </message>
    …
  <message name="Timeout">
      <part name="TimeoutVar"/>
  </message>

  <portType name="Timer">
      <operation name="SetTimer">
          <input message="tns:SetTimer"/>
          <output message="tns:SetTimerResponse"/>
      </operation>
```

Fig. 8. WSDL Specification of Timer Service

4.2 Develop the Aspect Component

We should develop the aspect component which can invoke a timer service to test the

performance of composite service. The component is specified using the extened WS-BPEL. The following steps are required to develop the component.

(1) We first define the timer service as a participant service using the <partnerLink> element.
(2) We identify the join points for the target service within the WS-BPEL specification. These join points can correspond to the basic activities such as receive, invoke, and reply within the WS-BPEL specification.
(3) The identified joint points are specified as <pointcut> elements in the component. The property "operation" of the <pointcut> element assigns a location of the target service to test the performance.
(4) To invoke the timer service, the invocation statements are defined using the <advice> element. The type "before" calls the operation "StartTimer" of the timer service before the execution of target service. And the type "after" calls the operation "StopTimer" after the service execution.

The developed aspect component is shown in **Fig. 9**. The property "type" of the aspect element is to represent whether the target service is composite or single.

```
<process ...> ...
  <aspect name="Aspect" type="composition" ... />
    <partnerLink name="TimerController" />
    <pointcut name="..."
              operation="..." />
    <advice type ="before" pointcutName= "...">
          <invoke partnerLink="TimerController"/
                  operation="StartTimer" ... /></advice>
    <advice type ="after" pointcutName="...">
          <invoke partnerLink="TimerController"/
                  operation="StopTimer" ... /></advice>
  </aspect> ...
</process>
```

Fig. 9. Aspect Component Specification for Performance Testing

4.3 Weave the Aspect with WS-BPEL

Fig. 10 shows the weaving process between aspect component specification and WS-BPEL process specification. The specification of WS-BPEL process and aspect component are parsed with XML parser. After the parsing is finished, the corresponding elements between two parsed results are identified with comparing of the values of the elements. With these corresponding relationships, the weaving engine proceeds their weaving to generate a testable WS-BPEL specification using the extended WS-BPEL. The woven specification will contain both the XML fragments for service composition and for performance testing.

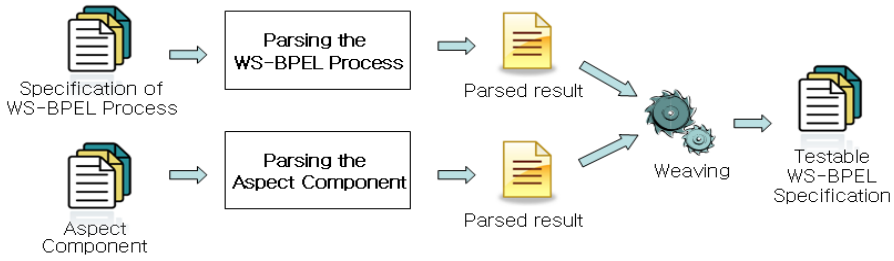

Fig. 10. Weaving Process of WS-BPEL and Aspect

4.4 Testing the Performance

In order to perform the test, the test driver is required to handle the client's request. At first, the test driver runs in the client's machine, and generates a message for client's service request, and then sends the message to WS-BPEL engine of SOA framework. The testable WS-BPEL specification is processed to test the service performance by WS- BPEL engine, as shown in **Fig. 11**.

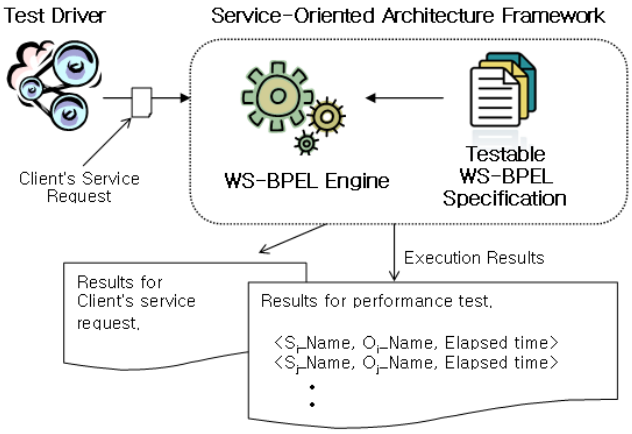

Fig. 11. Procedure of Performance Testing Execution

One of the execution results of the testable WS-BPEL specification is a list of the execution times measured for each single service or several services. Another results are responses for the service request of the client. The collected results will be stored in a file for all service invocation specified in WS-BPEL process. Using the results, we can calculate the performance of composite service, CSP by the reponse time in client's view as follow:

$$CSP = \sum_{i=1}^{n} \sum_{j=1}^{k} OT_{ij} \tag{16}$$

where i is a participant service and j is an invoked operation within a service. OT_{ij} is a response time for operation j of service i. Note that the total response time of services participating at service composition of the WS-BPEL means the performance of the composite service.

