ISSN 2347 - 3983
Volume 8. No. 1.2, 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter2681.22020.pdf
https://doi.org/10.30534/ijeter/2020/2681.22020

A Review on Fault Taxonomies for Web Testing

Suhaila Mohd. Yasin[1], Shuhaida Ismail[2]
[1]Department of Software Engineering, Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia, ysuhaila@uthm.edu.my
[2]Department of Mathematics and Statistics, Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia, shuhaida@uthm.edu.my

ABSTRACT
For functional testing of web applications, fault-seeding is a method in which a software tester creates several modified versions of a web application that contain errors. They are useful for discovering faults that can compromise a web application's behaviour. Using a fault taxonomy to guide the fault-seeding of a system increases the confidence that the seeded faults are realistic. This paper presents a systematic review of fault taxonomies that are introduced for testing web applications. Selection of relevant literature is performed by applying the inclusion and exclusion criteria. Six significant fault taxonomies for web testing are then selected and analysed. We suggest three fault classes or sub-classes that can be incorporated into a fault taxonomy namely the warning notification fault, media fault and orphan page fault. Finally, some major areas where the fault taxonomies are commonly applied are health software applications, cloud computing, Android applications and desktop applications.

Key words : Fault taxonomy, fault-seeding, software testing, web applications, functional testing.

1. INTRODUCTION

Web applications are software program that runs on a web server. They have become seemingly ubiquitous [1], frequently utilised to complete a myriad of tasks. Their existence has profoundly changed the society's traditional approach to almost any tasks due to their flexibility, availability and accessibility. Web applications enable the tasks to be completed with little restraint on time and physical boundaries, therefore increasing productivity and profitability. However, web applications' developers have been constantly challenged with incorporating these advantages to their released products while minimising their errors simultaneously [2].

A software is susceptible to errors particularly if it is complicated and consisting of an extensive amount of codes [3]. The heterogeneous nature of a web application poses a great challenge to testing [1] when a fault can be caused by either a web component or a number of them combined. Figuring out the characteristics of a severe bug is not a straightforward task. For functional testing, the main concern is on the faults that disrupt a web application's function.

In software testing, real faults are highly valuable compared to artificial faults. For a web application, the existence of a real fault is often discovered and reported by its user. Therefore, it is often easy to find a well-documented report of real faults for a web application. Nevertheless, artificial faults may substitute and provide a close indicator for real faults [4]. This is useful when a web application's bug report record is minimal or poorly maintained.

One of the ways of creating artificial faults is through fault-seeding. Proposed by Lipton et al. [5], this concept is achieved by creating several modified versions of a system that contain errors, which are then used when evaluating the fault detection rate of a set of test cases. However, the fault-seeding should apply an established fault classification to ensure that the produced artificial faults are realistic.Work that explores existing faults in web applications introduces a fault classification after exhaustively investigate the faults' characteristic [6]-[10]. These work use several terms to describe the fault classification such as fault taxonomy, bug taxonomy and mutation operator. In this paper, the term 'fault taxonomy' is used to describe the fault classification.

This paper presents various fault classifications that are proposed in the field of web application functional testing over the years. The aim of this paper is to investigate the applicability of these fault classifications for different kinds of web applications and improvements that are proposed. Additionally, several potential suggestions are also proposed to classify faults in web applications that do not seem to have an explicit class of their own.

The rest of this paper is organised as follows. Section 2 presents the web applications fault taxonomies that are covered in this paper. Section 3 describes the methodology of the fault taxonomy review. Section 4 presents the results and discusses the outcome of the review activity. Finally, Section 5 presents the conclusion of this paper.

2. THE FAULT TAXONOMY REVIEW METHODOLOGY

Budgen and Brereton [11] described the fundamentals and guideline of a systematic review in Software Engineering. With respect to this, a systematic review of existing fault taxonomies for web applications is performed to properly identify, document and analyse common characteristics of the relevant literature in fault taxonomies. To achieve this objective, this systematic review adopts the following review processes [12]: planning the review, conducting the review and reporting the outcomes from the review.

