Article

Testability as a Measure for Improving Software Quality in System Analysis and Design

Kardan Journal of Engineering and Technology
1 (1) 48–57
©2019 Kardan University
Kardan Publications
Kabul, Afghanistan
DOI: 10.31841/KJET.2021.5
https://kardan.edu.af/Research/Kardan_journal_of_engineering_and_technology.aspx#

Irfanullah Khan
Abdul Khaliq Khan Shinwari
Zaidullah

Abstract

Software quality assurance is being an emerging field of software engineering, advocating more stable solutions to real-world systems strongly. SQA is a formal procedure used to evaluate, document, and assure the quality of the work products at all phases of SDLC. Different factors may affect the quality of the system. The primary purpose of this research is to identify and understand various quality factors and their effect on software/system development. The relationships between these factors and SDLC phases are investigated, and another quality factor testability is proposed to the quality factor set of the system analysis & design. The expected results of the proposed solution have shown the importance of testability, specifically at system analysis & design phase of software/system development.

Keywords: *Software Quality, system analysis & design, testability*

Mr. Irfanullah Khan, *Assistant Professor at Kardan University, Kabul, Afghanistan.*
Mr. Abdul Khaliq Khan Shinwari, *Assistant Professor at Kardan University, Kabul, Afghanistan.*
Mr. Zaidullah, *Assistant Professor at Kohat University of Science and Technology, Kohat, Pakistan*

Introduction

Software quality assurance is becoming one of the emerging fields of software engineering and a formal technique to assure, evaluate, and document quality. This technique can be used in every phase of SDLC to assure the software product quality. In modern software development, SQA is very critical as it ensures the quality of processes and methods being used in software engineering. Software requirements that show conformance to the ANSI/IEEE standard are the base on which the quality can be measured [12]. However, the quality of the software may be affected by various factors and to achieve the highest level of quality; these factors need to be identified to speed up the activities involved in software development. Identification of these factors, finding the relationship among these factors and suitable mapping among various factors is a human-centric and a time-consuming job [2].

System analysis and design is a well-organized, planned, systematic approach used for developing information systems. SAD is used to understand and specify the detailed requirements of the system and the components of the system. Furthermore, how those system components should be implemented so that they work together accurately and efficiently. A system analyst is responsible for analyzing the requirements, design, and implement the system by using analysis and design concepts, methodologies, and techniques. While, in a case the system analysis & design concepts is not in use, the system may not just be able to solve the intended problem but also can be a significant cause of user dissatisfaction. System analysis & design also involves dealing with the current and ultimate users of the system to support them to work and get familiarity with new technologies [7]

Testability means the ability to test, the ease with which the software can be tested or the degree to which testing gives confidence of accuracy and helps to produce high-quality products. According to IEEE, "*testability is the specific defined standard to which a system or component facilitates the test overall procedure as well as the confirmation of the result validity.*" Testability must be performed as earlier in the development as possible because the higher the testability, the better tests [11].

The rest of the paper is organized as;

Section II includes the literature review, Section III presents the proposed solution, Section IV concluded the paper with the significant outcome, and finally, Section V outlined the directions and scope for future of the proposed solution.

2 Literature Review

An effective software project management means, to manage project not just only within time, budget, and resources but also with the highest level of quality. To assure software for its quality, testing is considered to be an effective mean, since testing requires 40-50% of the developmental efforts. Several test procedures, mainly under the black box, white box, and gray box, were highlighted. Furthermore, the testing phase of SDLC was also being overviewed as an integral phase to assure quality. Moreover, the testing technique need to be used not just only to measure the quality level of one phase of SDLC but also for a particular quality attributes. An extended model was proposed, which include testing in every phase of SDLC along with the direction of selecting a particular testing type to be applied to a particular phase [9].

