International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Flexible and Dynamic Failure Recovery Mechanism for Composite Web Services Using Subset Replacement

Shuchi Gupta[1], Prof. Praveen Bhanodia[2]

[1]Department of Computer Science & Engineering, Patel College of Science & Technology, RGPV, Indore, India

[2]HOD, Department of Computer Science & Engineering, Patel College of Science & Technology, RGPV, Indore, India

Abstract: ***Business organizations today are complicated and intricate and provide a varied number of services. These huge numbers of services cannot be provided by an individual web service and hence we require a collection of web services which can deal with these complex services. In this paper a mediator (broker) is developed to make the interaction between service providers and service requesters intelligible. Execution of composite web services deals with execution of all the individual web services. Hence, it is very likely that the composite web services are liable to failure. At the time of execution of the composed web services, even if an individual service fails or becomes inapproachable, it results in non-function of the entire composed web services. In this situation of total non-function the entire process needs to re- executed. Hence to tackle this situation we have implemented a mechanism to ensure flexible and smooth functioning of business organizations. In this paper we have thus developed a dynamic failure recovery model to replace the subset containing failed subset with QoS equivalent subset. This system thus has the ability to reconfigure and recover from failure at run time.***

Keywords: Web Services; Service Oriented Architecture; Dynamic Composition; Subset Replacement

1. Introduction

The Internet and the World Wide Web are flourishing daily. Today almost all organizations are fully utilizing the Internet for promotion of their business [1]. Web services can be termed as the services which are furnished over the Internet. In today's challenging world the biggest challenge for business organizations is not only to deliver new services and solutions but also to rapidly provide the customers with the services using less resources [21]. Web services are essential for almost all businesses because of its features like robustness, language neutral, support to loosely coupled environment and platform independent.

Web services are implemented using the Service Oriented Architecture (SOA). SOA can be imagined as a collection of various technologies and applications that can be located above the existing applications of the organizations [21]. At present, large number of Web Service are present on the World Wide Web. Most of these are designed to serve a specific type of business functionality. To cope up with the huge business enterprise set up a single web service is not sufficient. Hence we require a collection of various web services. This collection can be termed as web service composition.

Web services can be composed using two techniques i.e. statically or dynamically. Static composition is done by identification of the services before execution and dynamic composition is achieved at the time of execution or during run-time [5]. It is essential to compose the web services at runtime or dynamically because it is a tedious task to find atomic services for composition well in advance. Dynamic composition decreases the time complexity of a system.

Execution of a composite web service includes execution of all combining web services [3, 6, and 8]. Hence, the composed web services are more vulnerable to failure than single or atomic web services [8]. While the process of execution of the composed web services, a failure in a single component too results in failure of the whole set. And as a result the whole business will not respond to the users requirements. In such scenarios the whole composite service need to be run completely all over again. Therefore we require a technique to ensure no single process is hampered during its execution.

2. Related Work

Internet today plays an important role in meeting the varied demands of the users. Web services thus are the next big thing. There are various papers proposed and implemented in the field of web services, web service composition and failure recovery. Firstly as the demands of users are complex and varied single services are not capable of meeting the demands of the users. Composition of web services is required for meeting all these complex demands of the users. In [3, 5, and 6] the authors have proposed the need for web service composition stating the reason behind this that users demands cannot be fulfilled by single web services. Senkul et al. have put forward an infrastructure for the construction of composition of web services and this method is based on logically formulating the web services [3]. Ming et al. placed a method for composition of web services dynamically using the technique of breaking down the users' requirements into minute parts known as abstract services. Then create a set of existing atomic and composite services and finally an executable composition of web services is

achieved by matching the abstract, atomic and composite web services [5]. Boumhamdi et al. also proposed a solution for dynamic web service composition and also re-configuring the web services in case of failure [6].

In [14, 15, 16 and 17] the authors present the need of failure and fault detection for web service oriented systems. Vieira et al. recommended a solution to evaluate and compare the performance and the recovery time of web services in presence of faults [13]. Chen et al. put forward the importance of fault detection [14]. Mansour et al. placed a reliability model for dynamically calculating the reliability of the web services by using the technique of rollback [16]. Zhu et al. presented a testing tool for diagnosing faults for web service composition [15]. These papers had a major focus on failure and fault detection but they did not focus on recovery from failure.

