Research Article

Test Case Quality Factors

Samera Obaid Barraood[1], Haslina Mohd[2], Fauziah Baharom[3]

[1,2,3]Universiti Utara Malaysia, Malaysia
[1]Hadhramout University, Mukalla, Yemen
samera_obaid@ahsgs.uum.edu.m[1], sammorahobaid@gmail.com[1], haslina@uum.edu.my[2], fauziah@uum.edu.my[3]

Article History: Received: 10 November 2020; Revised: 12 January 2021; Accepted: 27 January 2021; Published online: 05 April 2021

Abstract: The guarantee of software quality is very important. Thus, before a software is released to the end users, the flaws in the software should be detected by using high quality test cases. Currently, gauging the quality of test cases is carried out without any particular model and the criteria for good test cases is still unclear. Therefore, this study studies the literature using Systematic Literature Review (SLR) technique to identify the criteria of good test cases. The SLR considered papers between 2010 and 2018 in IEEE Xplore, ACM Digital Library, and Science Direct databases. Through the searching, it was found 310 papers are related. After filtering using exclusion and insertion criteria, 14 papers were considered for analysis. As a result, the test managed to identify 30 quality factors from the selected articles. These quality factors were additionally inspected, arranged and finished to be incorporated as the quality factors of test cases evaluation metrics.

Keywords: Test case, test case evaluation metrics, software testing, Systematic Literature Review (SLR)

1. Introduction

The most important phase for detecting software defects in producing high-quality software is software testing phase [1]. It could determine the risk reduction. The effective and successful software testing has been a worth explored issue because it really affects the success of a project. In fact, Lai [2] found that the success of a software is always under 40%. The effectiveness of the testing relates to the quality of the test cases, which depends on the amount of errors being revealed [3], [4]. This implies that the testing should reveal as many errors as possible during the testing so that the requirements are not jeopardized and meeting the acceptable level of quality [5]–[7]. Various reasons have been identified leading to software failures, including misled understanding (among team members) upon different contexts, immature experience in designing test case, and immature understanding [8]–[10]. Those identified factors are clearly possible because designing good test cases is a complex art. There is no simple formula in generating test cases [11]. However, testers could focus on two things in improving the quality of software testing and productivity; identifying the most effective quality metrics and measuring the test case quality [2]. Both the quality and testing metrics are important [12]. In fact, various applications have used the test case quality metrics, especially in evaluating existing test suites in ensuring sufficient number of testing are performed [13].

In accordance, this study gathers previous works reported in papers published in IEEE Xplore, ACM Digital Library, and Science Direct for identifying appropriate and usable testing metrics in measuring and evaluating test cases quality. For this, Systematic Literature Review (SLR) has been executed. Altogether, 310 papers have been discovered meeting the purpose of this study. The procedure and details of the SLR protocol this study has gone through are discussed at length in the fourth section.

The structure of this paper is recognized as follows: Section 2 explains the background of test cases. It is followed with a review on related works in section 3. After that, section 4 describes the procedure for this study. Then, section 5 displays the results, and the last one summarized this paper.

2. Background of Test Case

Generally, testing a software is costly. Hence, it aims to gather maximum number of flaws [5]–[7], [14]. It has to be extensive, covering all possible ways the system can be used [15]. Accordingly, deciding on the adequate number of testing really matters. [16] recommends to continue testing covering both functional and non-functional aspects until all critical dangers are solved.

Among the major risks that are difficult to handle include incomplete analysis on the requirement, evolved technology and context of use, swiftly changed requirements, and imperfect and inflexible management of resources allocation. In conjunction to that, [2] recommends that designers and developers should plan for early detection and prevention from flaws. It could help reducing the possibilities of flaws in the developed software. In such situations, the test cases have to be well understood. Also, it has to be effectively designed [17].

2.1. A Test Case

A test case, which consists of expected results based on the inputs (including actions, where applicable), and a set of preconditions, is constructed in determining whether or not the specified part of the test item has been correctly implemented [18], [19]. It is a very significant asset in software testing and in the software development generally. It impacts best when it is able to detect flaws very confidently, especially flaws that are hardly found. On top of that, the test cases are better, in situations where they could come out with more reliable results, improved performance, and lowered cost in terms of scheduling reliability, testability, and productivity [2], [6], [20], [21].