5. Implementation and Experiment

As a proof-of-concept, we conduct the implementation and experiment of our approach for on-line transaction processing system.

5.1 Tool Implementation

To implement our proposed technique for the performance testing of composite service, we need to develop an aspect descripter which can generate the aspect component to measure the execution time of a service, and develop a weaving engine that can generate a testable WS-BPEL specification. It also needs a WS-BPEL engine to execute the testable WS-BPEL specification and to get the performance result from the execution. Therefore we implemented a performance test tool using the Netbeans IDE 6.7 [26] which provides main functionalities for SOA, such as WS-BPEL engine, WS-BPEL designer, and web service developer. The screen shot of our tool implementation is shown in **Fig. 12**.

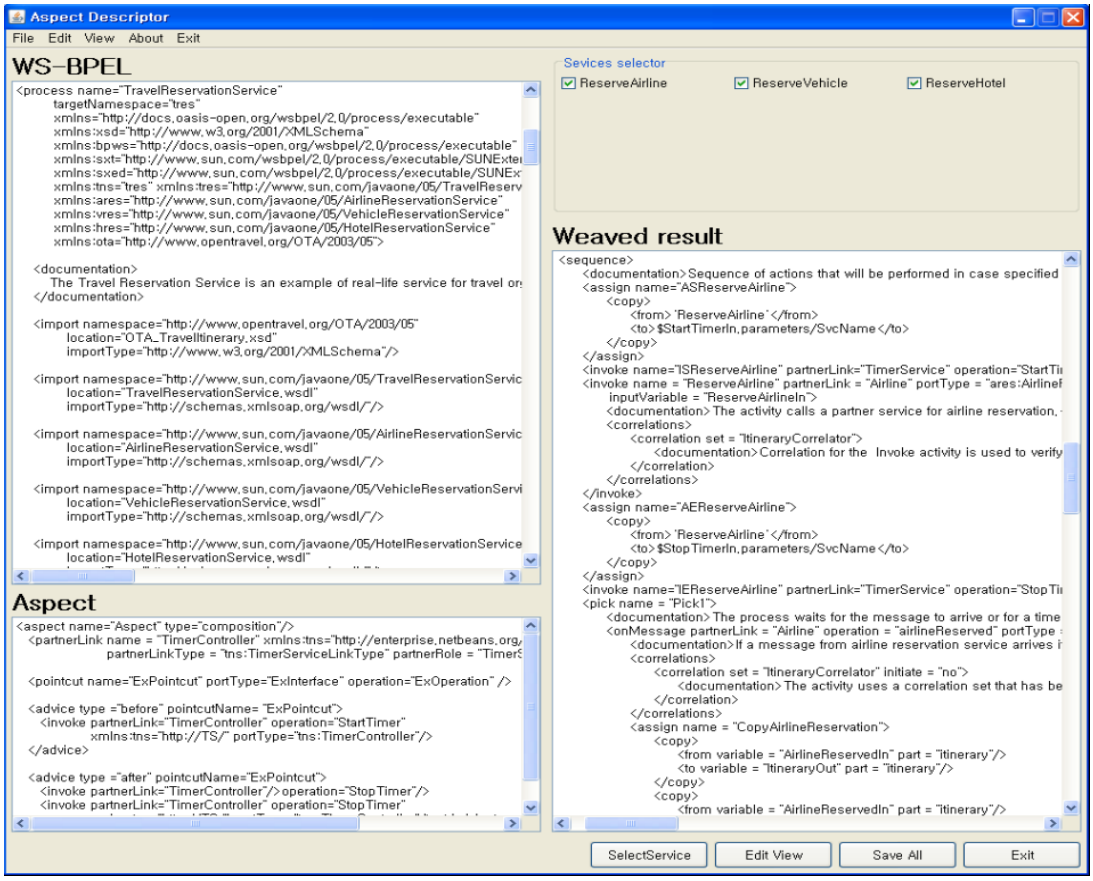

Fig. 12. Screen Shot of Aspect Descripter

From **Fig. 12**, the aspect descripter has three parts of WS-BPEL displayer, aspect component editer and weaved result displayer. The weaved result is generated by the weaving engine which is implemented within the aspect descripter.

