ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 12(43), DOI: 10.17485/ijst/2019/v12i43/148336, November 2019*

Web Services Failures and Recovery Strategies: A Review

Nimra Memon*, Muhammad Saleem Vighio and Zahid Hussain

Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah, Pakistan;
memonnimra03@gmail.com, saleem.vighio@quest.edu.pk, zhussain@quest.edu.pk

Abstract

Objectives: Due to heterogeneity, cross-boundary integration, and deployment over the Internet, Web services are highly vulnerable to a wide variety of failures. This study provides an overview of different types of failures, and recovery strategies for Web services. **Method/findings**: To conduct this study, we have reviewed several novel research studies to provide a precise and all-in-one summary of different types of failures and possible recovery solutions for Web services. The study reveals that, a clear understanding of different failures-types and possible recovery solutions will help to develop services which are highly reliable and dependable. **Applications**: Highly reliable and dependable Web services are the key focus of all sensitive and mission-critical applications like navigating systems in aircrafts, nuclear reactor systems, robotics, and so on.

Keywords: Web Services, Failures, Recovery Strategies, Fault-tolerance

1. Introduction

With the growing use of the Internet and mobile technologies, Web services have gained much popularity since last few years.[1,2] In one or another way, we use Web services in our daily lives, for example, paying our bills, booking a taxi, or reserving a table in a restaurant.[3–5] A task performed by the Web service can be as simple as converting one type of currency to another, or it can be a complex task requiring multiple services to coordinate and collaborate to perform that task jointly.[6,7] Performing a complex task jointly requires services to interact over the unreliable Internet and beyond their organizational boundaries under heterogeneous environments.[8,9] This makes Web services vulnerable to a wide variety of failures that may range from simple inconvenience to a significant financial or monetary loss. A service may fail due to many reasons like service unavailability or down-time, logic errors, inconsistent or incompatible inputs and so on.[10] However, because of their use in important and critical applications, services are required to be highly reliable.[11] Efforts to produce reliable Web services are under way,[12–15] but, it is a very challenging task due to the unreliability of Internet, heterogeneous and cross-boundary interaction, incompatible business logics and so on. This study presents a survey of different types of failures which affect the normal execution of Web services. Furthermore, different types of recovery strategies to safeguard from such failures have also been presented. It is believed that a thorough understanding of different types of failures and corresponding recovery strategies will help to design Web services which are resilient to failures.

The rest of the study is structured as follows: Section 2 presents an overview of different types of failures which often occur during the execution of Web services. Section 3 gives an overview of different types of strategies used to recover from services failures. Section 4 presents discussion, and finally, section 5 gives the conclusion of the work.

*Author for correspondence

2. Web Services Failures

Web services are actually software applications designed to perform specific task(s) using the Internet. In addition to the containment of all features of traditional software, Web services also contain some additional features like autonomy, heterogeneity, and interoperability. Like traditional software, Web services also suffer from errors and failures from development to execution.[16] Moreover, due to their heterogeneous and cross-boundary interaction, and deployment over the Internet, which is an unreliable media, Web services are more vulnerable to failures than their traditional counterparts. Different types of failures which affect the execution of Web services are categorized into three general categories: development, physical, and interaction faults.[10,17] Occurrence of any or all these failure-types can be transient or permanent, and can lead to service degradation, unavailability, or complete shutdown. All fault-types are described below:

2.1. Development Faults

Development faults are introduced during the development phase of Web services, but are exposed when services are actually executed. These faults are introduced by the environment, human developers, development tools, and production facilities.[10] Development faults are classified into parameter incompatibility and interface change faults as defined below:

- **Parameter incompatibility faults** arise when services receive incompatible input values other than the expected; for example, a service expects an integer value but is provided a string constant. In that case, the service will end-up in an error or invalid result message.
- **Interface change failures** (or inconsistency failures) occur when the interface or ontology of the service is changed (or updated), whereas, service invocation requests are forwarded to the old interface. This happens due to the unawareness of users from corresponding updates. In some cases, the interface of services is changed, but, the process (logic) is not updated accordingly. For example, in a hotel reservation service, a user requests for the booking of four rooms, but only two rooms are available at that time.

