An Autonomic Approach to Enhance Web Application Availability

Hussam N. Fakhouri (hu_ss_am@yahoo.com)
The University of Jordan

Research Article

Keywords: web applications, cloud applications, self-healing, auto restore, auto backup, autonomic computing, website availability

Posted Date: May 29th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-31808/v1

Abstract

Autonomic computing is one of the fascinating features that enable the system to automatically manage itself, diagnose itself, detect the error that cause the failure, then recover and reconfigure the system. The concept of software, systems and web self healing is widely used in many software such as windows operating system which restores and recover tools. Since of the aim of the self healing software feature is to fast recover the application and keep running it and available for 24/7 as optimal as possible then it will a suitable to apply this capability to the web applications to fast recover from any unexpected change that may happen. This paper proposes a self-healing system that monitor, diagnose, check and heal web applications automatically and immediately with unnoticeable recovery time. To test the practical applicability of the proposed methodology, an application has been developed to demonstrate the methodology and apply it for real time web applications. The results of experiments performed on different scenarios demonstrated the ability of the proposed approach to heal web applications and to increase its availability.

1- Introduction

The fast and recent web spread motivated large numbers of companies to offer their services and business through the World Wide Web (WWW). This wide spread of web business applications require more research to develop web applications that has autonomic features such as self healing. Self healing includes the ability of the application to recover itself by detecting any fault or unexpected unauthorized change that happened to the web application files. Self healing of web application require a 24/7 auto monitoring of the web application and a fast mechanism of recovery that can keep the online functionality and service offering to the customer available all the time. The importance of developing fast automated self-healing web applications was generated from the effect that may be caused if the web application for a business or a company is stopped to run for few hours. For example, an online business such as a bank may lose customer trust and lose financially if it is not functioning for a few hours. Many factors may affect a web application and cause it to stop. These factors may be either internal or external. The internal factor includes viruses, worms that may affect the server that hosts the web application. The external factors include attackers that attack the website and change the content of web pages for a web application for different reasons including the use of different methods such as xss, sql injection. You need to have references for this information?!

After publishing the web application on the hosting server many problems rise including deletion of a component, replacing of a component or modifying a component. The risk of having one of these three factors is very high. For example, replacing a web application checkout component by an attacker that function the same way as the main component but with minor changes would allow the attacker to steal credit card information which will cause a major problem and loss to the web application owner and customers. Most of web application owner do not perform tests to check if the component has been

changed or not and that is because of the complex architecture of the web application and lack of knowledge at the owner level. This paper will propose a solution for such problems and many others by proposing an approach for applying external self healing that has the functionality of self healing, self monitoring, self diagnosis and self recovering to keep the web application in good health.

In this research, we mainly focus on techniques for web applications healing from functional failures (self-healing) by automatically detecting failures, diagnose faults, and heal the web application to behave and run as supposed to before the failure happen.

In the self-healing mechanism we perform a black box testing for the software considering the whole web application page as one component and our aim is to ensure that this component run and it has not changed from its origin. Self-healing insures that a component stays the same as it was distributed by the system without any modifications or changes by any other external unauthorized effect and to ensure that this component have not been omitted or deleted from the server and that the web application directory does not contain any injected or added files to it.

Our research suggest the existence of an approach that analyze the content of the released web component, a mechanism for monitoring the web application, a mechanism for diagnosing and detecting of failure and a healing mechanism that bring back the software to its healthy status.

The major research contribution is defining a mechanism for web application fault recovery despite the cause of the fault. The mechanism is automatic and relay on the diagnosis of the web application component. In summary this research define a mechanism for healing software application by external self healing approach and recover from failure. Design an approach that can monitor, diagnose and detect failure automatically, efficiently. Develop an implementation of the framework for any web application regardless of the language used in the web application development and provide an experimental results that demonstrate the efficacy of the approach.

The reset of this paper is organized as follows: section two presents related work on self-healing mechanisms, and the mechanisms that are used for healing the different application; section three presents a full description of the proposed approach and the figures that illustrate it; section four contains the experimental results; and finally section five provides the conclusion and future work.

2- Related Work

Many researchers dedicated their efforts to the field of self healing and here we drop a light over some of those researchers and mention their methodology used in self healing software or systems. (Azim, et.al., 2014) presented an Android apps approach that uses patch construction and automatic error detection for providing a certain degree of self-healing. (Michael E. Shin, 2005) described an approach for distributed software architecture concurrent and designing self-healing component. (Montani et.al., 2008) described a CBR approach that gave capabilities of self- healing to distributed software systems, by means of real world application experimental results. (Park et.al 2009) proposed a self-healing

mechanism that diagnose, heal and monitor its internal error by contextual information and self-awareness. (Naftaly M., 2003) worked in complex systems and mainly for heterogeneous distributed software. And he put the conditions of self-repair and self-healing for those systems.

