A Novel Framework for Reliable and Fault Tolerant Web Services

Akhilesh Kumar Pandey, Abhishek Kumar, Shuchita Shukla

Abstract*— In the highly competitive software industry there is a significant needs for interoperable robust, reliable and secure services in almost every practical application. Reliable and fault tolerant web services have attracted increasing interest from researchers in recent years. Various components and architectures have been developed to enhance the communication between consumer and providers. In all those cases, web services need to be designed to accomplish robustness and reliability. After examining various approaches, we have introduced a framework which is designed by keeping the reliability and fault tolerance challenges into consideration. In this paper, web services are designed and created by Replication Manager. The Replication manager selects and registers a primary web service to provide services to consumer out of multiple replicas of web services. The consumer access the WSDL (Web Service Description Language) through UDDI (Universal Description, Discovery and Integration) repository. The consumer invokes the web service using WSDL. The Replication manager keeps track of availability of web services and switches & register accordingly to the Replication Manager in case of unavailability which again results in updating the WSDL for the next request from consumer. Result and analysis shows a high curve of performance and achieves a robust and reliable state of execution. Further, the framework has been enhanced using the Replication Manager as a fault tolerance mechanism.

***Index Terms*—Web service, Fault tolerance, Reliability, Replication Manager.**

I. INTRODUCTION

In real world, the development model of solution and infrastructure is built around internet based application to incorporate business requirements and to promote the availability of the business process.

To achieve this, various organizations started adopting web services in present to automate their business process [1]. Various E-commerce and Banking applications have exposed their resources using web services to retain their complex behavior across various platforms and different technologies. These complex applications need to be run continuously for years without any inconsistencies. To fulfill these business requirements, frameworks need to be automated and defined in such a way that it can achieve reliable and fault tolerant state. It should be designed in such a way that the business process doesn't get affected by additional changes. Web service provides a better solution to solve the problems of platform independence and compatibility issues across various technologies [2]. But, the technology is also lagging behind because of their quality of service during the web service selection. Hence, it is necessary to address those issues in order to retain its trust on consumers and providers while selection web services which make the researchers to realize the importance of fault tolerance techniques [1-3, 5, 6]. In this paper, we have employed fault tolerance as a solution to one of the key issues faced by a mass in web services.

II. TECHNOLOGY OVERVIEW

In a web service environment, a typical consumer-provider communication includes following steps:

Medium: To communicate between consumer and provider over network in order to exchange the wavelengths, a medium is required.

Protocol: Medium is not enough to initiate the communication. In order to standardize the communication to exchange the information with their independent bands, a physical independent transport protocol is required. In web services, HTTP (Hyper Text Transfer Protocol) is recommended to use.

Language: Even after having medium and protocol two parties need to exchange the information over a common language. i.e, csv, text, MS-Excel etc.

The problem with these formats is that it is not self explanatory and platform dependent. But to build an interoperable solution, only way of carrying data between nodes is XML (Extensible Markup Language). XML became defacto standard for exchanging data because it is self explanatory and it contains well defined structure & semantics [1].

Binding Protocol: Along with XML representing business specific data, a party may need to send transport specific or additional information. XML doesn't manage to differentiate between business data and helper data. SOAP (Simple Object Access Protocol) is also a XML which provide different sections to manage and differentiate data.

Revised Manuscript Received on December 22, 2018.

Akhilesh Kumar Pandey, West Corporation, Bangalore, India
Abhishek Kumar, Department of Computer Science and Engineering, MIET Meerut, U.P, India
Shuchita Shukla, ABB Ltd, Bangalore, India

All the helper data can be placed into SOAP header and business data can be send as part of SOAP body. It acts as a Application specific protocol because the receiver of the SOAP XML can identify the difference between different data [7, 8].