2.2. Physical Faults

Physical faults (also known as system faults) occur due to the failure of a server on which requested service is deployed, or the failure of a network connection. Physical faults result in service unavailability. Services become unavailable due to server shutdown or downtime, for maintenance and update purposes or in cases when the power supply to the server machines is discontinued due to the power breakdowns or natural faults.

2.3. Interaction Faults

Interaction faults are all operational or external faults, which popup during the execution, or the use phase of services. These fault-types are broadly classified into content and timing faults. **Content faults** also referred to as corrupt service faults are further classified into Service Level Agreement (SLA), Quality of Service (QoS), and incorrect service invocation faults, whereas, **timing faults** are classified into semantic, and timeout faults. All these faults-types are described below:

- **SLA faults** are actually the violation of non-functional properties of a service, that is, the service completes successfully but does not conform to predefined service level agreement. For example, the expected execution time of an operation-completion is 12 seconds; but, the service took 20 seconds to complete the task.
- **QoS faults** including also SLA faults occur due to the degradation of service in terms of quality: slow speed or delays in response time.
- **Incorrect service invocation faults** occur when a service is called with an incorrect name instead of the actual name.
- **Semantic faults** occur due to the incompatibility of composed services requested to perform a joint task, for example, in a joint booking of a hotel and a taxi, operation does not complete successfully due to the different time formats of these services.
- **Timeout faults** arise when a component service fails to complete execution within allocated time frame. This happens when the service is overloaded to process many requests at the same time. For example, too many requests for grabbing a cheap ticket may overload the booking service; this may result in excessive delays (timeouts) at the requester's end or even in the unavailability of the service.

All above fault-types can further be classified with respect to different viewpoints during the life time of services. These fault classes can be viewed as development, operational, internal, external, hardware, software, functional, and non-function faults.[10,17] A complete taxonomy of all fault-types with respect to different viewpoints is summarized in Table 1.

As it can be seen in Table 1, different fault-types may belong to different fault-classes and may occur in an overlapping fashion. For example, timeout faults of the interaction faults category can be viewed as operational, external and hardware faults. Occurrence of all or any of the fault-type can leave the service in a failure mode incapable of providing the required functionality.

3. Recovery Strategies

Fault-tolerance refers to the ability of system to detect and recover from failures.[18] Because of their increasing use in sensitive and mission-critical applications, Web services are required to provide desired functionality even in cases of failures. To provide reliable services, various failure-recovery strategies have been proposed in the literature (see Refs.[10,11,19,20]). However, the most commonly used recovery strategies for Web services are described below:

- **Ignore:** As its name suggests, this strategy ignores those faults which do not affect the primary goal of the service. For example, in a sight-seeing booking service, failure of (optional) getSalesInfo service may be ignored as important tasks like booking of flight and hotel have completed successfully.
- **Skip:** Under this strategy, if a service deviates from QoS and SLA logic, then its successive services are skipped to execute conditional to the fact that skipped services do not affect the primary goal of the service composition. For example, if computeDistance service of sight-seeing scenario deviates from its actual execution time, say from 5 sec to 8 sec then getSales Info service is skipped to execute in order to meet the promised execution time of the whole process.
- **Retry:** This strategy re-executes the faulty service to a particular number of times or till the service completes successfully. **Retry** is used to recover from temporary failures caused by the hardware, software, or the network.
- **RetryUntil:** With an addition of time-based re-invocation of faulty service, this strategy is an extension of the "retry" strategy. That is, each re-invocation is constrained to a particular time-stamp. For example, RetryUntil (bookFlight,5,10) re-invoke bookFlight service to a maximum 5 retries with each retry occurring after 10 time-stamps.
- **Wait:** This strategy delays the execution of a service to a specified time instant. For example, Wait (book Flight, 8:00) is used to invoke bookFlight service not before 8:00. This strategy is used to handle service unavailable faults.
- **Alternate:** This strategy selects another functionally equivalent service to perform some task when the first service encounters a failure. Alternative action invokes different service instead of the same service.

