Quality Factors of Test Cases: A Systematic Literature Review

Samera Obaid Barraood[1,2], Haslina Mohd[1], Fauziah Baharom[1] and Mazni Omar[1]
[1]Universiti Utara Malaysia, Malaysia, {sammorahobaid@gmail.com,, haslina@uum.edu.my, fauziah@uum.edu.my, mazni@uum.edu.my}
[2] Hadhramout University Mukalla, Yemen, (sammorahobaid@gmail.com}

ABSTRACT
The use of high quality test cases enable the detection of software defect which eventually helps in ensuring the quality of software before being released to end users. Unfortunately, at the moment, the criteria of good test cases are still vague without any specific model to measure the quality of the test cases. Therefore, this study aims to identify the criteria of good test cases based on the findings from studies conducted within the years 2010 to 2018. The Systematic Literature Review (SLR) by Kitchenham approach was adapted in order to comprehensively identify the related criteria. From the review, a total of 310 related articles were found from the IEEE Xplore, ACM Digital Library, and Science Direct databases. The search was then narrowed down using specific key-words and as a result the number relevant articles ended up to 14. From the review of these articles, 30 quality factors of the test cases were identified. These quality factors were further examined, categorized and finalized to be included as the quality factors of test cases evaluation metrics.

Keywords: Test case, test case evaluation metrics, software testing, systematic literature review.

I INTRODUCTION

Software testing is a vital phase of producing high-quality software in order to detect bugs. However, the effectiveness of testing depends on the quality of test cases whereby some test cases are better of detecting failures compared to others (Chauhan, 2010; Inozemtseva & Holmes, 2014). During the testing of software, errors should be revealed as many as possible so that it will not jeopardize its initial requirements and be up to the quality acceptable level (Lewis, 2009; Liu & Miao, 2010; Quadri & Farooq, 2010). There are many reasons lead to software failures such as lack of understanding, poor experience of test case design, and inaccurately tackling varying situational contexts among team members (Eldh, Hansson, & Punnekkat, 2011; Gómez, Monte, & Monte, 2016; Khan & Malik, 2017). Currently, there is no simple formula or prescription for generating good test cases since the designing of test cases is a complex art (Kaner, 2003). To improve the productivity and quality of software testing, testers or developers must be able to measure the quality of test cases and identify the most effective quality metrics (Lai, 2017).

Test case quality metrics are used in various applications particularly in evaluating existing test suites to ensure sufficient number of testing being performed (Noor & Hemmati, 2015). This indicates that the quality and testing metrics had some importance (Kupiainen, Mäntylä, & Itkonen, 2015). This study is conducted to identify appropriate and usable testing metrics for measuring and evaluating the quality of test cases. Three databases were scrutinized in this study namely the IEEE Xplore, ACM Digital Library, and Science Direct. From the review, 310 published articles were discovered to discuss on test case quality issue. After narrowing the search using a specific keyword, the articles were reduced to 14 primary studies of good test cases. Further discussions of the SLR protocol are presented in the following section.

This paper is structured as follows: Section 2 provides the background and related work of test cases quality. Section 3 describes how the research method was conducted. Section 4 presents the results and the last section provides the conclusion of the study.

II BACKGROUND AND RELATED WORK

This section presents the important concepts related to the quality of test cases.

It is important to note that testing should not show the absence of defects since testing should be exhaustive enough covering all possible ways in which a system can be used even though it is impossible in many cases (Kim, Hong, Bae, & Cha, 1999). This triggers the problem of deciding the sufficient number of testing. One of the evaluation suggestions was to ensure that the most significant risks have been addressed by executing test cases covering the most important functional and non-functional requirements of the system as specified by user (Aziz, 2017).

Two common concepts related to any unsuccessful software testing are *fault and failure*. Failure refers to the inability of a system or component to perform a required function according to its specification. This means that failure is the term used to describe the problems in a system on the output side (Bertolino & Marchetti, 2005; Chauhan, 2010). Fault, on the other hand, is a condition that in actual causes a system to produce failure. Fault is considered as having synonymous meaning to the word defect or bug. Therefore, fault represents reasons embedded in any phase of SDLC and results in failures. Failures can also be described as a manifestation of bugs (Chauhan, 2010). The terms fault and failure are strongly related to each other. For instance, some bugs/faults are hidden in the sense that these are not executed, as they do not get the required conditions in the system. As a result, hidden bugs/faults may not always produce failures whereby they may execute only in certain rare conditions (Bertolino & Marchetti, 2005; Chauhan, 2010).