WSDL (Web Service Description Language): Consumer always request for information and provider always provides information. To get the data from provider, consumer need to know about the provider. To provide interoperable distributed standard documentation, WSDL has been introduced. It acts as a contract between consumer and provider. WSDL has a predefined structure and has grammar. WSDL is used to provide the information about the resources such as service name, No. of operations, parameters and return types to the consumer [8-9].

UDDI (Universal Description, Discovery and Integration): UDDI is a registry which is distributed across network and can be accessed by any consumer in order to access the WSDL. Consumer locates the WSDL from UDDI registry. UDDI is interoperable and can be accessed using any programming language. Private UDDI's are internally maintained by an organization and only accessible within the organization. Whereas, public UDDI is accessed globally over the public network. Figure 1 depicts the basic architecture of consumer-provider communication in a web service environment [7, 8, 9].

III. RELATED WORK

Various researchers has been working since very long time to make their framework handle selection of multiple web services in order to reduce the errors during runtime in production environment. Soonwook Hwang and Carl Kesselman [3] had given a Grid Workflow to handle

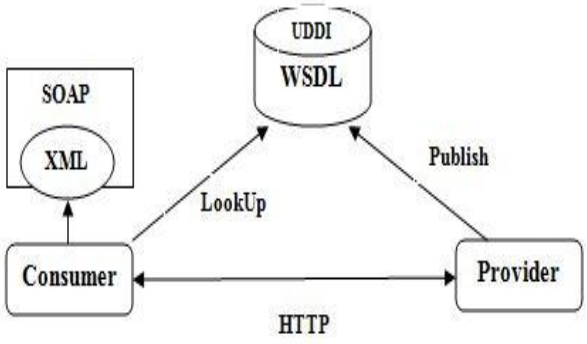

Figure 1: A Typical Web service Architecture.

failure for the grid in 2003. The structure of the workflow enables its users to identify failure handling techniques and also a way to change the workflow structure by modifying. The analysis done in the paper shows high fault recovery in grid system with greater performance during failures. Jorge Cardoso [2] proposed a predicted QoS model for workflows management system in 2004. This model has been implemented using different features in consideration to identify and analyze the QoS on runtime. Shuying Wang [10] had proposed an agent-based Web service workflow model in 2006 to manage the inter as well as intra- enterprise workflow process. Service agents are used to collaborate the existing heterogeneous in order to facilitate the flexibility and reusability of process. Jorge Salas et. al. [11] provided a framework to increase the availability of the web services using replication. The model ensures the availability of web services by distributing the application across various site which further uses web service multicasting. The model has been tested and analyzed as per with the standards defined by WS-I. Rohit Kumar Shukla [12] in 2006 provided an ideology to accept exception management as a enterprise solution in web service to avoid the usage of complicated frameworks and other utility components. Stefan Bruning [13] at 2007 provided a fault taxanomy for SOA (Service Oriented Architecture). This paper focuses on defining SOA specific faults on top of basic faults occurs in web service environment with an use case of airlines management system. Vassiliki Diamadopoulou [1] suggested various ways of supporting the selection and consumtion of web service with QoS characteristics in 2008 to support web service discovery and solution using tols & services. WSDL is used to identify QoS information from registry and store it and the compare them to identify the most valuable characteristics for the same web service. Walaa Nagy [15] suggested a flexible and effective tool for the selection of web services in SOA in 2012. The main idea behind the algorithm is to integrate legacy match making techniques with data ware housing techniques to track the history of preference of user for better selection approach. The feasibility of the framework is demonstrated and enhances the performance and reliability. Thirumaran .M [3] in 2012 presented a exception handling mechanism during runtime for web service. The paper is build around exception context by assuming time and policy constraints to determine whether the exception can be handled or not. A runtime exception handling has been proposed by taking QoS attributes as parameters which further yields to manage the runtime exceptions and performance upgradation. Quanyu WANG [16] had proposed a policy based exception handling technique for web services in 2012 by employing exception handling description language (EHPDL-P) on exception handling framework (EHP-F) for BPEL (Business Process Execution Language) process. Similarly, the same year JIANG Caoqing [17] proposed a model to handle exceptions in BPEL process based on petri net. The paper differentiates different process modules and provides an integration model for normal process and exception handling in BPEL Chien-Cheng Lin [18] in 2013 suggested a portable interceptor technique for SOAP (Simple Object Access Protocol). The paper proposes a mechanism consists of APIs (Application Programming Interface) on SOAP engine and a core component which receives the passing message. The module provides runtime lifecycle management.