Table 1. Web services failure types

Fault-classes	**Development faults**		**Physical faults**	**Interaction faults**				
				Content	**Timing**			
	Parameter incompatibility	**Interface faults**	**Service unavailability**	**SLA**	**QoS**	**Incorrect service**	**Semantic**	**Time-out**
Development	10,17,26	10,17,26,27		27	27			
Operational			17,28–30	10,17,27	10,17	10,17	10,17	10,17,27,28,24
Hardware			17,26,24,30-31	27	27		17,27	17,27
Software	12,13,31,32	12,13,31,18	12,32–35	12,32,18	12,32,18	12,32,18	32,18	32,18
Internal	32,18	32		12	12,32		32	32
External	33,36–39	33,36-38	33,36–38	18,33,37–39	32,36,37	18,36,37,39	18,36,39	33,36–40
Non functional				32	32,35,39,40			13,32
Functional	33,34	33,34	33,34				33,34	

All above recovery strategies can be used individually or in combination with others to handle different types of failures. Table 2 gives a review-summary of different types of failures and their possible recovery strategies.

Table 2. Failure-types and possible recovery solutions

Fault type		Recovery action
Unavailable/unresponsive		Retry,[7,26,30,33,36,38–40,41] RetryUntil,[29,33] Ignore,[26,31,33] Alternate,[31,41] Wait[33]
Syntactic faults	Parameter incompatibility	Ignore,[26,33,41] Alternate,[32,41] Retry[32,41]
	Interface change	Ignore,[26,32,41] Alternate,[31,41] Retry[33,41]
Content faults	QoS	Ignore,[26,31,33,41] Skip,[31,42] Retry,[33,41] Alternate[31,33,41]
	SLA	Ignore,[31,33] Skip,[31,33,41,40] Retry,[31] Alternate[31]
	Incorrect service	Ignore,[33] Retry,[41] Alternate[31]
Timing faults	Timeout	Ignore,[26,33,41] Skip,[31,33] Retry,[33,36,42,21] Alternate[31,33,41,42]
	Semantic	Ignore,[26,33,41] Retry,[33,41] Alternate[33,41]

4. Discussion

Though, much research has been conducted in the area of fault-tolerance of Web services, however, not all faults are avoidable.[21–23] Due to the dynamic, heterogeneous, and cross-boundary integration of Web services deployed over the unreliable Internet, faults become hard to predict and resolve.[4,9] It is possible that a number of faults occur at the same time during the execution of services; this may require more than one recovery strategy to be applied to recover from those failures. However, which combination of recovery strategies can provide best optimal solution, and in which order these strategies should be applied is a very cumbersome problem. The field of fault-tolerance is still maturing, and the introduction of advanced heuristic, AI, and other state-of-the-art techniques may further improve the reliability of Web services.[24,25]

5. Conclusion

A Web service offers its users a coarse-grained and value-added functionality using the Internet. In addition to the containment of all feature of traditional software, Web services contain some additional features like autonomy, heterogeneity, and interoperability. Furthermore, like their traditional counterparts, Web services may also suffer from errors and failures during their entire life (development to execution).

The issue of failures increases when Web services are deployed over the unreliable media and communicate under heterogeneous environments. Due to their use in important and critical applications, Web services are required to be highly available and reliable. Based on the importance of services dependability, this study presented an overview of different failures-types which affect the execution of Web services. Furthermore, an overview of different recovery strategies with respect to different failure types has also been present. Based on the discussion with references to the novel research, it is concluded that detecting and avoiding services failures is a cumbersome problem, specially, when many faults occur at the same time. Furthermore, in order to recover from complex failures, a combination of different recovery strategies may be applied at the same time; however, what is the best combination and best order in which these strategies need to be executed is a very difficult problem to resolve. The field of fault-tolerance is still maturing and the introduction of more sophisticated and state-of-the-art recovery techniques enriched with AI and heuristics is highly needed to make more reliable services.