5.2 Experiment Environment

The experiment is performed with travel reservation system (TRS) [27], as shown in **Fig. 13**. TRS, which provides total service for a travel, consists of flight reservation service, hotel reservation service and car rental service.

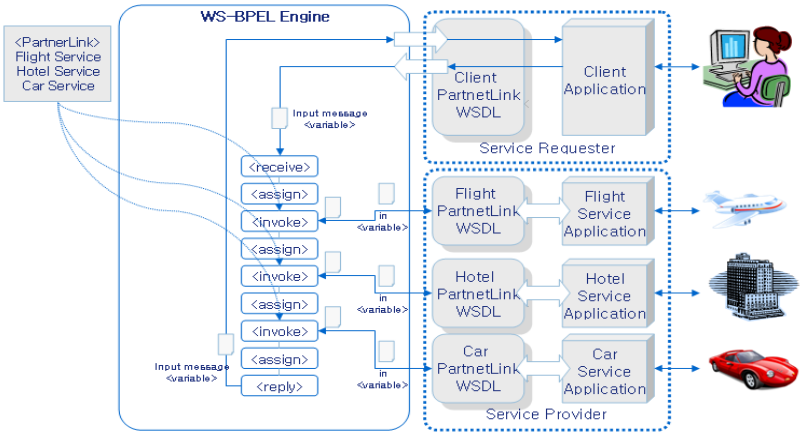

Fig. 13. Experimental construction for TRS

If a client sends a service request to the WS-BPEL engine, the three services are composited according to the WS-BPEL process to provide the integrated travel service. The specification and execution of WS-BPEL for the TRS are achieved by the Netbeans IDE 6.7, as shown in **Fig. 14**.

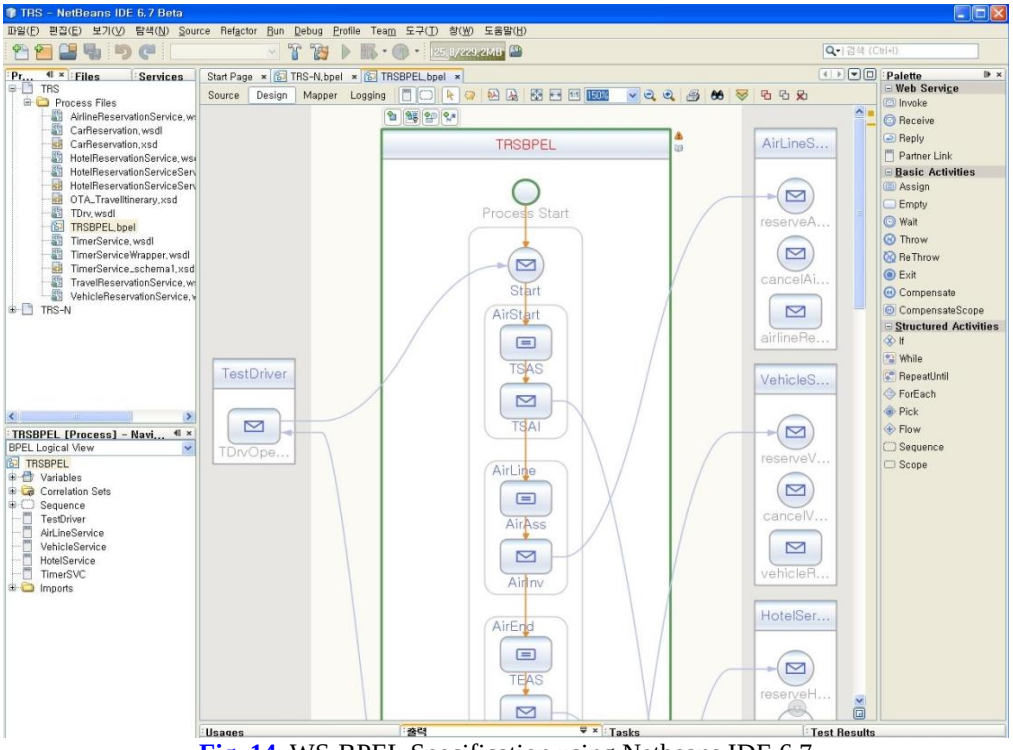

Fig. 14. WS-BPEL Specification using Netbeans IDE 6.7

The system configuration for our experimental environment is summarized in **Table 2**. The TRS that consisted of three services are installed at service provider and the Netbeans IDE is built at SOA server.