2.1 Systematic Review Plan

Table 1 presents the data extraction form that is used in this work. While the objective of this review have been described above, the rest is explained here. This systematic review considers published work from three categories: technical reports, journal articles and conference proceedings. The work must be sourced from a university's website and established electronic databases like IEEE, SPRINGER, ELSEVIER and ACM that range from 2000 to 2019.

Table 1: Data extraction form

Data ID	In number format
Type	(*Journal, conference, Technical report*)
Year	(*between 2000 to 2019*)
Publication source	(*Name of journal/conference/report*)
Search source	(*University's repository/Electronic database*)
Title	(*Title of the work*)
Author	(*All author*)
Citation	(*Number of last 5 years direct citations if available*)
Validation	(*Summary of validation activity*)
Extension or application	(*Data* ID, title *of work, extension/application*) Arranged by newest if more than one work exists.

In terms of quality, a published work on fault taxonomy that receives significant direct citations in the last 5 years is preferred as it indicates its visibility and applicability among the research community. During data extraction, the focus is directed to the following information: Year of the published article, the number of citations, description of the fault taxonomy (title, author, publication details), types of web applications, support tools and additional remarks that describe existing applications, criticisms, extensions or modifications.

Aside from the usual information that the data extraction form specifies, the data extraction form is also designed to consider the following requirement:

1. What research questions that the work is addressing?
2. What research methodology is applied?
3. Does the outcome of the work fully or partially satisfies the requirement or scope of the review?
4. Has the work been validated empirically? If not, why?

After establishing the plan for this systematic review, the review process begins.

2.2 Conducting the Review

Fig. 1 illustrates the review activity. It consists of five main activities namely identification of research, selection of studies, assessment of quality and finally data extraction and synthesis. These activities are discussed according to the Software Engineering procedure for systematic review [12].

Capitalize only the first word in a paper title, except for proper nouns and element symbols. For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [8].

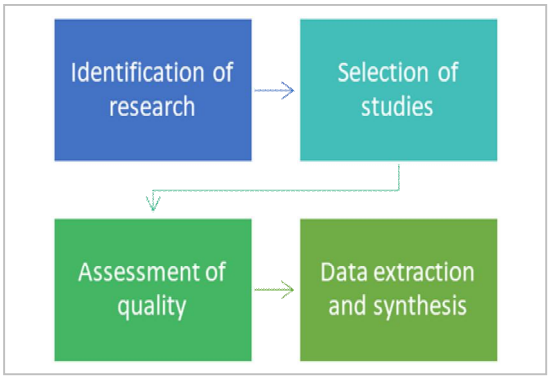

Figure 1: The review activity

A. Identification of research

Several search strategies are employed. Firstly, a list of keywords is developed. Table 2 presents the keywords that are considered during this activity. The keywords are searched exclusively or using Boolean AND's and OR's to form various combinations of keywords. Multiple searches on various sources such as the electronic databases and university's resource sites are performed. Subsequently, some search activities are performed on Google Scholar to verify the citation status of a work and other relevant work that cite and extend the primary work. Aside from this, the reference section of the published work is analysed to identify other work that is also relevant to this systematic review. Negative as well as positive publication of a work is reviewed equally. EndNote is used for bibliography management. The search activity is documented to track the progress of the search activity. Table 3 presents the search activity results.

Table 2: Keywords used in the search activity

No.	Keyword	Alternative spelling
1.	Fault taxonomy	Fault taxonomies, bug taxonomy, bug taxonomies, defect taxonomy, defect taxonomies
2.	Fault classification	Fault classes, failure classification, failure classes, bug classes
3.	Fault analysis	Bug analysis, failure analysis
4.	Mutation operator	Mutation operators, mutation analysis, mutation testing
5	Vulnerability taxonomy	Vulnerability taxonomies, vulnerability, vulnerabilities

Table 3: Results of the search activity

Database	Number of articles	Number of selected articles
ACM	7	0
Elsevier	3	1
IEEE	22	3
Research Gate	6	1
Springer	6	0
University	9	1
Total	53	6

B. Selection of studies

Several criteria for the selection of studies are proposed after the following research questions (RQ) are formulated:

RQ1. What types of fault taxonomy exists for web application testing from 2000 to 2020?