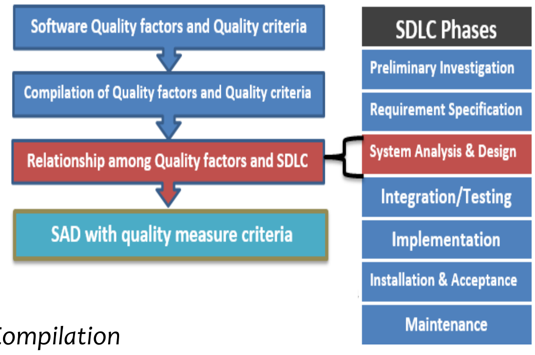
Software testing is more tangible than SQA and is, therefore, not more challenging; this means problems solution visibility is higher than avoiding problems. SQA is planned and well-organized series of activities to produce the highest degree of confidence in a software product, and therefore able to qualify the criteria for which it was intended to be developed. To enhance and assure software quality, the standardized plan would be required, activities performed must be according to the project, and based on the standard. Moreover, Role of SQA team, SQA architecture, and QA activities was proposed [6].

Software Quality Assurance Process (SQAP) were activities planned to assess and assure the process by which the software systems were engineered and also making sure that the work done was according to the standards. For software quality assurance, the associated factors that affect the quality of the software product were identified, which would reasonably be helpful in the future to speed up the software development practices and procedures. The relationship among several factors needs to be identified. To achieve the highest level of quality in a software product, SQAP was required to be involved in four phases, including requirement engineering, converting requirement into the design, implementation, and maintenance. The identified factors after extensive research were being listed and further being classified as influencing factors and variable factors [3].

To assure quality in SDLC phases, a quantitative analysis method including seven essential tools of quality, Statistical process control, Six Sigma, has been highlighted. The proposed methods can be used to manage and control the quality of software during SDLC. However, the relevant statistical technique was required to identify testability and reliability [8]

The quality factors and its quality criteria were compiled along with its description in popularity order based on four quality models. A model is a group of factors, and every factor is a collection of various criteria. The quality factors occurrence and the frequency of occurrence of quality criteria in quality models were highlighted. To develop a quality software product, quality attribute description is required, which was based on some quality factors and its associated criterion. It was essential to know the relationship among factors, criteria, and quality attributes. As a result, portability and reliability were the most popular one as it exists in all four quality models as shown in figure 1. The least popular factors were also being identified [4].

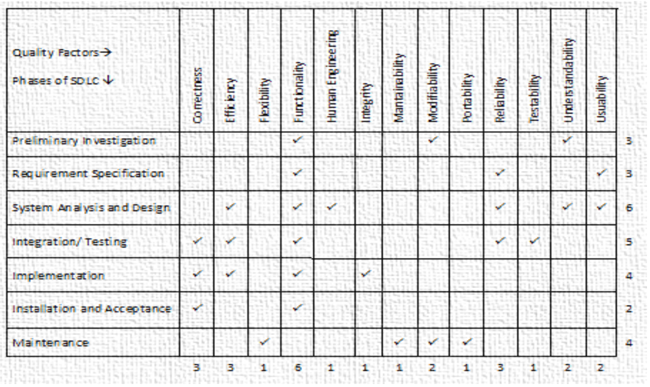
Figure 1: Compilation of Quality Factors and Criteria


Software Quality factors and Quality criteria
Compilation of Quality factors and Quality criteria
Relationship among Quality factors and SDLC
SAD with quality measure criteria
SDLC Phases
Preliminary Investigation
Requirement Specification
System Analysis & Design
Integration/Testing
Implementation
Installation & Acceptance
Maintenance

Source: Author's Compilation

The four quality models that are McCall's, Boehm's, Dromey's, and ISO/IEC 9126 Standards, along with SDLC Phases, were discussed. The relationship between SDLC phases and quality factors of various quality models were highlighted. Based on the relationship, factors relevant to a particular phase of SDLC will eventually give a thorough understanding that would lead us towards quality software development [5].

