LC-JSTEM
Your Research Partner

ISSN: 2708-7123 | Volume-**01**, Issue Number-**02** | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

SOFTWARE QUALITY MODELS AND RELIABILITY MEASUREMENT METRICS

Muhammad Shahid
CESAT, Islamabad, Pakistan,
mshahid61@gmail.com

ABSTRACT— "Software reliability is the probability that the software will run without failures for an unlimited time until and unless changed intentionally". It is a crucial characteristics of quality with other essential characteristics. Software reliability measurement is an important concern for any organization to master its software before reaching to end users. A literature review shows that much wok has not been done in this direction. Although achieving software reliability is hard due to software's complex nature but different techniques can be applied to enhance its reliability. Every organization has its own metrics for reliability measurement. There is a need to define metrics according to international standards. In this paper we will analyze software quality and reliability metrics thoroughly.

Keywords— Software reliability measurement, reliability techniques, reliability models, reliability metrics.

I. INTRODUCTION

"Reliability is an external software quality attribute defined by the ISO/IEC 25010:2011" [1]. "Software reliability is the probability that the software system will function properly without failure over a certain time period" [2]. "It is an external quality attribute, which relates internally to the notion of program faults or defects". "Software Reliability is also an important factor that affect system reliability"[1]. It is different from hardware reliability in such a way that it shows software design precision, while hardware reliability covers manufacturing perfection. Hardware reliability have a tendency to be steady or fixed while passes time while software reliability has trend to alter in the testing life cycle.

The failing causes are totally diverse for both hardware and software reliabilities. "Hardware faults arise mostly from wear and physical deterioration, while software faults come mostly from design issues"[10]. They may be incorrect requirements, deviation of code from requirements, sudden modifications in operational control or incorrect changes in the code.

Researchers have developed different models to correlate software reliability and time. But it is obvious that software reliability is not related to time. The modeling techniques for software reliability are now very much matured. However, we should select the technique very carefully before using in any case. "Measurement in software is still in its infancy. No good quantitative methods have been developed to represent software reliability without excessive limitations"[11]. We can use different approaches to enhance the reliability of software. However, it is a difficult task to get software reliability with a balanced cost against the software development time.

The remaining paper is outlined as following. Section II describes software and system quality models. Section III discuss about the software reliability measurement and reliability matrices. Conclusion is mentioned in the final Section IV.

II. SOFTWARE QUALITY MODELS

There are different quality models available in industry. "Software quality models and their metrics may be used in many contexts, i.e. during the development of a new application [3, 4] or when selecting commercial components" [5]. "Software quality is described by a number of attributes which are further classified into two categories, internal and external". Reliability is one of the vital and fundamental quality factor, criteria or characteristic between the others. Farooq et al. [6] shows that "it is the only common factor of different quality models i.e., Bohem's, FURPS, McCall's, ISO 9126 and ISO 25010:2011". Currently, ISO/IEC 9126 [7] is one of the prevailing quality standards used in the world. There are two software quality models according to ISO/IEC 25010 standard.

1. Quality in use model
2. Product quality model

The quality in use model consists of five characteristics, a few of these are further classified into sub-characteristics. "These characteristics relate to the outcome of interaction when a product is used in a particular context of use". This quality model covers the whole computer system that includes in use computer system and software products. Table 1 below represents characteristics and sub-characteristics of this quality in use model.

The product quality model has eight characteristics. All of these are more splitted into different sub characteristics. These characteristics and sub-characteristics explain static and dynamic features of the software and the computer system respectively. Most of the characteristics of this model are also relevant to wider systems and services. Table 2 shows characteristics and sub-characteristics of the product quality model.

Table 1: "Quality in Use Model" by ISO/IEC

Characteristics	Sub-Characteristics
Effectiveness	None
Efficiency	None
Satisfaction	Usefulness, Trust, Pleasure, Comfort
Freedom from Risk	Economic Risk Mitigation, Health and Safety Mitigation, Environmental-Risk Mitigation
Context Coverage	Context Completeness, Flexibility

Table 2: "Product Quality Model" by ISO/IEC

Characteristics	Sub-Characteristics
Functional Suitability	Functional Completeness, Functional Correctness, Functional Appropriateness
Performance Efficiency	Time-Behavior, Resource Utilization, Capacity
Compatibility	Co-existence, Interoperability
Usability	Appropriateness Recognisability, Learnability, Operability, User Error Protection, User Interfaces Aesthetics
Reliability	Maturity, Availability, Fault Tolerance, Recoverability
Maintainability	Modularity, Reusability, Analyzability, Modifiability, Testability
Portability	Adaptability, Installability, Replaceability
Security	Integrity, Confidentiality, Accountability, Authenticity

Both the models can be used by different stake holders like developers, evaluators, end-users, quality assurance and control staff. They also present an outline to facilitate trade-offs between software capabilities.