RQ2. How is the fault taxonomy formulated?

RQ3. How is the fault taxonomy applied, reviewed or validated?

RQ4. How is the fault taxonomy extended?

With respect to the inclusion criteria, the following are considered:

1. Include published work from 2000 to 2019 that received at least 1 direct citation.
2. If a work that fits inclusion criterion 1 is unpublished, include the work if it is sourced from a university's repository or resource site.
3. Include work from established electronic databases like IEEE, SPRINGER, ELSEVIER and ACM

In contrast, the exclusion criteria apply to work with the following issue:

1. It is published before 2000.
2. It is a preliminary work that lacks empirical evidence and/or, poorly documented.
3. In the case of multiple publications, the publication with the least citation is eliminated.

An expert panel is consulted throughout the review activity for feedback on the included and excluded work. However, we are responsible for the final selection of the fault taxonomies after careful deliberation of the feedback is made.

C. Assessment of quality

To organise the reviewed literature, several quality criteria are considered during the quality assessment activity of the included work. The quality criteria are arranged from the most important to the least important:

1. The number of quality citations: A quality citation is defined as a direct citation received from a journal article or a conference proceeding.
2. Empirical evidence: A work is superior in quality if it is supported with an empirical evaluation.
3. Extension: A work is superior in quality if it has been extended by other work.

D. Data extraction and synthesis

The data collection and synthesis are performed simultaneously. Each work is analysed to extract information that is specified in the data extraction form presented in Table 1. The activities are performed on 9 work that has been selected. Duplicate work is excluded based on the third exclusion criterion. This systematic review provides only a descriptive synthesis of the collected data that is presented in Table 3. Quantitative synthesis is beyond the scope of this review as the main interest is focused on the applicability of the fault taxonomy.

2.3 Review Report

The outcome of the exhaustive data extraction and synthesis are presented in the next section. Ultimately, a conclusion is presented to shape the future direction of this work.

3. FAULT TAXONOMIES FOR WEB APPLICATIONS

For testing web applications, different fault taxonomies have been proposed for different aspects and kinds of web applications. Our review revealed six fault taxonomies that are selected. For brevity, each fault taxonomy is assigned with an abbreviation that best describes it. The following is a brief description of the selected fault taxonomies.

3.1 The E-Commerce Bug Taxonomy

Vijayaraghavan and Kaner [6] introduce E-Commerce Bug Taxonomy (EcBT). EcBT describes e-commerce failures that

are discovered by using several approaches such as analysing electronic bug databases, brainstorming additional types of problem and refinement through peer review. The failures are categorised into 45 top-level fault classes that are arranged under two main lists: component failures and qualitative failures. While only a brief example is reported [6], the full fault taxonomy is available online[1].

3.2 The Causes of Failures in Web Applications

Pertet and Narasimhan [7] compile the Causes of Failures in Web Applications (CFWA) after investigating various actual incidents of website outages. Additionally, several case studies of system failures are examined. The results of these activities lead to the release of a report for common failures observed in web applications. This report contains an appendix that serves as a quick reference for an observed web application's failure.

3.3 The Object-Oriented Web Fault Taxonomy

The Object-Oriented Web Fault Taxonomy (OOWFT) is introduced for mutation-testing of the .NET web applications. Mansour et al. [8] identify and propose three distinct classes of the .NET web applications: the code-behind file which contains VB.NET and C#.NET codes, the presentation file which targets the HTML and layout tags of the web application and event feature which considers the interactions between the code-behind file and the presentation file classes. In these classes, several mutation operators that can be applied while performing mutation testing to the .NET web applications are introduced.

