AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Fault Analyser Model to Detect Faulty Web Service During Composition of Web Service

[1]Jayashree, K., [2]Chithambaramani, R., [3]Babu, R.

[1]Assistant Professor C.S.E. Department, Rajalakshmi Engineering college, Anna University, India.
[2]Assistant Professor I.T. Department, Rajalakshmi Engineering college, Anna University, India.
[3]Assistant Professor I.T. Department, Rajalakshmi Engineering college, Anna University, India.

ARTICLE INFO

Article history:
Received 10 October 2014
Received in revised form
22 November 2014
Accepted 28 November 2014
Available online 1 December 2014

Keywords:
Composite web service, Faulty Analyzer Model, Monitoring, Runtime faults

ABSTRACT

Web services are program to program interaction over the internet. A composite web service invokes one or more other web services and combines their functionality. Web service composition is an important problem in web service based systems. It is to build a new value-added web service using existing web services. A web service fails due to software bugs, unstable communication over the Internet, and complete crash of service servers. When a single web service fails, the entire composite service execution also fails. To detect faults in web services workflow is a challenging task. This paper presents an fault analyzer model for detecting faults in composite web services workflow. The proposed model has been tested with a sample web service application.

© 2014 AENSI Publisher All rights reserved.

To Cite This Article: Jayashree, K, Chithambaramani, R, Babu, R, Fault Analyser Model to Detect Faulty Web Service During Composition of Web Service. ***Aust. J. Basic & Appl. Sci.,*** *8(18): 163-170, 2014*

INTRODUCTION

Service Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. The web service is implemented using SOA concepts, where the basic unit of communication is a message, rather than an operation. Web services are powered by Extensible Markup Language (XML) and three other core technologies: Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration (UDDI). The web services are self-contained and self-describing and can be discovered using UDDI. Web service interface is described using WSDL. Other systems interact with WS in a described manner via SOAP messages and this is conveyed using HTTP with an XML serialization. Web services provide several technological and business benefits, such as Application and data integration, Versatility, Code re-use and Cost savings.

A single web service is not sufficient to fulfill the user's request, services should be combined together. The process of developing a composite web service is referred to as a web service composition. Business Process Execution Language (BPEL) defines a notation for specifying business process behavior based on web services. Composite web services fail when one of the services in the workflow fails and runtime faults will be thrown. Monitoring web services in a workflow is more difficult than monitoring a centralized application because web services are distributed in nature.

This paper proposes a generic adaptive method for composing web services and to find the faulty web services in an effective manner. The rest of the paper is organized as follows. The review of related work is presented in Section 2. The proposed work is given in section 3. Implementation and results are given in section 4 and section 5 concludes the paper.

Literature Survey:

In this section we discuss work related to web service composition and fault detection. While there has been considerable work relating to web service testing, debugging and fault tolerance, there has been less research in the area of composite web service monitoring.

Monitoring of web services require additional issues because it is loosely coupled and dynamic in nature. Runtime monitoring of business processes can be treated as a dynamic analysis of runtime events. Such events can be analyzed as online that occur during the execution of the application, or offline that occur after the

Corresponding Author: Jayashree, K., Assistant Professor C.S.E. Department, Rajalakshmi Engineering college, Anna University, India.
E-mail: k.jayashri@gmail.com

execution has been terminated (Bratanis *et al* 2007). (Lazovik *et al*, Baresi, pistore *et al*, lohman *et al*, li *et al*, barbon *et al*) are examples of online techniques; (Aalst & Pesic) are offline techniques.

Dynamic composition of web services is a promising approach and at the same time a challenging research area for the dissemination of service-oriented applications. It is widely recognized that service semantics is a key element for the dynamic composition of web services, since it allows the unambiguous descriptions of a service's capabilities and parameters. Silva *et al* introduces a framework for performing dynamic service composition by exploiting the semantic matchmaking between service parameters (i.e., outputs and inputs) to enable their interconnection and interaction. The basic assumption of the framework is that matchmaking enables finding semantic compatibilities among independently defined service descriptions. They also developed a composition algorithm that follows a semantic graph-based approach, in which a graph represents service compositions and the nodes of this graph represent semantic connections between services. Moreover, functional and non-functional properties of services are considered, to enable the computation of relevant and most suitable service compositions for some service request.