References

1. Web services: concepts, architectures and applications. [cited 2004]. https://www.springer.com/gp/book/9783540440086.
2. Sheng QZ, Qiao X, Vasilakos AV, Szabo C, Bourne S, Xu X. Web services composition: A decade's overview. Int J Inf Sci. 2014;280(2):218-38.
3. Service-oriented architecture: a field guide to integrating XML and web services. [cited 2004]. https://www.arcitura.com/wp-content/uploads/2017/08/Erl_SOABook1_Ch01-2.pdf.
4. Papazoglou MP, Traverso P, Dustdar S, Leymann F. Service-oriented computing: state of the art and research challenges. Computer. 2007;40(11):38-45.
5. Service-oriented architectures, and cloud computing. [cited 2013]. https://www.w3schools.in/service-oriented-architecture/.
6. Lemos AL, Daniel F, Benatallah B. Web service composition: a survey of techniques and tools. ACM Comput Surv. 2015;48(3):1–41.

7. Dustdar S, Schreiner W. A survey on web services composition. Int J Web Grid Serv. 2005;1(1):1–30.
8. Schäfer M, Dolog P, Nejdl W. An environment for flexible advanced compensations of web service transactions. ACM Trans Web. 2008;2(2):1–14.
9. Yu Q, Liu X, Ouguettaya AB, Medjahed B. Deploying and managing web services: issues, solutions, and directions. VLDB J. 2008;17(3):537–72.
10. Avizienis A, Laprie JC, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Depend Secure Comput. 2004;1(1): 11–33.
11. Towards fault tolerance in web services compositions. [cited 2007 Sep 14]. https://dl.acm.org/citation.cfm?id=1316552.
12. Zeng L, Le H, Jeng JJ, Chung JY, Benatallah B. Policy-driven exception-management for composite web services. In: Proceeding of the 7th IEEE international conference on e-commerce technology (CEC 2005); 2005. P. 355–63.
13. Angarita R, Cardinale Y, Rukoz M. Reliable composite web services execution: towards a dynamic recovery decision. In: Proceedings of the XXXIX Latin American computing conference (CLEI'3); 2014. vol. 302. P. 5–28.
14. Gupta S, Bhanodia P. A fault tolerant mechanism for composition of web services using subset replacement. Int J Adv Res Comput Commun Eng. 2013;2(8):3080–85.
15. Issarny V, Tartanoglu F, Romanovsky A, Levy N. Coordinated forward error recovery for composite Web services. In: Proceedings of the 22nd international symposium on reliable distributed systems; 2003. P. 167–76.
16. Mariani L. Fault taxonomy for component-based software. Electron Notes Theor Comput Sci. 2003;82(6):55–65.
17. Chan KSM, Bishop J, Steyn J, Baresi L, Guinea S. Fault taxonomy for web service composition. In: Di Nitto E, Ripeanu M, editors. Service-oriented computing – ICSOC 2007 workshops. ICSOC 2007; 2009. vol. 4907. P. 363–75.
18. Dependability: basic concepts and terminology. [cited 1992]. https://www.springer.com/gp/book/9783709191729.
19. Tartanoglu F, Issarny V, Romanovsky A, Levy N. Dependability in the web services architecture. In: de Lemos R, Gacek C, Romanovsky A, editors. Architecting dependable systems. Lecture notes in computer science; 2003. P. 2677.
AQ1 20. Zhang J, Zhou A, Wang S, Yang F. Overview on fault tolerance strategies of composite service in service computing. J Wirel Commun Mob Comput. 2018;1:1–8.
21. Kopp O, Leymann F, Wutke D. Fault handling in the web service stack. In: Maglio PP, Weske M, Yang J, Fantinato M, editors. Service-oriented computing. ICSOC 2010. Lecture notes in computer science; 2010. P. 6470.