Table1: Relation Among SDLC Phases & Quality Factor



Quality Factors→ Phases of SDLC ↓	Correctness	Efficiency	Flexibility	Functionality	Human Engineering	Integrity	Mantainability	Modifiability	Portability	Reliability	Testability	Understandability	Usuability	
Preliminary Investigation				✓				✓				✓		3
Requirement Specification				✓						✓			✓	3
System Analysis and Design		✓		✓	✓					✓		✓	✓	6
Integration/ Testing	✓	✓		✓						✓	✓			5
Implementation	✓	✓		✓		✓								4
Installation and Acceptance	✓			✓										2
Maintenance			✓				✓	✓	✓					4
	3	3	1	6	1	1	1	2	1	3	1	2	2	

Source: Relation among SDLC phases & quality factor [5]

The above table showing the importance of system analysis & design phase during which six factors which were Efficiency, Functionality, Human Engineering, Reliability, Understandability and Usability must be considered to achieve the quality of the work product [5].

The details of these factors are as under:

Efficiency: The software should utilize system resources in the best possible manner. These resources are namely, time and space.

Functionality: The software should be capable of following rules, laws, and policies and in the same way offers functionality as per customer requirement.

Human Engineering: including the internal strength of the software, dependability, user-friendliness, and human-computer interface (HCI).

Reliability: The software should be able not to fail under various conditions. The software should perform as per customer expectation and satisfaction.

Understandability: The amount of effort needed to understand the system. The intention of the software should be clear, consistent use of variables, modules, and control structure should be in accordance with the standards.

Usability: The amount of effort required to use the software and on the valuation of such use by end-users. The software should work according to the standards, under-allocated resources, and robust. These factors shall be considered at the preliminary phases of the SDLC to achieve the quality being desired in the end product.

3 Proposed Solution

According to the research question, quality factor testability is proposed to be added to the quality factor set to achieve quality work product in system analysis & design phase of SDLC [4] [5] [9] [10].

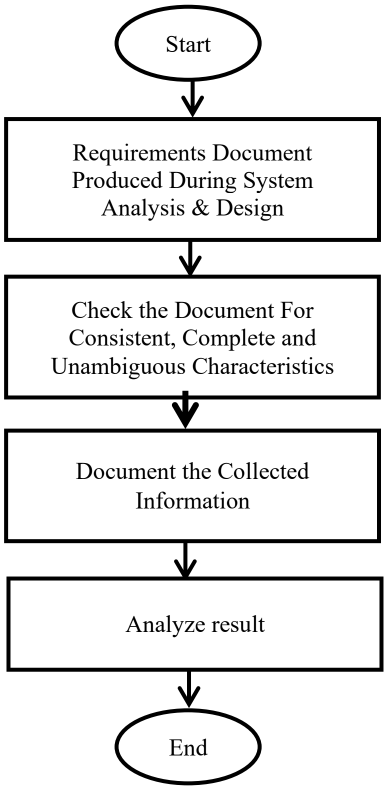
To fulfill the criteria for testability, the requirements document need to be consistent, complete, and unambiguous. According to IEEE std-830-1998.

Consistent: *"Explains the phenomena of extremely interrelated aspect, that makes a logical organize sense that all activities written in the requirements document should be interconnected with each other [1]."*

Complete: *"Completeness in the requirements document encircle all the required options that must fulfill the overall information [1]."*

Unambiguous: *"Unambiguous in requirements document refers to the concept that the related document must not be interpreted repeatedly [1]."*

Figure 2: Generic View of Proposed Solution


Start
Requirements Document Produced During System Analysis & Design
Check the Document For Consistent, Complete and Unambiguous Characteristics
Document the Collected Information
Analyze result
End