Mohammed Alodib *et al* (2009) aims to present a method of creating architectures which allow monitoring occurrence of failure in Service Oriented Architectures (SOA). The presented approach extends Discrete Event Systems techniques to produce a method of automated creation of Diagnoser Service which monitors interaction between the services to identify if a failure has happened and the type of failure. To do so, a formal representation of business processes is introduced, which allows modeling of Observable/Unobservable events, failure and the type of failure. The paper puts forward a set of algorithms for creating models of Diagnoser Service. Such models are then transformed into new Services implemented in BPEL, which interact with the existing services to identify if a failure has happened and the type of failure. The approach has been applied to an example of diagnosis of Right-first-time failure in Services used in telecommunications. Web service composition highlighted for its capability to compose autonomous services to achieve new functionality. Assuming the failure of any individual web service will cause the failure of the composite service, the overall reliability of composite service is the product of the reliability of constituent web services. Even all the other web services are reliable, one unreliable web Service could decrease the overall reliability to a very low level. The upper bound of overall reliability is often determined by the weakest constituent web services. We believe that web service composition can even help to achieve much higher reliability by leveraging the redundancy intrinsic to the Web.

Mahbub and Spanoudakis (2005) have developed a framework to support the run time verification of requirements for service based systems whose composition process is specified in BPEL. The requirements to be monitored are specified using event calculus. Event logs are recorded during the execution of a service based system which is then checked against the specifications. Moser *et al* (2008) introduced an aspect-oriented extension for existing BPEL environments that allows the monitoring of existing BPEL processes according to certain QoS criteria and an adaptation strategy to replace existing partner services based on various selectors that implement different replacement strategies.

The redundancy of web service holds a great key to avoid the failure problem. This redundancy can be easily discovered by the UDDI registry. When failures occur in any constituent web service, a composite web service can discover redundant web services, and replace the failed ones. Shin *et al* (2007) have described about how to obtain the optimization. In this Optimization processes consist of two phases: one is to remove unnecessary web services and the other is to find the best starting point of a composition. They have suggested some composition algorithm that they choose the services based on requirement.

Proposed Architecture:

Service providers who offer web services for public use, register their services in service registries using the publish operation. These web services are expressed in WSDL format. A consumer would look up the registry to find an appropriate web services and requests the service from the service registry using find operation. The service registry returns a known provider (Service URL) for the requested service. There may be multiple registries for multiple services depending upon the needs of the client. The user service then binds itself to the service provider for service execution. The service input parameters are sent to the service provider who executes the service and returns the results to the consumer. These interactions happen through the use of SOAP messages. The overall proposed system architecture is shown in Figure 1.

In the proposed monitoring architecture, Fault Analyzer has been used to find the faulty web services in the process flow during runtime. The flow of the program execution is taken into consideration and the inputs are gathered from the end user and it is provided to the service client. The service client requests for the required WSDL URL in the service registry based on the inputs obtained. Whenever any action is performed by the client, internally the process flow is constantly checked using the Business Process Execution Language which in turn checks for the correct process flow. The process flow for the client is checked by various parameters such as business id, registry URL and service id. After receiving the required WSDL URL the service client performs the binding operation to the required service provider.

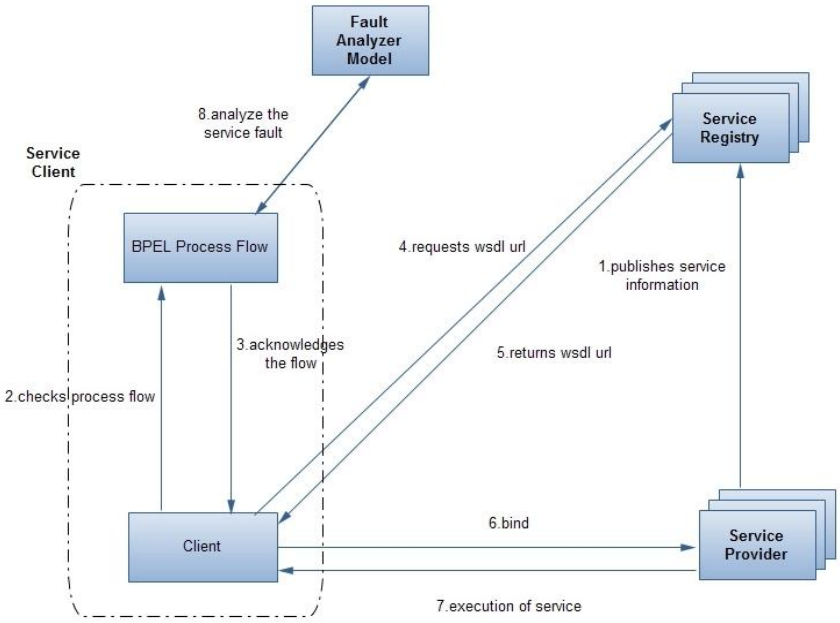

Fig. 1: Fault detection Model.