2.2. A Test Case Design

The quality of test cases is paramount in software testing. It substantially determines the wellness of the tests, the flaws discovered and the ultimate achievements. They eventually leading to the discovery of flaws, especially in the coding [2], [10]. This implies that it has to be well-designed, and comprehensive for the desired software being tested [17]. There are common tests being carried out in varying software [11]. On top of the common ones, writing test cases from scratch is very important. However, it is very difficult. In designing test cases, it is notable for ensuring that testing could achieve a certain level of thoroughness [22]. Hence, software testers must have sufficient skills to write good test cases [21], [23], [24], which really requires them to have a transparent knowledge on the system being tested [25]. According to Paruch et al. [24], the testers should be creative, curious, structured, able to understand the big picture, friendly and providing constructive feedback. Regarding the reasons for the difficulties in writing test cases, [11] believes it includes:

- According to some styles of testing, test cases generating by people, for example, domain or risk-based testing.
- There are different ways for good test cases. But not found test case that will be good in all of them.
- Test cases help to discover information. Different test types are more effective for different information classes.

Besides, [10] found that a deep understanding and avoiding test case construction is necessary for producing good test cases. Meanwhile, [23] discovered that the difficulties also come from unclear requirements. This has to be avoided, because the better the test cases, the more flaws are discovered, and it eventually results in higher quality [9], [13].

2.3. A Test Case Quality

The failure of software development around the world, which consequences in tremendous losses in monetary and time has increased the awareness on software quality. It creates a major research area and should be unavoidable [26]. As a result, a universal standard has been stipulated regarding the software quality. Specifically, ISO/IEC 9126 and ISO/IEC 25010 define quality as “the extent to which the system satisfies the stated and implied needs of its various users” [27]–[29]. In testing, a test case quality is the attribute for fault level in testing phase [14].

As there are standards for software quality, the tasks in measuring it are daunting [30]. The difficulties in software testing vary depending on the size and complexity of the software being tested [31].

For every software testing, the software tester must regard the quality of the test cases as a very important goal [9], [32], [33]. They have to be carefully generated. In generating them, the tester has to carefully select [34] and prioritize [13], [35] so that the software is free of failure when in operation [36], [37]. In such situation, it could increase software productivity [38] and reliability [26], [33], [39].

There are many criteria for the quality of test cases. One of them lies on the breadth coverage of the functionalities in the system being tested [40]. Then, [14] added that various dimensions have to be considered in ensuring the quality of test cases. Among the common dimensions include code defect density, failure rate, cumulative failure profile, coverage factor, fault days number, fault density, modular test coverage, minimal unit test case determination, and requirement specification change request. Additionally, user satisfaction is also a quality attribute [41].

The standards (ISO/IEC 9126 and ISO/IEC 25010) can be used to validate the test cases, as to ensure they are acceptable [42]. Some quality characteristics can be referred to in ISO-IEC 25010:2011. However, [43] found that applying them is quite challenging for some testers due to some operational complications. This implies that there is a need for quality factors/metrics that can be easily referred to by beginner testers in producing high-quality test cases. As a response to that, this study takes the challenge, aims at identifying good quality factors/metrics for test cases.

3. Related Work

A metric is a function assigned to a value of an attribute [44]. Meanwhile, software metric refers to the way of measuring software, including its development process [45] that utilizes a metric. Further, the IEEE 1061-1998 defines a software quality metric as “A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality” [46]. [2] Emphasizes that effective quality metrics of a test case is paramount in uplifting the quality and productivity of a software. Various researchers have investigated the related perspectives of quality and quality metrics. One of the common example is the work by [47]. They concentrated on examining test case quality features generated by using test-first method. They used for comparison of software development approaches the quality of test cases. They gauged the produced code quality by test-first and test-last approaches and examined the variance of the quality of test cases in these two approaches. Total number of failing assertions, mutation score, and code coverage were used as three quality indicators in measuring the designed test cases. Moreover, the interface was also enforced. It allowed for the execution of test cases of a participant on the other participants code.

Regarding that, [2] has proposed a measurement model for the quality of test cases called Iterative and Incremental Development (IID). The model comprises thirteen features. They are classified into manageability, qualified documentation, reusability quality characteristics and maintainability indicators.

[30] proposed for a quality of test cases a multi-dimensional measuring. For them, not just the detected flaws number is important but also other features such as source code and usage profiles.

