2008 IEEE International Symposium on Service-Oriented System Engineering

A Reliability Evaluation Framework on Composite Web Service[1]

Jiang Ma, Hao-peng Chen,
School of Software, Shanghai Jiao Tong University, 200030 Shanghai, P.R.China
ma.jiang@hotmail.com, chen-hp@sjtu.edu.cn

Abstract

The composition of web-based services is a process that usually requires advanced programming skills and vast knowledge about specific technologies. How to carry out web service composition according to functional sufficiency and performance is widely studied. Non-functional characteristics like reliability and security play an important role in the selection of web services composition process. This paper provides a web service reliability model for atomic web service without structural information and the composite web service consist of atomic web service and its redundant services. It outlines a framework based on client feedback to gather trustworthiness attributes to service registry for reliability evaluation.

Keywords: Reliability Evaluation, Composite Web Service

1. Introduction

The development of software systems frequently entails the need to integrate diverse applications within an enterprise and across enterprises. Different kinds of application integration technologies, such as object-oriented middleware, message-oriented middleware, and, more recently, the Web services platform, have been proposed for this purpose [3].The Web service technology has captured the attention of practitioners and academia as a promising solution to cost-efficient and manageable application integration.

According to the Stencil Group, Web services are “loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols” (www.stencilgroup.com).

A web service is characterized by its flexibility to encapsulate discrete business functionalities and its interoperability to support universal application integration [2]. Composition of computational resources and web-based services into integrated solutions is a key activity to enhance offerings and allow for a smooth process from the point of view of customers. Such integration process has been greatly simplified with the advent of web services technology.

On composing a dependable and reliable web service, we must ensure this composite web service has the capacity to monitor every atomic service, evaluate and enhance the reliability. The Internet where WSs operate in is a failure-prone environment. A component WS may become unavailable at runtime causing failure to the execution of the composite WS. In particular, executing a WS operation on a (temporarily) unavailable WS triggers an exception, and the handling of an exception may involve rolling back the earlier operations of the composite WS and re-executing them on other component WSs, resulting in a waste of computing resources and slow response time [1].

The quality of a web service based application fully dependents on the services of which it makes up. In a near future, the amount of services on the internet will increase greatly. It becomes an important problem that how to guarantee that the services using by one application all have good quality among the huge number of services on the internet.

The Universal Description, Discovery, and Integration specifications offer users a unified and systematic way to find service providers through a centralized registry of services that is roughly equivalent to an automated online "phone directory" of Web services [6]. Service registry is not only a static centralized storage pool of service descriptions, but also should verify the service qualities based on the static service descriptions to improve the creditability of the services registered on it, and analyze the feedbacks of applications.

The main structure of this paper is: the second part of the paper introduces some related works on reliability modeling in software, grid computing and web service. The third part provides the model of reliability evaluation. The fourth part puts forward a feedback framework based on above-mentioned model and then case study in the fifth part with discussion. Finally, there are conclusions and future work.

2. Related work on reliability modeling

The research of modeling and evaluation software reliability is along with the software evolution. IEEE defines the key concepts of software reliability [7]

[1] This paper is supported by the National High-Tech Research Development Program of China (863 program) under Grant No. 2007AA01Z139

- **Failure** is the inability of the software to perform its mission for function within specified limits. Failures are observed during testing and operation.
- **Software reliability** is the probability that software will not cause the failure of a product for a specified time under specified conditions. This probability is a function of the inputs to and use of the product as well as a function of the existence of faults in the software.

Actually web service reliability is not just concerned of software and web service itself. Many factors could affect the functional execution of web service such as churn of network, stability of web service server. [8] categorized failures in grid computing into: blocking failures, time-out failures, matchmaking failures, network failures, program failures, resource failures. Those kinds of failures also exist in web service domain. The hierarchical modeling maps the physical and logical architecture of the service system and makes the evaluation and analysis clear and simple by identifying the independence among layers.[9] suggested a SOA(service oriented architecture) reliability evaluation model using two attribute: availability, which is the quality attribute of whether the web service is present or ready for immediate use, and accessibility, which is the quality attribute of service that represents the capable of serving a web service request.