The service provider provides the client with the execution of the required service as requested by the service client. This is the execution phase of a single service. On completion of the execution of the current service, the next service may be invoked in the same way. So in order to check for the correctness of the workflow from one service to another the Fault Analyzer Model is used, which in turn checks and verifies the correct workflow. If any problem occurs with the workflow of the services, the model provides recovery of that service by replacing it with an alternative service to the service client.

When there are n services as a set of states, the set of actions for each service and the transition is as shown in Figure 2. The first service will be the initial state and if the service is successful means it will go to the next service state otherwise it will be in any one of the error state namely input or output error.

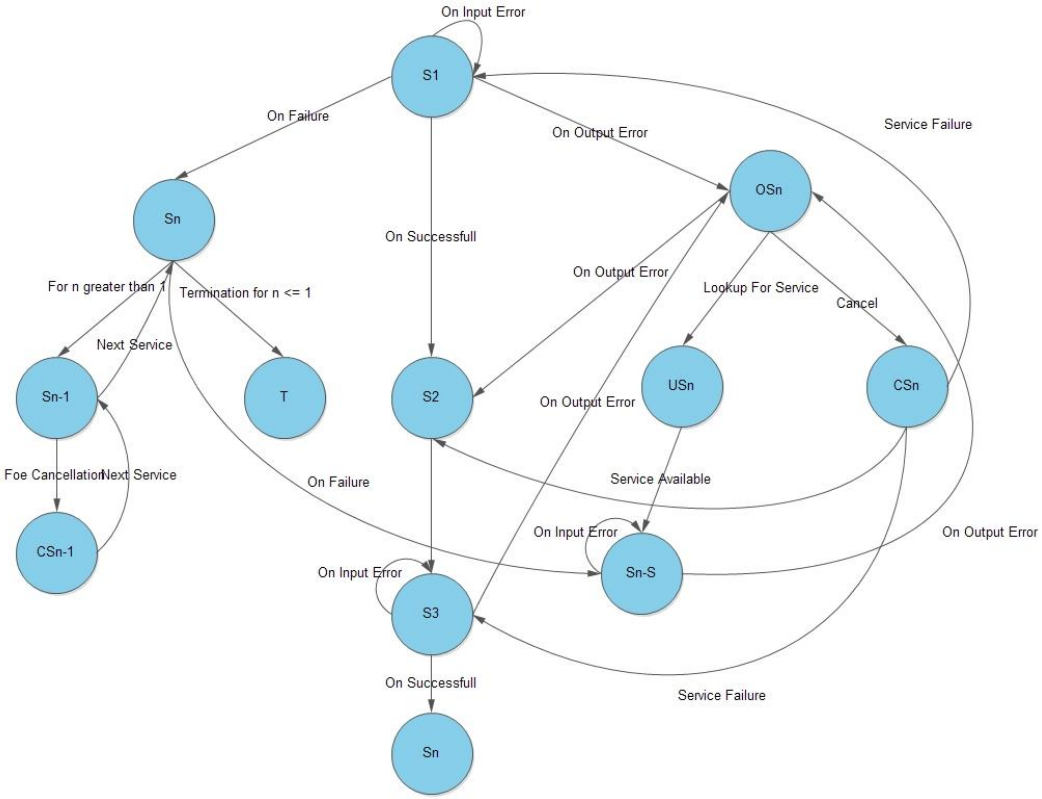

Fig. 2: LTS for process flow validation.

When service S1 is failed due to input error, the error message is reported to the user to correct the specific input. The user can correct the error or can cancel the service. Output error occurs when there is a deviation from the excepted output, and it is reported to the user. For an output error the user can use other available service by the lookup method or can cancel the service. When the other similar service is available in the service registry, the service is selected and used. The similar service is checked again for both input and output error. For n number of services in a composite service the faults are detected as similar to service S1.