(Diaconescu A., 2003) worked a system for component-based software systems to have self-optimization, and self-healing and to enable dynamic adaptation in those systems. (PARK, et.al, 2008) introduced the code generation in an automated way for self healing using design level productions and he made software architecture to support generation of automatic code for self-healing in the internal and external system environment. (Brumley et.al., 2007) introduced software systems self-healing architecture where programs detect exploits, self-diagnose and self-monitor the main cause of the vulnerability, self-recover from attacks and self-harden against future attacks. (Dinkel et.al., 2008) presented a novel approach for distributed embedded systems self-healing that contain black-box application software. (Katti et.al., 2015) used Gossip protocols and are inherently fault-tolerant and scalable to compares two novel failure detection and consensus algorithms (Fakhouri et al, 2020).

(Stelios Sidiroglou, 2009) developed a tool named ASSURE that Dynamically patches the running production application to self-checkpoint at the rescue point, it work on Linux operating system to a specified rescue point. (Hervé Chang , 2013) proposed to use Exception Handlers for Healing Component-Based Systems that heal failures activated by exceptions raised in the OTS components actually deployed in the system.

Angelos D. Keromytis focusing on systems that effect structural changes to the software under protection, as opposed to block level system reconfiguration provided a first attempt to characterize self-healing software systems by surveying some of the existing work in the field. (Locasto et.al., 2006) demonstrated the feasibility of the scheme using Selective Transactional EMulation (STEM) as both the monitoring and remediation mechanism for low-level software faults. (Goutam S., 2007) illustrated critical points of the emergent research topic of Self – Healing Software System, described various issues on designing a self-healing software application system that relies on repair of web application or service agent code and data, and the the on-the-fly error detection. Stehle et.al., 2010 presented a computational geometry technique and a supporting tool to tackle the problem of timely fault detection during the execution of a software application. Huang et.al., 2010 investigated the effects of using an unsupervised log data abstraction method to aid the supervised learning processes of problem determination. (Kramer et .al 2007) focus on architectural approaches to self-management because the architectural level seems to provide the required level of abstraction and generality to deal with the challenges posed. (Harald and Dustdar,2011) focused on the self healing branch of the research and gave an overview of the current existing approaches. (Dabrowski et.al., 2002) during communication failure he used architectural models to characterize how architecture, consistency-maintenance mechanism, topology, and failure-recovery strategy each contribute to self-healing. (Elkorobarrutia et.al., 2006) in cases such as forcing the component to some state and rolling back one transition , he described an approach of inserting a self-healing mechanism in components that are specified according to a state chart and whose implementations also offer the possibility to act in terms of state (Fakhouri et al, 2019).

3- Proposed Approach: Autonomic Approach To Enhance Web Application Availability (Aaeaa)

Our approach consist of automatic exterior healing approach that monitor the website files and try to keep it in good health 24/7 working time. The proposed approach is concerned with black box testing of the web application files. Our main concern is the finally compiled web application component. We do not check the internal code of the web file or the flow of its internal functions, rather we check the file itself as one component and test its features such as existence of the component, size, hash key, manufacturer and its correct place. The approach will monitor, diagnose, and heal the web application immediately at the time of the external or internal effect that could cause the web application to change unexpectedly. The proposed system will have a life cycle run to guarantee the full time running of the web.

Figure 1 shows the three main phases of the approach which consist of phase 1: pre-healing, phase 2: healing process and phase 3: post healing. A full details for the description of the main components of each phase and the procedure in each phase is illustrated in sections 3.1 to 3.3.

3.1- Pre-healing phase:

Pre healing phase is the phase of preparing the system for the healing phase and it start by Initializing the system and goes throw building the database and get to identify the website files and after that creating the backup copy of the website; the initializing process consist of running the implemented approach settings for the first time to determine and choosing the specific folder for the web application to monitor and set the initial parameters needed., the Analysis: in this step the approach will gather information about the chosen web application including file size, date of creation, manufacturer of the file and the hash key. The output of the released application is the input of the self healing approach. The output component is mainly the web applications files and other files such as the assemblies of the bin folder. The approach is programming language independent. This means that the system can analyze any type of web application files for any language such as PHP, ASP, HTML etc., Building database: this step is done by storing all of the information about the web application that result from the analysis phase in a database that contains the major and necessary information for the diagnosis process. The aim of using a database is to keep a fast and organized method for diagnosis and a reference for all the web application component for any time access for review or diagnosis. Creating a copy of original components: to reuse and fix the web application later in the healing process we need first to have a copy of the original components and this copy will be stored as a compressed folder on a specific directory determined by the implemented approach and not on the published web directory.

3.2.1 Main processes

The main process of the proposed approach are listed in figure 3.

Monitoring: which keeps a 24/7 monitoring status for any change that may happen to the folder of the web application components (files). It keeps track of all the web application component as well as the web application folder directory for any change that may cause any changes that will be detected and monitored includes deletion of the component, replacing the component, modification of the component and addition of any file to the web directory folder.

Comparing: this process provide the diagnosis process with the results of comparing the monitored component with record stored on the database by the analysis phase. The database contain a full details of all components of the web application that are required for the diagnosis.

Diagnosis: decide wither the system needs to be healed or it is in good health. In this step a solution to the system will be required and suggested if the system is infected or faulty state or in good health.

Healing: it is the process of reusing the original component of the system and replacing or compensating the affected component in order for the system to be in good health. In this step the solution to the problem is applied; when a fault is detected during the diagnosis phase, the latest application saved copy will be restored to the web application directory. The restoration is triggered by detecting a change. This process is in the order of seconds. The changes along with the healing event will be stored in the database for further analysis and the process is automated.