The effective mechanism of reducing the software development risks has been a worth issue to be explored further since it is reported that software project success rate is always under 40% (Lai, 2017).

Lai (2017) summarizes the related risk events into four types; incomplete requirement analysis, new technology and environment evolution, frequent requirements change, and imperfect and inflexibility resources allocation management. These types of risk are frequently difficult to avoid or exclude. Hence, detection and prevention planning is considered the best way to reduce the risk of software developments and produce good quality projects The quality is defined by ISO/IEC 9126 and ISO/IEC 25010 as the extent to which the system satisfies the stated and implied needs of its various users (Hussain & Mkpojiogu, 2015; ISO-IEC 25010:2011, 2011; ISO/IEC 9126-1, 2000). The following section will provide a background on test cases quality.

A. **Test Case Quality**

A test case, a set of preconditions, inputs (including actions, where applicable) and expected results, is developed to determine whether or not the covered part of the test item has been implemented correctly (ISO/IEC/IEEE, 2015; Lin, Tang, & Kapfhammer, 2014).

In testing, the absence of all errors cannot be guaranteed. Therefore, the most important consideration in program testing is the design and creation of effective test cases. In addition, the design of test-case is necessary because a complete testing is impossible to achieve. The most significant strategy is to try to conduct the testing as complete as possible (Myers, 2006).

A test case is an important asset in a software development. In the software industry, a test case design has been the principal since the quality of the test case substantially affects how well the system is tested, what failures will be found and what coverage can be achieved. In addition, the major purpose of test cases is to find the undiscovered code errors or defects (Eldh et al., 2011; Lai, 2017).

There is no simple formula or prescription for generating "good" test cases because the procedure too complex. Nevertheless, there are tests that are good for software development purposes in determining the type of information required (Kaner, 2003). A good test case is one that has a high probability of finding an as-yet undiscovered error (Liu & Miao, 2010). The complexity of designing good test cases comes from three sources (Kaner, 2003):

- *Test cases help to discover information*. Different types of tests are more effective for different classes of information.
- *Test cases can be "good" in a variety of ways.* No test case will be good in all of them.
- *People tend to create test cases according to certain testing styles*, such as domain or risk-based testing. *Good domain tests are different from good risk-based tests*.

The number of revealed failures by test cases can determine its effectiveness. If a test case reveals more failures, then the quality of it will be higher (Gómez et al., 2016)**.**

Test case quality is a desirable and important goal in test case generation (Gómez et al., 2016; Palomba, Panichella, Zaidman, Oliveto, & Lucia, 2016; Sharma, Gupta, & Singh, 2015). Therefore, the selection (Kazmi, Jawawi, & Mohamad, 2017) and prioritization (Noor & Hemmati, 2015, 2017) have to be prioritized since poorly design tests have been proven to negatively impact future maintenance activities (Karanikolas, Dimitroulakos, & Masselos, 2017; Palomba et al., 2016), software reliability (Sharma et al., 2015; Yadav & Yadav, 2015), and software productivity (Munir, Wnuk, Petersen, & Moayyed, 2014).

Quality of test cases depends on the coverage of all the functionalities in a system under testing. The coverage of a quality test cases is often described using certain criteria (Salman & Hashim, 2016).

The test cases should then be validated against some known quality standards to ensure that they are in an acceptable form as well as (Boghdady, Badr, Hashem, & Tolba, 2011). The quality characteristics that can be used in the test design techniques as included in the ISO/IEC 25010: 2011. However, the model is difficult to be applied due to its operational complications (Lampasona, Heidrich, Basili, & Ocampo, 2012). This proves that the quality factors/metrics need to be identified in producing high-quality test cases. Therefore, this study is conducted to identify the good quality factors/metrics of test cases. The following section provides an overview of the quality metrics used in previous studies.

B. **Quality Metrics**

Software quality is essential because not only the software is used in diverse area of various applications but also several historical events have indicated the impact of software failures around the world. The consequences of software failures may result in monetary and human losses. Thus, issues related to software quality becomes a major research area and should be unavoidable (Yadav & Yadav, 2015).