[4] examines reliability of an entire application created from a set of web services. In that context, it places greater emphasis on combining reliability measures for different web services, than on determining the antecedents of reliability for each web service. It views reliability of web services as a measure of failure-free operation, without modeling the source and type of failure. Back to the definition of failure, when a web service could not accomplish its mission for function, it will be considered as a failure occurred. We adopted this idea based on our web service consumer feedback models. Consumer encounter an inability to fulfill the functional behavior, it will send a failure report to data collector in UDDI whose data is used for reliability evaluation.

[10] suggested a web service evaluation model using group testing technique which is originally developed for testing large samples of blood. It uses this technique to test the contamination of an entire group of services by applying one test. Test phrase is often maintained by the service provider, and it's a challenge for a service consumer to test a web service when the service is on site especially if not free. We believe it is easier for a client to compose web services using the data collected in the operation phase.

3. Reliability computing model

Based on research of web service, many studies aim at combination of web service. In this paper, composed web service is defined as,

$$C = \{N_i\}\left(i \in [1 \ldots n]\right) \tag{3-1}$$

C defines the composite web service for reliability evaluation. The basic unit of composite web service is defined as a node N_i, n is the number of nodes in the composite service. Service node is defined as,

$$N_i = \{S_i, B_{ij}\}\left(i \in [1 \ldots n], j \in [0 \ldots q]\right) \tag{3-2}$$

S_i is the primary service in the composite service. Each node could only contain one primary service. When composite service process to a particular node, it will always invoke primary service first. As primary service may fail during particular time, backup services could substitute to act the desired behavior. So B_{ij} is the backup service for S_i, and q is the number of backup service. 0 means there is no backup service for the primary service, whereas q>1 means a redundancy service pool.

Each single web service in the node is regard as atomic web service. an atomic web service is a service that should be treated as a unit that is not to be broken.

During this section, we will first discuss a reliability model for atomic web service and then this model will extend to apply for a single node, at last we will reference a business process reduction algorithm for computing composite web service.

3.1 Atomic web service reliability

The following are the assumptions concerning this system that will be used throughout this paper.

- Each web service has only 2 states, working state and failure state.
- The single web service is assumed to have a failure intensity parameter λ.
- When web service is at failure state, service provider will try to fix this failure, and the fix time which including locating time and fixing time, follows an exponential distribution with a parameter μ.
- Each failure occurred is mutually independent.
- The initial state is working state.

Note that we assume each failure occurred is mutually independent which means reliability of each web service is independent. It can be argued that this assumption does not always hold, in that multiple web services from a vendor may prove to be unavailable if the vendor's site is down temporarily. For the purpose of web service evolution, numbers of web service will decentralized by deferent service provider, we adopt the standard characterization of independence, though.

Based on those assumptions, we use time-dependent Markov model which is widely used in reliability evaluation field.

For this single web service, we define $P_0(t)$ as the probability when the service is at working state and $P_1(t)$ as the probability when the service is at failure state. The corresponding Kolomogorov's equations are

$$P'_0(t) = \mu P_1(t) - \lambda P_0(t) \quad (3\text{-}3)$$

And

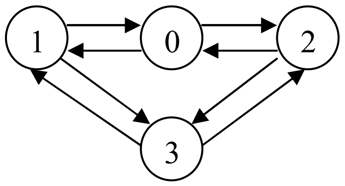
$$P_1(t) = 1 - P_0(t) \quad (3\text{-}4)$$

With the initial condition $P_0(0)=1$, $P_1(0)=0$,it can be shown that $P_0(t)$ is the reliability of the single web service.

3.2 Node reliability

In order to enhance web service reliability, the most practical way is to build a service with backup services. Backup services are usually heterogeneous, and provided by different service provider, so the characteristics of these services is different. In our model, it means those service have different failure intensity parameter λ and exponential distribution.

We use Markov chain to evaluate those web services. We first describe a simple one-backup service case, and the general case will be discussed later. The figure 1 depicts the Markov state transitions.



State 0: initial state, all service working
State 1: service *S* down, *B* working
State 2: service *B* down, *S* working
State 3: both services *S*, *B* down.