The full details for the steps for the proposed approach and the flowchart is shown in figure 4.

3.3- Phase 3 post process:

This phase contain many steps a full description of these steps are listed in figure 5, the first step is Storing change in the database, this step aims to record all the information that has been done in the healing process including storing the date and time of healing and the component that has been restored. Storing this information will give the administrator a clear summary about the history of the application after releasing it.

The second step is Storing affected component and analyzing reasons; if the healing process resulted from a change in the component itself either for any of the mentioned reason then keeping this file will give us indicator about the reason that caused the fault and this will help us avoid such situations and to enhance or develop mechanisms to update the software or the server so that it can face such cases. For example if the reason of change was illegal access to the server then certain policy could be taken into consideration or if the change to the component has been done by a virus then the server itself should be treated by clearing the virus.

The final step is the Update of All Web Application Components: The analysis process is an important step to get a future enhanced healthy web application because of the previously mentioned reasons and because of the fact that the analysis process results can be used to enhance and update the web application itself and in case that the application has been distributed to may servers then the updated component can be distributed to other servers as a precaution to avoid been infected by the same way.

AAEAA heal the following points:

- deletion of a component that cause the system to fail to run
- change of a component by external factor either human or non human
- original component replacing
- addition of external component to the software folder

AAEAA dynamically modifies the website to correct the failure. The changes that has happen will be stored to be analyzed in the future by the admin and if this error happened often then the recorded information will guide the admin for the reason by analyzing the stored information or the affected component.

Analyzing the changes, the result of checking, diagnosis and monitoring give us indicator about the reasons that cause the system to fail. It also gives a brief overview about the main causes and its indicators. By defining the reason the system administrator can find a suitable solution for the reason that caused the problem.

4- Evaluation

In this section, we evaluate the ability of AAEAA to recover from different factors in web applications. More specifically, through an experimental study, we try to find answers to the following main questions and then we make different scenarios to illustrate each case:

- What advantages can we get when using AAEAA heal websites that are affected by different performance scenarios?
- What is the time to heal using AAEAA when compared with other healing approaches?

To evaluate our approach we need to evaluate the effectiveness and the ability of the proposed approach. Three experiments scenarios will be presented: deletion of a component, replacing of a component and modifying a component. We initialized the implemented self healing approach and selected the web application directory to be monitored, analyzed the testing website directory and built the database. The second phase of the experiments start for testing the effectiveness of the approach to heal the deletion case by executing the application and after that we deleted a file from the web directory. To test the effectiveness and performance of the approach in detecting the problem of replacing a component we have created a file name with the same name and extension of a specific file on the web application directory. For testing the final case of modifying a component we considered manual modification of the component and this is the human modification.

4.2.1 First scenario deletion of component:

AAEAA responded to this case by restoring the deleted page from the original copy that has been prepared in the initialization stage.

AAEAA recorded the problem in the database including the dime, date, type of problem and the name of the file that was deleted and replaced.

AAEAA was very efficient to recover different web application component extension that were tested including PHP, HTML, ASPX, DLL, and CSS.

4.2.2 Second scenario Replacement with similar component

AAEAA responded to this case by deleting the full directory of the web application and restoring the original copy of the website that has been prepared in the initialization stage.

AAEAA recorded the problem in the database including the dime, date, and type of problem and the name of the file that was replaced and recovered.

AAEAA was very efficient to recover different web application component extension that were tested including PHP, HTML, ASPX, DLL, and CSS.

4.2.3 Third scenario modifying a component

AAEAA responded to this case by deleting the full directory of the web application and restoring the original copy of the website that has been prepared in the initialization stage.

AAEAA recorded the problem in the database including the dime, date, and type of problem and the name of the file that was recovered

AAEAA was very efficient to recover different web application component extension that were tested including PHP, HTML, ASPX, DLL, and CSS.

Causes that may have affected the website components are as follows:

- external non human factors
    - virus
    - Wom
- software
    - defect in the components
    - conflict with other software
    - operating system
- external human factors
    - attacker
    - spy
    - fraud

Second experiment aimed to measure the time required for the healing process

We added website files of size 10 g and then we deleted 5 m of the file as shown in table 1. The aim of the experiment is to measure the time needed to heal for both systems, windows server system restore and AAEAA.

Table 1: illustrate the comparison of AAEAA with windows system restore (windows server 2012).

Method	Size of server	Size of website files	Time for healing
AAEAA	12 g	10 m	5 seconds
windows server 2012	12 g	10 m	40*60 second

Since the AAEAA heals by recovering the website files (components) and not the full system restore or recovery we can notice that windows system restoration work by restoring all files in windows **server 2012**. This took about 2400 second while in AAEAA took only 5 seconds.

Table 2 shows the comparison between our healing approach and Microsoft Windows healing approach.