There are some other tailored quality models by different researchers like Berota Model, GEQUAMO Model, Rawashdeh Model and Alvaro Model. Most of these follow ISO 9126 and ISO 25010 models.

III. SOFTWARE RELIABILITY MEASUREMENT

Software is considered an essential part of everyday usage products. Therefore, a lower degree of software reliability can be profoundly costly for the supplier in the form of displeased customers, loss in market share, extra work and effort to make the necessary changes and corrections in the faulty system.[8].

Software measurement is the introductory set of steps towards better software quality. The multiplicity and complexity of software products, functional processes and working conditions render it impractical to identify a single metric system that can be utilized as a determinant of overall software reliability. A number of characteristics, therefore, have to be measured. For the product itself, attributes such as complexity, size, reusability, data flow, control flow and modularity etc. are crucial whereas the development process consists of productivity, schedule and an estimation of cost and effort required. At the same time, the quality and reliability sector requires measurement of failures, time taken to fail, corrections applied and fault density etc. Last but not least, maintenance and upgrade sector demands for proper documentation. Reliability metrics are acquired from formulae of failure frequency and relevant data.

Typical reliability metrics include:

- ***Probability of Failure on Demand (POFOD)*** refers to the possibility of a transaction request failing.
- ***Rate of Occurrence of Failures*** (***ROCOF)*** refers to the frequency of failures.
- ***Mean Time to Failure*** (***MTTF)*** refers to the average time between successive system breakdowns.
- ***Availability*** refers to the possibility of a system to be working at a given time

We are considering different metrics defined by ISO/IEC. Following three types of metrics are used with ISO/IEC 9126-1, ISO/IEC 25010:2011 model.

- External metrics
- Internal metrics
- Quality in use metrics

The external metrics can be utilized for the measurement of software product quality by evaluating the response of the system to which it belongs. They are only usable during a software's testing and operational stages.

The internal metrics refer to requests for proposal, requirement definition, design criteria or source code. They are applicable only to a non-executable software which is still in its development stages.

The quality-in-use metrics determine whether a product satisfies the requirements of target users to accomplish goals efficiently, securely and appropriately in a specific context of use. Accomplishment of this is only possible in a realistic environment of system.

External Metrics

Following metrics have been defined in this category.

1. Functional Metrics
2. Reliability Metrics
3. Usability Metrics
4. Efficiency Metrics
5. Maintainability Metrics
6. Portability Metrics

We will look into Reliability Metrics to measure software reliability. Every organization can follow metrics defined by ISO or they may modify the metrics according to their systems and environment. Reliability metrics are further composed of following metrics.

Reliability Metrics

1. Maturity metrics
2. Fault tolerance metrics
3. Recoverability metrics
4. Reliability compliance metrics

1. Maturity Metrics

These metrics evaluate the existence of attributes like software failures that are triggered by flaws in the existing software. Following are sub-matrices in this category.

- **Failure Density against Test Cases:**

This matrix calculates the number of failures detected during a specified trial session. Failure intensity is very important for reliability.

- **Failure Resolution:**

This matrix confirms how many failure conditions are resolved.

- **Fault Density:**

Fault density matrix finds out the number of faults that were detected during a specified trial session. One must know difference between fault and failure clearly.

- **Fault Removal:**

This matrix counts the number of faults eliminated whilst testing and compares the sum of faults with the sum of those predicted.

- **Mean Time between Failures (MTBF)**

It calculates the frequency of failures that eventuated during a specified period of operation and sums up the average duration between the failures.

- **Test Coverage:**

What number of essential test scenarios have been concluded whilst testing? This confirms test case coverage. We can set threshold according to our own requirement.

- **Test Maturity:**

Purpose of this matrix is to check whether the product has been well tested.

Similarly, Fault Tolerance matrices pertain to the software's potential of sustaining a specific degree of performance in case of functional flaws.

2. Fault Tolerance Matrices

There are two selected matrix in this category.

- **Software Breakdown & Avoidance:**

How frequently the software trigger disruption in the production environment.

- **Mean Down Time:**

The average time of system unavailability when a malfunction ensues prior to its start.

3. Recoverability Metrics

These metrics evaluate attributes such as whether the software system is able to re-establish its appropriate degree of efficiency and retrieve the data that was directly influenced, for instance during a failure.

- **Mean Recovery Time:**

This metric shows the typical average time that the system takes to recover completely.

- **Restartability:**

Restartability matrix assesses how many times the process can restart offering services to the customers inside a required time.