Table 2. Hardware Configuration for Experiment

	Service Client	Service Provider	SOA Server
Processor	Intel Pentium 4 (3.0GHz)	Intel Pentium 4 (3.0GHz)	Intel Quad Core (2.5GHz)
Memory	1GB	2GB	4GB
Java	Java 2 SE	Java 2 EE	Java 2 EE
Bandwidth	100Mbps	100Mbps	100Mbps
SOAP Engine	Apache Axis 2	Apache Axis 2	Apache Axis 2

5.3 Experiment 1: Performance Test of the TRS

The experiment in this section conducts the performance test of the TRS. The experimental results is captured after the performance test of 10 times for each service as illustrated in **Fig. 15**. From **Fig. 15**, the totals of average elapsed times for the composite three services are 1674.8ms. The response time of the car rental service takes much more than the time of other services. This results show that the car rental service is inefficient in the view of performance. The inefficiency may be caused by the high complexity of service implementation. The fluctuation in the response times of car rental service occurred from the service execution status. High response time of car rental service was caused by normal booking which meets the customer requirements, and low response time was caused by normal execution but booking was not succeeded. Thus the car rental service needs to be substituted to a performance-efficient service with identical functions.

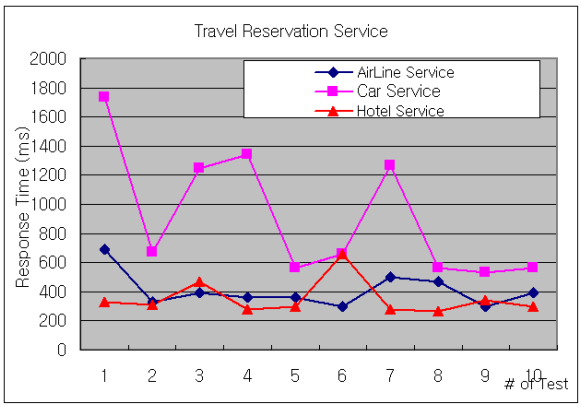

Fig. 15. Performance Test Result for TRS

5.4 Experiment 2: Performance Test for Variation of the TRS

As the worst performance of the car rental service in the experiment 1, we implemented a TBS (Travel Booking System) which is the variation of TRS, and then performed a testing with the TBS. The TBS shared the flight reservation(airline) service and hotel reservation service of the TRS, and its car rental service is altered with a vehicle rental service [28] that is another service application with identical functionalities of the car rental service. **Fig. 16** shows the results of the performance test for the TBS. From Fig. 16, the response times of airline service increase in some degree than the time of the experiment 1. This phenomenon can be guessed from that multiple service requests are occurred.

Fig. 16. Performance Test Result for TBS

The totals of average elapsed times for the TBS are 1058.0ms when the experiments were performed in similar environments of the TRS. The response time of the vehicle service, shown in **Fig. 16**, takes less than the time of car rental service in the TRS. So, we can composite the travel services in the same manner with TBS for performance-efficient service.

5.5 Experiment 3: Performance Test of the Service Load by Requests

In this experiment we conducted the performance testing with consideration of the service load by client requests, and compared the performances between the TRS and the TBS by the increment of the load. The load is increased by augmenting the number of clients (from 1 to 30 clients). Each client sends a request to a service and as soon as it gets the reply, it submits a new request again. The test results for the loads of the TRS and TBS is summarized in **Table. 3**. The response time is measured by the test of the composite serivce for one request. From **Table 3**, we can confirm that the performance of the TBS is much better than the TRS as increasing the requests of the compsite service. The responset time for two composite services is graphically represented in **Fig. 17**. We can still show that the TBS is performance-efficient service for the increment of the service load.