Effective quality metrics of a test case measurement is vital in improving the productivity and quality of software (Lai, 2017). A *metric* is a function assigned to a value of an attribute (Kaner & Bond, 2004). 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” (Software Engineering Standards Committee, 1998). Several studies have looked into the related aspects of quality metrics. For example, Kaner et al. (2004) provide ten questions to explain about the software engineering metrics and a framework on how to perform the evaluation whilst, Kupiainen, Mäntylä, and Itkonen, (2015) present the reasons for and effects of using metrics in industrial agile development. In their study, Kupiainen et al. have extracted 102 metrics from the primary studies in their SLR by focusing only on the metrics used by the agile teams and analyzing on the influence of the identified metrics. They hypothesized that Agile methods do not provide any special protection from the dysfunctional use of metrics even when using the core metrics of Agile development. The hypothesis based on the results of their study revealed that the use of metrics can have negative effects and drive dysfunctional behavior. A metric-driven approach proposed by Behkamal, Kahani, Bagheri, and Jeremic (2014) consists of 20 metrics for evaluating the inherent quality characteristics of a dataset before it is released to the Linked Open Data Cloud. Based on a SLR and the ISO/IEC 25012 standard they selected five inherent quality characteristics, which are semantic accuracy, syntactic accuracy, uniqueness, consistency, and completeness. The test case quality measurement model is proposed by (Lai, 2017) for enhancing the efficiency of Iterative and Incremental Development (IID) continuous testing. This model consists of four indicators; qualified documentation, manageability, maintainability and reusability quality characteristics.

III **RESEARCH METHODOLOGY**

Systematic Literature Review (SLR) was chosen as a research method because the study is more about trying to understand a problem than trying to find a solution to it (Kupiainen et al., 2015). Five reasons for conducting an SLR include, first, to aggregate and synthesize existing knowledge regarding a research topic. Second, to identify gaps in the earlier research. Third, to provide background information to start investigating a new research topic. Fourth, to provide a repeatable research method which, when applied properly, should provide sufficient detail to be replicated by other researchers. Fifth, the detailed documentation of the performed steps within the SLR enables in-depth evaluation of the conducted study (Kupiainen et al., 2015). In this study, SLR is used to perform an extensive study on the quality of test cases as well as identifying the factors and metrics that produce good test cases within the period of 2010 to 2018. The guidelines provided by Kitchenham (2007) were used as a basis to develop the SLR protocol. In the following subsections, the research questions, search and selection process, inclusion and exclusion criteria, and data extraction are described.

A. **Research Questions**

The main objective of this study is to determine the factors that affect the quality of test cases. In addition, the focus is more on the metrics and measurements of the test cases that can produce a good quality testing. Hence, the research questions are:

RQ1: How much are the research activities conduected related to the quality of test cases for the last 8 years (2010-2018)?

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?

B. **Search and Selection Process**

The search and selection process was performed for determining related primary studies. The process contains three steps as shown in Table 1.

Step 1: *Select Source Repositories:*

In this step, appropriate databases for this study were selected. Three databases were used which include IEEE Xplore, ACM Digital Library, and Science Direct. Based on Kazmi et al. (2017) recommendation, the first two were chosen because both databases covered almost all important conferences, while Science Direct include almost all important journals in the domain of testing. The process began by entering the keywords that are related to the research questions.

To obtain the most relevant search results, the string with (OR, AND) operators was switched according to the time span between 2010 and 2018. In this study two stages of searching were done; the first one with string ("test case" OR "test case quality") AND ("metrics" OR "factors" OR "indicators"). The total papers in this stage were 268, as presented in Table 1. However, once the selected articles were fully read, it is observed that some of the studies have used the term "effectiveness of test cases" instead of the "quality of test cases". Therefore, a second stage of searching was performed inside the selected data depositories with this string; "test case effectiveness" OR "the effectiveness of test case". The total papers in this stage were 42, as portrayed in Table 1.

Step 2: *Read Titles and Abstracts*

As shown in Table 1, papers were included and excluded based on their titles and abstracts. The content of each paper was skimmed through in case of unclear abstracts. The selected papers in this step were 39 from the first stage of research and 15 from the second stage, so the total is 54 based on the inclusion and exclusion criteria (section C). One of these papers was replicated.