Implementation and Results:

A sample web service application from the Travel agency domain has been used. Web services have been created for Registration, Flight Enquiry, Ticket Availability, Reservation, Reservation Status Enquiry and Ticket Cancellation. Various web services are composed together to provide web service composition.

Registration Service:

It allows the user to register themselves for booking the tickets and the sequence diagram for the registration service is shown in Figure 3.

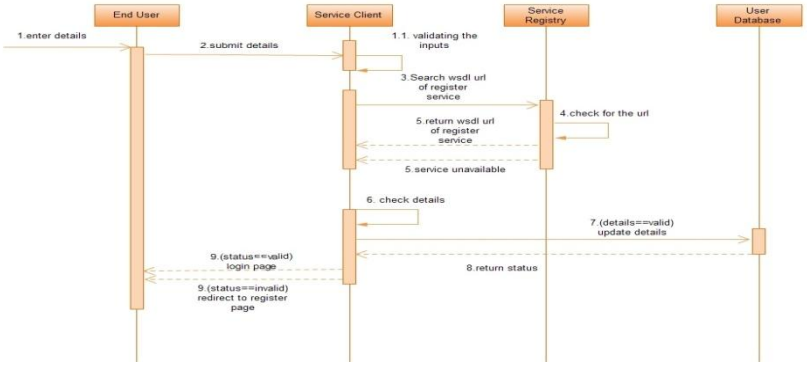

Fig. 3: Sequence diagram for registration service.

Search Service:

It allows the passenger to key-in the Flight details for their search and it returns the flight list. The interactions between the components is shown in figure 4.

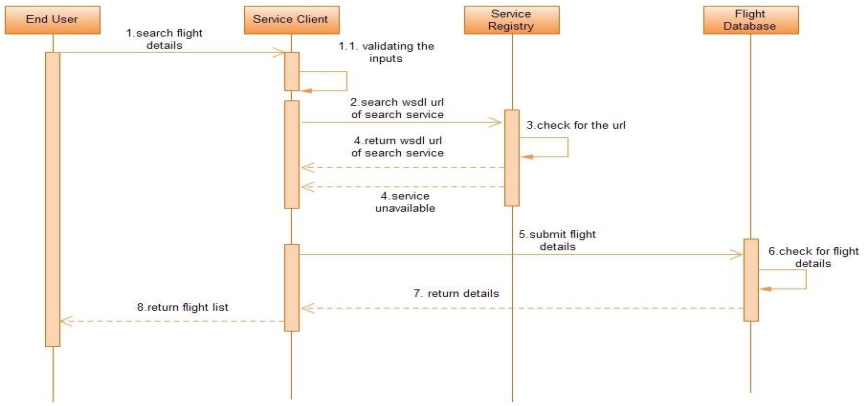

Fig. 4: Sequence diagram for registration service.

Booking Service:

Booking service is a composite web service, comprising of the Payment and Booking Details services and its interaction is shown in Figure 5. The user first provides details for booking the tickets by providing details which includes Date of Travel, Flight Number, Flight Name, Departure place, Destination place, Class, Name, Age etc.

Payment Service:

Payment service is used to process the payment amount. The user is required to give the credit card details for processing the payment through an authorized payment gateway. The interactions between the various components are shown in Figure 6.

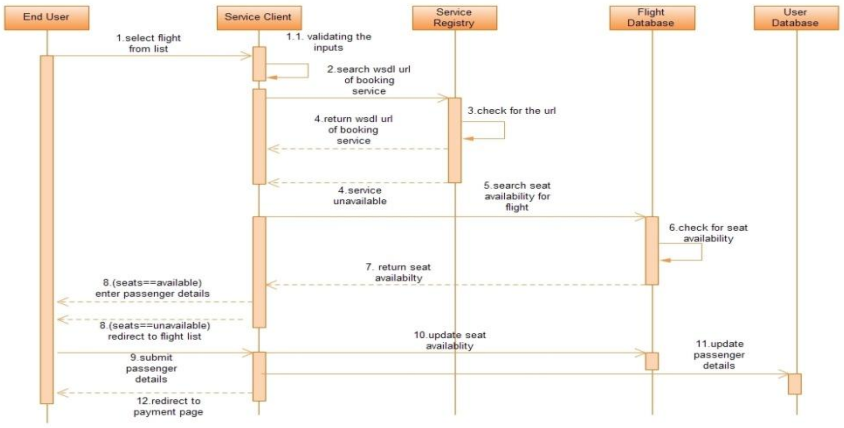

Fig. 5: Sequence diagram for booking service.