Earlier, [44] came out with a set of ten questions regarding software engineering metrics. It is coupled with a framework on the procedure to perform the evaluation. Meanwhile recently, [48] proposed a metric-driven approach comprising 20 20 metrics in order to assess the inherent quality features of a dataset before released to the Linked Open Data Cloud. Based on an SLR and the ISO/IEC 25012 standard, they selected five inherent quality characteristics, which are syntactic accuracy, semantic accuracy, consistency, uniqueness, and completeness.

Later, [12] underlined the reasons for and effects of using metrics in industrial agile development. They extracted 102 metrics from previous works reported in the literature. In their study, they only considered on the metrics used by agile teams. They found that the use of metrics may lead to behavior functional damage due to negative effects that it had.

Although those metrics have been shared, researchers believe they are debatable. Hence, researchers keep studying for appropriate metrics for ensuring the quality of test cases [49]–[52].

4. Research Methodology

This study decided to use Systematic Literature Review (SLR) as the research methodology. It is appropriate as this study aims at knowing a problem, but not at making an attempt to address it [12]. Regarding that, this study particularly intends in order to distinguish the former research gaps, synthesize the existing research topic knowledge, provide a continues research method which may provide sufficient details when applied in a suitable way to be used by other researchers, and supply background information to start exploring a new research topic [12]. For such purpose, this study adapted the guideline provided by [53]. Generally, the guideline acts as a basis for developing the protocol of the SLR. In the execution, this study collected and reviewed works on test case quality between 2010 and 2018 and produced good test cases by identifying their factors and metrics.

4.1. Research Questions

The core purpose of this study is to determine the factors that affect the quality of test cases. Particularly, this study focuses on the metrics and measurements of the test cases in making high-quality testing. In supports for that, the following research questions need to be answered:
RQ1: How much are the conducted research activities between 2010 and 2018 related to the quality of test cases?
RQ2: What are the quality factors/metrics for producing a good test case?
RQ3: Is the effectiveness of test case affected by the quality factors/metrics?

4.2. Search and Selection Process

The search and selection process have been carried out to select the primary studies. It contains three steps as detailed in Table 1.

Step 1: Selecting Source Repositories
Suitable databases were selected in this step. This study considered IEEE Xplore, ACM Digital Library, and Science Direct only, which are the most appropriate for the field of study, software engineering. It was decided based on the recommendation by [34] that IEEE and ACM cover almost all prominent conferences in software engineering, while Science Direct covers nearly all important journals in software engineering. The execution was begun with entering the reserved words related to the research questions. To obtain the most relevant search results, this study switched the string with (OR, AND) operators suitable with the time span between 2010 and 2018. Two stages of searching were used in this study. Firstly, with string (“test case” OR “test case quality”) AND (“metrics” OR “factors” OR “indicators”), which resulted in 268 papers, as detailed in Table 1. Having read the articles, this study discovered that some of the studies use the term “effectiveness of test cases” instead of the “quality of test cases”. Therefore, the second stage was performed with the string "test case effectiveness" OR "the effectiveness of test case". It resulted in 42 papers, as detailed in Table 1.

Step 2: Reading Titles and Abstracts
In this step, according to inclusion and exclusion criteria (section C), 39 papers were extracted from the first stage and 15 were selected from the second stage. The titles and abstracts of the included and excluded papers had been read. In case the abstract is unclear, the content of the paper is scanned. Through this process, 54 papers were selected, as detailed in Table 1.

Step 3: Reading Full Text
The full paper of the selected abstract was then gathered. They were carefully read. Eventually, considering the selection criteria, 14 of them were selected, as they meet the requirement for this study.

Table 1. Studies Distribution after Applying Inclusion/Exclusion Criteria

Data Repositories	**First Stage**			**Second Stage**		
	Step 1	**Step 2**	**Step 3**	**Step 1**	**Step 2**	**Step 3**
IEEE	52	12	4	11	5	1
ACM	201	23	8	3	3	

Science Direct	15	4	1	28	7	0
Total	268	39	13	42	15	1
	13 Relevant articles			1 Relevant article		

4.3. Inclusion and Exclusion Criteria

Exclusion Criteria:
- Papers without contents on quality factors or metrics.
- Papers written not in English.
- Books and workshops.
- Papers with contents not related to testing.