Table 3. Test Result by Service Load Increment

Load (# of clients)	Response Time of TRS (ms)	Response Time of TBS (ms)	Difference (ms)
1	1710.32	1046.19	664.13
5	1820.12	1145.36	674.76
10	1910.95	1330.29	580.66
15	2120.96	1490.29	630.67
20	2290.56	1625.16	665.40
25	2510.78	1880.89	629.89
30	2820.92	1998.53	822.39

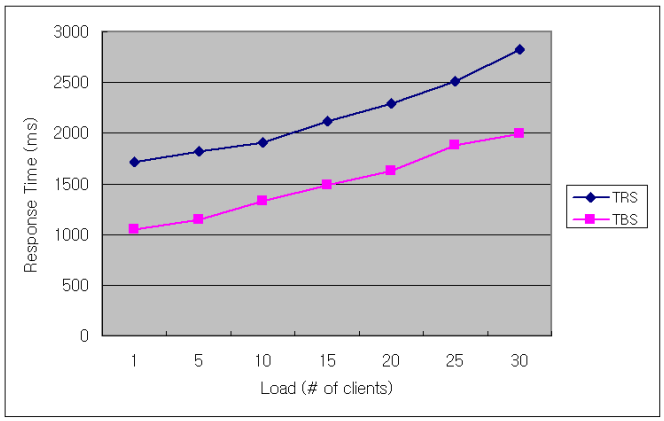

Fig. 17. Performance Test Result for Service Load

5.6 Experiment 4: Performance Test of Parallel Service Invocation

The flight reservation service and vehicle rental service of the TBS can be invoked separately, regardless of invocation sequence because those do not have any effect each other. In this experiment we perform the performance testing for the parallel invocation of the two services. The two services are invoked at the same time and then the hotel reservation service is invoked. The response time of the parallel invocation is estimated at a point that is received the replies of two service. **Fig. 18** shows the test result of this experiment. As one expects the performance improvement of parallel execution, the composite service integrated with the three services yields the good response time.

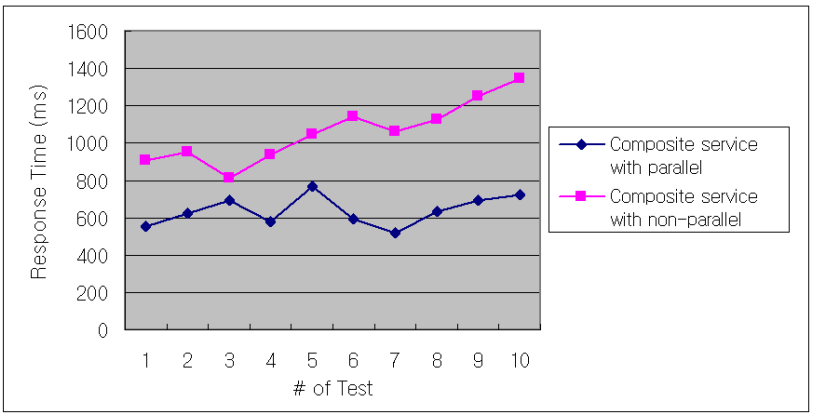

Fig. 18. Performance Test Result for Parallel Service Invocation

5.7 Discussions: Results from the Experiments

As noted in the previous sections, we performed various experiments to test the performance of the travel reservation service. In the experiment 1 and 2, we can see that the composite service can have a low performance due to the car rental service with a bad performance and the service needs to be replaced to other identical service, the vehicle rental service, with a good performance. So the results of the experiments can be used to select a service which can improve the performance of composite service.

In the experiment 3, the critical response time defined in Service-Level Agreement (SLA) for the TRS and TBS is 2000ms. The TRS violates the response time of SLA from the load of 15 clients. The TBS always satisfies the response time about the load from 1 client to 30 client. As a result, for selecting a participant service in a WS-BPEL process, we should consider whether the response time of SLA is satisfied by the increment of the service loads.

In the result of the experiment 4, we can show that the performance of the composite service was improved by the parallel invocation of WS-BPEL process. In the WS-BPEL process, the two services are invoked in parallel, as shown in **Fig. 19**.