3.4 The Web Fault Taxonomy

The Web Fault Taxonomy (WFT) [9] begins from an initial effort to construct a top-down fault taxonomy by two software testers through an analysis of the high-level characteristics of web applications. The fault taxonomy is refined by a bottom-up validation effort using real faults collected from several bug reports in SourceForge[2]. WFT consists of six main fault classes: the multi-tier architecture, the Graphical User Interface (GUI) based on browser functionalities, the session-based user interaction, the hyperlinked structure, the HTTP protocol-based communication between client and server and the mechanisms for user authentication.

3.5 Web Application Fault Classification

Sampath et al. [13] develop the Web Application Fault Classification (WAFC) as a guide for a group of graduate and undergraduate students who participate in a fault detection experiment. The experiment requires the students to manually seed realistic faults into three web applications.

Interestingly, WAFC is introduced after extending another work that proposes an initial fault classification that is used for user-session based testing [14]. In WAFC, faults are classified into five types, namely data store faults, logic faults, form faults, appearance faults and link faults. Another work supports the applicability of WAFC through an exploratory study on two actual open-source web applications [10].

3.6 Web Mutation Operators

Praphamontripong et al. [15] introduce the Web Mutation Operators (WMO) that addresses the testing of control and state connections between a web application's software components. Using a support tool called WebMuJava, 11 mutation operators for HTML and JSP are applied to create artificial faults in the transition of these web application's components. Five types of transitions are considered: simple link transition, form link transition, component expression transition, operational transition and redirect transition.

4. RESULTS AND DISCUSSION

The results are presented according to the research questions posed in Section 2.2. The salient and concise manner of the presented result intends to offer a higher comprehension of the work that has been completed and provide a base for the future direction of the research.

4.1 RQ1: What types of fault taxonomy exists for web application testing from 2000 to 2020?

Fig. 2 shows the distribution of articles that are discovered during the systematic literature review activity. The final selection of six fault taxonomies is derived after the exclusion criteria are applied to the relevant literature. The analysis of the relevant literature indicates that existing fault taxonomies cover a myriad of fault classes that were briefly described in Section 3. The most extensive fault taxonomies are EcBT and WFT respectably, with an overlap observed on several fault classes. Overlaps are observed on component-related and database-related fault classes. Even though EcBT focuses its fault classes on E-commerce web applications, we do not see any issues if a significant number of its fault classes that cover component failures are applied to other types of web applications. Further, we note similarities between the qualitative fault classes of EcBT and WFT's protocols and authentications fault classes, with the former choose to align its fault classes with existing software quality attributes, whereas the latter focuses on web application's architecture attributes.

[1] A Taxonomy of E-Commerce Risks and Failures - http://www.testingeducation.org/a/tecrf.pdf
[2] SourceForge - https://sourceforge.net/

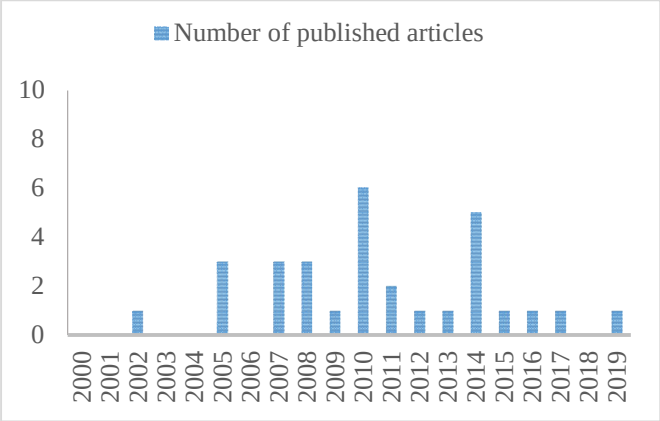

Figure 2: Number of articles discovered in the systematic review

Next, we observed that several fault taxonomies are interested in real faults (EcBT, CFWA, WFT and WAFC) while others proposed specific mutation operators for artificial faults and use a real web application to validate these mutation operators (OOWFT and WMO).