4. **Reliability Compliance Metrics**

These metrics measure attributes such as the amount of functions that are unable to meet the standard as per the guidelines set for the required compliance.

- **Reliability Compliance:**

It is the capability of the software that shows how it observes regulations, conventions and standards of reliability. This is an important issue. Developers should follow standards strictly to obtain good reliability results.

All the matrices mentioned above to measure software reliability have been taken from ISO/IEC 9126 standard. Any organization may define their own matrix or they can exactly follow above matrices to measure software reliability. One sample matrix, Fault Removal, is shown in Table 3 on next page.

Similarly Internal Matrices and Quality in Use Matrices have further sub matrices to improve the software quality. We can select desired matrices from these categories to fulfill our requirements. As we know, reliability is an attribute of quality so there is a need to measure the attributes of quality to build a high reliable software. Some issues may arise when we use standards introduced by the ISO/IEC 9126. Some of these experiences are shared by Botella et al. [9].

IV. CONCLUSION

This paper focuses primarily on software reliability measurement and matrices. Achieving software reliability is considered an important task of any organization. Measurement of software reliability is difficult resulting from the complexity of the software. Quality of the software might be improved by applying metrics at each and every phase of software development life cycle (SDLC). We recommend to follow reliability matrices defined by ISO/IEC.

V. DISCUSSION

Supporting or contradicting them. Students should also justify their results with evidence-based reasoning and not merely stating the supporting and contradicting studies. Discussion should end with implications or suggestions for future research or

Table 3: Fault Removal Matrix by ISO/IEC

Metric name	Objective	Application Method	Measurement Formula	Explanation	Scale used	Measure type
Fault Removal	It tells about the corrected faults.	"Shows total number of faults that were removed during testing and then compare with the total number of detected faults and total number of predicted faults."	1) X = A1 / A2 2) Y = A1 / A3 Where: A1 = total faults corrected A2 = total number of detected faults A3 = number of predicted faults in the software	0 <= X <= 1 The closer to 1 is better as fewer faults remain. 0 <= Y The closer to 1 is better.	1) Absolute value 2) Absolute value	A1= Count A2= Count A3= Count

REFERENCES

[1] ISO/IEC CD 25010: Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Systems and Software Quality Models, 2011.

[2] Michael R. Lyu, Handbook of Software Reliability Engineering, McGraw-Hill publishing, 1995, ISBN 0-07-039400-8.

[3] Dromey, R.G., "Cornering the Chimera", IEEE Software, 13(1), 1996, pp. 33-43.

[4] Kitchenham, B., Pfleeger, S.L., "Software Quality: The Elusive Target", IEEE Software, 13(1), 1996, pp. 12-21.

[5] Franch, X., Carvallo, J.P., "A Quality-Model-Based Approach for Describing and Evaluating Software Packages", in 10th IEEE International Conference on Requirements Engineering(RE), Germany, September, 2002.

[6] Fsrooq SU, Quadri SMK., Ahmad N, Metrics, Models and measurements in software reliability, IEEE 10th International Symposium on Applied Mchine Intelligence and Informatics (SAMI), Slovakia; 2012, p. 441-449.

[7] International Organization for Standardization, "ISO Standard 9126: Software Engineering – Product Quality, part 1, 2 and 3", Geneve, 2001 (part 1), 2003 (part 2 and 3).

[8] Noman F, Schneidewind, "Reliability Review", The R & M Engineering Journal, Volume 23, No. 2, June 2003.

[9] Botella, P., Burgués, X., Carvallo, J.P., Franch, X., Pastor, J.A., Quer, C., "Towards a Quality Model for the Selection of ERP Systems", in Cechich, A., Piattini, M., Vallecillo, A. (eds.), Component-Based Software Quality: Methods and Techniques, Springer-Verlag, LNCS 2693, 2003, pp. 225-246.

[10] www.ece.uvic.ca

[11] sawaal.ibibo.com

BIOGRAPHY

Dr. Muhammad Shahid is currently working as a Deputy Chief Manager and Director at the Software Quality Assurance Department, Centre of Excellence in Science and Applied Technologies (CESAT), Islamabad. He did his Masters from Punjab University Lahore and PhD in Computer Science from University Technology Malaysia. He is the member of various national and international academic bodies including IEEE and IAENG. He has visited different countries to present his research work.

Dr. Shahid has made many scholarly contributions including peer-reviewed research papers for international conferences and journals. He is also a certified software tester (CTFTL) from International Software Testing Qualifications Board (ISTQB).

As an experienced Testing and Quality professional with a demonstrated history of working in the computer software industry, he is providing the intellectual inputs regarding different projects, supervising software development and testing activities.