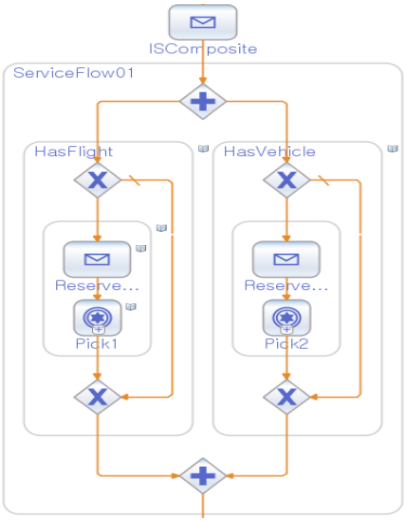

Fig. 19. Parallel Service Invocation of TBS

The parallel invocation bring out the the performance improvement of compsite service, but it can waste the computing resource for parallel processing when the processing does not finished by the failure of service invocation. From **Fig. 19**, even if the invocation of the flight reservation service is fail, the vehicle reservation service can be still executing. The result of the processing is invalid. That is, the vehicle rental service is no longer applicable. Therefore the parallel invocation can cause resource waste due to unnecessary works.

Results of these tests can be used to optimize WS-BPEL processes and SLA, and to discover potential performance bottlenecks. In our experiments, the timer service can cause the overhead in the response time measurement. However, the overhead can be ignored because the timer service is processed in the local host which handles the composition of services and occurs in very low delay (less than 4ms).

6. Conclusion

The performance issue of real-time transaction system is one of major concerns in software development based on SOA approach. The existing researches about WS-BPEL have been focused on the functional or structural features of a service process. Although WS-BPEL specifies the control logic of behavioral process for service composition, it does not support the performance issue in its specification.

Therefore, in this paper, we extend the meta-model of WS-BPEL using the aspect-oriented paradigm for considering the performance property in WS-BPEL process. And also we

proposed the performance test procedure to explain how to develop the aspect-based WS-BPEL spec. and perform the performance testing. To support the test procedure, we implemented the aspect descriptor and weaving engine to generate a testable WS-BPEL specification. Finally, we conducted experiments on the TRS, on-line travel reservation system, to evaluate the applicability of our approach. Our technique can apply to choose an adequate service among various candidate service components with considering the performance. One of further works is applying our technique to mobile smart-phone applications, and developing the performance testing framework to support service composition among smart devices.

References

[1] T. Erl, "Service-Oriented Architecture: Concepts, Technology, and Design," Prentice-Hall, 2005.

[2] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Computer*, vol. 40, no. 11, pp. 38-45, Nov. 2007. Article (CrossRef Link)

[3] OASIS Standard, "WS-BPEL Ver. 2.0," Apr. 2007.

[4] C.H. Liu, S.L. Chen, X.Y. Li, "A WS-BPEL Based Structural Testing Approach for Web Service Compositions," in *Proc. of IEEE Int. Symp. on Service-Oriented System Engineering*, pp. 135-141, Dec. 2008. Article (CrossRef Link)

[5] E. Zeeb, A. Bobek, H. Bohn, S. Pruter, A. Pohl, H. Krumm, I. Luck, F. Golatowski, D. Timmermann, "WS4D: SOA-Toolkits making embedded systems ready for Web Services," in *Proc. of Int. Workshop on Open Source Software and Product Lines*, pp. 33-42, 2007. Article (CrossRef Link)

[6] Y. Kakumoto, H. Terada, S. Sekino, N. Komoda, "Component Oriented Software Framework for Train Car Systems," in *Proc. of Int. Conf. on Computational Intelligence for Modelling, Control and Automation*, pp. 587-593, Nov. 2005. Article (CrossRef Link)

[7] N. Komoda, "Service Oriented Architecture (SOA) in Industrial Systems," in *Proc. of IEEE int. Conf. on Industrial Informatics*, pp. 1-5, Aug. 2006. Article (CrossRef Link)

[8] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C.V. Lopes, J.M. Loingtier, J. Irwin, "Aspect-Oriented Programming," in *Proc. of European Conf. on Object-Oriented Programming*, pp. 220-242, 1997. Article (CrossRef Link)

[9] W3C, "WSDL Ver. 2.0," TR-REC-wsdl20-primer-20070626, 2007.

[10] J. Offutt, W. Xu, "Generating Test Cases for Web Services Using Data Perturbation," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 5, pp. 1-10, Sep. 2004. Article (CrossRef Link)

[11] X. Bai, W. Dong, W.T. Tsai, Y. Chen, "WSDL-Based Automatic Test Case Generation for Web Services Testing," in *Proc. of IEEE Int. Workshop on Service-Oriented System Engineering*, pp. 207-213, October, 2005. Article (CrossRef Link)

[12] W.T. Tsai, R. Paul, Y. Wang, C. Fan, D. Wang, "Extending WSDL to Facilitate Web Services Testing," in *Proc. of IEEE Int. Symp. on High Assurance Systems Engineering*, pp. 171-172, 2002. Article (CrossRef Link)

[13] W.T. Tsai, R. Paul, L. Yu, A. Saimi, Z. Cao, "Scenario-Based Web Service Testing with Distributed Agents," *IEICE Transaction on Information and System*, vol. E86-D, no. 10, pp. 2130-2144, 2003.