Lastly, we notice that a fault class from EcBT [6] can benefit from further extension due to the advancement in modern web applications. Consequently, this review suggests on extending an existing fault class and introducing new fault classes: error/warning notification faults, media faults and dead-end page faults.

A. Error/warning notification faults

While there is an existing fault class that covers faults relating to an error or warning notification (i.e. error messages/exception handling fault class) [6], it is reasonable to extend the existing fault class to cover more faults. An example from several actual observations of a web application like *OpenBiblio*[3], for instance, includes a warning for an out-of-range date selection. In the existing fault class, it addresses the incorrect expiration date fault for payment error-handling, but there is no specific mention on out-of-range date error-handling, which can occur in forms not exclusive to payment only. In the case of *OpenBiblio*, it can occur when a staff member changes the due date for a loaned library material, which in turn can automatically trigger a fine for late return of the material. Overlooking this fault can cause a series of faults in reporting a user's outstanding balance, which indirectly leads to a less desirable user impression on the efficiency of a web application.

B. Media faults

With the recent technology advancement, modern web applications are more visually attractive and imitate a desktop application look. Such a feature is prevalent in Rich Internet Applications (RIA) that use a variety of media to achieve its functionality.

While the term 'media faults' seems generic, this is merely an initial suggestion that covers common media elements like an interactive text, image, audio and video. With sufficient effort to research this fault category, it is necessary to expand it into several sub-categories that cover more media elements that can be manipulated by the user. Some examples of common faults related to a web application's media include unsupported types of media (e.g. using contemporary, less common font types), corrupted media, and explicit media that lacks an appropriate warning for the user.

C. Dead-end and orphan page faults

A dead-end page is a term in Search Engine Optimisation (SEO) that describes a technical flaw of a web application when a web page does not provide an optional outgoing navigation element (e.g. links or buttons) [16]. An orphan page also shares this characteristic with an additional issue of lacking an incoming navigation element [16]. While this fault does not seem serious, it causes a serious functional issue if the affected web page contains a significant task or if it should direct the user to another significant web page. For instance, if a user accesses a web application to purchase an item and completes the payment instruction, the web application should provide the user with a receipt and a link to either return to the main page or log out of the web application. If the web page that displays the receipt is a dead-end page, it does not allow the user to proceed to the aforementioned choices, forcing the user to stay even after the task is complete.

4.2 RQ2: How is the fault taxonomy formulated?

As shown in Table 4, there are similarities and also disparities between these fault taxonomies with respect to the approach taken to formulate them. Typically, fault taxonomies that do not consider mutation-testing like EcBT, CFWA, WFT and WAFC are more inclusive of real faults in the effort of validating and enhancing the fault classes. Whereas OOWFT and WMO are more focused on a domain-based solution that can be applied to the test problem at hand. The latter's enhancement is also customised towards the specific domain of testing, as opposed to the former that favours generalisation.

4.3 RQ3: How is the fault taxonomy applied, reviewed or validated?

Table 5 summarises the number of citations for each fault taxonomies to date. CFWA has the highest citations to date while WMO is the least cited fault taxonomy. This is expected as WMO is the newest fault taxonomy introduced compared to the rest. We suspected the high citations for CFWA is due to the comprehensiveness of the fault taxonomy. With respect to WAFC, the updated work showed considerable effort was made to improve the fault taxonomy.