IV. PROBLEM STATEMENT

There are various techniques which can be applied to web services in order to provide reliable and fault- tolerant services to its consumers using replication and diversity. Replication and diversity are one of the efficient mechanisms to build reliable and fault tolerant systems. Redundancy has been used quite long ago to ensure availability of resources and components in distributed systems.

In this paper, our main goal is to focus on the systematic analysis of replication and diversity mechanism when applied to web services in order to insure reliability. Here, we also investigate the performance and web service availability using various replication techniques. A novel framework for web services has been proposed that insure reliability and fault tolerance of web services.

V. TECHNIQUE FOR RELIABLE AND FAULT TOLERANT WEB SERVICES

In web services when an fault occurs, it goes into various stages [18] . When an error occurs in web services, it should make sure the error or faults through various fault detection mechanism to know the failure causes so that failed components can be repaired of recovered from an error. The flow of web service failure responses shown in figure 2.

A) **Error Confinement**: Error confinement stage prevents an error effect on web services. It can be gain with the help of error detection within a service by multiple checks.
B) **Error Detection:** Error detection stage helps in identifying unexpected error in a web service.
C)**Error Diagnosis:** Error diagnosis stage helps to diagnose the fault that has been traced in error detection stage. Error diagnosis stage comes into picture when error detection doesn't provide enough information about fault location.
D) **Reconfiguration:** Reconfiguration comes into picture when and error is detected and located in the error detection and error diagnosis stage.
E)**Recovery:** Recovery is used to recover fault from web service using retry and rollback approaches.
F) **Restart:** Restart comes into picture after the recovery of web service. Restart can be done either using hot start or cold start.
G) **Repair:** In Repair, failed component has to be changed in order to work properly.
H) **Reintegration:** In the reintegration stage repaired component has to be integrating.

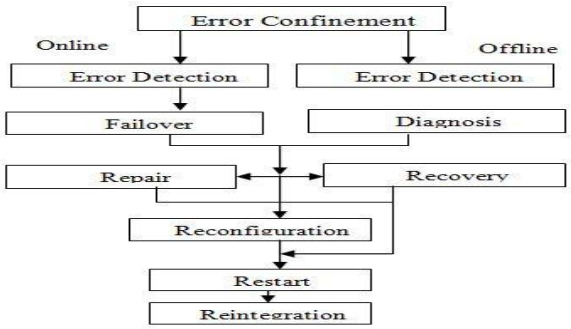

Figure 2: Failure Stages of Web services

In web services, there are many fault tolerant techniques that can be applied such as replication. Replication is a more efficient technique for handling exception in a distributed application. Services can resume more effectively by maintaining the global state of the application. For instance, let's assume if one service needs the assistance of another service to provide the desired result to the customer then service needs to communicate with other service. Suppose, while communicating with other service, at certain point of time if a fault occurs in a service, then there is no need to continue service with faults. Then the state manager has to rollback the state of the application at that point where the fault occurred so that service can resume without fault and response can be given to the consumer more effectively.

VI. PROPOSED WORK

A. FAULT PEVENTION (FP) ALGORITHM

In *FAULT_PREVENTION* (FP) algorithm, S represents services, T represents time intervals and F represents faults. Here S_i & S_j are root and sub services directories. Root service directory S_i is communicating with its sub service directory S_j (where i=0 to n-1 & j=1 to n) which takes time T_k (k= 1 to n). During service communication when a faults occurs rollback the service from its previous state otherwise service will be executed normally.