[14] H. Mei, L. Zhang, "A framework for testing Web services and its supporting tool," in *Proc. of IEEE Int. Workshop on Service-Oriented System Engineering*, pp. 207-214, 2005. Article (CrossRef Link)

[15] T. Lertphumpanya, T. Senivongse, "A basis path testing framework for WS-BPEL composite services," in *Proc. of Int. Conf. on Software Engineering, Parallel and Distributed Systems*, pp. 107-112, 2008. Article (CrossRef Link)

[16] Z. Li, W. Sun, Z. B. Jiang and X. Zhang, "BPEL4WS Unit Testing: Framework and Implementation," in *Proc. of IEEE Int. Conf. on Web Services*, pp. 103-110, July 2005. Article

(CrossRef Link)
[17] Y. Yuan, Z. Li, W. Sun, "A Graph-Search Based Approach to BPEL4WS Test Generation," in *Proc. of IEEE Int. Conf. on Software Engineering Advances*, pp. 14-14, Oct. 2006. Article (CrossRef Link)
[18] H. Cao, S. Ying, D. Du, "Towards Model-based Verification of BPEL with Model Checking," in *Proc. of 6th IEEE Int. Conf. on Computer and Information Technology*, pp. 190-190, Sep. 2006. Article (CrossRef Link)
[19] OASIS Standard, "Test Assertions Guidelines Version 1.0," June 2011.
[20] OASIS Standard, "XTemp: XML Testing and Event-driven Monitoring of Processes Version 1.0," Mar. 2011.
[21] L. Peng, M. Koh, J. Song, S. See, "Performance Monitoring for Distributed Service Oriented Grid Architecture," *Lecture Note in Computer Science*, vol. 3719, pp. 351-356, 2005. Article (CrossRef Link).
[22] A. Bertolino, G.D. Angelis, A.D. Marco, P. Inverardi, A. Sabetta, M. Tivoli, "A Framework for Analyzing and Testing the Performance of Software Services," *Communications in Computer and Information Science*, vol. 17, no. 6, pp. 206-220, 2009. Article (CrossRef Link).
[23] S. Chandrasekaran, J.A. Miller, G.S. Silver, B. Arpinar, A.P. Sheth, "Performance Analysis and Simulation of Composite Web Services," *Journal of Electronic Markets*, vol. 13, no. 2, pp. 120-132, Jan. 2009.Article (CrossRef Link).
[24] J. Clark, "XML path language (XPath) 2.0," December, 2010.
[25] D. Berardi, D. Calvanese, G.D. Giacomo, "Reasoning on UML class diagrams," *Artificial Intelligence*, vol. 168, pp. 70-118, July 2005. Article (CrossRef Link).
[26] Oracle Corporation, "Netbeans IDE 6.7," 2010.
[27] A. Koval, "Understanding the Travel Reservation Service," July, 2010.
[28] IBM, "Web Sphere Studio Information Sample," 2010.

Jong-Phil Kim is a Ph.D. candidate in the department of computer science at Chungbuk National University, Cheongju, Korea. He received the B.S. and M.S. degrees in computer science from Chungbuk National University, in 2006 and 2008, respectively. His research interests are aspect-oriented programming, service-oriented architecture, embedded software modeling and software testing.

Dong-Hyuk Sung is received the B.S. and M.S. degrees in the department of computer science from Chungbuk National University, Cheongju, Korea, in 2008 and 2011, respectively. His research interests are service-oriented architecture, embedded software modeling and software testing.

Jang-Eui Hong is an associate professor of Computer Engineering at the school of Electrical and Computer Engineering, Chungbuk National University, Cheongju, Korea. He received his Ph.D. in computer science from KAIST, Korea, in 2001. He served as a research member at ADD (Agency for Defense Development) from 2001 to 2004. His research interests include software quality, aspect-oriented programming, embedded software modeling, low-energy software model, and software process improvement.