Journal of Theoretical and Applied Information Technology
20th July 2014. Vol. 65 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RELIABILITY ANALYSIS OF WEB SERVICES BASED ON RUNTIME FAULT DETECTION

[1]K.JAYASHREE, [2]DR. SHEILA ANAND, [3]R. CHITHAMBARAMANI
Rajalakshmi Engineering College, Chennai
[1]k.jayashri@gmail.com , [2]sheila.anand@gmail.com , [3]r.chithambaramani@gmail.com

ABSTRACT

Web service technology is being increasingly used for commercial business applications. Reliability of the web services provided is an important criterion to enable the wide spread deployment of such services. This paper proposes analyzing the reliability of web services based on runtime fault information. Monitoring of web service interactions enable runtime faults to be detected and recorded. A sample application was developed and logs were designed for recording the service usage and faults information. Faults were injected in the sample application and faults during publishing, discovery, binding, execution and composition were recorded. Reliability estimation and prediction was carried out using Jelinski-Moranda Binomial model and Goel-Okumoto Non-Homogenous Poisson model.

Keywords: *Reliability for web services, Monitoring Component, Fault Diagnoser, Fault Log, Service Usage Log, Reliability analysis.*

1. INTRODUCTION

Web services are loosely-coupled, platform-independent, self-describing software components that can be published, located and invoked using the standards, such as, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery Integration (UDDI) [1].

Web services can be either simple or composite. Web service that work on a stand-alone basis and do not invoke other web services are called simple or atomic web services. A composite web service combines the functionality of two or more simple web services by invoking them dynamically at runtime. Web services can be composed by using three methods namely static composition, semi static and dynamic composition. Web services composed at design time is called as static composition. When web services are composed during runtime it is termed as dynamic composition.

Reliability, security, cost, and performance are criteria that are identified as relevant non-functional QoS requirements when selecting different web services [2]. Reliability is a specific aspect of the broader concept of dependability. Dependability is defined [3] as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers. Reliability can be defined as the probability, that the system will correctly deliver services as expected by the user over a given period of time. "Reliability on demand" is defined as the probability that the system will successfully complete its tasks when invoked [4].There is essentially two types of software reliability models. The first model attempts to predict software reliability from design parameters such as lines of code, nesting of loops etc. The second model also known as "software reliability growth" attempts to predict software reliability from test or real data [5].

As web services are heterogeneous, distributed and dynamic in nature, faults can occur in different places, for example faults can occur in the software application or in the network connection. For atomic services, faults can occur during publishing, binding, discovery and execution. In composite services, failure in any one service can decrease the overall reliability of composite service. Also, since operations of a composite service often span multiple individual web services, web service composition itself also brings unreliability with the presence of the failures [6].

This paper focuses on the reliability analysis for single and composite web services. Reliability of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

web services is analyzed based on runtime monitoring and fault occurrence information. The rest of the paper is organized as follows. Section 2 presents the related work. In Section 3, presents the Reliability Analysis based on Runtime Fault Information. In Section 4, Implementation details are explained. Section 5 concludes the paper.

2. RELATED WORK

Software reliability prediction is a task to determine future reliability of software systems based on the past failure data [7]. Musa et al [8] defines software reliability as the probability of failure-free operation of a computer program for a specified time in a specified environment. Compared with traditional software reliability [8], reliability of a web service contains unique requirements of timeliness (availability and performance) [9]. Since web services are remote web applications that can only be invoked remotely, timeliness is of paramount importance: if a web service cannot be delivered to a service requester within a predefined time threshold with promised performance, the service will become useless.

Tsai et al [10] have proposed Service-Oriented software Reliability Model (SORM) to evaluate the reliability of atomic services and composite services. They have proposed group testing and majority voting for software testing of atomic web services. The voting system detects faults by comparing the output of each service with the weighted majority output. They have also suggested the use of a remote test agent to perform distributed testing on data collected at runtime. The reliability of the composite services is estimated based on the reliabilities of the individual atomic services. Senol Arikan [11] has argued the basic assumptions for computing reliability in SORM model. His contention is that assignment operation never generates new failures; a condition fails when any data associated with it fails and there is no cyclic dependency on the parameters or entities used for reliability analysis. The author has defined the research challenges in this area and given an approach to solve the reliability problem of Service Oriented Architecture (SOA). The author has suggested the use of a monitoring module to monitor the system components and inputs to identify failed components; a reliability validator module to calculate and analyze the reliability of the components and a redundancy module to deploy redundancy. Implementation and results of the proposed approach has not been given.