22. Chan PPW, Lyu MR, Malek M. Making services fault tolerant. In: Proceedings of the third international conference on service availability (ISAS'06); 2006. P. 43–61.
23. Architecting dependable systems. [cited 2003]. https://www.springer.com/gp/book/9783540407270.
24. Muthusami A. Fault prediction in web services. Int J Sci Eng Technol Res (IJSETR). 2014;3(4):1015–9.
25. Immonen A, Pakkala D. A survey of methods and approaches for reliable dynamic service compositions. Serv Orient Comput Appl. 2014;8(2):129–58.
26. A recovery mechanism based on a rewriting process for web service compositions. [cited 2008 Jul]. https://pdfs.semanticscholar.org/cca9/7c42e9fff0f706de45fe52112596cf4fb5d8.pdf.
27. Bhandari GP, Gupta R. Extended fault taxonomy of SOA-based systems. J Comput Inf Technol. 2017;25(4):237–57.
28. Maheshwari EP, Tosic T. Recovery policies for enhancing web services reliability. In: IEEE international conference on web services (ICWS'06); 2006. P. 189–96.
29. Diwase D, Vidap P. A testing framework for fault tolerant of transactional web services, Indian J Comput Sci Eng (IJCSE). 2013;3(6):839–43.
30. He W, Recovery in web services applications. In: IEEE international conference on e-technology, e-commerce and e-service (EEE'04); 2004. P. 25–28.
31. Wang Q, Lv G, Ying S, Wen J. A policy-driven exception handling approach for service-oriented processes. In: IEEE 16th international conference on computer supported cooperative work in design (CSCWD'12); 2012. P. 49–455.
32. Saboohi H, Kareem SA. Requirements of a recovery solution for failure of composite web services. Int J Web Semant Technol (IJWesT). 2012;3(4):13–19.
33. Liu A, Li Q, Huang L, Xiao M. FACTS: a framework for fault-tolerant composition of transactional web services. IEEE Trans Serv Comput. 2010;3(1):46–59.
34. Gulcu JK, SozerH, Aktemur B. FAS: introducing a service for avoiding faults in composite services. In: Proceedings of the 4th international conference on software engineering for resilient systems (SERENE'12); 2012. P. 106–20.
35. Tamak J. A review of fault detection techniques to detect faults and improve the reliability in web applications. Int J Adv Res Comput Sci Softw Eng. 2013;3(6):14–21.
36. Lau KK, Tran CM. Server-side exception handling by composite web services. In: Binder W, Dustdar S, editors. Emerging web services technology volume III. Whitestein series in software agent technologies and autonomic computing. Birkhäuser Basel; 2010. P. 37–54.
37. WS-FTM: a fault tolerance mechanism for web services. [cited 2005]. https://www.semanticscholar.org/paper/WS-FTM-%3A-A-Fault-Tolerance-Mechanism-for-Web-Looker/08de8f4c3498d0efa058af2617d98e71a21c410f.
38. Fault management of web services. [cited 2009]. https://harvest.usask.ca/handle/10388/etd-08182009-231403.
39. Arul U, Prakash S. Towards fault handling In B2b collaboration using orchestration based web services

composition. Int J Emerg Technol Adv Eng Int Conf Inf Syst Comput Eng (IJCSE). 2013;3(6):839–43.

40. Weidong W, Liqiang W,Wei L. A resilient framework for fault handling in web service oriented systems. In: IEEE international conference on web services; 2015. P. 663–70.
41. Liu A, Li Q, Huang L, Xiao M. A declarative approach to enhancing the reliability of BPEL processes. In: IEEE international conference on web services (ICWS 2007); 2007. P. 272–79.
42. Zeng L, Lei H, Benatallah B. Policy-driven exception-management for composite web services. In: IEEE international conference on e-commerce technology; 2005. P. 355–63.