Software Reliability Model

Drishti Sompura
Software Engineering
Shrinathji Institute of Technology & Engineering
Nathdwara, India
Sompurad89@gmail.com

Pankaj Dalal
Software Engineering
Shrinathji Institute of Technology & Engineering
Nathdwara, India
pkjdalal@gmail.com

Abstract: The software quality contains a set of different properties. One of the most important is represent by the reliability. A straight approach of the reliability supposes its orientation rather to the user than to the software development process. This approach derives from the users' point of view, which determinates an easier understanding of the reliability by the clients. Also, refers more at the execution than to the design, which makes it more dynamic than statically. One advantage is the fact that reliability is being calculated for the both components: hardware, software, and the entire system. This Paper describes the basic of software reliability and the software reliability Models used to calculate reliability of software.

Keywords— software reliability ; reliability model ; reliability metrics.

I INTRODUCTION

The means of software reliability is that the probability of failure-free operation of a computer program in a specified environment for a specified time.
The expected curve for software is given in Figure 1

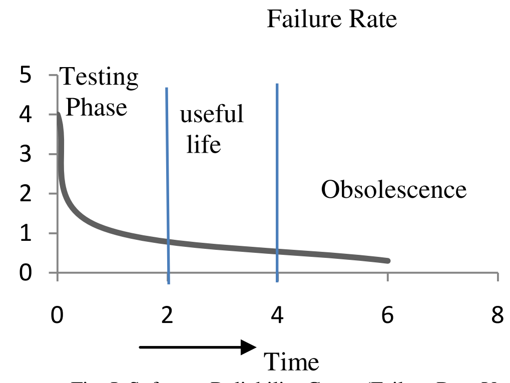


Fig: I Software Reliability Curve (Failure Rate Versus Time)

A . Software-Reliability Specifications

Reliability is always considered at the system level rather than the individual component level. Because the components of a system are interdependent i. e. a failure in one system component can be affect the operation of other components. In a computer-based system, the software reliability is discussed through three dimensions which are:

1. Hardware Reliability. What is the probability of a hardware component failing and how long does it take to repair that component?

2. Software Reliability. How likely is it that a software component will produce an wrong outcome? Software failures are different from hardware failures in that software doesn't wear out. This can continue operating correctly after an incorrect result has been produced.

3. Operator Reliability. How likely is it that the operator of a system will make an error?

The reliability of a system depends upon a number of factors rather than the
total of all the probabilities (failure probabilities of each factor).
PS = PA + PR + ... + PN,
where
PS = Probability of system failure
PA = Probability of component A failure
PB = Probability of component B failure
PN = Probability of component N failure
As the number of factors increase, the overall probability of system failure increases.

B. Reliability Terminologies

Table :I Reliability Terminology

Term	Description
System failure	An event which occur at some point in time when the system does not deliver a service as expected
System error	When the system behavior does not conform to its specification
Syetem Fault	An incorrect System State
Human error or mistakes	Human behavior that results the fault in system

1. Classification of Failures

When a system specifies the reliability then the engineer should identify the failure type and consider whether it should be treated differently in specification. Many large systems are divided into small subsystems and each subsystem has different reliability requirements. It is easy and cheap to assess the reliability requirements of each subsystem separately.[3]

Table:II Failure Classification

Failure Class	Description
Recoverable	The System can recover with or without operator intervention
Transient	Occurs only with certain inputs
Unrecoverable	The system cannot recover without operator intervention or the system may need to be restarted
Corrupting	Failure corrupts system data
Non-corrupting	Failure does not corrupt system data
Permanent	Occurs for all input values while involving a function of the system.

C. Reliability Metrics

Reliability metrics are used to quantitatively express the reliability of a software product. Some reliability metrics, which can be used to quantify the reliability of a software product are:

1. MTTF (Mean Time to Failure). Components have an average lifetime and this is reflected in the most widely used hardware reliability metric, Mean Time to Failure (MTTF). The MTTF is the mean time for which a component is expected to be operational. The MTTF is the average time between observed system failures. An MTTF of 500 means that one failure can be expected every 500 time units. The time units are totally dependent on the system and it can even be specified in the number of transactions. MTTF is relevant for systems with long transactions, where the system processing takes a long time. It should be longer than transaction length.