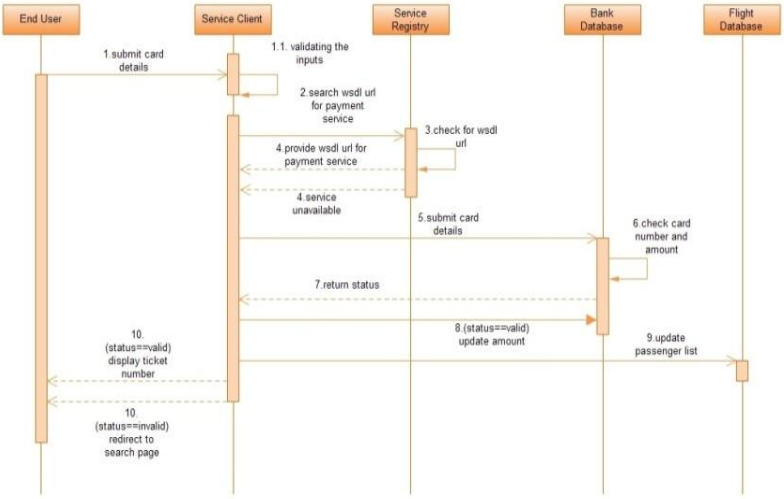

Fig. 6: Sequence diagram for payment service.

Cancellation Service: Allows the users to cancel their seats of their choice. The sequence diagram is shown in Figure 7.

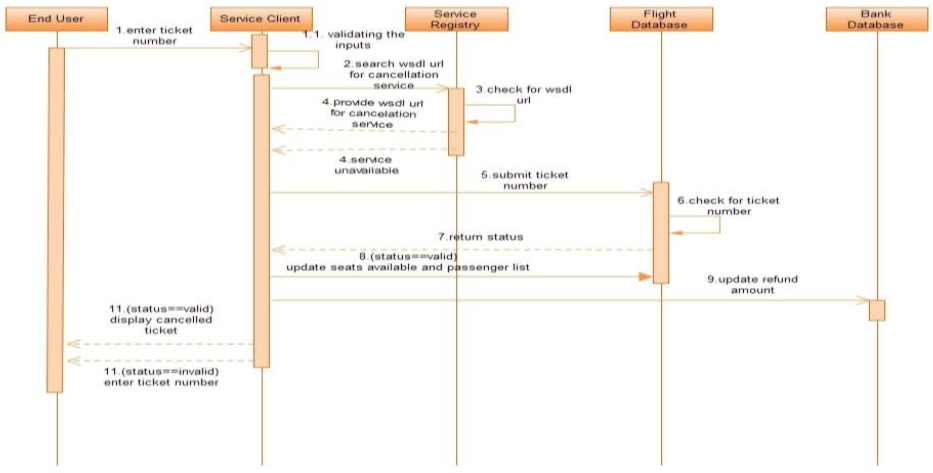

Fig. 7: Sequence diagram for cancellation service.

Consider a sample such as booking a flight ticket. Then the BPEL will contain all the required WSDL files such as login, booking, register, etc. So when the service clients access any of these service files then the BPEL checks the flow of access of the services by the client and validates it. The BPEL flow shown in a diagram format given below:

The process flow for the specific travel agency domain is as shown in figure 8. In the process flow of travel agency first the user is allowed to invoke Registration web service for the registration process. If the user is

already a registered user means Login web service is used to invoke to access the particular web service. The user can search for the Flight Availability and Rooms Availability by the corresponding web services that are stored in different jUDDI registry or in same jUDDI. As according to the user choice if the tickets for either flight or hotel or for both are available means they will invoke Ticket Reservation service and Room Reservation service as of their choice.

Fig. 8: Process flow for Travel Agency Service.

