Methods to Determine Software Reliability

Kiran Raj Krishnakumar.
Master's in Automation Technology (Mtr No.22213304), FH Westküste, Germany,
Email:kiranrajkrishnakumar@gmail.com

Mahdi Rajabi.
Master's in Automation Technology (Mtr No.22213316), FH Westküste, Germany,
Email:mahdirajabi23@yahoo.com

Martial Domche.
Master's in Automation Technology (Mtr No.22213315), FH Westküste, Germany,
Email:martialdomche@gmail.com

Abstract
Most of the current systems depend on software. Maintaining the quality of the software is very crucial. Reliability is one of the significant features affecting software quality. So, acquiring a high level of software Reliability is crucial regarding user experience and finances. Software Reliability is dynamic and stochastic. It differs from hardware reliability because it reflects design perfection rather than manufacturing embodiment. But quantitively assessing software reliability characteristics is always a challenge. Because software reliability depends on various factors, and no other model covers all these parameters. So, during the software development phase, one must select the most suitable metrics for each development step. The Methods to Determine Software Reliability article overviews the definition of reliability and different software reliability Metrics and models. On the topic of Reliability Metrics article analyses in-depth Fault and failure metrics, and on the subject of Reliability model article tries to cover all the different types of reliability models and deeply explain a few of the essential models.

Keywords: *Reliability, Software reliability metrics, Fault and failure metrics, Software reliability models, Deterministic model, Probabilistic model.*

1. Introduction

Reliability is defined as the probability of success or the probability that the system will perform its intended function under specified design limits (Dr.Pham 2000).The Higher probability of occurrence of software defects in a system, the lesser the reliability of the system. Several other factors affect software reliability, such as System Configuration, Operation characteristics, Product Characteristics, and System Performance. These factors are again considered under different effects, such as in the case of fault analysis. The fault itself is analysed under various parameters such as

- Fault Frequency.
- Kind of Fault.
- Module Interacted with that fault.
- Frequency of Faulty Module Execution
- Effect of Fault Occurrence.

To analyse the software reliability, it is required to examine all factors and consider all aspects associated with each element.

2. Software Reliability Metrics

Reliability metrics help us to quantify the reliability of software. The software can be determined using a wide variety of metrics. One selects software metrics according to the characteristics of the software that is in development. Below are some of the software reliability metrics explained. (Jatain and Mehta 2014)

2.1. *Product metrics*

Product metrics help us to create artifacts such as requirement specification documents, system design documents, etc. It is a code-oriented metric that checks if a product is good enough through attributes like usability, reliability, maintainability, and portability by analysing the source code. The code matrices contain LOC (line of code), KLOC (Thousand Line of Code), Functional Point Metrics, etc. These metrics estimate the software system's respective Number of lines, Number of functions, Number of modules, Number of variables, etc.

2.2. *Project metrics*

Project Metrics are used by the project management team in which it is used to define project characteristics and execution. It includes the analysis of the availability of resources and the effect of software from external factors based on user consideration. There is a correlation between the development process and the project completion time within the desired quality objectives. Higher reliability can be achieved by using a better development process. Project Metrics include the Number of software developers, Staffing pattern over the life cycle of the software, cost and schedule, and productivity.

2.3. *Process metrics*

Process metrics keep a continuous watch on the development process to measure the software reliability and software quality. This process metric aims to do the right job on the first cycle of the process. so, it is indirectly related to the development process. All the factors associated with the development process also come under the process metrics. E.g., The Effort required in the process, Time to produce the product, Effectiveness of defect removal during development, Number of defects found during testing, etc.

2.4. *Fault and failure metrics*

These matrices are associated with the software testing stage, which ensures the fault-free execution of the software.. These metrics are more customer-oriented, where fault analysis data during the software operation is collected using these data to calculate various parameters like Mean time to failure (MTTF)or some other parameter to predict software reliability. The below subsection explains some of these metrics.