Fault_Prevention(FP) Algorithm

```
FP(S,T,F)
1.      {
2.        if(Si = = Sj)              // i  <-  0 to n-1
3.          {                        // j   <-  0 to n
4.        T = Tk                  // k   <-  1 to n
5.    if (F = = Fl )        //  l  <-  1 to n
6.       {                  // rollback
7.        T= Tk-1;
8.        FP(S,T,F);
9.        }
10.   else
11.      {
12.       Continue ;
13.       }
14.    }
15.  }
```

B. FLOW CHART FOR FAULT PREVENTION ALGORITHM

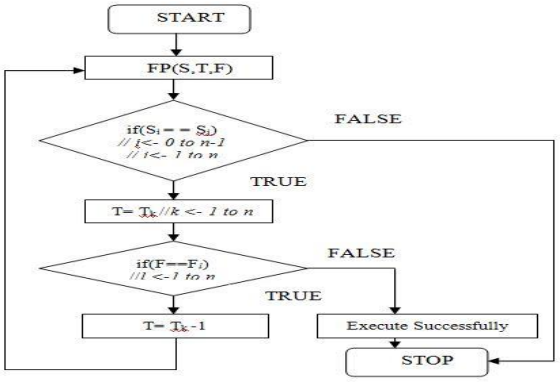

Figure 3: Flow chart diagram Fault Prevention (FP) Algorithm

C. PROPOSED FRAMEWORK

In web services whenever a fault encounters, these responses of the web services can be categorized into various types of stages [19] [20] such as fault detection, diagnosis, reconfiguration, recovery, repair etc.

In the proposed framework (figure 4) usually first Application Server(WebLogic) works as an active server and in the case of failure, remaining can be used for backup purpose. Here the web services are also hosted on all the servers, therefore whenever the primary web service fails to provide desired result then other web services can provide the desired result. Replication Manager that runs on Application Server keeps checking web service fault reported by the application. Then it will take an appropriate action based on the type of fault. In this case when Replication Manager doesn't get a reply from the active primary web service, then it will choose another web service and map the new address to WSDL document and register it with UDDI registry thus consumer can still access the web service with the same URL.

As shown in the Figure 4, Replication Manager is the key component of the proposed framework. It is responsible for performing following fundamental operations in order to provide fault tolerance in services:

a. Developing and marking a primary web service.
b. Registering WSDL with the UDDI registry.
c. Continuously checking faults reported by the application.
d. Whenever Replication Manager will get an faults reported by web service application, then it will transmit a request to an another web service application to ensure FT (Fault Tolerance) else reconfigure the WSDL document and maps it to a UDDI registry.
e. Replication Manager will also replace services; those are faulty to ensure FT (Fault Tolerance).