Sasikaladevi et al [12] have presented the design of reliability evaluation framework for composite web services. This paper gives the simple framework model for the selection of reliable web service among the available equivalent services. Reliability evaluation would be carried out by a broker or agent on behalf of the service consumer. Data collection unit uses service registry and feedback data obtained from prior consumers to evaluate availability and accessibility. The focus of the paper is on design of the framework and implementation of the proposed approach has not been given.

Chunli et al [13] have proposed a reliability prediction model for composite services combining control-structure such as sequence, if/switch, while and for composition condition. They presented an Extended Reliability Block Diagram (ERBD) in which composition conditions are regarded as atomic services and the reliability of composite services are evaluated according to their structures. The testing has been done using simulation and not real data.

Zibin Zheng et al [14] have proposed a collaborative reliability prediction approach for service-oriented systems. They have used past failure data of other similar users for predicting web service failure probabilities for the current user. Pearson correlation coefficient is employed for similarity computation. However, if there is insufficient number of similar users then dissimilar users are also considered.

Earlier work has primarily focused on testing or reliability prediction. Real or simulated data has been used for reliability analysis. This paper proposed reliability computation and prediction using real fault data obtained by runtime monitoring of web services. Reliability analysis of atomic and composite services has been proposed.

3. RELIABILITY ANALYSIS BASED ON RUNTIME FAULT INFORMATION

Web services runtime monitoring is an important component of web service management. To ensure availability and reliability of these web

Journal of Theoretical and Applied Information Technology
20th July 2014. Vol. 65 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

services, it is necessary to monitor the runtime behavior of web services and ascertain whether their behavior is correct, that is, as per their intended behavior. Monitoring the web service interactions at runtime would enable faults to be detected and recorded for reliability analysis.

We extend our earlier work on runtime monitoring to perform the reliability analysis [15]. We had proposed WISDOM (Web Service Diagnoser Model), which describes a generic architecture for runtime monitoring of web service interactions and handling fault detection. Independent Monitoring Components (MC) located in service registries and service providers are used to intercept the web service interactions at these entities. The request and response messages between the web service components would be first intercepted by these monitoring components and analyzed to detect faulty behavior.

The Fault Diagnoser is designed to be an independent external entity that would co-ordinate the individual

monitoring components. Policies were used to describe the intended behavior of web services and
faults are recorded as deviations from the intended behavior. Policies support standard assertions and provide a simple method to express the capabilities, requirements and characteristics of web services. The extended model with the Reliability Analysis Component is shown in Figure 1.