Figure 1. Markov chain state transitions

Service *S* is primary web service and *B* is backup service. Their intensity parameter is λ_S and λ_B , μ_S and μ_B stands for repair rate. Let $P_i(t)$ denotes the probability that the system is in state *i* at time *t*. The corresponding Kolomogorov's equations are

$$P_0'(t) = \mu_S P_1(t) + \mu_B P_2(t) - (\lambda_S + \lambda_B) P_0(t) \quad (3\text{-}5)$$

$$P_1'(t) = \lambda_S P_0(t) + \mu_B P_3(t) - (\mu_S + \lambda_B) P_1(t) \quad (3\text{-}6)$$

$$P_2'(t) = \lambda_B P_0(t) + \mu_S P_3(t) - (\mu_B + \lambda_S) P_2(t) \quad (3\text{-}7)$$

$$P_3'(t) = \lambda_B P_1(t) + \lambda_S P_2(t) - (\mu_S + \mu_B) P_3(t) \quad (3\text{-}8)$$

With initial condition

$P_0(0)=1$ and $P_j(0)=0$ for j=1,2,3.

With this initial condition, the differential equations are solvable numerically.

So the reliability $R = 1 - P_3(t)$

In general case, usually a web service is build up with a set of backup services. We define this web service architecture as a node. The failure intensity parameter of service *S* is λ_S and repair rate is μ_S . For backup service set, *B* down means all backup services failed. So the failure intensity $\lambda_B = \prod_{i=1}^{n} \lambda_{b_i}$ .when all backup services are not working, recovering an arbitrary backup service will make the backup service set working. So the repair rate of S is μ_f , μ_f denotes first recovered backup web service repair rate.

So we can derive the corresponding Kolomogorov's equations for web service in general case,

$$P_0'(t) = \mu_S P_1(t) + \mu_f P_2(t) - (\lambda_S + \lambda_B) P_0(t) \quad (3\text{-}9)$$

$$P_1'(t) = \lambda_S P_0(t) + \mu_f P_3(t) - (\mu_S + \lambda_B) P_1(t) \quad (3\text{-}10)$$

$$P_2'(t) = \lambda_B P_0(t) + \mu_S P_3(t) - (\mu_f + \lambda_S) P_2(t) \quad (3\text{-}11)$$

$$P_3'(t) = \lambda_B P_1(t) + \lambda_S P_2(t) - (\mu_S + \mu_f) P_3(t) \quad (3\text{-}12)$$

With initial condition

$P_0(0)=1$ and $P_j(0)=0$ for j=1,2,3.

So the reliability $R = 1 - P_3(t)$

3.3 Aggregated Reliability of composite web service

In the above section, we discussed the reliability evaluation method for atomic service and a web service with a set of backup services. In this section, we will propose a way to evaluate a business process composed with web services.

A composite web service usually make use of those functional web service to accomplish a business process, it is much similar as a work flow process. The aggregated reliability apparently depends on the structure of the business process. The independence degree between services and redundancy of a service is also the factors that affect aggregate reliability. These two factors we already discussed in above 2 sections. Composite web service structure could be viewed as nodes and the relationship between nodes. For the aggregated reliability, as we defined, the basic unit of a composite service is nodes, and operation relationships between nodes determine behavior of the composite service. So the composite web service reliability could define as

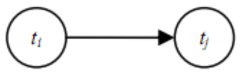
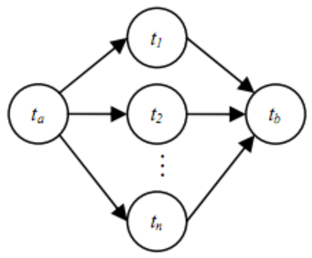
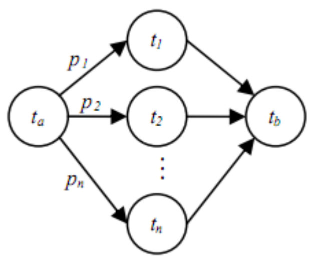
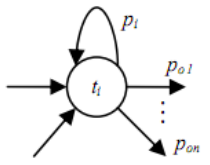
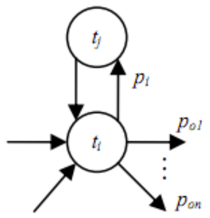
$$R_c = \left\{ R_{N_i}, O_s \right\} \tag{3-12}$$

R_c denotes the aggregate reliability, R_{N_i} denotes reliability of each node and O_s is the operation relationship of sub set of node set.

From above discussion, we could measure every node of this business process. For the operation relationship, we adopt Jorge Cardoso's Stochastic Workflow Reduction (SWR) algorithm [5]. Cardoso identify the following relationships: sequential, parallel, conditional, simple loop, and dual loop. Each activity is a data assignment, exchanging an event, doing an action, or executing a sub-process, applying those relationships, we can compute composite service reliability.