Inclusion Criteria:
- Papers talking about good test cases.
- Papers presenting the factors or metrics of testing quality.

4.4. Data Extraction

This study extracted data by carefully and critically reading through the full papers. This data extraction involved two phases. Firstly, standard information [53] was collected, which include the publication year, author names, title, and summary of the study. Secondly, information that is directly related to the research questions of this study was collected.

5. Results

This section provides the research questions answers together with the SLR results.
RQ1: *How much are the conducted research activities between 2010 and 2018 related to the quality of test cases?* The answer for this question is depicted in Tables 1 and 2. The total number of papers that are related to quality testing cases is 310. However, only 14 papers are deemed to be the most related as listed in Table 2. Subsection 5.1 provides more details about the selected studies.

5.1. Overview of Studies

This section details the overview of the primary studies related to quality test cases. It was found that there are 14 papers in IEEE Xplore, ACM Digital Library, and Science Direct databases between 2010 and 2018 reporting on quality test cases (as detailed in Table 1). Most of the studies (8) are published in the ACM Digital Library, followed by IEEE Xplore (5), and Science Direct (1).
Further, Table 2 presents the details of the 14 papers. The most similar study is S3, which was conducted in 2017. However, the study only focuses on the test case selection techniques instead of the quality of test cases. Thus, for the past eight years, this was the first study performed to identify the quality factors and metrics in producing high-quality test cases as well as good testing.

The Table 2 further explains that most of the studies (28.57%) were conducted in 2017. The others were mostly carried out in 2015 (21.40%) and 14.28% in 2010, 2014, and 2016, followed by one in 2011. Pertaining to the emphasized issue (column five), it seems that there is no study focusing exactly on the quality of test cases. Most of these studies generally either focus on the use of or proposing quality metrics for specific purposes. Among the purposes include test case selection [S3], test case generation [S5, S9, and S11], software maintenance [S7, S9], test case prioritization [S4, S6], productivity [S8], software reliability [S1, S5], test case design mistakes analysis [S13], and diagnosability of a test suite [S14]. Additionally, the table also portrays that almost all studies describe the quality of test cases in terms of structural design (code-based), whilst only one provides the test case generation quality in the specification (black box) and white box methods.

5.2. Test Case Quality Metrics

RQ2: *What are the quality factors/metrics for producing a good test case?* The answer to this question is described in Table 3. This study discovers that between 2010 and 2018, the quality of test cases is important in various domains and techniques particularly in software maintenance [S7, S9], software reliability [S1, S5], reusability [S2], software productivity [S8], test case generation [S9, S10, and S11], test case selection [S3], test suite diagnosability [S14]. And test case prioritization [S4, S6].

Referring to Table 3, it could be seen that 30 of quality metrics have been identified from the 14 primary studies. The most used metric is Coverage [S3, S4, S6, S8, S9, S12 and S14], which has various types such as branch, statement, condition, and method. Among all 14 studies, there is only one has used coverage metric [S3], while others used only some of it [S4, S6, S8, S9, S12, and S14]. Coverage is considered as a good indicator to be used as a proxy for evaluating the quality and the completeness of test suites [34]. However, S3 and S12 do not recommended other studies to merely use coverage because it is insufficient as it is not a good quality measurement for testing suite's effectiveness. For them, coverage has to be used together with other metrics. Meanwhile, S9 and S14 used branch coverage metric for comparison with their proposed metrics. On the other hand, S10 used mutations rather than coverage because the former not only know where to test but also what to test for. In contrast, S13 tried to improve the quality of test cases by analyzing the mistakes of test cases based on the knowledge of the test case writers instead of providing any quality metrics for usage. They found that most of the test cases have a deficiency quality in the light of the absence of understanding in regard to the relating knowledge, which is essential for test case design.

In general, all identified quality metrics from the selected primary studies are used for producing good test cases. The metrics were identified either based on current release of the system, the previous release, experience of the test team, diagnosability of the test cases, or similarity.

RQ3*: Is the effectiveness of test case affected by the quality factors/metrics?* Referring to S4 and S11, the test case effectiveness refers to the test case ability to detect more flaws or determine the number of flaws revealed. By revealing more failures, the chances of producing a more quality test cases will be higher. Thus, the results show that the effectiveness of test cases is affected by the quality of test case metrics. However, the coverage metric should not be used alone due to it is poor predictor of test case effectiveness [S3, S12].