Ticket Reservation service is a composite service comprising of the two services Ticket Booking details and Payment. Room Reservation service is a composite service comprising of the two services Room Booking details and Payment. Ticket Booking details service is executed first to obtain the ticket and flight details and block the tickets for the specified flight. The payment is added to the cart. Room Booking service is executed next to the ticket reservation of flight and it is used to obtain the room number and block the rooms for the specified date. The payment is added to the cart. After the execution of both Ticket Booking details and Room Booking details services. Payment is automatically invoked to process the payment for both the tickets and rooms. Once the payment process is successfully completed, ticket booking details are updated in the database. To the user, however, it would appear as one service only. Error can occur in execution of any of the three services. The user can cancel any of the services namely Ticket Cancellation and Room Cancellation of their choice.

Conclusion:

The proposed work thus provides an adaptive method to compose web services. During the composition of the various services the workflow between the services is verified using the fault analyzer model. After verifying the workflow the fault analyzer model checks for the fault in the input provided by the user and also checks for service fault along with providing the required alternative service for the faulty service. The alternative service provided is based on the first come first serve ranking. The semantics can be included in future so that automatic discovery of services can be done.

REFERENCES

Aalst, W.M.P. and M. Pesic, 2007. "Specifying and Monitoring Service Flows: Making Web Services Process-Aware," in Test and Analysis of Web Services, pp: 11-55.

Alodib, M. and B. Bordbar, 2009. "A model-based approach to Fault diagnosis in Service oriented Architectures", Seventh IEEE European Conference on Web Services, pp: 129-138.

Barbon, F., P. Traverso, M. Pistore and M. Trainotti, "Run Time Monitoring of Instances and classes Web Service Compositions" IEEE ICWS06, pp: 66-71.

Baresi, L., C. Ghezzi and S. Guinea, 2004. "Smart Monitors for Composed Services," in Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC'04), pp: 193-202.

Baresi, L. and S. Guinea, 2005. "Towards Dynamic Monitoring of WS-BPEL Processes," in Proceedings of 3rd International Conference on Service Oriented Computing (ICSOC'05), pp: 269-282.

Bratanis, K.D., A. Dranidis, Simons "An extensible Architecture for run-time monitoring of conversational web services", proceedings of the 3rd international workshop on monitoring, Adaptation and Beyond' MONO'10, pp: 9-16.

Lazovik, A.M., Aiello and M.P. Papazoglou, 2004. "Associating Assertions with Business Processes and Monitoring Their Execution," in Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC'04), pp: 94-104.

Lohmann, M., L. Mariani and R. Heckel, 2007. "A Model-Driven Approach to Discovery, Testing and Monitoring of Web Services," in Test and Analysis of Web Services, pp: 173-204.

Li, Z., Y. Jin and J. Han, 2006. "A Runtime Monitoring and Validation Framework for Web Service Interactions," in Proceedings of the 17th Australian Software Engineering Conference (ASWEC'06). IEEE Computer Society, pp: 70-79.

Mahbub, K. and G. Spanoudakis, 2004. "A Framework for Requirements Monitoring of Service Based Systems," in Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04). New York, NY, USA: ACM, pp: 84-93.

Mahbub, K. and G. Spanoudakis, 2005. "Run-time Monitoring of Requirements for Systems Composed of Web-Services: Initial Implementation and Evaluation Experience, "in Proceedings of International Conference on Web Services (ICWS'05), pp: 257-265.

Moser, O., F. Rosenberg and S. Dustdar, 2008. Non-Intrusive Monitoring and Service Adaptation for WS-BPEL WWW 2008 / Refereed Track: Web Engineering - Web Service Deployment, pp: 815-824.

Pistore, M. and P. Traverso, 2007. "Assumption-Based Composition and Monitoring of Web Services," in Test and Analysis of Web Services, pp: 307-335.

Shin, K. and S. Han, 2007. "Efficient Web Services Composition and Optimization Techniques", 2007 IEEE International Conference on Web Services (ICWS 2007).

Silva, E. and Lu´ıs Ferreira Pires, "A Framework for Dynamic Web Services Composition", European IST SPICE project (IST- 027617) and the Dutch Freeband A-MUSE project (BSIK 03025).

"Will Reliability Kill the Web Service Composition?"

World Wide Web Consortium (W3C), Simple Object Access Protocol, Version 1.2, 2007, available at http://www.w3.org/TR/soap12.

World Wide Web Consortium (W3C). Web Services Description Language (WSDL), Version 1.1. March, 2001, available at http://www.w3.org/TR/wsdl.

http://www.uddi.org/pubs/uddi_v3.htm.