[4] summarized reliability SWR algorithm which shows in Table 1 to evaluate the aggregated reliability of a business process.

Table 1. Combining reliability measures

Reliability	Notation
Sequential $R = R(t_i)R(t_j)$	
Parallel (AND split – AND join) $R = R(t_a)R(t_b)\prod_i R(t_i)$	
Conditional (XOR split – XOR join) $R = R(t_a)R(t_b)\sum_i p_i R(t_i)$	
Simple-loop $R = \frac{(1-p_i)R(t_i)}{1-p_i R(t_i)}$, where $p_i + \sum_{j=1}^{n} p_{oj} = 1$	
Dual-loop $R = \frac{(1-p_i)R(t_i)}{1-p_i R(t_i)R(t_j)}$, where $p_i + \sum_{j=1}^{n} p_{oj} = 1$	

4. Consumer Feedback based composite service framework

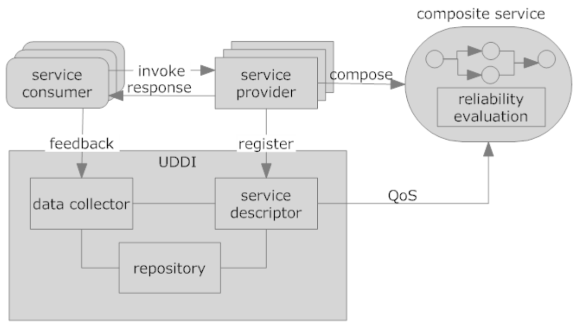


Figure 2. Reliability evaluation framework

Figure 2 shows our evaluation framework. All service providers register their web services on the UDDI which holds functional and non-functional information of these web services in service descriptor. When each client invokes and consumes web service, it will automatically feedback QoS information to UDDI. The repository stores QoS information from data collector which is the feedback interface of UDDI. Based on those QoS information, a composite service could evaluate its reliability and when this composite service applies to other web service or backup services pool, the reliability will be reevaluated.

Notice that in our model, the feedback information we collect is from client. The common way for QoS collecting is from service provider side which adds extra QoS metrics when data transport between consumers and providers. UDDI acquires this information from service provider and utilize it to recommend. However, the challenge of this model is the trustworthiness and accuracy of the information the provider feedbacks. We believe this pivot data for recommendation and evaluation should be took by a third party UDDI framework.

UDDI in our model collecting feedback from all those web service registered and stores that information for reliability evaluation. As a web service invoked by a great number of service consumer, feedback reflect the service provider reality. Service consumer feedback invocation count number n_i in a specified period time to determine the total invocation count number $N = \sum_{i=0}^{m} n_i$, where m denotes the service consumer instances feedback. Every time a service consumer faced a failure during service processing it reports that to UDDI to determine the total failure number *f* during a specified time. RC_i is the time interval between a web service failure and the most recent successful invocation. C is the count number a failure get repaired in a specified time interval which could be configured to adapt different network state.

The reliability evaluation framework needs to record the successor web service node set and count number S_i to determine the conditional probability between nodes.

Up to those discussions, we get all the parameter needed in our revaluation framework,

$$\lambda = \frac{f}{N} \qquad (4\text{-}1)$$

$$N = \sum_{i=0}^{m} n_i \qquad (4\text{-}2)$$

$$\mu = \frac{C}{\sum_{i=0}^{p} RC_i} \qquad (4\text{-}3)$$

$$p_i = \frac{S_i}{\sum_{j=0}^{q} S_j} \qquad (4\text{-}4)$$

5. Case study

Suppose we have a composite web service in a specific scenario. For simplification, this scenario just shows a simulation of web service composition. Figure 3 shows the composite web service.

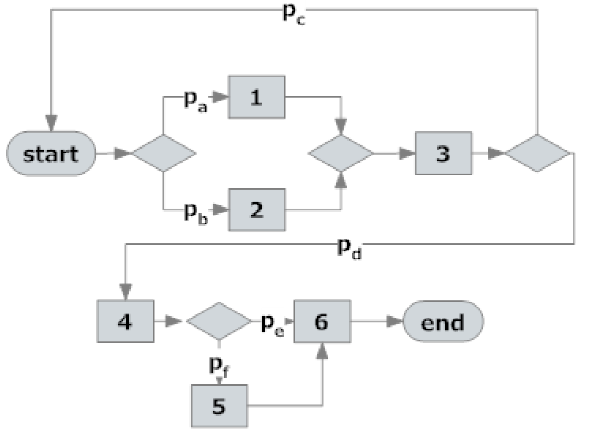


Figure 3. A typical composite web service

Suppose when a client first composed this service, all the node is atomic web service. Evaluating the composite service, our framework retrieved client feedback of each service from UDDI and applying it to our evaluation model. Table 2 shows the details.