Step 3: *Read Full Text*

The studies included in this step were selected by reading the full text. The output of this step, which was only 14 papers (13 from the first stage and only one from the second stage of search steps), was the papers that were related to this study based on the selection criteria.

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 articles		

C. **Inclusion and Exclusion Criteria**

Inclusion Criteria:

- Papers that present the factors or metrics of testing quality.
- Papers that talk about good test cases.

Exclusion Criteria:

- Papers that are not in English.
- Papers that do not contain the quality factors or metrics.
- Papers that do not relate to testing.
- Books and workshops.

D. **Data Extraction**

The data extraction was performed by reading the complete text of all the selected papers. The data collected from the selected papers were extracted in two phases. In the first phase, the standard information (Kithcenham, 2007) was collected, which include the title, authors name, publication year, and summary of the study. The second phase contains the information that directly related to the research questions of this study.

IV **RESULTS**

This section presents the results of the SLR and provides answers for the research questions. The following subsections describe an overview of the primary studies and present the quality factors/ metrics of the test case.

RQ1: How much are the research activities in the quality of test cases for the last 8 years (2010-2018)? 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 A provides more details about the selected studies.

A. **Overview of Studies**

This section presents the overview of the primary studies related to quality test cases. The 14 selected studies were discovered from the three data depositories (IEEE Xplore, ACM Digital Library, and Science Direct) within the period of 2010-2018 (as in Table 1). Most of the studies (8) were published in the ACM Digital Library, followed by IEEE Xplore (5) and Science Direct (1).

Table 2 presents the details of the 14 selected studies. 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.

Table 2. Details of the Selected Studies.

Study	Reference	Year	Study Type	Study Focus	Apply on
S1	Yadav & Yadav	2015		Software reliability	
S2	Juan, Lizhi, Weiqing, & Song,	2010		Reusability of test cases	
S3	Kazmi et al.	2017	SLR	Test case selection	Regression testing
S4	Noor & Hemmati	2015		Test case quality for prioritization	Five open source systems (java projects)
S5	Sharma et al.	2015	Online survey	Software reliability	
S6	Noor & Hemmati	2017	Empirical study	Test case prioritization	five open source systems (java projects)
S7	Karanikolas, Dimitroulakos, & Masselos	2017		Predicting software maintenance.	Object-oriented software
S8	Munir et al.	2014	a controlled experiment	TDD (Test Driven Development) on internal, external code quality and productivity	Professional java developers
S9	Palomba et al.	2016	Empirical study	Automatic test case generation	110 Open source projects from SourceFroge
S10	Fraser & Zeller	2010		Test case generation	Object oriented classes
S11	Gómez et al.	2016	Empirical study	Impact of computer science programs on the quality of test cases generation.	Black box and white box methods
S12	Inozemtseva & Holmes	2014		fault detection effectiveness	Five systems (large java programs)
S13	Eldh et al.	2011	Empirical study	Analysis of test case mistakes in test design phase	500 test cases by novice testers
S14	Perez, Alexandre; Abreu, Rui; van Deursen	2017		Diagnosability of a test suite for spectrum-based fault localization approaches	

As shown in Table 2, 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 focused exactly on the quality of test cases. Most of these studies are generally either focusing on the use of or proposing quality metrics for specific purposes. Among the purposes include test case generation [S5, S9, S11], test case selection [S3], test case prioritization [S4, S6], software maintenance [S7, S9], software reliability [S1, S5], productivity [S8], diagnosability of a test suite [S14], and test case design mistakes analysis [S13]. Furthermore, the table portrays that almost all studies described on the quality of test cases in terms of structural design 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. The results show that in the period of 2010 to 2018, the quality of test cases is important in various domains and techniques particularly in software reliability [S1, S5], software maintenance [S7, S9], software productivity [S8], reusability [S2], test case selection [S3], test case generation [S9, S10, S11], test case prioritization [S4, S6], and test suite diagnosability [S14].