The most important step towards reliable execution of web service system is recovery from failure. The authors of [8, 7, 9, 10 and 17] have put forward different techniques for failure recovery. He et al. implemented an infrastructure to statically recover from failure of web services [8]. Yin et al. presented a technique for replacing the failed component web service with the help of composition [7]. Erradi et al. put forward a policy for web service recovery in presence of various faults. The authors reviewed various types of faults and their effects on the system [9]. Saboohi et al. proposed a technique for recovery from failure of composite web services based on sub graph calculation. The major drawback of this system is the calculation of all sub graphs before the execution of the system. This process results in making the system time consuming in its execution as calculation of sub graphs is done even in absence of failure [10]. Moller et al. put forward the technique for dynamically recovering from web service failure. The authors have also focused rightly on the replacement in case of failure situations only [17].

Few papers also focused on the importance of QoS factor calculation in the web service environment [7 and 16]. Yin et al. focused on the use of QoS factors during the replacement of composite web services [7]. Mansour et al. also proposed the reliability model to enhance the QoS of the web services [16]. Thus we can say that we require a mechanism which can dynamically replace the composite set of web services with QoS calculated equivalent or better web service set.

3. Proposed Solution

In this paper we have implemented a mechanism which deals with reliable execution of a composite web services in the presence of faults. We have developed a mediator or say broker, whose responsibilities are to firstly find the component web services according to the user's demands and then compose the web services together. After composing the web services the next responsibility is to execute the composite set and then if failure persists, recover from the failure. Finally after all these steps are carried out the results are sent.

Initially the service requestor requests for some services. These requests are forwarded to the broker. The broker searches for the available services in the UDDI registry [2] and then the available services are sent to the web service composer module. This module composes all the available services and then forwards them to the execution monitor where execution is carried out. On occurrence of failure the web services are recovered from the recovery manager. The goal of the recovery manager is to calculate the possible subsets for replacements. The recovery manager first calculates all possible subsets of composite web service, in which failed service is presented. For example, let's take a sample set {S1, S2, S3} If the service S3 is failed then the number of subset = 4, and the subsets formed are: {{S3}, {S2, S3}, {S1, S3}, {S1, S2, S3}}. It is the responsibility of the broker to rank the subsets according to the QoS factors and then find the equivalent match.

The architecture of the proposed system is shown in Figure 1. The components included in the system are as follows: Service Requester, Service Provider, UDDI Registry and the Broker.

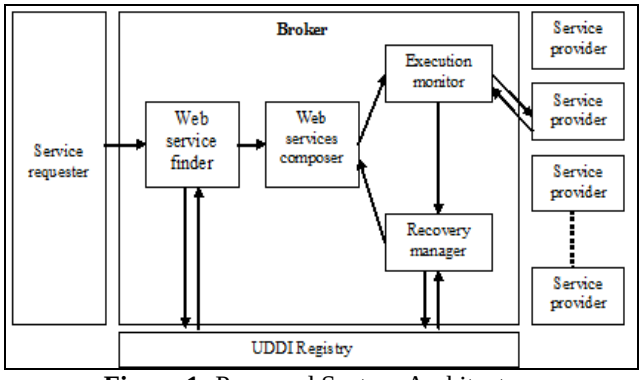

Figure 1: Proposed System Architecture

The Broker or the middle-agent is composed of various sub modules i.e. the Web Service Finder, Web Service Composer, Web Service Execution Monitor and the Web Service Recovery Manager.

The Supply chain management system considered for our implementation is for searching of raw materials. Firstly the system comprises of the customers and the manufacturers. On registering into the system the customer has to furnish with details of the Product required, supply location and supply within days**.** In our system the product comprises of various raw materials. For e.g. Product A is a collection of raw material A, B and C. On the basis of the customers' requirements the available services are looked upon by the web service finder in the UDDI registry. Web Service Finder finds the web and displays the service providers along-with their access point URL and cost of these services. This list is then forwarded to the composer where composite sets of the available web services are generated. Then a list of the composed web services along with their cost per unit price is displayed for the customer. The customer then enters the cost he wishes to pay along with the quantity of raw materials required. Then firstly all the available composite sets are ranked on the basis of the QoS factors i.e. availability, throughput and response time and then the set with the best QoS factor is sent for execution. The execution monitor executes these web services and if no failure persists then the results are displayed to the customer. On appearance of failure the service subsets are transferred to the recovery manager where the equivalent subsets in replacement of failed subsets are searched for and then the best possible equivalent subset is executed. And thus finally the results are displayed to the customer.