Table 2 Computing reliability for composite web service

service	λ	μ	reliability
1	0.003	0.2	0.9852
2	0.01	0.2	0.9524
3	0.002	0.18	0.9890
4	0.025	0.15	0.8572
5	0.005	0.15	0.9677
6	0.006	0.1	0.9434
7	0.025	0.15	
8	0.025	0.15	

The composite web service aggregated reliability R_c is,

$$R_c = \frac{(1-p_c)(p_a R_1 + p_b R_2) R_3}{1 - p_c (p_a R_1 + p_b R_2) R_3} R_4 \left(p_e R_6 + p_f R_5 R_6 \right)$$

$$p_a + p_b = 1,\ p_c + p_d = 1,\ p_e + p_f = 1$$

By assuming p_a=0.6, p_c=0.6, p_e=0.7, R_c=0.7278.

We can see R_4 is the week point of this composite web service. In order to enhance this composite service, we add a backup service pool made up of S_7 and S_8 .applying our evaluation model, the reliability of node 4 will improve to R_4 =0.9994, so the aggregated reliability is improved to R_c=0.8485.

6. Conclusion

The paper presented a reliability model of web service and its redundant services which could be apply to business process reduction algorithm to evaluate composite web service reliability. We also introduced our evaluation framework which is based on collecting consumer feedback for trustworthiness QoS to service registry for evaluation framework. As for any attributes of quality, this framework can add existed attributes and even more new attribute through the suitable model.
We also descript how the evaluation model we put forward works with a study case. Actually, to demonstrate the correctness and practicability of this mechanism, we should develop a simulation application which runs on our UDDI with data collector gathering client feedbacks. It is also the next research step that we will take in future.

7. Reference

[1]ASan-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, Cheng-Hung Chen, "On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection", IEEE International Conference on Web Services (ICWS 2007) pp. 184-191.
[2]R. T. Rust and P. K. Kannan, "E-service: A new paradigm for business in the electronic environment," Commun. ACM, vol. 46, no. 6, pp.36–42, 2003.
[3] Thomas Mikalsen, Stefan Tai, Isabelle Rouvellou, "Transaction alattitudes: Reliable composition of autonomous Webservices", http://www.research.ibm.com/AEM/pubs/wstx-WDMS-DSN2002.pdf
[4]Hangjung Zo, Nazareth, D.L., Jain, H.K., "Measuring Reliability of Applications Composed of Web Services", System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on Jan. 2007 Page(s):278c - 278c
[5]J. Cardoso, J. Miller, A. Sheth, and J. Arnold, "Modeling Quality of Service for Workflows and Web Service Processes," Technical Report #02-002, LSDIS Lab, Computer Science, University of Georgia, 2002.
[6]F. Curbera et al., "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," IEEE Internet Computing, vol. 6, no. 2, Mar./Apr. 2002, pp. 86-93.
[7]Krajcuskova, Z. "Sotware Reliability models", Radioelektronika, 2007. 17th International Conference 4-25 April 2007 Page(s):1 - 4 Digital Object Identifier 10.1109/RADIOELEK.2007.371428
[8]Dai, Yuan-Shun; Pan, Yi; Zou, Xukai; "A Hierarchical Modeling and Analysis for Grid Service Reliability" Transactions on Computers Volume 56, Issue 5, May 2007 Page(s):681 - 691 Digital Object Identifier 10.1109/TC.2007.1034
[9] Zhenyu Liu; Ning Gu; Genxing Yang; "A Reliability Evaluation Framework on Service Oriented Architecture" Pervasive Computing and Applications, 2007. ICPCA 2007. 2nd International Conference on 26-27 July 2007 Page(s):466 - 471
[10]W. T. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul*, N. Liao , "A software reliability model for Web services", The 8th IASTED International Conference on Software Engineering and Applications, Cambridge, MA, September 2004