2.4.1. *Mean time to failure (MTTF)*

MTTF is a key measure of the mean time to irreparable failure of a product. This calculation is used to understand the typical life span of a system, to determine whether a new version of the product is better than the old one, and to give users or customers information about expected lifetimes and when they should schedule a change, upgrade or maintenance. The MTTF works well when you are trying to assess the average life of products and systems with short lifetimes. It also only works when you are assessing the complete failure of a technology product. Unrepairable system can only break once. MTTF can be calculated in the following ways: (OPYRCHAŁ 2021)
Hazard function h(t) should be estimated using experimental data. Then the reliability function R(t) can be calculated using the Weiner formula (see Eq. (1)).:

$$R(t) = \exp\left(-\int_0^t h(\tau)d\tau\right) \qquad (1)$$

Knowing the reliability function R(t), MTTF can be calculated(see Eq. (2)).

$$MTTF = \int_0^\infty R(t)dt \qquad (2)$$

or

$$MTTF = \frac{Total\ Hours\ of\ Operation}{Total\ Number\ of\ Assets\ in\ use} \qquad (3)$$

2.4.2. *Mean time to repair(MTTR)*

MTTR can be defined as the time taken to complete the repair after the incident of the breakdown. By simplifying all the information, we can gather on MTTR, we can find out which of our equipment takes the longest time to get back into service. MTTR takes the downtime of the system (or assets) and divides it by the number of failures (see Eq. (4)).**:** (Aquitas Solutions 2018)

$$MTTR = \frac{Total\ Downtime}{Number\ of\ Failures} \qquad (4)$$

2.4.3. *Mean time between failure (MTBF)*

MTBF, or Mean Time Between Failures, is a basic measure of system reliability and is represents the average time between two failures for a repairable system. It is usually represented in units of hours. The higher the MTBF, the greater the reliability of the product. MTBF is largely based on assumptions and definition of failure and attention to these details is paramount for correct interpretation. The MTBF is calculated as follows (see Eq. (5)).: (Aquitas Solutions 2018)

$$MTBF = \frac{Total\ Uptime}{Number\ of\ Failures} \qquad (5)$$

Failure Rate (λ) is the inverse of MTBF or how often something fails per hour (see Eq. (6)).:

$$\lambda = \frac{1}{\text{MTBF}} \qquad (6)$$

Equation 1 illustrates the assessment of reliability through MTBFs (see Eq. (7)).

$$Reliability = e^{-\left(\frac{Time}{MTBF}\right)} \qquad (7)$$

2.4.4. *Rate of occurrence of failure (ROCOF)*

For a repairable system, if $N_f(t)$ is the number of failures of a system that have occurred by time t, then the expected number of failures that have occurred at time t is (see Eq. (8)). (Yeh, The Rate of Occurrence of Failures 1997)

$$M_f(t) = E\left(N_f(t)\right) \qquad (8)$$

This derivation is called the rate of occurrence of failures (ROCOF) or the frequency of system failures at time t. Clearly, if $m_f(t)$ increases, the system is deteriorating.

2.4.5. Probability of failure on demand (POFOD)

POFOD is defined as the probability that the system will fail when a service is requested. It is the number of system failures given several systems inputs. POFOD is the likelihood that the system will fail when a service request is made. A POFOD of 0.1 means that one out of a ten service requests may result in failure. POFOD is an important measure for safety critical systems. POFOD is appropriate for protection systems where services are demanded occasionally (Bahl and Kaur 3, May 2014.).

2.4.6. Availability (AVAIL)

Availability helps us to determine if the system is available for a particular time interval. We can find out the total amount of time a system is unavailable by determining the frequency of system outages within the timeframe of calculation, The duration of outages and scheduled uptime.

3. Software Reliability Models

Software reliability models estimate errors in a system by testing the system for a period of time. These errors or failures found during the testing process are solved iteratively until the design objectives are met.There are two main types of software reliability: The deterministic model, The Probabilistic model.

3.1 *The deterministic model*

The deterministic model is used to study the number of distinct operators and operands in a program, the number of errors, and the number of machine instructions in the program. (Dr.Pham 2000)

3.1.1. Halstead's software metric

Halstead (1977) uses the number of distinct operators and the number of distinct operands in a program to develop expressions for the overall program length, volume, and number of remaining defects in a program. (Dr.Pham 2000), The following notations are used: n_1= number of unique or distinct operators appearing in a program; n_2 = number of unique operands.
N_1 = Total number of operators occurring in a program; N_2 = total number of operands; N = length of the program; V = Volume of the program; E = number of errors in the program; I= number of machine instructions.
Halstead showed length and the volume measures of the program can be estimated (see Eq. (9,10)).:

$$\mathrm{N} = \mathrm{N}_1 + \mathrm{N}_2 \tag{9}$$
$$V = Nlog_2\,(n_1 \;+\; n_2) \tag{10}$$

Were

$$N_{1=}n_1 log_2 n_1 \tag{11}$$
$$N_1 = n_2 log_2 n_2 \tag{12}$$

Halstead proposed two empirical formulae to estimate the number of remaining defects in the (see Eq. (13,14)).:

$$E = V/3{,}000 \tag{13}$$
$$E = A/3{,}000 \tag{14}$$

Where

$$A = \left({}^{V}\!\Big/_{{}^{2n_2}\!/_{n_1 N_2}} \right)^{\frac{2}{3}} \tag{15}$$