Fault Injection and Detection [20]

Fault injection technique is the artificial insemination of faults into a system. It is carried out for analyzing the actual execution of the system. Fault injection results in reviewing the exact functionality of the system in presence of faults. In this method some variables and values are changed at run time which fulfils the condition that makes failure in the web service. We injected various types of faults into our system as listed below:

- Software and development fault
- Network fault or connection loss
- Operational fault
- Unexpected server crash

The following techniques were applied to inject faults:

- Code Modification: Using this technique faults are injected into the source code at the time of compilation.
- Code Insertion: In this type of fault injection technique, a few instructions are added to the main program that will permit the faults to be injected before the occurrence of the particular instructions.
- Change variable value: This technique enables changing of the value of a variable dynamically but the condition must be predefined. If the supply location is changed at the time of service execution that may occur a fault.
- Network Disconnection: The connection cable is removed to simulate the network or connection loss fault.

The algorithms for implementation of the system have been presented in our previous work [20]. The algorithms implemented are: Algorithm to Identify Failed Subset, Algorithm to Identify Equivalent Subset and Algorithm for Alternative Subset Replacement. The policy of replacement is also explained in our previous work [20].

4. Testing and Results

Most web service applications developed today use static binding to call specific web services known at design time. Supply chain management system is however inherently dynamic, as conditions change over time, the functionality provided by the system changes. Web services are composed transparently in the supply chain management system to access more complex functions. So we use the supply chain management system (SCMS) to validate our method.

The whole system has been tested with different services, suppliers and faults. We calculated overall response time for composition and execution of Composite Web service. Then the response time was compared to offer overall Complex service to the user.

This comparison shows that the proposed method is very useful in case of failure in a composite web service and improves nearly 50% response time as compared to technique that composes the entire process all over again.

Saboohi et al. proposed a method in which they identify all the possible sub graphs for the replacement and their alternative before the execution of composite web service [10]. Identifying all the sub graphs and their alternatives are time consuming. Even if failure doesn't occur, this method still identifies all the subsets and their alternatives for replacement. So we are successful in reducing the response time by identifying the failed subset and their alternatives after the failure occurs. The total number of composite sets generated are also reduced due to identification of only the failed subset. We have compared the total number of subsets generated in the proposed method with the previous method.

The total number of subset in a composite set is= (2^n) - 1.
The total number of subset in a composite set using proposed method is = 2^ (n-1).

Where n is the number of service in the composite set.

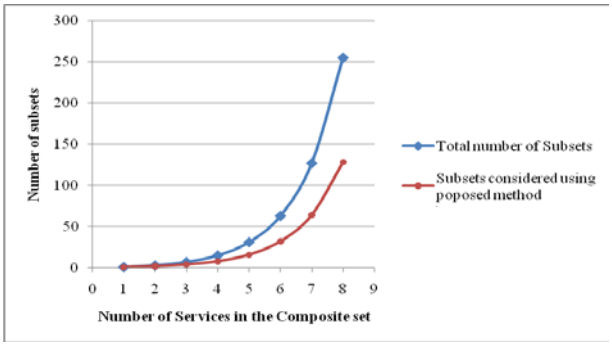

Figure 3

Figure 2 shows that our method reduces the number of subsets of total subset and so that we have to explore very less alternative. In table 1 depicts the comparison between the response times for identifying total number of subsets by both methods is shown.

Table 1: Comparison between the response times to identify subsets

No. of services in Composite Web Service	Response time to identify all possible subsets using earlier method[10] (in millisecond)	Response time to identify subsets using our method (in millisecond)	% Improvement
2	5	3	40%
3	17	5	70.58%
4	29	15	48.28%
5	63	35	44.44%
6	185	57	69.19%

The table 1 shows the comparison between the response times to identify subsets. The table shows that the response time taken by our method is less and improves up to 70% as compared to the earlier method in which all possible subsets are identified. Thus the above result shows the effectiveness of the proposed method and its usefulness in case when failure occurs in a composite web service environment.

5. Conclusion

With this paper we present a mechanism for dynamically recovering failure of composite web services. This method is based on the policy of subset replacement in a set of composite web services. This method implemented, enables the user to avail the entire complex and intricate services offered by today's business organizations and thus helps in meeting the user's requirements. In the case of failure or non-functioning of a web service, the proposed method covers up the failure by replacing the failed web services by their equivalent subset calculated using QoS parameters.

In future, different failure reasons can be incorporated in the proposed method for composite web service. In this paper we have only dealt with sequential web service composition. In future various other types of compositions can also be considered. A method may be proposed to identify failed subset and equivalent subsets in more efficient way that increases the system efficiency and reduce the response time of the system.