Table 2: Comparison between the proposed approach and Microsoft Windows

	Microsoft Windows (System Restore)	Proposed approach
Recover error resulting from deleting software component	Yes	Yes
Heal Replaced component that has same functionality	No.	Yes
Heal at run time	No	Yes
Generate reports of the diagnosis of the problem and the healing process	Yes	Yes
Store the affected component for future analysis	No	Yes
Methodology of repairing	System Restore	Automatically including Comparing, analyzing, diagnosing and healing to return the software to its original status of the manufacturer
State of the healing	To a specified restore point.	To the manufacturer state either the original or with updates
Knowing the structure of the software and its component	Yes	Yes
Healing time	After release	After release
Building time	During the development of the software	During the development of the software
software	Windows Operating system	Application Software

Conclusion

Integrating self healing approaches into websites introduce a very efficient improvement for the website performance. Different methods and approaches aims at reducing the cost and time needed for the rerun of the websites after failure and to try to have a software system that run and has the ability to heal itself.

This research introduced an approach for self-healing for websites that monitor the software 24/7 and it has the ability to capture information about the specific website components that are monitored. It has the ability to diagnose and heal the software. The experimental results have shown the efficiency of the proposed approach to detect the failure or error and heal it.

Future work

Our future work is inspired from biological software engineering processes and this aims to improve the self learning of the proposed approach and to generalize the concept of healing so that the healing of website will have the features of self-learning and self-adaptation.

Declarations

Hussam n. FAKHOURI declare no competing interests

6- References

1. Aladwan, F., Fakhouri, H. N., Alawamrah, A., & Rababah, O. (2018). Students Attitudes Toward Blended Learning among students of the University of Jordan. Modern Applied Science, 12(12).
2. Alessandra Gorla, Mauro Pezz`e, Jochen Wuttke , Achieving cost-effective SOFTWARE reliability through self-healing , Computing and Informatics, Vol. 2, 2010, 1001–1022, V 2009-Sep-19
3. ammo Krueger, Christian Gehl, Konrad Rieck, Pavel Laskov , TokDoc: A Self-Healing Web Application Firewall, SAC'10 March 22-26, 2010, Sierre, Switzerland. Copyright 2010 ACM 978-1-60558-638-0/10/03
4. Angelos D. Keromytis , Characterizing Self-Healing Software Systems , 2007 Computer Network Security Volume 1 of the series Communications in Computer and Information Science pp 22-33. Online ISBN 978-3-540-73986-9
5. Boris Koldehofe, Ruben Mayer, Umakishore, Kurt Rothermel, Marco Völz, Rollback-Recovery without Checkpoints in Distributed Event Processing Systems, DEBS'13, June 29–July 3, 2013, Arlington, Texas, USA. Copyright 2013 ACM 978-1-4503-1758-0/13/06
6. Brumley David, Newsome James, and Song Dawn, 2007. Sting: An End-to-End Self-Healing System for Defending against Internet Worms Book chapter in "Malware Detection and Defense", Editors Christodorescu, Jha, Maughn, Song.
7. Dabrowski , K. Mills, 2002,Understanding Self-healing in Service-Discovery Systems, WOSS '02 Proceedings of the first workshop on Self-healing systems. Pages 15-20 , ACM, ISBN:1-58113-609-9

8. Dashofy Eric M., André van der Hoek , Richard N. Taylor, 2002, Towards Architecture-based Self-Healing Systems , WOSS '02, Nov 18-19, 2002, Charleston, SC, USA.
9. Diaconescu Ada, 2003, A Framework for Using Component Redundancy for Self-Adapting and Self-Optimizing Component-Based Enterprise Systems , Copyright is held by the author/owner(s). OOPSLA'03, October 26–30, 2003.
10. Dinkel, M. (2008). A Novel IT-Architecture for Self-Management in Distributed Embedded Systems, PhD thesis, TU Munich.
11. Edward Stehle, Kevin Lynch, Maxim Shevertalov, Chris Rorres, and Spiros Mancoridis, 2010. On the use of Computational Geometry to Detect Software Faults at Runtime. Proceeding ICAC '10 Proceedings of the 7th international conference on Autonomic computing. Pages 109-118, SBN: 978-1-4503-0074-2
12. Qin, J. Tucek, J. Sundaresan, and Y. Zhou. Rx: Treating Bugs As Allergies—A Safe Method To Survive Software Failures. In Proceedings of the 20 th ACM Symposium on Operating Systems Principles (SOSP 2005), pages 235–248, Oct.
13. Fakhouri, H. N., & Al-Sharaeh, S. H. (2018). A Hybrid Methodology for Automation the Diagnosis of Leukemia Based on Quantitative and Morphological Feature Analysis. Modern Applied Science, 12(3).
14. Fakhouri, H.N., Hudaib, A. & Sleit, (2019), A. Multivector particle swarm optimization algorithm. Soft Comput doi:10.1007/s00500-019-04631-x
15. Fakhouri, H.N., Hudaib, A. & Sleit, A., (2020), Hybrid Particle Swarm Optimization with Sine–Cosine Algorithm and Nelder–Mead Simplex for Solving Engineering Design Problems, Arabian Journal for Science and Engineering, DOI: 10.1007/s13369-019-04285-9
16. Fakhouri, S. N., Hudaib, A., & Fakhouri, H. N. (2019). Enhanced optimizer algorithm and its application to software testing. Journal of Experimental & Theoretical Artificial Intelligence, 1-23.
17. Fatima Aladwan, A. A., Ali, E. M. M., Fakhouri, H. N., & Alzghoul, I. (2018). Service Composition in Service Oriented Architecture: A Survey. Modern Applied Science, 12(12).
18. Fuad Mohammad M., Deb Debzani, Baek Jinsuk, 2011, Self-Healing by Means of Runtime Execution Profiling. Proceedings of 14th International Conference on Computer and Information Technology (ICCIT 2011) 22-24 December, 2011, Dhaka, Bangladesh
19. Fuad Mohammad M., Debzani Deb , 2009, Similarity Mapping of Software Faults for Self-Healing Applications , WSSU RIP grant no. 121225/2009.
20. Candea and A. Fox. Crash-Only Software. In Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS IX), pages 12–20, May 2003
21. George Selvin, Evans David, Davidson Lance, A Biologically Inspired Programming Model for Self-Healing Systems WOSS '02, Nov 18-19, 2002, Charleston, SC, USA.
22. Goutam Kumar Saha, 2007, Software - Implemented Self-healing System, CLEI ELECTRONIC JOURNAL, VOLUME 10, NUMBER 2, PAPER 5.