Table 3. 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 cases that are designed to expose defects.	S1
3.	Fault Density		S5
4.	Code Defect Density		S5
5.	Mean Time to Failure		S5
6.	Test Case Understandability	How easy to understand a test case in terms of its internal and external descriptions?	S2
7.	Test Case Changeability	Changeable structure and style of a test case which allows changes to be made easily, completely, and consistently.	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 reflected from test scenarios and test fields in which a test case can be executed.	S2
10.	Test Cohesion (Lack of Cohesion of a Test Method)	Textual similarity among the tested methods.	S9
11.	Test Coupling (Coupling Between Test Methods)	Methods with high coupling have higher textual similarity with the other methods contained in the test suite.	S9
12.	Size of Test Case	It refers to the LOC (Line of Codes) in the test method or the number of assertions in a test case.	S4, S6
13.	Historical Fault Detection	It considers 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 that are called by the test and have been changed since the previous version.	S4, S6
15.	Method Coverage	Refers to the number of unique methods called from the test case (directly or indirectly) during the test execution.	S3, S4, S6
16.	Similarity-Based Metric	The similarity between test cases is defined based on their sequences of method calls, extracted from execution traces.	S4, S6
17.	Mutation Analysis	It seeds artificial defects (mutations) into programs; a non-detected mutation indicates a weakness in the test suites.	S10, S12
18.	Coverage-based Test Adequacy Criteria	Refers to how much of the program is executed when the test case run.	S3, S6
19.	Fault-based Test Adequacy Criteria	Measures the quality of a test case by their ability 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	Refers to the fraction of decisions (branches) in the program that are executed by its test suite.	S3, S12, S8, S9, S14
22.	Modified Condition Coverage	For a test suite to be modified based on adequate condition.	S3, S12

23.	Test Suite Size	The number of test cases in the test suite.	S9, S12
24.	McCabe's Cyclomatic Complexity	Indicates how difficult 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 provides an assessment of its efficiency by pinpointing the root cause of failure given when an error is detected.	S14

Thirty quality metrics are identified from the 14 primary studies as shown in Table 3. The most used metric is Coverage [S3, S4, S6, S8, S9, S12 and S14]. The Coverage metric has various types such as statement, branch, method, and condition. Only one of the studies used all these types [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 a completeness of test suites (Kazmi et al., 2017). However, some studies [S3, S12] do not recommended using coverage as the only measure because it is insufficient and not a good quality measurement for test suite effectiveness. The studies recommended that it would be better to combine the use of coverage with other metrics. [S9, S14] used branch coverage metric for comparison with their proposed metrics. [S10] commented the used of mutations rather than coverage because the former not only know where to test but also what to test for. On the other hand, [S13] instead of providing any quality metrics for usage, the authors try to improve the quality of test cases by analyzing the mistakes of test cases based on the knowledge of test cases writers. They found that most of the test cases have a low level of quality because of the lack of understanding regarding the corresponding knowledge, which is important for test case design.
In general, all identified quality metrics from the selected primary studies are used for producing good test cases. The metrics are identified either based on the previous release of the system, current release, similarity, diagnosability of the test cases, or experience of the test team.

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

V CONCLUSION

This study provides an overview of the test case quality metrics. In particular, this study identifies good test cases suggested by previous researchers, which consequently may lead towards high-quality of software testing. This SLR had identified 30 quality measures based on the 14 relevant articles as stated in Table 2. Based on the previous studies, software quality metrics significantly affected the effectiveness of the test cases in revealing the software defects in most system applications. In addition, these metrics can be used not only for evaluating the quality of test cases for different applications but also for being able to generate good quality of the test cases. This SLR will be expanded in the future by including more articles from various data depositories that are related to software quality metrics and test cases. The plan will also include the construction of standard for quality of test cases that can be applied in various applications.

ACKNOWLEDGEMENT

The co-author wishes to 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

Aziz, Y. (2017). *Exploring A Keyword Driven Testing Framework: A Case Study At Scania IT.*

Barbara Kitchenham. (2007). *Guidelines For Performing Systematic*

Literature Reviews In Software Engineering (Keele Univ). Durham, UK.

Behkamal, B., Kahani, M., Bagheri, E., & Jeremic, Z. (2014). A Metrics-Driven Approach For Quality Assessment Of Linked Open Data. *Journal Of Theoretical And Applied Electronic Commerce Research*, *9*(2), 64–79. Http://Doi.Org/10.4067/S0718-18762014000200006

Bertolino, A., & Marchetti, E. (2005). A Brief Essay On Software Testing. *Software Engineering, The Development Process*, *3*.