References

[1] W3C, "Web Services Architecture," 2006; http://www.w3.org/TR/2004/NOTE-ws-arch- 20040211/
[2] "WSDL and UDDI", w3schools, Available at: http://www.w3schools.com/wsdl/wsdl _uddi.asp.
[3] Pinar Senkul, "Composite Web Service Construction by Using a Logical Formalism", Proceeding of 22nd International Conference on Data Engineering Workshops (ICDEW'06), IEEE 2006, pp 56-65.
[4] Antonio Bucchiarone, Stefania Gnesi, "A Survey on Services Composition Languages and Models", International Workshop on Web Services Modeling and Testing, WS-MaTe 2006, pp 51-63.
[5] Wang Qing-Ming, Tang Yong, Zhang Zan-Bo, "Research in Enterprise Applications of Dynamic Web Service Composition Methods And Models", Preceding of Second International Symposium on Electronic Commerce and Security, IEEE 2009, pp 146-150.
[6] Kaouthar Boumhamdi, Zahir Jarir, "Yet another Approach for Dynamic Web Service Composition", International Conference for Internet Technology and Secured Transactions, 2009. ICITST 2009.
[7] Keting Yin, Bo Zhou, Shuai Zhang, Bin Xu, Yixi Chen, "QoS-aware Services Replacement of Web Service Composition" 2009 International Conference on Information Technology and Computer Science pp271-274.

[8] Weiping He, Virginia Tech, "Recovery in Web Service Applications", International Conference on e-Technology, e-Commerce and e-Service (EEE'04), IEEE, 2004.
[9] Abdelkarim Erradi, Piyush Maheshwari, Vladimir Tosic, "Recovery Policies for Enhancing Web Services Reliability", International Conference on Web Services (ICWS'06), IEEE, 2006.
[10] Hadi Saboohi, Amineh Amini, Hassan Abolhassani, "Failure Recovery of Composite Semantic Web Services using Subgraph Replacement" Proceedings of the International Conference on Computer and Communication Engineering 2008 May 13-15, 2008 Kuala Lumpur, Malaysia, pp489-493.
[11] K. Christos, V. Costas, G. Panayiotis, "Towards Dynamic, Relevance-Driven Exception Resolution in Composite Web Services", 4th Int. Workshop on SOA & Web Services.
[12] Roman Vaculín, Kevin Wiesner, Katia Sycara, "Exception handling and recovery of semantic web services", Fourth International Conference on Networking and Services, IEEE, 2008, pp217-222.
[13] Marco Vieira, Nuno Laranjeiro, "Comparing Web Services Performance and Recovery in the Presence of Faults", International Conference on Web Services (ICWS 2007),IEEE,2007.
[14] Hao-Peng Chen, Cheng Zhang, "A Queueing-Theory-Based Fault Detection Mechanism for SOA-Based Applications" The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007), IEEE, 2007.
[15] Zekun Zhu, Jianxin Li, Yongwang Zhao, Zhuqing li, "SCENETester: A Testing Framework to Support Fault Diagnosis for Web Service Composition," The 11th IEEE International Conference on Computer and Information Technology, IEEE, 2011.
[16] H. Elfawal Mansour and T. Dillon, Fellow, IEEE, Dependability and Rollback Recovery For Composite Web Services," IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 4, NO. 4, OCTOBER-DECEMBER 2010
[17] Thorsten Moller, Heiko Schuldt, " OSIRIS NEXT: Flexible Semantic Failure Handling for Composite Web Service Execution," 4th IEEE International Conference on Semantic Computing, 2011.
[18] Sean Baker and Simon Dobson, "Comparing Service-Oriented and Distributed Object Architectures", Proceedings of the International Symposium on Distributed Objects and Applications, Springer, 2005
[19] IDC White Paper, "IB and the strategic Potential of Web Services", 2002,http://www3.ibm.com/software/solutions/webservices/pdf/May13_IBM_WebServices.pdf
[20] Shuchi Gupta, Praveen Bhanodia, "A fault tolerant mechanism for composition of web services using subset replacement", International Journal of Advanced Research in Computer and Communication Engineering, Vol 2, Issue 8, August 2013
[21] Peter Classon, "The Business Value of Implementing a Service Oriented Architecture Discussion on the business drivers and benefits of SOA," http://www.liquidhub.com/docs/Horizons_SOA_BusValue_v4.pdf