23. HANDSEN, J. G., CHRISTIANSEN, E., AND JUL, E. 2006. The laundromat model for autonomic cluster computing. In Proceedings of 3rd IEEE International Conference on Autonomic Computing (ICAC).
24. Harald Psaier , Schahram Dustdar,2011 A survey on self-healing systems: approaches and systems , Computing DOI 10.1007/s00607-010-0107
25. Hervé Chang, Leonardo Mariani, Mauro Pezzè, 2013, Exception handlers for healing component-based systems, ACM Transactions on Software Engineering and Methodology (TOSEM) - Testing, debugging, and error handling, formal methods, lifecycle concerns, evolution and maintenance. Volume 22 Issue 4, October 2013. Article No. 30
26. Hudaib, A. A., & Fakhouri, H. N. (2018). Supernova optimizer: a novel natural inspired meta-heuristic. Modern Applied Science, 12(1), 32-50.
27. Huebscher, M. C. and McCann, J. A. 2008. A survey of autonomic computing—degrees, models, and applications. ACM Comput. Surv., 40, 3, Article 7 (August 2008), 28 pages DOI = 10.1145/1380584.1380585 http://doi.acm.org/10.1145/1380584.1380585
28. Etoh. GCC Extension for Protecting Ppplications from Stack-smashing Attacks. http://www.trl.ibm.com/projects/security/ssp/
29. Kramer and J. Magee. Self-managed systems: an architectural challenge. In FOSE ’07: 2007 Future of Software Engineering, pages 259–268, 2007. IEEE Computer Society.
30. Newsome, D. Brumley, and D. Song. Vulnerability–Specific Execution Filtering for Exploit Prevention on Commodity Software. In Proceedings of the 13 Th Annual Symposium on Network and Distributed System Security (NDSS 2006), pages 1–15, Feb. 2006.
31. O. Kephart and D. M. Chess. The vision of autonomic computing. Computer, 36(1):41–50, 2003.
32. Tucek, J. Newsome, S. Lu, C. Huang, S. Xanthos, D. Brumley, Y. Zhou, and D. Song. Sweeper: A Lightweight End-ToEnd System for Defending Against Fast Worms. In Proceedings of the 2ndEuropean Conference on Computer Systems (EuroSys 2007), pages 115–128, Mar. 2007.
33. Jeff Kramer and Jeff Magee , Self-Managed Systems: an Architectural Challenge , Future of Software Engineering(FOSE'07) 0-7695-2829-5/07 © 2007 IEEE
34. Jehad Alameri, Haifa Bani Ismail, Amal Akour, and Hussam N. Fakhouri, (2019), Blended Learning and the use of ICT Technology Perceptions among University of Jordan Students, Volume 9 of the Learning and Analytics in Intelligent Systems series, springer
35. Jens Ehlers, André van Hoorn, Jan Waller, Wilhelm Hasselbring, 2011, Self-Adaptive Software System Monitoring for Performance Anomaly Localization , ICAC’11, June 14–18, 2011, Karlsruhe, Germany.Copyright 2011 ACM 978-1-4503-0607
36. Jiang Michael, Zhang Jing, Raymer David, and Strassner John, 2007, A Modeling Framework for Self-Healing Software Systems
37. Katti Amogh, Fatta Giuseppe Di, Naughton Thomas, 2015, Scalable and Fault Tolerant Failure Detection and Consensus , EuroMPI ’15, September 21-23, 2015, Bordeaux , France ISBN 978-1-4503-3795-3/15/09.