3.1.2. Cyclomatic complexity

The measurement of cyclomatic complexity by McCabe (1976) was designed to indicate a program's testability and understandability (maintainability) (Kan 2002).It's the number of linearly independent paths comprising the program. As such it can be used to indicate the effort required to test a program. To determine the paths, the program procedure is represented a strongly connected graph with a unique entry and exit point. The general formula to compute cyclomatic complexity is (see Eq. (15)).

$$M = V(G) = e - n + 2p \tag{15}$$

V(G) = cyclomatic number of G; e = number of edges; n= number of nodes; p= number of unconnected parts of the graph.

3.2. *The probabilistic model*

The Probabilistic model Represents the failure occurrences and the fault removals as probabilistic events. (Dr.Pham 2000)

3.2.1. Error seeding models.

The Error seeding group of models estimates the number of errors in a program by using the multistage sampling techniques. Errors are divided into indigenous errors and induced errors (seeded errors). The unknows number of indigenous errors is estimated from the number of induced errors and the ratio of the two types of errors obtained from the debugging data. Models included in this group are :

- Mils' error seeding model.
- Cai's model.
- Hypergeometric distribution model.

3.2.1. Failure rate models

The failure rate group of models is used to study the program failure rate per fault at the failure intervals. Models included in the group are:

- Jelinski and Moranda.
- Schick and Wolveraton.
- Jelinski- Moranda geometric.
- Moranda geometric Poisson.
- Negative-binomial Poisson.
- Modified Schick and Wolverton.
- Goel and Okumoto imperfect debugging.

3.2.1. Reliability growth models

The reliability growth group of models measure and predicts the improvement of reliability programs through the testing process. The growth model represents the reliability or failure rate of a system as a function of time or the number of test cases. Models included in this group are:

- Coutinho model.
- Wall and Ferguson model.

3.2.1. Non- Homogeneous poisson process models (NHPP)

The non-homogenous Poisson Process (NHPP) group of models provides an analytical framework for describing the software failure phenomenon during testing. The main issue in the NHPP model is to estimate the mean value function of the cumulative number of failures experienced up to a certain point in time. Models include in this group are:

- Musa exponential.
- Goel and Okumoto.
- S- shaped growth.
- Hyper exponential growth.

3.2.1. Curve fitting models

The curve fitting group of models uses statistical regression analysis to study the relationship between software complexity and the number of faults in a program, the number of changes, or failure rate. This group of models finds a functional relationship between dependent and independent variables by using the methods of linear regression, nonlinear regression, or time series analysis.

- Estimation of errors.
- Estimation of complexity.
- Estimation of failure rate.

3.2.1. Markov structure models

A Markov Process has the property that the future behavior of the process depends only on the current state and is independent of its history. The Markov structure group of models is a general way of representing the failure process of software. This group of models can also be used to study the reliability and interrelationship of the modules. Models included in this group are:

- Markov model with imperfect debugging.
- Littlewood Markov.
- Software safety.

4. Conclusion

Software reliability plays a significant role in defining the software limitations and analysis. This article focused on overviewing software reliability metrics and models. Since software reliability depends on various factors, no one model can precisely predict the reliability of a system. so, we can conclude that during the software development phase, one must select the most suitable metrics for each development step. Through sufficient testing and proper maintenance, software reliability can be improved.

5. Reference

Jatain, Aman, and Yukti Mehta. 2014. "Metrics and Models for Software Reliability: A Systematic Review." *IEEE* 6.

Dr.Pham, Hong. 2000. *Software Reliability.* New Jersey: Springer-Verlag Singapore Pte.Ltd.

OPYRCHAŁ, Leszek. 2021. "IS MEAN TIME TO FAILURE (MTTF) EQUAL TO." *Journal of KONBiN* 255-256.

Kan, Stephen H. 2002. *Metrics and Models in Software Quality Engineering.* Boston: Addison-Wesley.

Aquitas Solutions. 2018. "Mean Time Between Failur & Mean Time To Repair." 6.

Yeh, Lam. 1997. "The Rate of Occurrence of Failures." 14.

Bahl, Kailash, and Gurpreet Kaur. 3, May 2014. "ISSN 2348 - 7968 Software Reliability, Metrics, Reliability Improvement Using Agile Process." *International Journal of Innovative Science, Engineering & Technology.*