[3] OpenBiblio - https://obiblio.sourceforge.net/

Table 4: Characteristics of the fault taxonomies

Fault taxonomy	Characteristics
EcBT [6]	• Fault classes are initially formulated by brainstorming. • Further refinements made by searching in electronic databases and bug databases for related open-source software sites and adopting of the Software Quality Characteristics (ISO-9126) and finally from peer review.
CFWA [7]	• Fault classes are formulated when categorising real web application's faults that were collected by a survey. • The fault classes contain non-malicious failures and security violations. • A quantitative estimation of the failure, sample symptom and recovery recommendation are also included.
OOWFT [8]	• Introduces system specific mutation operators. • Mutation operators are formulated by separating a .NET file into two code-level files that distinguish the .NET file's code-behind and event features.
WFT [9]	• Fault classes are formulated by identifying web application's features. • Real faults are then collected and matches with the fault classes. • The fault classes are available publicly for further contributions.
WAFC [13]	• Fault classes are formulated by distinguishing a fault's physical location in a web application and how it manifested. • Real faults from two web applications were used to refine the fault classes.
WMO [15]	• Introduces mutation operators for control and state connections testing between a web application's components. • Faults are created for five types of transitions between a web's components. • Further refinement is suggested after a validation activity using a support tool.

It also indicates that the fault taxonomy has reached a level of maturity that is often seek by researchers who wish to apply an established fault taxonomy instead of creating a new one. Nevertheless, looking at the number of citations alone is not sufficient to determine the applicability of the fault taxonomy. Next, the scope and year of these citations are analysed.

Table 6 describes the scope of research where the fault taxonomy has been cited, whereas Fig. 3 illustrates the distribution of the citations from 2015 to date. It is evident that CFWA is the most applied fault taxonomy across various software disciplines, particularly in cloud computing. While there is a lack of application for EcBT and WMO in the last 5 years, they have been actively reviewed by work that either adopt them or strive to develop a new fault taxonomy for a different software disciplines. This is observed in EcBT [17]-[19], WMO [20], CFWA [21]-[23], OOWFT [24]-[25], WFT [25]-[26] and WAFC [27] respectively.

Table 5: The number of citations for each fault taxonomy to date

Fault Taxonomy (abbreviated)	Citations
EcBT [6]	37
CFWA [7]	181
OOWFT [8]	48
WFT [9]	35
WAFC [13]	137
WMO [15]	9
Total	447

Table 6: Scope of research where citations exist

Fault Taxonomy	Scope (software discipline)
EcBT [6]	Web applications, Android applications, blockchain systems, health software applications, service oriented architecture (SOA) based systems and video games.
CFWA [7]	Web applications, web services, cloud computing systems, enterprise applications, health software applications, large scale systems, space missions software systems and video streaming applications,
OOWFT [8]	Web applications, web services, database applications, large scale systems, mobile applications and spreadsheet applications.
WFT [9]	Web applications, rich Internet applications and desktop applications.
WAFC [13]	Web applications, graphical user interface (GUI) applications and rich Internet applications
WMO [15]	Web applications, Android applications, hybrid systems, mobile applications and smart contract programs.

4.4 RQ4: How is the fault taxonomy extended?

A fault taxonomy is considered as extended when one or more changes were suggested by other work. The suggested changes can either be removing existing fault classes or adding new fault classes (or subclasses). This review discovered that out of these 6 fault taxonomies, WAFC and WMO were extended in later work. One of WAFC's fault classes, logic fault is expanded to cover more sub-classes of faults. Aside from this, a new type of fault class called compatibility faults is also introduced [10]. With respect to WMO, recent work suggested a modification of the fault taxonomy by removing three mutation operators that are found to have less significant on the outcome of the testing in terms of the fault detection capability [28].