38. Liang Huang, Xiaodi Ke, Kenny Wong, and Serge Mankovskii,2010 Symptom-based Problem Determination Using Log Data Abstraction, CASCON '10 Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research , Pages 313-326
39. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham. Vigilante: End-To-End Containment of Internet Worms. In Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP 2005), pages 133–147, Dec. 2005.
40. Rinard, C. Cadar, D. Dumitran, D. M. Roy, T. Leu, and J. William S. Beebee. Enhancing Server Availability and Security through Failure-Oblivious Computing. In Proceedings of the 6thSymposium on Operating Systems Design and Implementation (OSDI 2004), pages 303–316, Dec. 2004.
41. Michael E. Locasto, Stelios Sidiroglou, and Angelos D. Keromytis, 2006 .Software Self-Healing Using Collaborative Application Network and Distributed System Security Symposium: February 2-3, 2006, San Diego, California: Proceedings
42. Michael E. Shin, 2005 , Self-Healing Component in Robust Software Architecture for Concurrent and Distributed Systems. Science of Computer Programming. Volume 57, Issue 1, July 2005, Pages 27–44
43. Montani Stefania, Cosimo Anglano 2008, Achieving Self-Healing in Service Delivery Software Systems by Means of Case-Based Reasoning. Applied Intelligence. April 2008, Volume 28, Issue 2, pp 139-152.
44. Naftaly H. Minsky, 2003, On Conditions for Self-Healing in Distributed Software Systems , Work supported in part by NSF grants No. CCR-98-03698. Autonomic Computing Workshop. 2003. Proceedings. 25 June 2003.
45. Horn. Autonomic computing: IBM perspective on the state of information technology. In AGENDA 01, Scottsdale, AR, 2001.
46. Pankaj Thorat, S. M. Raza, Dung T. Nguyen, Giyeol Im, Hyunseung Choo, Dongsoo S. Kim, 2015 Optimized self-healing framework for software defined networks, Proceeding, IMCOM '15 Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, Article No. 7 , ISBN: 978-1-4503-3377-1
47. Park Jeongmin, Youn Hyunsang, Lee Joonhoon, Lee Eunseok, 2009, Automatic Generation Techniques of a Resource Monitor based on Deployment Diagram, International Conference on Convergence and Hybrid Information Technology.
48. Philip Koopman 2003, Elements of the Self-Healing System Problem Space , ICSE WADS03
49. Rababah, O. M., Alzaghoul, E. F., & Fakhouri, H. N. (2018). An Enhanced Approach for Using Data Visualization for Sentiment Analysis and Auto Summarization Data. Modern Applied Science, 12(9).
50. Sidiroglou, M. E. Locasto, S. W. Boyd, and A. D. Keromytis. Building a Reactive Immune System For Software Services. In Proceedings of the 2005 USENIX Annual
51. Sidiroglou, Y. Giovanidis, and A. Keromytis. A Dynamic Mechanism for Recovery from Buffer Overflow Attacks. In Proceedings of the 8thInformation Security Conference (ISC2005), pages 1–15,

Sept. 2005.

52. Technical Conference (USENIX 2005), pages 149–161, Apr. 2005.
53. Tom Janssen Rui Abreu Arjan J.C. van Gemund , 2009, Zoltar: A Spectrum-based Fault Localization Too. SINTER'09, August 25, 2009, Amsterdam, The Netherlands. Copyright 2009 ACM 978-1-60558-681-6/09/08
54. Xabier Elkorobarrutia, Alberto Izagirre, and Goiuria Sagardui , A Self-Healing Mechanism for State Machine Based Components, proceedings of the I International Conference on Ubiquitous Computing: Applications, Technology and Social Issues. Alcal de Henares, Madrid, Spain, June 7-9, 2006.