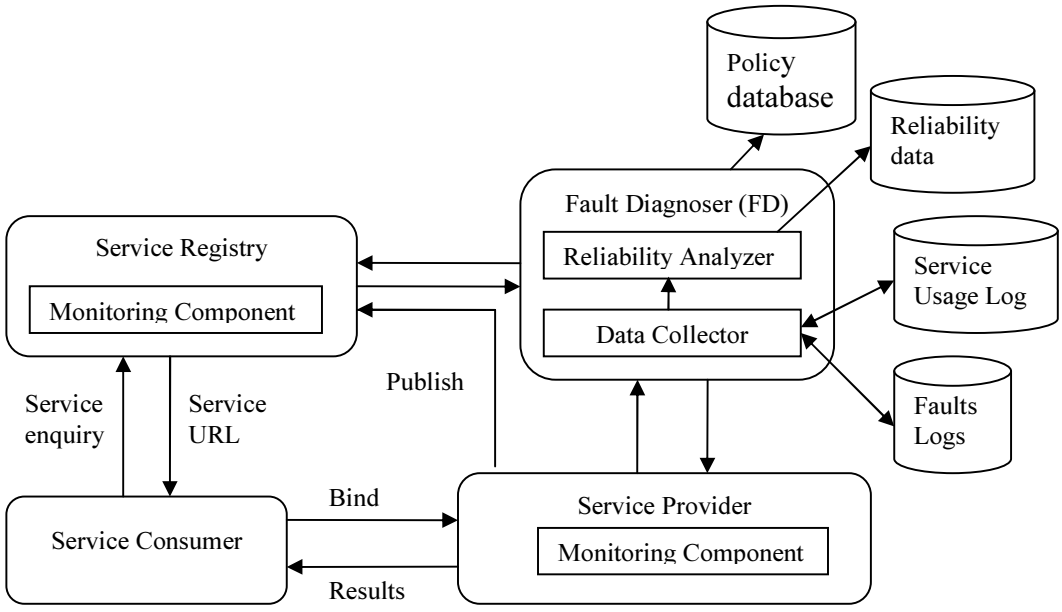

Figure 1: Runtime Fault Detection And Reliability Analysis Architecture

Service providers publish services in service registries to make their services available for use by service consumers. Users would then be able to discover the services by searching the service registry. The service details obtained are used to invoke (bind) and execute the service offered by the service provider.

Faults can occur during publishing if the service description given by the service provider is incorrect. Service discovery faults can occur during the search process if the service name is not listed or the search criteria is incorrect. Error can also occur if the service description does not match the service specifications given in WSDL. Fault can occur during binding if

there are authentication failures or if the service invocation details, like port number, are incorrect. Faults can occur during execution because the input parameters given by the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

service user are incorrect. Faults can also occur if the execution has errors or if the output returned to the user is incorrect as it does not match the service specifications. Network failure would cause any of the above operations to be timed out. Faults recorded during publishing, discovery, binding, execution and composition has been used for reliability analysis.

Reliability analysis is concerned with failure rate and the time between failures. These parameters are calculated from the real fault data obtained using the above model. Reliability prediction considers the history of failures to predict the probability that the web services will work without failure for a given time period under the prevailing operational conditions. Reliability Analyzer component located in the Fault Diagnoser performs the reliability analysis: computing the failure rates and also predicating reliability using standard models. The Data Collector sorts and aggregates the data from the fault log and service usage log. The Reliability Analyzer performs the reliability analyses and reliability estimation.

3.1 Runtime Log Data details

Two logs were designed to maintain service usage and fault data in the Fault Diagnoser. The format of the logs and details of data maintained are now described in detail.

3.1.1 Service Usage Log

Service Usage log was designed to store details of usage of all the web services. The data gets updated when a web service is invoked and executed. Details of Atomic (Single) and Composite web services are updated in this table. The details of data stored in the Service Usage Log are given in Table 1.

Table 1: Service Usage Log

Data Field	Description
Service_ID	Web Service Identifier
Service_Provider_Name	Service Provider name
Service_name	Name of the specific web service
Service_status	Executed successfully or Failed
Date_Time_Stamp	Date and time of invocation of web service
Execution_Time	Total time taken for execution
Flag	Atomic service or Composite service
Previous_Service_ID	ID of Previous Service invoked, for Composite service
Next_Service_ID	ID of Next Service invoked, for Composite service

3.1.2 Fault Log

Fault logs were used to keep a record of all faults that have happened during the runtime execution of web services. Faults which occurred during publishing, discovery, binding, execution and composition of web services were recorded in this log.

Faults were classified under the major heads, Publishing, Discovery, Binding, Execution and Composition. Each fault class was given a unique numeric code which was recorded as Fault_Code. Each fault in the class was given a unique fault identifier which is recorded in the field Fault_ID. The description of the fault is stored in Fault_String. For example, for Publishing Fault, Fault_Class was given as "Publishing", Fault_code was given the code 100, Fault_String was given as "Format Fault" and Fault_ID was given as 101.

The data gets updated whenever a web service fault occurs during runtime. Fault details of Atomic (Single) and Composite web services are updated in this table. The details of data stored in the Runtime Fault Log are given in Table 2.

Table 2: Runtime Fault Log

Data Field	Description
Service_ID	Web Service Identifier
Service_Provider_Name	Service Provider name
Service_name	Name of the specific web service
Fault_Class	Fault Classification description
Fault_Code	Fault Class ID
Fault_ID	Fault ID
Fault_String	Fault Description
Date_Time_Stamp	Date and time of invocation of web service

3.2 Reliability Analysis

Reliability is quantified using Mean Time Between Failures (MTBF). The formula for

Journal of Theoretical and Applied Information Technology
20th July 2014. Vol. 65 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

calculating the MTBF is given in Equation 1 [Musa et al [16] .

$$\Theta = T/R \quad (1)$$

Where Θ represents MTBF, T is the time slot for which reliability is calculated and R gives the number of failures within the time slot T.