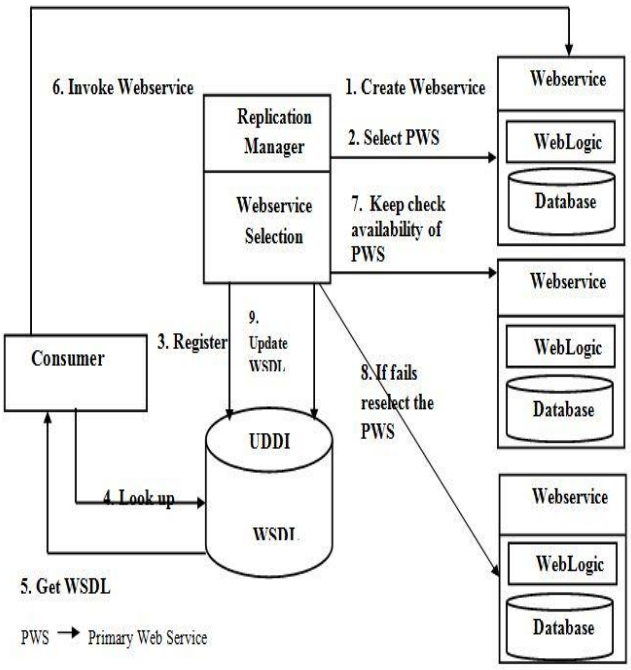

Figure 4: Novel Framework for Reliable and Fault Tolerant Web services

Working flow of Replication manager has been shown in figure 5, when an primary web service (PWS) encounters an exception then the replication manager selects an most appropriate web service to continue providing services to consumer. Here the replication manager (RM) maps the new address of web service to the WSDL so that still consumer will be able to call provider with same URL. This process is transparent to the consumers. The replication manager that is up and running on the application server keeps checking availability of web service continuously. Replication manager (RM) sends a message to web services repeatedly, if replication manager doesn't get a reply from a primary web service (PWS) with an time interval, it'll select another web service and will map the address of newly selected web service to a WSDL and mark it as a primary web service (PWS).

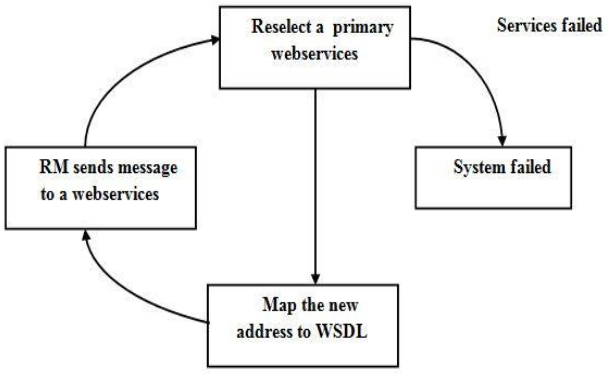

Figure 5: Working flow of Replication Manager

STEPS FOR REPLACING FAULTY SERVICES

In order to replace faulty web services, Replication Manager is used as a key component of proposed framework as shown in Figure 4. Here following steps are taken into consideration for replacing faulty services,

[1] Lookup a service from the service registry (UDDI) , which returns WSDL documents. Let's assume, it is not returning *null*, where the 'while' loop in figure 4 becomes true.

[2] For each service registered with UDDI calculate the degree of fault using Replication Manager. Let's take a range of *degree_of_ fault* between 0 to 1. If the value *of degree_of_ fault* is equal to 1, this means web services are free from error.

[3] If the value of *degree_of_ fault* is less than 1 then roll back the previous state of the web service by taking the help of Fault Notifier. It can be made possible by managing state of the web service.

[4] If the *degree_of_ fault* is less than 1, then the service is likely to be error prone.

[5] Reconfigure and again invokes the web service which is exactly same as the invocation of the primary web service.

E PSEUDOCODE FOR REPLACING FAULTY SERVICES

There are following process required to replace faulty services as shown follows-

Pseudocode for Replacing Faulty Services

```
1.while(service == ServiceRegistry)
2. {
3. Consumer <- WSDL Document;
4. if(df == 1) // degree of fault
5. {
6.     message("service is error free");
7. }
8. else
9. {
10.       message("service is error prone");
11.       rollback();
12.       reconfigure();
13.       reinvoke();
14.   }
15.   }
```

VII. EXPERIMENT RESULTS AND ANALYSIS

A number of experiments are designed and performed in order to evaluate reliability and fault tolerance of web service. Our framework is implemented in Apache CXF that runs with java technology on Web Logic application server. The web service is deployed on different boxes(machines) and the primary web service (PWS) is chosen by replication manager (RM).In the experiments, faults are injected in the framework[21][22].In an web service environment a number of faults may occurs such as business fault , network fault including resources problem , entry point failure problem and etc. These are the faults injected into the framework in order to evaluate reliability and fault tolerance of the proposed framework. In an our experiments , we compare total five approaches for measuring reliability and fault tolerance for proposed framework , They are as follows:

a. Single Web service without Retry and Reboot
b. Single Web service with Retry
c. Single Web service with Restart
d. Spatial Replication with Failover
e. Spatial Replication with Retry and Failover

The web service is experimented for 720 hours by creating a 720X60 request per hour (43200) from the consumer. Faults in a web service considered when a framework cant responds to the consumer. (In the case of retry, a fault is considered when a consumer retrieves five times but yet can't get response back. A total experimental analysis is presented in figure 6, 7, 8 & 9 shows the number of failures with respect to the time.