Boghdady, P. N., Badr, N. L., Hashem, M., & Tolba, M. F. (2011). A Proposed Test Case Generation Technique Based On Activity Diagrams. *International Journal Of Engineering & Technology IJET-IJENS*, *11*(03), 37–57.

Chauhan, N. (2010). *Software Testing: Principles And Practices*. Oxford University Press.

De S Campos Junior, H., Araújo, M. A. P., David, J. M. N., Braga, R., Campos, F., & Ströele, V. (2017). Test Case Prioritization: A Systematic Review And Mapping Of The Literature. In *Proceedings Of The 31st Brazilian Symposium On Software Engineering* (Pp. 34–43). ACM.

Eldh, S., Hansson, H., & Punnekkat, S. (2011). Analysis Of Mistakes As A Method To Improve Test Case Design. In *Fourth IEEE International Conference On Software Testing, Verification And Validation Analysis* (Pp. 70–79). IEEE Computer Society. Http://Doi.Org/10.1109/ICST.2011.52

Fraser, G., & Zeller, A. (2010). Mutation-Driven Generation Of Unit Tests And Oracles. In *ISSTA'10* (Pp. 147–157). Trento, Italy: ACM.

Gómez, O. S., Monte, B., & Monte, B. (2016). Impact Of CS Programs On The Quality Of Test Cases Generation : An Empirical Study Categories And Subject Descriptors. In *ICSE '16 Companion*. Austin, TX, USA: ACM. Retrieved From Doi: Http://Dx.Doi.Org/10.1145/2889160.2889190

Hussain, A., & Mkpojiogu, E. O. C. (2015). An Application Of ISO/IEC 25010 Standard In The Quality-In-Use Assessment Of An Online Health Awareness System. *Jurnal Teknologi*, *77*(5), 9–13.

Inozemtseva, L., & Holmes, R. (2014). Coverage Is Not Strongly Correlated With Test Suite Effectiveness Categories And Subject Descriptors. In *ICSE'14* (Pp. 435–445). Hyderabad, India: ACM. Retrieved From Http://Dx.Doi.Org/10.1145/2568225.2568271

ISO-IEC 25010:2011. (2011). *ISO-IEC 25010: 2011 Systems And Software Engineering-Systems And Software Quality Requirements And Evaluation (Square)-System And Software Quality Models*. ISO.

ISO/IEC/IEEE. (2015). ISO/IEC/IEEE International Standard - Software And Systems Engineering--Software Testing--Part 4: Test Techniques. *ISO/IEC/IEEE 29119-4:2015*. Http://Doi.Org/10.1109/IEEESTD.2015.7346375

ISO/IEC 9126-1. (2000). Information Technology — Software Product Quality — Part 1: Quality Model. *Iso/IEC Fdis 9126-1*. ISO; IEC. Retrieved From Http://Www.Cse.Unsw.Edu.Au/~Cs3710/Pmmaterials/Resources/9126-1 Standard.Pdf

Juan, Z., Lizhi, C., Weiqing, T., & Song, Y. (2010). Test Case Reusability Metrics Model. In *2nd International Conference On Computer Technology And Development* (Pp. 294–298).

Kaner, C. (2003). What Is A Good Test Case ?, 1–16.

Kaner, C., & Bond, W. P. (2004). Software Engineering Metrics : What Do They Measure And How Do We Know ?, 1–12.

Karanikolas, C., Dimitroulakos, G., & Masselos, K. (2017). Early Evaluation Of Implementation Alternatives Of Composite Data Structures Toward Maintainability. *Transactions On Software Engineering And Methodology*, *26*(2), 44. Retrieved From Https://Doi.Org/10.1145/3132731

Kazmi, R., Jawawi, D. N. A., & Mohamad, R. (2017). Effective Regression Test Case Selection : A Systematic, *50*(2).

Khan, H. H., & Malik, M. N. (2017). Software Standards And Software Failures : A Review With The Perspective Of Varying Situational Contexts, 17501–17513.

Kim, Y. G., Hong, H. S., Bae, D.-H., & Cha, S. D. (1999). Test Cases Generation From UML State Diagrams. *IEE Proceedings-Software*, *146*(4), 187–192.