Then Failure Rate is calculated as the inverse of the MTBF. The formula for Failure Rate (λ) is given by Equation 2.

$$\lambda = 1/\Theta \quad (2)$$

Reliability (R) is calculated as Exponential of Failure Rate and is given by Equation 3.

$$R = e^{-\lambda T} \quad (3)$$

4. IMPLEMENTATION

IBM Rational Application Developer IDE has been used for the implementation of runtime Fault Detection system. jUDDI (Java Universal Description Discovery and Integration) has been used to configure the web service registry. The web services are published in the jUDDI registry which is implemented using MySQL database. SOAP handlers have been configured using JAX-RPC to detect faults in the execution of web services.

The policies which define the intended behavior of web services have been generated using WS-Policy standard. Web services have been composed using BPEL for static composition and semi dynamic composition and OWL-S has been used for dynamic composition.

A sample web service application for Railway Reservation was developed to record fault behavior. The list of services used and their description is given in Table 3.

Table 3: List of Web Services for Railway Ticketing Application

S.No	Web Services	Description
1.	Train Enquiry	Allows the users to determine the list of trains to the intended destination
2.	Ticket Availability	Allows users to find out whether seats or berths are available in the train of their choice. They can also obtain information about the number of seats available in the required class.
3.	Ticket Reservation	Ticket Reservation service is a composite web service, comprising of the Ticket Booking and Ticket Payment services. The user is returned a PNR(Passenger Name Record) number, that is used to uniquely identify the reserved tickets
	Ticket Booking	The user first provides details for booking the tickets which includes Date of Travel, Train Number, Train Name, Departing station, Destination station, Class, Name, Age, Payment details etc.
	Ticket Payment	Ticket 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
4.	Reservation Status Enquiry	Requires the passenger to key-in the PNR number and the reservation status is displayed.
5.	Ticket Cancellation	Allows the users to cancel their seats of their choice by giving the PNR number

Faults were injected in the application and faults during publishing, discovery, binding, composition and execution were recorded in the proposed log formats. For example, Figure 2 shows the faults injected in services offered by a particular service provider.

Journal of Theoretical and Applied Information Technology
20th July 2014. Vol. 65 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

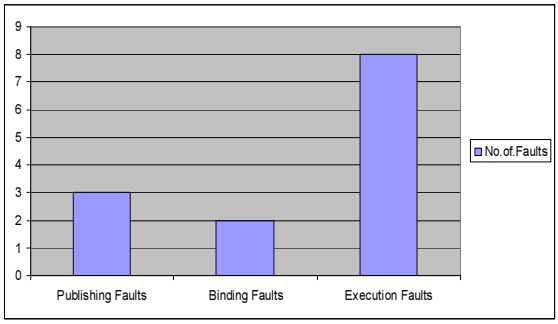

Figure 2. Faults occurrence in specific service provider

Figure 3 shows the fault injected in a particular service; PNR service which is used for Ticket Reservation Enquiry.

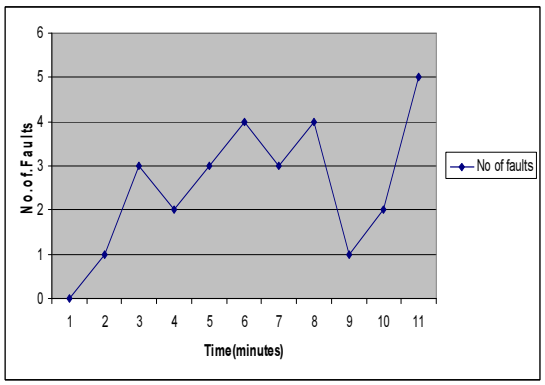

Figure 3: Fault occurrence in Execution of PNR Service

Ticket Reservation Service is a composite web service which comprises of two atomic services; Ticket Booking and Ticket Payment. Figure 4 shows the fault injected in the two component services.

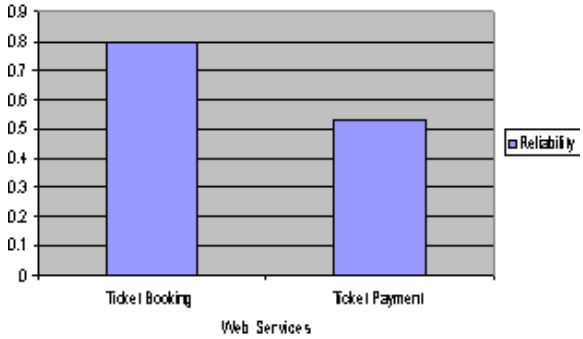

Figure 4. Faults in Composite Ticket Reservation Service

Multiple instances of the web services were created for reliability analysis and the service details are given in Table 4.