Table 1. Details of the Selected Studies

Study	**Reference**	**Year**	**Study Type**	**Study Focus**	**Apply on**
S1	[26]	2015		Software reliability	
S2	[54]	2010		Reusability of test cases	
S3	[34]	2017	SLR	Test case selection	Regression testing
S4	[13]	2015		Test case quality for prioritization	Five open source systems (java projects)
S5	[33]	2015	Online survey	Software reliability	
S6	[35]	2017	Empirical study	Test case prioritization	five open source systems (java projects)
S7	[36]	2017		Predicting software maintenance.	Object-oriented software

S8	[38]	2014	a controlled experiment	TDD (Test Driven Development) on productivity, internal, and external code quality.	Professional java developers
S9	[32]	2016	Empirical study	Automatic test case generation	110 Open source projects from SourceFroge
S10	[55]	2010		Test case generation	Object oriented classes
S11	[9]	2016	Empirical study	Impact of computer science programs on the quality of test cases generation.	Black-box and white-box techniques
S12	[4]	2014		fault detection effectiveness	Five systems (large java programs)
S13	[10]	2011	Empirical study	Analysis of test case mistakes in test design phase	500 test cases by novice testers
S14	[56]	2017		Diagnosability of a test suite for spectrum-based fault localization approaches	

Table 2. Test Case Quality Metrics used in the Primary Studies

No	**Metric**	**Description**	**Studies**
1.	Test Team Experience	Skills and experience of test team on software testing.	S1, S11
2.	Quality of Document Test Cases (QDT)	Test case documentation is included a general information of a test case such as the name of the pattern, scope and expected results.	S1
3.	Fault Density	Derived measure defined as faults per KLOC (Thousand Line of Code).	S5
4.	Code Defect Density	It measures the defects relative to the software size expressed as lines of code i.e., it measures code quality per unit.	S5
5.	Mean Time to Failure	It is the time between failures	S5
6.	Test Case Understandability	How easy to understand a test case regarding the internal and external descriptions?	S2
7.	Test Case Changeability	Changeable structure and style of a test case which allows changes to be ended easily, consistently, and completely.	S2, S7

8.	Test Case Independency	The measurement of the degree of dependency among one test case to other test cases.	S2
9.	Universal	It is considered from test fields and test scenarios in which a test case can be executed.	S2
10.	Test Cohesion	The similarity of text among the methods of test.	S9
11.	Test Coupling (Coupling Between Test Methods)	The high coupling methods have higher textual similarity with the else methods in the test suite.	S9
12.	Size of Test Case	It refers to the line of codes in the test method or the number of assertions in a test case.	S4, S6
13.	Historical Fault Detection	It sees a test case to be effective in the current release if the same test was also able to detect faults in previous releases.	S4, S6
14.	Code Change-Related Metrics (Changed Method Coverage)	Refers to the number of unique methods calls which are the test called and in the previous version had been changed.	S4, S6
15.	Method Coverage	Refers to the inimitable methods number called from the test case through the test execution.	S3, S4, S6
16.	Similarity-Based Metric	The test cases similarity is identified based on their method calls sequences, pulled out from execution traces.	S4, S6
17.	Mutation Analysis	It seeds mutations into programs; the mutation which is non-detected refers to test suite weakness.	S10, S12
18.	Coverage-based Test Adequacy Criteria	Refers to executed of the program when the test case run.	S3, S6
19.	Fault-based Test Adequacy Criteria	Measures the quality of a test case by their capability to detect known faults, as an estimate for their ability for detecting unknown faults.	S3, S6
20.	Statement Coverage	The degree to which a software is being tested.	S3, S12
21.	Decision/Branch Coverage	point to the decisions fraction in the program that is running by its test suite.	S3, S12, S8, S9, S14
22.	Modified Condition Coverage	For a set of test cases to be altered based on a suitable condition.	S3, S12
23.	Test Suite Size	Refers to how many test cases within the test suite.	S9, S12
24.	McCabe’s Cyclomatic Complexity	Refers to the difficulty of a program or module to be tested and maintained.	S8