2. MTTR (Mean Time to Repair). Once failure occurs, some time is required to fix the error. MTTR measures the average time that it takes to track the errors that leads to failure and then to fix them.

3. MTBF (Mean Time Between Failures). We can combine the MTTF and MTTR metrics to get the MTBF metric:

MTBF = MTTF + MTTR

Thus, a MTBF of 300 hours indicates that once a failure occurs, the next failure is expected to occur only after 300 hours. In this case, the time measurements are real time and not the execution time as in MTTF.

4. POFOD (Probability of Failure on Demand). POFOD is the likelihood that the system will fail when a service request is made. A POFOD of 1/1000 means that one out of a thousand service requests may result in failure. POFOD is an important measure for safety critical systems and should be kept as low as possible.

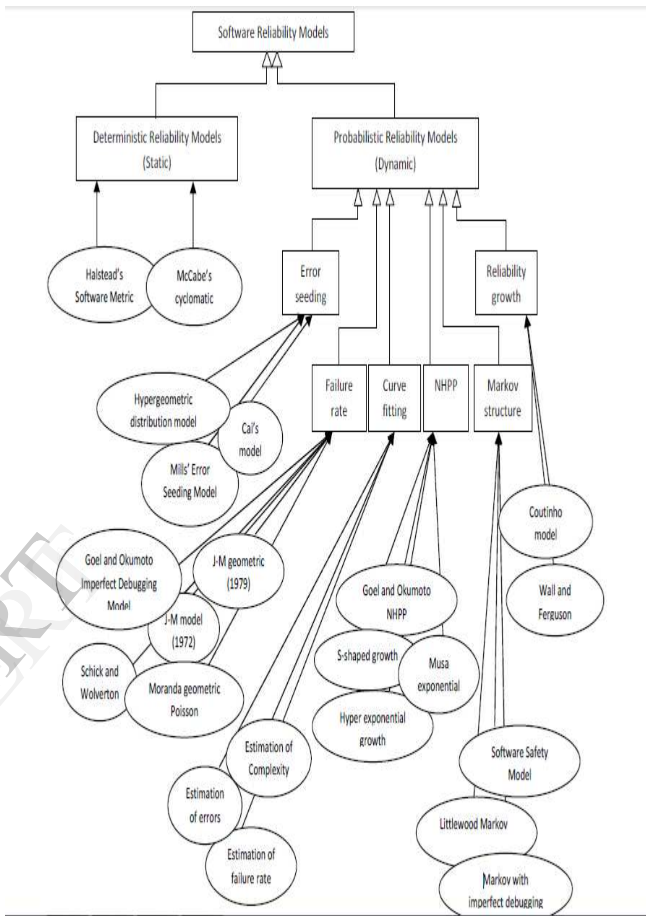
5. ROCOF (Rate of Occurrences of Failure). ROCOF is the frequency of occurrence with which unexpected behavior is likely to occur. A ROCOF of 2/100 means, two failures are likely to occur in each 100 operational time units. This is called the failure intensity. It is relevant for operating systems and transaction-processing systems where the system has to process a large number of similar requests that are relatively frequent; for example, credit-card processing systems, airline booking systems, etc.

6. AVAIL (Availability). Availability is the probability that the system is available for use at a given time. An availability of 0.998 means that in every 1000 time units, the system would be available for 998 of these.

II SOFTWARE RELIABILITY MODEL

The taxonomy shows different types of reliability models from previous works in the field of software reliability.

Fig:II Taxonomy of software reliability models



A. Deterministic Reliability Model vs. Probabilistic Reliability Model

In deterministic reliability models, we will discuss about two popular models: Halstead's Software Metric and McCabe's cycloramic complexity, and also probabilistic reliability models, which are categorized into six different branches: Error seeding, Failure rate, Curve Fitting, Reliability growth and Markov structure. Each of these software reliability models have some methods to estimate software reliability and other performance measures such as software complexity, software safety, and the number of remaining errors[9],[10].