Source: Author's Compilation

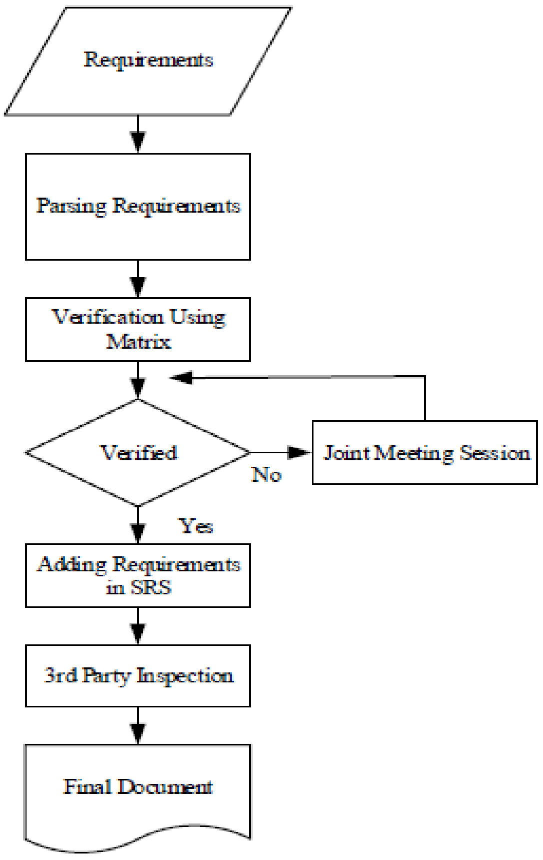
Several studies have been conducted that resulted in producing a quality requirements document.

One the studies proposed four processes as shown in the following figure 3. These processes can be applied to assure that the requirements document was consistent, complete, and unambiguous.

As shown in the above figure3, as a first step, the requirements document was taken as an input for parsing requirements which was helpful in identifying real needs of the customer, domain requiring extraction and to ascertain the requirement for completeness. The output of parsing will then complete requirement and could be expressed as $R(Total) = N + \alpha$.

In second step, verification using matrix was applied to the output of the first step. The primary reason of this mapped requirement activity was to assure correctness of the error, unambiguity and consistency of the requirements document. In this step, 1 would be assigned in a case when stakeholder agreed on the requirement otherwise 0 would be assigned. Finally all the issues or all 0's were discussed, verified, revised and replaced them to 1's. As a result, Ri will be equal to sum of all the requirements. The output requirement was then incorporated in the requirements document as per IEEE SRS std-830-1998 [1]. In third step, the requirements document was submitted to 3rd party inspection, where IEEE SRS std-830-1998 models i.e. Pragmatic Quality Model (PQM) and Perspective based Reading (PBR) were used to inspect the document as a whole. Total quality score was obtained as a result of 3rd party inspection [13].

Figure 3: Four Process Flow Diagram



Source: Four Process Flow Diagram [13]

Case Study:

To experiment with the proposed method, the researchers have examined testability that was consistent, complete, and unambiguous. According to the proposed solution, the requirements document produced during system analysis and design needed to be tested by using the four process approach [13] as shown in figure 3.

According to figure 2, we took a Pass/PIN safe requirements document as a case. Pass/PIN safe is an open source OSI certified application offering solution to the problems of password memorization.

In parsing method, three new functional requirements were discovered. These functional requirements were missed during the system analysis and design phase. As a result, completeness of the requirements was assured. As per parsing rules, following three requirements were being extracted.

- Primary Password must be maximum in length possibly more than 150 characters.
- Search data related to the word entered either in user name or in password.
- Database should be unlocked by providing primary password and must not be opened until the primary password is entered.

In verification using matrix, two of the requirements had some inconsistencies. The anomalies were identified because two customers were involved in in providing the requirement, resulting an unclear and ambiguous situation as shown in table 2. The first issue was about change of TAN. Both customers were not satisfied. The second issue was about TAN expiry, where first customer said "yes" and second said "No".

Table 2: Issues Traced During Verification Using Matrix

Requirements	C 1	C 2	RE	Status
Title or username can't be changed during TAN entry.	0	0	1	0
After using TAN entry must expires automatically and shall not be used again.	1	0	1	0

Source: Author's Compilation

As all these requirements are accepted by stakeholder which need to be corrected. Using joint meeting session, the requirements corrected as.

- In first case, the extracted requirement was incomplete so both customers were not agreed about the change during TAN entry, but after joint meeting session both the customers satisfied and agreed.
- In second case, customer 2 was not agreed about the TAN entry, but after negotiation agreed upon the requirement.

The requirement after verification is resolved during joint meeting session as given in table 3. The output of this step is then incorporated according to the standard [1] in the requirements document.