Figures

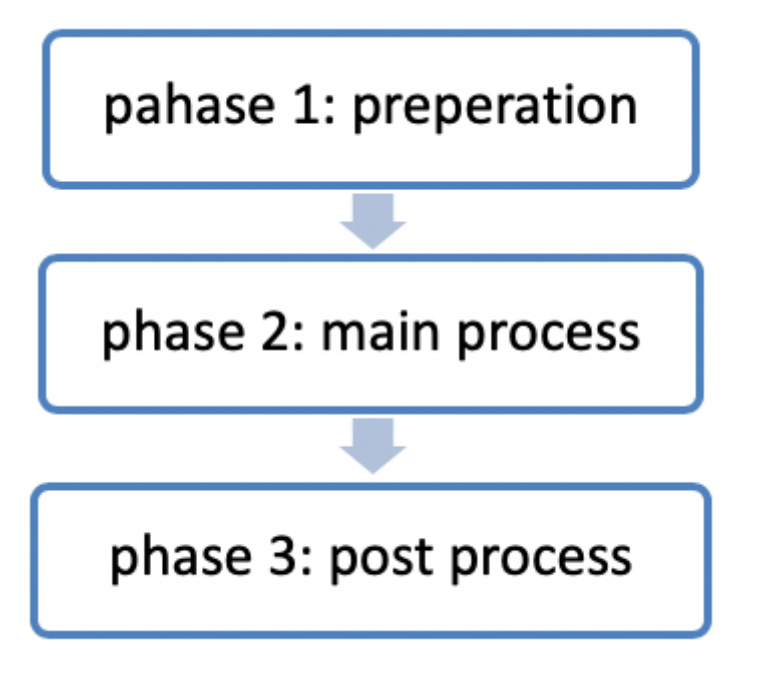

Figure 1

AAEAA Main Phases

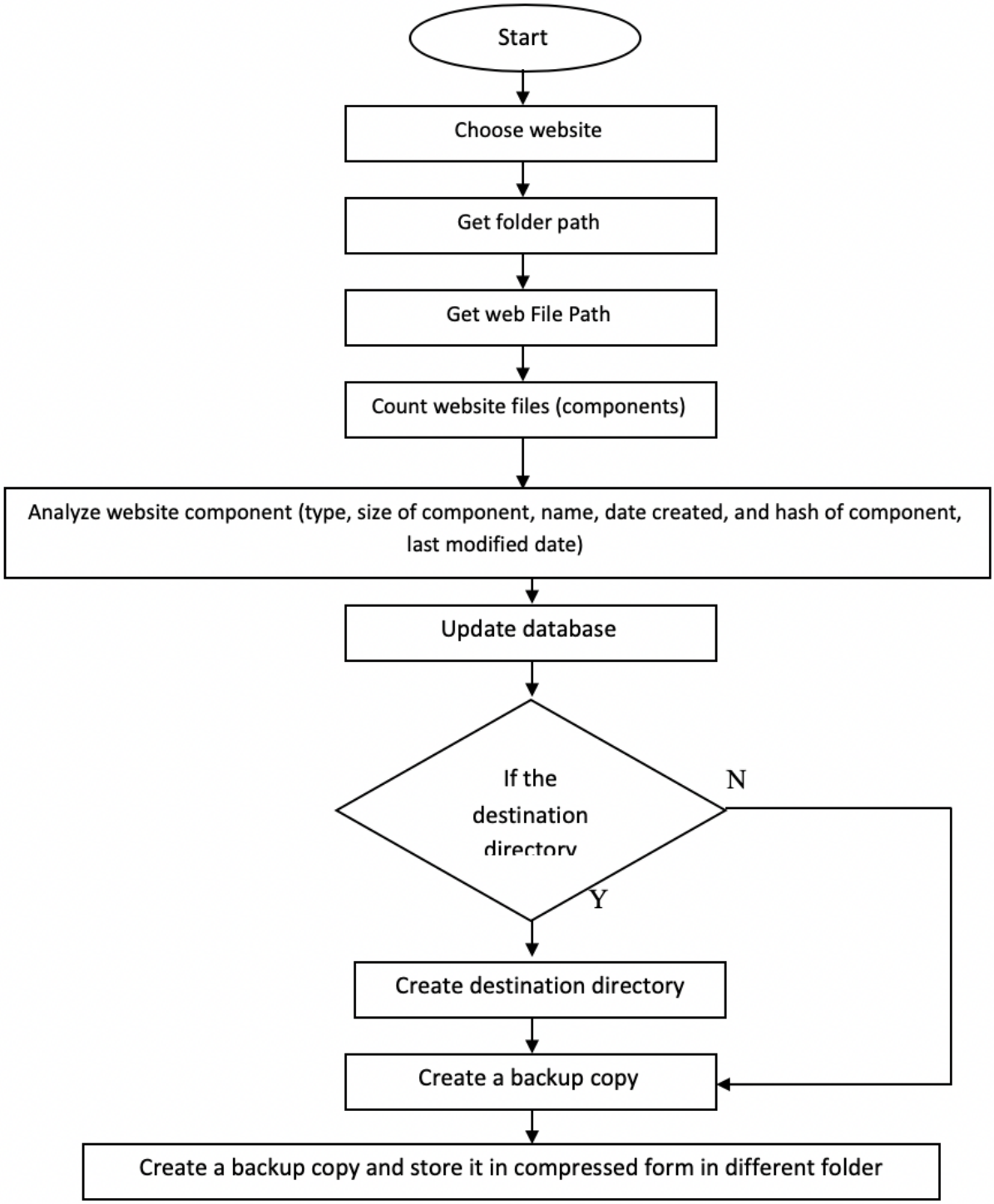

Figure 2

AAEAA preprocess procedure flowchart

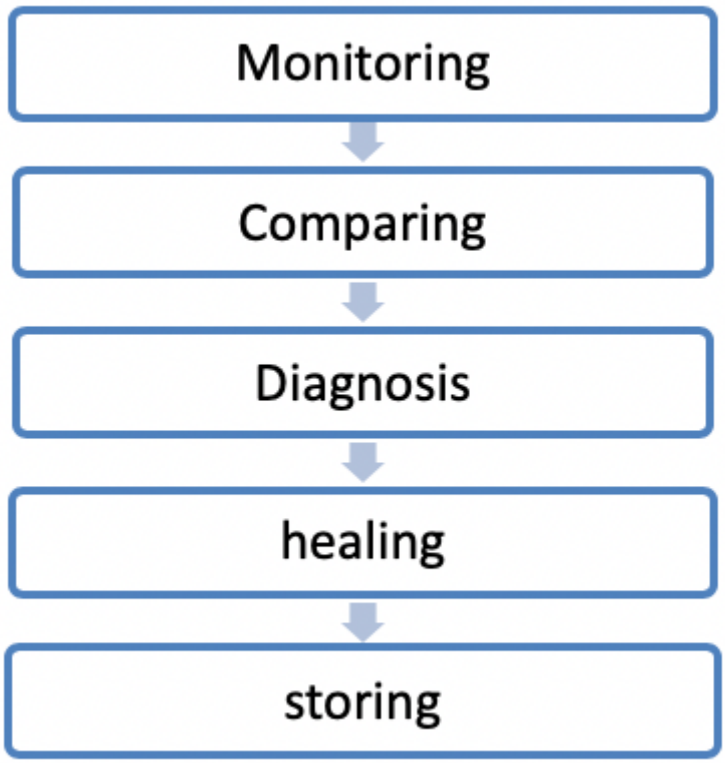

Figure 3