Kupiainen, E., Mäntylä, M. V, & Itkonen, J. (2015). Using Metrics In Agile And Lean Software Development – A Systematic Literature Review Of Industrial Studies. *Information And Software Technology*, *62*, 143–163. Http://Doi.Org/10.1016/J.Infsof.2015.02.005

Lai, S.-T. (2017). Test Case Quality Management Procedure For Enhancing The Efficiency Of IID Continuous Testing. *Journal Of Software*, *12*(10), 794–806. Http://Doi.Org/10.17706/Jsw.12.10.794-806

Lampasona, C., Heidrich, J., Basili, V. R., & Ocampo, A. (2012). Software Quality Modeling Experiences At An Oil Company. *Proceedings Of The ACM-IEEE International Symposium On Empirical Software Engineering And Measurement - ESEM '12*, 243. Http://Doi.Org/10.1145/2372251.2372296

Lewis, W. E. (2009). *Software Testing And Continuous Quality Improvement Third Edition*. Boca Raton London New York: Taylor & Francis Group, LLC.

Lin, C., Tang, K., & Kapfhammer, G. M. (2014). Test Suite Reduction Methods That Decrease Regression Testing. *INFORMATION AND SOFTWARE TECHNOLOGY*. Http://Doi.Org/10.1016/J.Infsof.2014.04.013

Liu, P., & Miao, H. (2010). A New Approach To Generating High Quality Test Cases. *IEEE Computer Society*, *1*, 71–76. Http://Doi.Org/10.1109/ATS.2010.21

Munir, H., Wnuk, K., Petersen, K., & Moayyed, M. (2014). An Experimental Evaluation Of Test Driven Development Vs. Test-Last Development With Industry Professionals. *Proceedings Of The 18th International Conference On Evaluation And Assessment In Software Engineering - EASE '14*, 1–10. Http://Doi.Org/10.1145/2601248.2601267

Myers, G. J. (2006). *The Art Of Software Testing*. John Wiley & Sons.

Noor, T. Bin, & Hemmati, H. (2015). A Similarity-Based Approach For Test Case Prioritization Using Historical Failure Data. In *26th International Symposium On Software Reliability Engineering (ISSRE)* (Pp. 58–68). IEEE.

Noor, T. Bin, & Hemmati, H. (2017). Studying Test Case Failure Prediction For Test Case Prioritization. In *PROMISE*. Toronto, Canada: ACM. Retrieved From Https://Doi.Org/10.1145/3127005.3127006

Palomba, F., Panichella, A., Zaidman, A., Oliveto, R., & Lucia, A. De. (2016). Automatic Test Case Generation : What If Test Code Quality Matters ? In *ISSTA'16* (Pp. 130–141). Saarbrücken, Germany: ACM. Retrieved From Http://Dx.Doi.Org/10.1145/2931037.2931057

Perez, Alexandre; Abreu, Rui; Van Deursen, A. (2017). A Test-Suite Diagnosability Metric For Spectrum-Based Fault Localization Approaches. In *The 39th International Conference On Software Engineering (ICSE)* (Pp. 654–664). IEEE. Http://Doi.Org/10.1109/ICSE.2017.66

Quadri, S. M. K., & Farooq, S. U. (2010). Software Testing – Goals, Principles, And Limitations. *International Journal Of Computer Applications*, *6*(9), 7–10.

Salman, Y. D., & Hashim, N. L. (2016). Advanced Computer And Communication Engineering Technology: Proceedings Of ICOCOE 2015. *Lecture Notes In Electrical Engineering*, *362*(December). Http://Doi.Org/10.1007/978-3-319-24584-3

Sharma, H., Gupta, D., & Singh, R. (2015). Ranking Based Software Quality Assessment Using Experts Opinion. In *2015 International Conference On Computational Intelligence And Communication Networks*. IEEE. Http://Doi.Org/10.1109/CICN.2015.277

Software Engineering Standardscommittee. (1998). *IEEE Standard For A Software Quality Metrics Methodology, IEEE Std 1061-1998*. USA: IEEE.

Yadav, H. B., & Yadav, D. K. (2015). A Fuzzy Logic Based Approach For Phase-Wise Software Defects Prediction Using Software Metrics. *INFORMATION AND SOFTWARE TECHNOLOGY*. http://doi.org/10.1016/j.infsof.2015.03.001