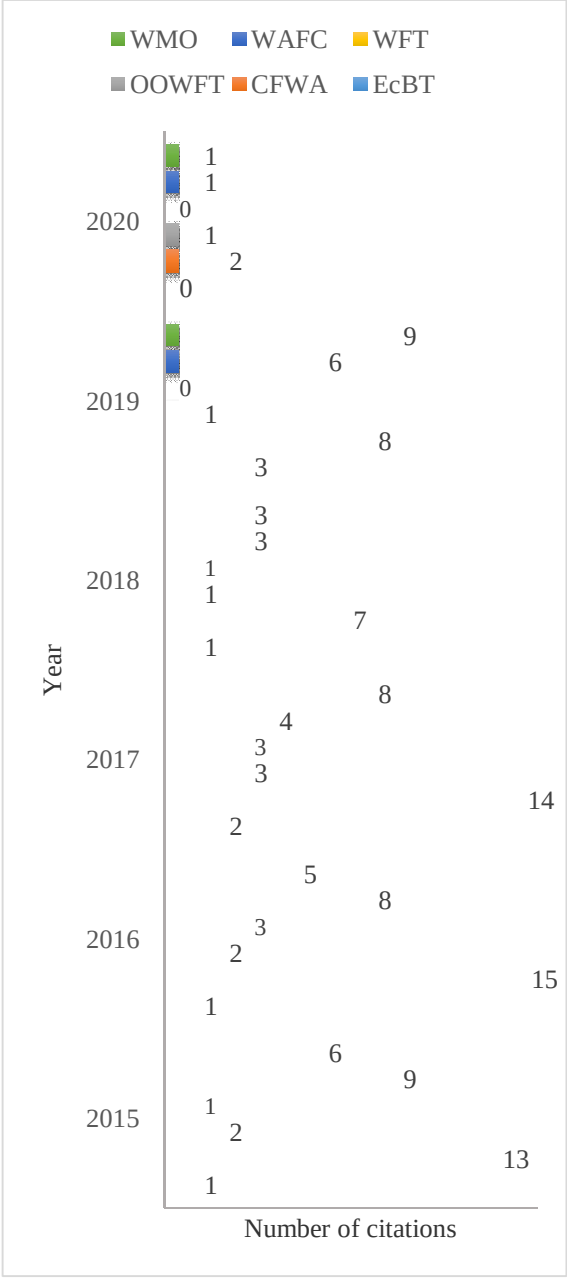

Figure 3: The number of citations from between 2015 and now (2020)

5. CONCLUSION

This paper presents fault taxonomies for web application functional testing that were selected for their applicability and extendibility. In this paper, we systematically reviewed existing fault taxonomies that are proposed for testing web applications. We described a plan to review the existing literature. Next, we conducted the review and reported the outcome. We identified six fault taxonomies that fulfil the inclusion and exclusion criteria of the systematic review. We investigated identified their characteristics and application on web applications. We also discussed on past effort on extending these fault taxonomies. In addition, we proposed several suggestions on extending a fault taxonomy. Finally, based on the review of current work that extends the fault taxonomies, we suggest the same effort should be made to improve the usability and compatibility of these fault taxonomies in state-of-the-art web applications.

APPENDIX

Appendixes, if needed, appear before the acknowledgment.

ACKNOWLEDGEMENT

The authors would like to thank the Ministry of Higher Education Malaysia (MOHE) for supporting this research under Fundamental Research Grant Scheme Vot No. FRGS/1/2018/STG06/UTHM/03/3 and partially sponsored by Universiti Tun Hussein Onn Malaysia.

REFERENCES

1. Y.-F. Li, P. K. Das, and D. L. Dowe, "Two decades of web application testing—A survey of recent advances," *Information Systems*, vol. 43, pp. 20-54, July 2014.
2. J. Goyal, and B. Kishan, "Progress on Machine Learning Techniques for Software Fault Prediction," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 2, pp. 305-311, April 2019.
3. B. Yetukuri, J. S. Miriyala, S. Mundru, and Y. Sangeetha, "Software Bug Prediction," *Journal of Critical Reviews*, vol. 7, no. 11, pp. 3952-3956, July 2020.
4. R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes, and G. Fraser, "Are mutants a valid substitute for real faults in software testing?," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Hong Kong, China, 2014, pp. 654-665.
5. R. Lipton, "Fault diagnosis of computer programs," *Student Report, Carnegie Mellon University*, 1971.
6. G. Vijayaraghavan and C. Kaner, "Bug taxonomies: Use them to generate better tests," in *Software Testing Analysis & Review Conference*, Orlando, USA, 2003.
7. S. Pertet and P. Narasimhan, "Causes of Failure in Web Applications," Carnegie Mellon University, Parallel