Table 4: Web Services Details Used For Reliability Analysis

Log Data	200
Number of Web services	20
Number of service providers	10
Number of service clients	10
Number of web service execution instances	200
Publishing faults injected	20
Discovery faults injected	13
Binding faults injected	9
Execution faults injected	35
Composition faults injected	45

4.1 Reliability Estimation and Prediction

Metrics-based Software Reliability Assessment Tool (M-SRAT) has been used for software reliability assessment. M-SRAT is a JAVA application that offers multiple models for software reliability. Reliability estimation and prediction was carried out using Jelinski-Moranda (J-M) and Goel-Okumoto (G-O) models.

4.1.1. Jelinski-Moranda model (J-M)

This is one of the earliest and the most commonly used model for assessing software reliability. In this model the failure time is calculated as proportional to the remaining faults. This is taken as an exponential distribution. Reliability estimation is given by Equation 4.

$$(MTBF)\ t = 1 / (N – (I – 1)) \qquad (4)$$

where, N is the total number of faults, I is the number of fault occurrences, MTBF is the Mean Time between failures and T is the time between the occurrence of the (i-1).

The sample data obtained in the fault log is given as input in.csv format and the graph obtained using the estimation procedure of the model is shown in the Figure 5.

Journal of Theoretical and Applied Information Technology
20th July 2014. Vol. 65 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

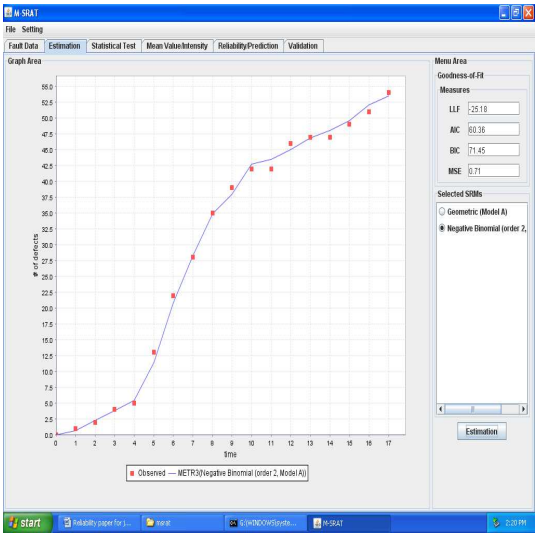

Figure 5: Reliability Estimation of Jelinski-Moranda model

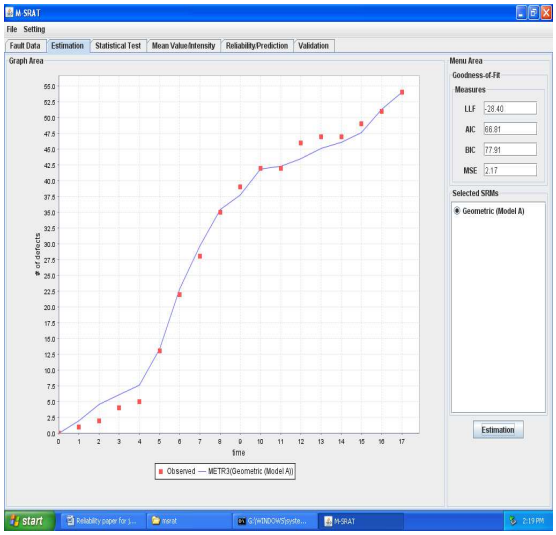

Figure 6: Reliability Estimation of Goel-Okumoto's model.

Red dots show the observed errors in the graph. It can be observed that the number of defects observed map closely with the curve obtained with the model.

4.1.2 Goel-Okumoto Model (G-O)

Goel-Okumoto's model assumes that the cumulative number of failures follows a Poisson process [15]. The data required by Goel-Okumoto's model consists of :

1. (n1,n2,….,nk), the number of faults detected for each time interval

2. (t1,t2,…..tk), the time intervals for which the.(n1,n2,….,nk) are observed.

The graph obtained using the sample data and the estimation procedure in the tool is shown in the Figure 6. The cumulative number of faults detected in the web services is shown in the graph as red dots.