The main process of the proposed approach

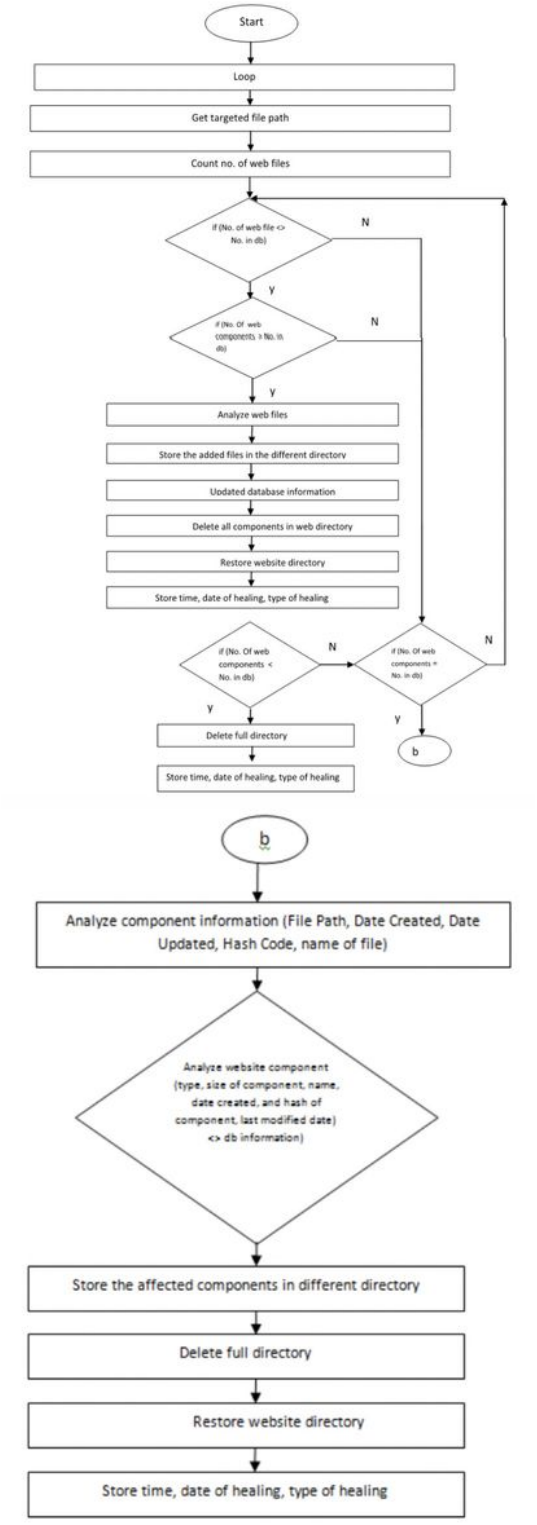

Figure 4

steps for the proposed approach and the AAEAA.

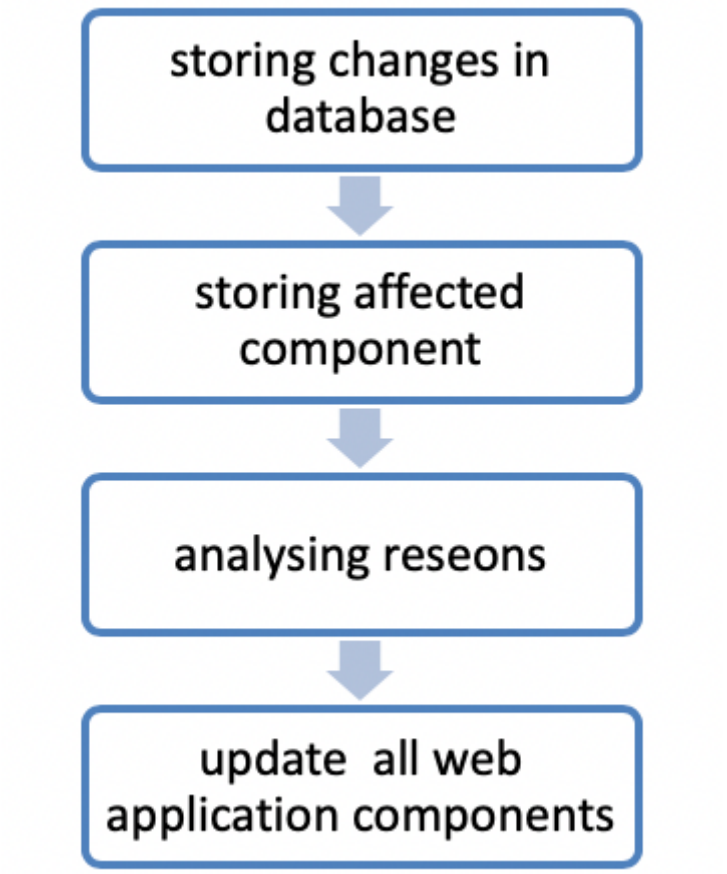

Figure 5

AAEAA post process main components