The reliability and fault tolerance of proposed framework tested under various circumstances such as resources problem, network problem including normal operation as well. Resources problem crated in the framework by increasing loads on the application server (*Weblogic*) and entry point failure created in the framework by restarting the application server.

Experiments over 720 hour period(720X60= 43200 requests)	Normal	Resource Problem	Network Level Faults
Single service without retry & reboot	4928	6130	5324
Single web Service with retry	2210	2327	2289
Single web Service with restart	2561	3160	5211
Spatial replication with failover	1350	1711	5258
Spatial replication with retry & failover	1089	1148	2210

Figure 6: Experimental Results

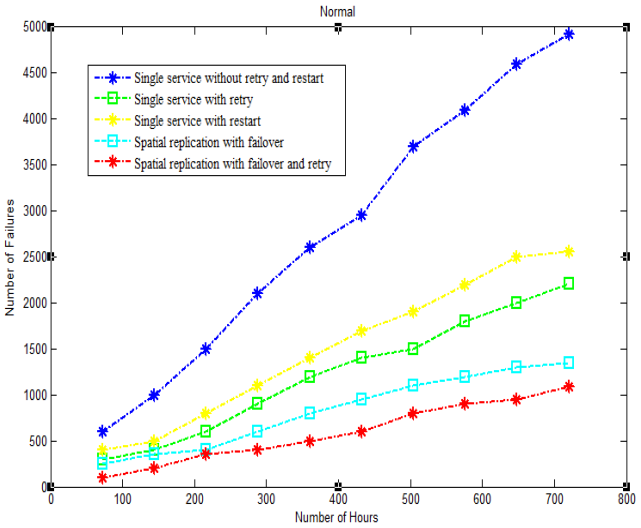

Figure 7: Number of failures when application server up and running normally.

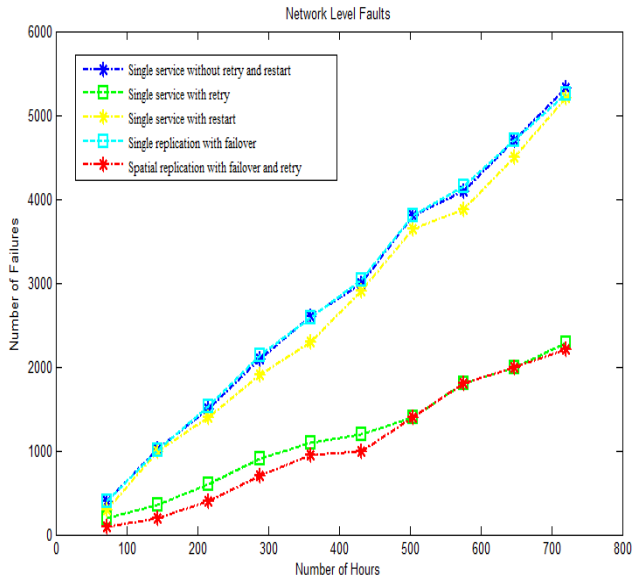

Figure 8: Number of failures under network problem

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The results of experiments narrowly denotes rather than linear increase. Therefore the ratio of number of failures with respect to time all most constant for each and every experiment. In the proposed framework, when there is no redundancy (single web service without retry and restart) mechanism applied then it shows the failure ratio is more. Hence here we try enhancing reliability and fault tolerance of framework using redundancy techniques such as spatial redundancy with replication and temporal with retry or reboot.