25.	Fault detection capability	The function call profile with the fault detection capability with the goal to reduce cost is used as an effective measure.	S3
26.	Fault revealing capability	Defect discovery capability is measured and compared with retest-all for effective indicator.	S3
27.	Failure frequency rate	Most frequent failures with relationship to test cases are used as effective measure.	S3
28.	Fault detection rate	Fault detection rate with the cost of analysis used as effective measure.	S3
29.	Defect Discovery Time	Test case execution profile with defect discovery time used as effective measure.	S3
30.	DDU (Density- Diversity-Uniqueness)	It gives an assessment of its efficiency by pinpointing the root driver of defect given when the fault is recognized.	S14

6. Conclusion

Towards building a high-quality software testing, thirty quality metrics have been identified from 14 primary studies through SLR. As stated by former studies, the test cases effectiveness in discovering flaws in most applications are influenced significantly by software quality metrics. In addition, for different applications the metrics may be able to create good test cases quality besides evaluating test case quality. In future, the scope of the research will be expanded to include extra data repositories to obtain as many related articles as possible. In addition, the plan will include the construction of standard for quality of test cases that can be utilized in different applications.

7. Acknowledgement

The authors thank Universiti Utara Malaysia in funding this study under the PBIT grant, S/O code 12319, and Research and Innovation Management Centre, Universiti Utara Malaysia, Kedah for the administration of this study.

References

1. L. Rajamanickam, N. Azlia, B. Mat, S. Norbaya, and B. Daud, “Software Testing : The Generation Tools,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 2, pp. 231–234, 2019.
2. S.-T. Lai, “Test Case Quality Management Procedure for Enhancing the Efficiency of IID Continuous Testing,” *J. Softw.*, vol. 12, no. 10, pp. 794–806, 2017, doi: 10.17706/jsw.12.10.794-806.
3. N. Chauhan, *Software Testing: Principles and Practices*. Oxford university press, 2010.
4. L. Inozemtseva and R. Holmes, “Coverage Is Not Strongly Correlated with Test Suite Effectiveness Categories and Subject Descriptors,” in *ICSE’14*, 2014, pp. 435–445.
5. W. E. Lewis, *Software Testing and Continuous Quality Improvement Third Edithion*. Boca Raton London New York: Taylor & Francis Group, LLC, 2009.
6. P. Liu and H. Miao, “A New Approach to Generating High Quality Test Cases,” *IEEE Comput. Soc.*, vol. 1, pp. 71–76, 2010, doi: 10.1109/ATS.2010.21.
7. S. M. K. Quadri and S. U. Farooq, “Software Testing – Goals , Principles , and Limitations,” *Int. J. Comput. Appl.*, vol. 6, no. 9, pp. 7–10, 2010.
8. H. H. Khan and M. N. Malik, “Software Standards and Software Failures : A Review With the Perspective

of Varying Situational Contexts," pp. 17501–17513, 2017.