Table 3: Issues Traced During Verification Using Matrix

Requirements	C 1	C 2	RE	Pre-Status	Ri
Title or username can't be changed during TAN entry.	1	1	1	0	1
After using TAN entry must expires	1	1	1	0	1

automatically and shall not be used again.

Source: Author's Compilation

Furthermore, the requirements document before and after the four process approach shown in figure 3 are handed over for 3rd party inspection. Further suggestion included customer feedback, ensuring customer satisfaction.

Table 4: Result of 3rd Party Inspection

Project Name	Req. discovery by parsing (α)	Req. correction using mapping	3rd party inspection	TQS (avg)
Pass/PIN safe before approach	2	2	1	79
Pass/PIN safe after approach	3	2	2	92

Source: Author's Compilation

After implementation, the total quality score of simple inspection was 79 while after the approach was 92 as shown in the given table 4.

4 Result and Conclusion

To improve software quality, testability for system analysis & design is critical. The proposed approach can directly or indirectly effect the software quality. The proposed solution added more requirements, confirming completeness, traces the inaccuracies, incorrectness and ambiguities confirming requirements are corrected.

The result obtained as given in table 4, shows the Average Total Quality Score (TQS) obtained before and after the approach, which was 79 and 92 respectively. This shows significant improvement in the requirements document after being modified according to IEEE STD-830-1998[1] which is a clear indication of testability significance in system analysis & design. It has been concluded that the requirements document must be tested; otherwise, it cannot ever be reliable and may eventually produce a system which may not be according to the expectation of the customer. According to the research, it has been found that testability must be included to the quality factor set that is efficiency, functionality, human engineering, understandability, and usability [5] specifically in system analysis & design to assure the quality of the software/system to be developed.

5 Future Work

Work is required to identify other factors in system analysis & design. Another research is required to find out the relationship of testability with other quality factors in system analysis & design.

References

[1] ANSI/IEEE Standard Glossary of Software Engineering Terminology, IEEE STD-830-1998.

[2] Al-Qutaish, R. E., "Quality Models in Software Engineering Literature: An Analytical and Comparative Study", Journal of American Science, 2010. Vol. 6, no. 3, pp. 166-175.

[3] Ashwin Tomar & V M Thakare, "Identification and Listing of Factors Affecting Software Quality Assurance", International Journal of Internet Computing, 2012.

[4] Basit Habib, "Compilation of Software Quality Factors and Criteria along with their Description for a Quality Product", International Journal of Computer Applications, December 2013.

[5] Basit Habib Rana Aamir Raza Ashfaq, "Relationship between Factors of Quality Models and the System Development Life Cycle", International Journal of Computer Applications, November 2013

[6] Divya Bindal, Jyoti Tamak, "Enhancing Software Quality Using Quality Assurance Practices in the Project Life-Cycle" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 10, October 2013.

[7] Kendall & Kendall, Systems Analysis and Design, 10/E, ©2019 Prentice Hall, ISBN-13: 9780134817361

[8] Manar Abu Talib, Adel Khelifi, Alain Abran, Olga Ormandjieva, "Techniques for Quantitative Analysis of Software Quality throughout the SDLC", The SWEBOK Guide Coverage, conference paper, January 2010

[9] Maneela Tuteja, Gaurav Dubey, "A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models", International Journal of Soft Computing and Engineering (IJSCE), Volume-2, Issue-3, July 2012

[10] MZ Khan, MA Khanam, MH Khan, "Software Testability in Requirement Phase: A Review", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 4, April 2016

[11] PA Laplante, "Requirements Engineering for Software and Systems", eISBN: 9781315303710, https://doi.org/10.1201/9781315303710, 24 October 2017

[12] Pressman, R. S., Software Engineering: A Practitioner's Approach, Eight Edition, McGraw-Hill International Edition, 2015.

[13] SW Ali, QA Ahmed, I Shafi, "Process to Enhance the Quality of Software Requirement Specification Document", International Conference-ieeexplore.ieee.org, 2018.