Under resource problem our proposed framework improves the reliability and fault tolerance of web service, Here the experimental result shows that the temporal redundancy helps to improve the availability of web service. For the web service with retry the % of failure decreased from 10.25% to 3.93%, this indicates that the temporal redundancy with retry improves the reliability and fault tolerance of the web service. In the case of web service with reboot also failure ratio is decreased from 10.25% to 5.32%. In the case of spatial redundancy failure rate is reduced from 10.25 to 3.32 %. In the case of spatial and temporal the failure ratio is decreased from 10.25% to 2.25%. In the case of hybrid approach (spatial and temporal redundancy) the highest reliability and fault tolerance is achieved. Under Network faults temporal redundancy decreases the failure ratio from 11.22% to 4.14%. When there is a fault occurred into SOAP message then consumer request can be processed and it stops with *SOAPFault &* entered into failed state.

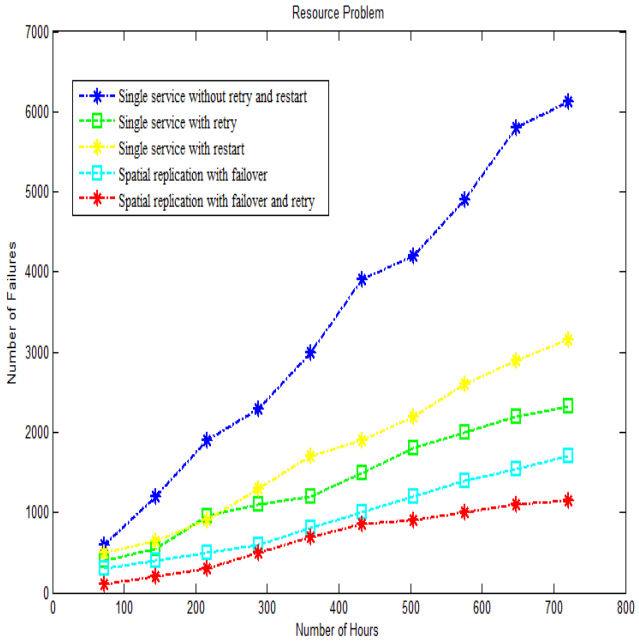

Figure 9: Number of failures under resource problem

VIII. CONCLUSION

In this paper, various techniques for FT (Fault Tolerance) in web services application are surveyed and addressed that take care of reliability and extensibility of web services into consideration. A proposed methodology reduces the number of failures to ensure FT (Fault Tolerance) in compared to traditional approaches. In this paper's methodology for replacing faulty services to ensure reliability and FT (Fault Tolerance) services to consumers are also discussed.In this paper web service stack of standards is in use, in addition to this FT (Fault Tolerance) is added to the web service architecture to insure reliability of web services. Future work includes an approach for repairing faulty web services in an optimal time and selecting appropriate web services to resume casual workflow of services.