Data Laboratory, Technical Report CMU-PDL-05-109, 2005.
8. N. Mansour and M. Houri, "Testing web applications," *Information and Software Technology*, vol. 48, no. 1, pp. 31-42, January 2006.
9. A. Marchetto, F. Ricca, and P. Tonella, "Empirical validation of a web fault taxonomy and its usage for fault seeding," in *9th IEEE International Workshop on Web Site Evolution*, Paris, France, 2007, pp. 31-38.
10. Y. Guo and S. Sampath, "Web application fault classification - An exploratory study," in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, Kaiserslautern, Germany, 2008, pp. 303-305.
11. D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, 2006, pp. 1051–1052
12. B. Kitchenham, "Procedure for undertaking systematic reviews," *Computer Science Department, Keele University (TRISE-0401) and National ICT Australia Ltd (0400011T. 1), Joint Technical Report*, 2004.
13. S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, and A. S. Greenwald, "Applying concept analysis to user-session-based testing of web applications," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 643-658, October 2007.
14. S. Elbaum, G. Rothermel, S. Karre, and I. Fisher, "Leveraging user-session data to support web application testing," *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 187-202, March 2005.
15. U. Praphamontripong and J. Offutt, "Applying mutation testing to web applications," in *Third International Conference on Software Testing, Verification, and Validation Workshops*, Paris, France, 2010, pp. 132-141.
16. M. Dudharejia. SEO Terminology: Orphan and Dead-End Pages. [online] Available at: <https://www.searchenginejournal.com/seo-terminology-orphan-and-dead-end-pages/7373/> [Accessed 17 July 2020].
17. H. K. Rajaram, J. Loane, S. T. MacMahon, and F. Mc Caffery, "Taxonomy□based testing and validation of a new defect classification for health software," *Journal of Software: Evolution and Process*, vol. 31, no. 1, p.e1985, January 2019.
18. M. Jimenez, M. Papadakis, T. F. Bissyandé and J. Klein, "Profiling Android Vulnerabilities," in *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Vienna, 2016, pp. 222-229.
19. Z. Wan, D. Lo, X. Xia and L. Cai, "Bug Characteristics in Blockchain Systems: A Large-Scale Empirical Study," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, Buenos Aires, 2017, pp. 413-424.
20. L. Deng and J. Offutt, "Reducing the Cost of Android Mutation Testing," in *Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering*, California, USA, 2018, pp. 542-547.
21. T. Wang, J. Wei, W. Zhang, H. Zhong, and T. Huang, "Workload-aware anomaly detection for web applications," *Journal of Systems and Software*, vol. 89, pp. 19-32, March 2014.
22. T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske, "Hora: Architecture-aware online failure prediction," *Journal of Systems and Software*, vol. 137, pp. 669-685, March 2018.
23. T. A. Majchrzak, M. Sakurai, and N. Serrano, "Conceptualizing and Designing a Resilience Information Portal," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, Hawaii, USA, 2018, pp. 45-54.
24. S. M. Yasin, P. A. Strooper, and J. R. Steel, "A pseudo-genetic algorithm for optimising test cases," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, China, 2017, pp. 500-507.
25. M. S. Biçer and B. Diri, "Predicting defect prone modules in web applications," in *International Conference on Information and Software Technologies*, Vilnius, Lithuania, 2015, pp. 577-591.
26. D. K. Jain, A. Kumar, S. R. Sangwan, G. N. Nguyen, and P. Tiwari, "A particle swarm optimized learning model of fault classification in Web-Apps," *IEEE Access*, vol. 7, pp. 18480-18489, February 2019.
27. R. Bhan, M. S. Ahmad, M. Jain, A. Singh, R. Pamula and P. Faruki, "VM Availability in Presence of Malicious Attacks in Open-Source Cloud," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2019, pp. 26-30.
28. U. Praphamontripong and J. Offutt, "Finding redundancy in web mutation operators," *in Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops*, Tokyo, Japan, 2017, pp. 134-142.