It can once again be observed that the number of defects observed map closely with the curve obtained with the model.

Figure 7 shows the comparison of the reliability prediction curves of both the models.

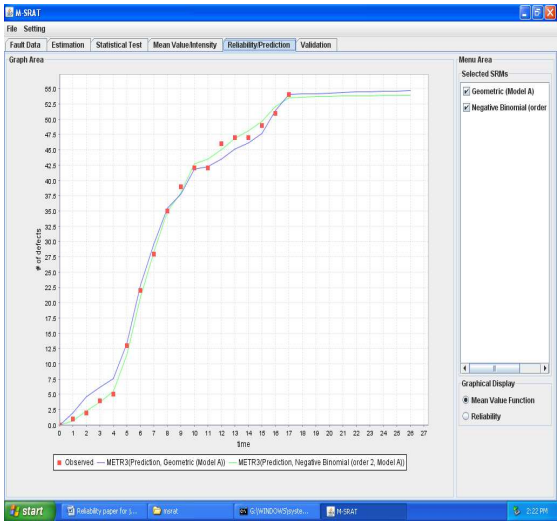

Figure 7: Comparison of Reliability Prediction by Jelinski-Moranda and Goel-Okumoto models.

Reliability prediction helps the service providers to understand the performance of their web services offered by them. The service providers can correct their faulty service and improve the reliability of their web services.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. CONCLUSION

This paper presents a reliability analysis of web services based on runtime faults. Runtime monitoring of web services has been carried out for a sample application which was injected with faults. Log formats were designed to store the service usage and fault occurrences in the services. Reliability estimation was performed on the fault data using Jelinski Morando model and Goel-Okumoto models and the results obtained have been presented. It was observed that the number of defects observed was found to map closely with the reliability curve obtained with the model. This helps to establish the validity of the fault detection process. Reliability prediction was carried out for the two models and compared. This analysis helps the service providers to understand and improve the reliability of the web services offered by them.

REFERENCES

[1] Papazoglou, M. P., Georgakopoulos, D.: Service-oriented computing, Communications of the ACM, Vol. 46, No. 10, 2003, pp. 25–28.

[2] L. Arockiam and N. Sasikaladevi Simulated Annealing Versus Genetic Based Service Selection Algorithms International Journal of u- and e- Service, Science and Technology Vol. 5, No. 1, March, 2012.

[3] Laprie 1992 J-C Laprie editor Dependability: Basic concepts and Terminology, Springer-verlog 1992.

[4] Alan Wood, software reliability Growth Model, Technical Report 96.1,September 1996, Part Number: 130056.

[5] Karanta, Ilkka Methods and problems of software reliability estimation VTT Technical Research Centre of Finland 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)

[6] Y.Pan Will Reliability Kill the Web Service Composition? citeseerx.ist.psu.edu/ viewdoc /download? doi=10.1.1.136.5893

[7] M. R. Lyu. Handbook of Software Reliability Eng. McGraw-Hill, New York, 1996.

[8] J.D. Musa, A. Iannino, and K. Okumoto, Software Reliability Measurement Prediction Application, 1987: McGraw-Hill.

[9] Criteria Analysis and Validation of the Reliability of Web Services-oriented Systems JiaZhang, Liang-Jie Zhang Proceedings of the IEEE International Conference on Web Services (ICWS'05)

[10] W. T. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul, N. Lia A Software Reliability model for web services

[11] Senol Arikan Automatic Reliability Management in SOA-based critical systems

[12] N.Sasikaladevi, Dr.L.Arockiam Reliability evaluation model for composite services International Journal of Web & Semantic Technology Vol 1 No. 2 April 2000.

[13] XIE Chunli LI Bixin, LIAO Li WANG Xifeng Combining Control Structure and Composition Condition for Web Services Reliability Prediction Chinese Journal of Electronics Vol.21, No.3, July 2012.

[14] Zibin Zheng, Zheng, Z, Michael, R Lyu 'Collaborative Reliability Prediction of Service-Oriented systems 2010 , proceedings of the 32nd ACM/IEEE International conference on software Engineering, pp 35-44.

[15] K.Jayashree, Sheila Anand, Policy Based Distributed Runtime Fault Diagnoser Model for Web Services, LNICST 2012 pp 9-16.

[16] Software Reliability Engineering: Techniques and Tools CS130 Winter, 2002.