REFERENCES

1. Vassiliki Diamadopoulou, Christos Makris, Yannis Panagis, Evangelos Sakkopoulos: Techniques to support Web Service selection and consumption with QoS characteristics. Elsevier, Journal of Network and Computer Applications 31 (2008) 108-130.
2. Jorge Cardoso, Amit Sheth, John Mille, Jonathan Arnold, Krys Kochut: Quality of service for workflows and web service processes, Elsevier, Web Semantics: Science, Services and Agents on the World Wide Web 1 (2004) 281-308
3. Thirumaran.M, Dhavachelvan.P, Shanmugapriya.R, Kiran Kumar Reddy: A Novel Approach for Web Service Run Time Exception Handling. 2nd International Conference on Communication, Computing & Security [ICCCS-2012]
4. Soonwook Hwang; Kesselman, C., "Grid workflow: a flexible failure handling framework for the grid," High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on , vol., no., pp.126,137, 22-24 June 2003
5. Shaofei Wu, "A New Method of Exception Handling in Workflow," Intelligent Ubiquitous Computing and Education, 2009 International Symposium on , vol., no., pp.420,422, 15-16 May 2009
6. Zhao Kaiu; Zhang Linlin; Ying Shi, "A flexible exception handling framework for Semantic Programming Language," Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on , vol., no., pp.2199,2204, 19-22 Aug. 2011
7. Walaa Nagy, Hoda M. O. Mokhtar, Ali El-Bastawissy: A Flexible Tool for Web Service Selection in Service Oriented Architecture. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 12, pp. 191-201, 2011
8. Georgios John Fakas, Bill Karakostas: A peer to peer (P2P) architecture for dynamic workflow management. Information and Software Technology. 46 (2004) 423–431
9. Cavanaugh E.: Web services: Benefits, Challenges, and a Unique, Visual Development Solution (White Paper). 2006.
10. Shuying Wang, Weiming Shen, Qi Hao: An agent-based Web service workflow model for inter-enterprise collaboration. Expert Systems with Applications 31 (2006) 787–799
11. Jorge Salas, Francisco Perez-Sorrosal , Marta Patiño-Martínez , Ricardo Jiménez-Peris: WS-replication: a framework for highly available web services. Proceedings of the 15th international conference on World Wide Web, May 23-26, 2006, Edinburgh, Scotland
12. Rohit Kumar Shukla: Exception Handling in Service-Oriented Architecture. SAP Developer Network, Nov. 2006.
13. Bruning, S.; Weissleder, S.; Malek, M., A Fault Taxonomy for Service-Oriented Architecture. High Assurance Systems Engineering Symposium, 2007. HASE '07. 10th IEEE , vol., no., pp.367,368, 14-16 Nov. 2007 .
14. Shaofei Wu: A New Method of Exception Handling in Workflow. Intelligent Ubiquitous Computing and Education, 2009 International Symposium on, vol., no., pp.420 422, 15-16 May 2009
15. Quanyu Wang; Guobin Lv; Shi Ying; Jing Wen: A policy-driven exception handling approach for service-oriented processes. Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on , vol., no., pp.449,455, 23-25 May 2012
16. Jiang Caoqing; Ying Shi; Hu Shanming; Xu Hui; Qiang Yueming: A formal model for exception handling in BPEL process. Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on, vol., no., pp.2072,2077, 29-31 Dec. 2012.

17. Chien-Cheng Lin, Chen-Liang Fang, Deron Liang: A portable interceptor mechanism for SOAP frameworks. Computer Standards & Interfaces 36 (2013) 209–218
18. M. Lyu and V. Mendiratta, “Software Fault Tolerance in a Clustered Architecture: Techniques and Reliability Modeling,” In Proceedings' of IEEE Aerospace Conference, Snowmass, Colorado, vol.5, pp.141-150, 6-13 Mar. 1999.
19. Lyu, M.R. Mendiratta, V.B., "Software fault tolerance in a clustered architecture: techniques and reliability modeling," Aerospace Conference Proceeding sof IEEE , vol.5, no., pp.141,150 vol.5, 1999.
20. T. Anderson and P. A. Lee. Fault Tolerance - Principles and Practice. Prentice-Hall, 1981.
21. M. Merideth, A. Iyengar, T. Mikalsen, S. Tai, I. Rouvellou, and P. Narasimhan,“Thema: Byzantine-Fault-Tolerant Middleware for Web-Service Application, ”Proceedings of IEEE Symposium on Reliable Distributed Systems, Orlando, FL, Oct. 2005.
22. Y. Yan, Y. Liang and X. Du, “Distributed and collaborative environment for online experiment system using Web services,” Proc. the Ninth International Conference on Computer Supported Cooperative Work in Design ,vol.1, pp.265-270, 24-26 May 2005.