9. O. S. Gómez, B. Monte, and B. Monte, "Impact of CS programs on the quality of test cases generation : An empirical study Categories and Subject Descriptors," in *ICSE '16 Companion*, 2016.
10. S. Eldh, H. Hansson, and S. Punnekkat, "Analysis of Mistakes as a Method to Improve Test Case Design," in *Fourth IEEE International Conference on Software Testing, Verification and Validation Analysis*, 2011, pp. 70–79, doi: 10.1109/ICST.2011.52.
11. C. Kaner, "What Is a Good Test Case ?," in *In Software Testing Analysis & Review Conference (STAR) East.*, 2003, pp. 1–16.
12. E. Kupiainen, M. V Mäntylä, and J. Itkonen, "Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies," *Inf. Softw. Technol.*, vol. 62, pp. 143–163, 2015, doi: 10.1016/j.infsof.2015.02.005.
13. T. Bin Noor and H. Hemmati, "A similarity-based approach for test case prioritization using historical failure data," in *26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 58–68.
14. S. Kumar and P. Ranjan, "A phase wise approach for fault identification," *J. Inf. Optim. Sci. ISSN*, 2017, doi: 10.1080/02522667.2017.1380420.
15. Y. G. Kim, H. S. Hong, D.-H. Bae, and S. D. Cha, "Test cases generation from UML state diagrams," *IEE Proceedings-Software*, vol. 146, no. 4, pp. 187–192, 1999.
16. Y. Aziz, "Exploring a keyword driven testing framework: a case study at Scania IT," 2017.
17. G. J. Myers, *The art of software testing*. John Wiley & Sons, 2006.
18. C. Lin, K. Tang, and G. M. Kapfhammer, "Test Suite Reduction Methods that Decrease Regression Testing," *Inf. Softw. Technol.*, 2014, doi: 10.1016/j.infsof.2014.04.013.
19. ISO/IEC/IEEE, "ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing--Part 4: Test techniques," *ISO/IEC/IEEE 29119-4:2015*. pp. 1–149, 2015, doi: 10.1109/IEEESTD.2015.7346375.
20. M. F. Granda, N. Condori-fernández, T. E. J. Vos, and O. Pastor, "A Model-level Mutation Tool to Support the Assessment of the Test Case Quality," in *25TH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS DEVELOPMENT (ISD2016 POLAND)*, 2016, pp. 42–54.
21. P. M. Kamde, V. D. Nandavadekar, and R. G. Pawar, "Value of test cases in software testing," in *Management of Innovation and Technology, 2006 IEEE International Conference on*, 2006, vol. 2, pp. 668–672, doi: 10.1109/ICMIT.2006.262303.
22. R. Romli, S. Sarker, M. Omar, and M. Mahmod, "Automated Test Cases and Test Data Generation for Dynamic Structural Testing in Automatic Programming Assessment Using MC/DC," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, no. 1, p. 120, Feb. 2020, doi: 10.18517/ijaseit.10.1.10166.
23. S. R. Jan, S. T. U. Shah, Z. U. Johar, Y. Shah, and F. Khan, "An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," *Int. J. Sci. Res. Sci. Eng. Technol. (IJSRSET), Print ISSN*, pp. 1990–2395, 2016.
24. L. Paruch, V. Stray, and C. B. Blindheim, "Characteristic traits of Software Testers," in *Evaluation and Assessment in Software Engineering (EASE 2020)*, 2020, doi: https://doi.org/10.1145/3383219.3383270.
25. W. M. McKeeman, "Differential Testing for Software," *Digit. Tech. J.*, vol. 10, no. 1, pp. 100–107, 1998.
26. H. B. Yadav and D. K. Yadav, "A fuzzy logic based approach for phase-wise software defects prediction using software metrics," *Inf. Softw. Technol.*, 2015, doi: 10.1016/j.infsof.2015.03.001.
27. ISO-IEC 25010:2011, *ISO-IEC 25010: 2011 Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models*. ISO, 2011.
28. A. Hussain and E. O. C. Mkpojiogu, "An application of ISO/IEC 25010 standard in the quality-in-use assessment of an online health awareness system," *J. Teknol.*, vol. 77, no. 5, pp. 9–13, 2015.
29. ISO/IEC 9126-1, "Information technology — Software product quality — Part 1: Quality model," *Iso/Iec*

Fdis 9126-1, vol. 2000. ISO; IEC, pp. 1–26, 2000.

30. C. Pfaller, S. Wagner, T. Universit, and M. Wiemann, "Multi-Dimensional Measures for Test Case Quality," in *Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference on*, 2008, pp. 364–368.
31. Z. Nayyar, N. Rafique, N. Hashmi, N. Rashid, and S. Awan, "Analyzing Test Case Quality with Mutation Testing Approach," in *Science and Information Conference 2015*, 2015, pp. 902–905.
32. F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, and A. De Lucia, "Automatic Test Case Generation : What If Test Code Quality Matters ?," in *ISSTA'16*, 2016, pp. 130–141.
33. H. Sharma, D. Gupta, and R. Singh, "Ranking Based Software Quality Assessment Using Experts Opinion," in *2015 International Conference on Computational Intelligence and Communication Networks*, 2015, doi: 10.1109/CICN.2015.277.
34. R. Kazmi, D. N. A. Jawawi, and R. Mohamad, "Effective Regression Test Case Selection : A Systematic," vol. 50, no. 2, 2017.
35. T. Bin Noor and H. Hemmati, "Studying Test Case Failure Prediction for Test Case Prioritization," in *PROMISE*, 2017.
36. C. Karanikolas, G. Dimitroulakos, and K. Masselos, "Early Evaluation of Implementation Alternatives of Composite Data Structures Toward Maintainability," *Trans. Softw. Eng. Methodol.*, vol. 26, no. 2, p. 44, 2017.
37. D. Spadini, F. Palomba, A. Zaidman, M. Bruntink, and A. Bacchelli, "On The Relation of Test Smells to Software Code Quality," 2018.
38. H. Munir, K. Wnuk, K. Petersen, and M. Moayyed, "An experimental evaluation of test driven development vs. test-last development with industry professionals," *Proc. 18th Int. Conf. Eval. Assess. Softw. Eng. - EASE '14*, pp. 1–10, 2014, doi: 10.1145/2601248.2601267.
39. D. Latha and P. Premchand, "Estimating Software Reliability Using Ant Colony Optimization Technique with Salesman Problem for Software Process," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 7, pp. 20–29, Mar. 2018, doi: 10.30534/ijatcse/2018/04722018.
40. Y. D. Salman and N. L. Hashim, "Automatic Test Case Generation from UML State Chart Diagram: A Survey," in *Advanced Computer and Communication Engineering Technology*, Switzerland: Springer International Publishing, 2016, pp. 123–134.
41. B. Maqbool, F. U. Rehman, M. Abbas, and S. Rehman, "Implementation of Software Testing Practices in Pakistan's Software Industry," in *ICMSS 2018*, 2018, doi: 10.1145/3180374.3181340.
42. P. N. Boghdady, N. L. Badr, M. Hashem, and M. F. Tolba, "A Proposed Test Case Generation Technique Based on Activity Diagrams," *Int. J. Eng. Technol. IJET-IJENS*, vol. 11, no. 03, pp. 37–57, 2011.
43. C. Lampasona, J. Heidrich, V. R. Basili, and A. Ocampo, "Software quality modeling experiences at an oil company," *Proc. ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas. - ESEM '12*, p. 243, 2012, doi: 10.1145/2372251.2372296.
44. C. Kaner and W. P. Bond, "Software Engineering Metrics : What Do They Measure and How Do We Know ?," pp. 1–12, 2004.
45. A. Ahmad, T. Siddiqui, and N. A. Khan, "A Detailed Phasewise Study on Software Metrics : A Systematic Literature Review," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 3, no. 3, pp. 1696–1705, 2018.
46. Software Engineering StandardsCommittee, "IEEE Standard for a Software Quality Metrics Methodology, IEEE Std 1061-1998," IEEE, USA, 1998.
47. A. Causevic, D. Sundmark, and S. Punnekkat, "Test Case Quality in Test Driven Development : A Study Design and a Pilot Experiment," in *the EASE 2012*, 2012, pp. 223–227.
48. B. Behkamal, M. Kahani, E. Bagheri, and Z. Jeremic, "A Metrics-Driven Approach for Quality Assessment of Linked Open Data," *J. Theor. Appl. Electron. Commer. Res.*, vol. 9, no. 2, pp. 64–79, 2014, doi: 10.4067/S0718-18762014000200006.

49. K. Juhnke, M. Tichy, and F. Houdek, “Quality Indicators for Automotive Test Case Specifications,” in *SEERTS 2018: Workshop on Software Engineering for Applied Embedded RealTime Systems @ SE18*, 2018, pp. 96–100.
50. A. Adlemo, H. Tan, and V. Tarasov, “Test case quality as perceived in Sweden,” in *5th International Workshop on Requirements Engineering and Testing (RET’18)*, 2018, pp. 9–12, doi: https://doi.org/10.1145/3195538.3195541.
51. K. Park, D. MG, and M. N, “Statistical Analysis of Metrics for Software Quality Improvement,” *arXiv Prepr. arXiv1802.05865*, 2018.
52. C. Jones, *A Guide to Selecting Software Measures and Metrics*. Boca Raton London New York: Taylor & Francis Group, LLC, 2017.
53. Barbara Kitchenham, *Guidelines for performing Systematic Literature Reviews in Software Engineering*, Keele Univ. Durham, UK, 2007.
54. Z. Juan, C. Lizhi, T. Weiqing, and Y. Song, “Test Case Reusability Metrics Model,” no. Icctd, pp. 294–298, 2010.
55. 55. G. Fraser and A. Zeller, “Mutation-driven Generation of Unit Tests and Oracles,” in *ISSTA’10*, 2010, pp. 147–157.
56. A. Perez, Alexandre; Abreu, Rui; van Deursen, “A Test-suite Diagnosability Metric for Spectrum-based Fault Localization Approaches,” in *the 39th International Conference on Software Engineering (ICSE)*, 2017, pp. 654–664, doi: 10.1109/ICSE.2017.66.