B. Deterministic reliability models

Halstead's theory is probably the best-known technique to measure the complexity in a software program and the amount of difficulty involved in testing and debugging the software.

1. Halstead's Software Metric

This model uses the number of distinct operators and the number of distinct operand in a program to develop expressions for the overall volume , program length and the remaining no. of defects in a program.[8][9][10]

2. *McCabe's Cyclomatic Complexity Metric*

McCabe's cyclomatic complexity metric is software metric that provides a quantitative measure of the logical complexity of a program by counting the decision point. Example: one should start with 1 for the straight path through the module or subroutine. Then each time it should be added to one of the following keywords of selection and iteration. Also, 1 should be added for each case in a case statement lacks a default case. By using the McCabe's measure, the code is considered to be of high quality software. McCabe has suggested that a program with a high level of the metric is very difficult to produce and maintain. He recommended a total score of 10 as an upper limit for the FORTRAN environment. [11]

C. *Probabilistic reliability models*

These reliability models are deterministic ones which use some static parameters of program, apply some statistical methods to estimate measures such as the number of errors in a program, failure rate, relationship between software complexity and the number of faults in a program, and so on.[8]

1. *Error Seeding Models*

Error seeding is a technique, which can be used to evaluate the amount of residual errors after the system software test phase. The error seeding group of models estimates the number of errors in a program by using the multistage sampling technique. Multistage sampling is a complex form of cluster sampling. Models included in this group are:

- Mills' error seeding model; [13]
- Cai's model; and [14]
- Hyper geometric distribution model [16]

a. Mills' error seeding model: This model is one of the error seeding methods which estimate the number of errors in a program by introducing seeded errors into the program. In this method some of known error types are inserted into the program, and the program is executed with the test cases under test conditions. The ratio of found seeded errors to the total number of seeded errors is approximate equal to the ratio of found real errors to total number of errors.[15]

b. Cai's Model: Cai modified Mill's model by dividing the software into two parts: Part0 and Part1. There are following assumptions for this model:

- The software can be divided into two parts: Part0 and Part1.
- There are N defects remaining in the software, where Part0 contains N0 and Part1 N1 remaining defects. That is N= N0 + N1.
- At any time, that is, no matter how many remaining defects are contained in the software, each of the remaining defects has the same probability of being detected.
- Upon being detected, the defect is removed.
- Each time, one only one and remaining defect is removed and no new defects are introduced.
- There are n remaining defects removed.

Sometimes these assumptions are not reasonable and applicable for real software.

2. *Failure Rate Model*

Based on classification scheme for models described by Singpurwalla and Wilson in 1994, most of the probabilistic software reliability models fall into one of two broad categories .I: Modeling the times between successive failure of the software which is broken down into I-1. Those which use the failure rate as a modeling tool I-2. Those which model an inter-failure time as a function of previous inter-failure times II: Modeling the number of failures of the software up to a given time.

This group of models is used to study the program failure rate per fault at the failure intervals.

- Jelinski-Moranda Model – 1972. [18]
- Jelinski-Moranda Geometric Model – 1979. [19]
- Geol and Okumoto imperfect debugging Model – 1979. [20]

*a. Jelinski-Moranda Model (1972):*The Jelinski-Moranda (1972) model is a basic model of type I-1.As the software runs for longer and longer, more bugs are caught and purged from the system, and so the error rate declines and the future reliability increases.One basic assumption in the Jelinski-Moranda model is that faults are perfectly repaired when encountered. Normally however, any valid software fault must pass a regression test before it is categorized as resolved, and in practice regression testing uncovers a significant number of imperfect fixes and in some cases new faults. [2][3]

*b. Jelinski-Moranda Geometric Model (1979):*The J-M geometric model (1979) assumes that the program failure rate function is initially a constant and decreases geometrically at failure times.

The Jelinski-Moranda and J-M Geometric models failed the consistency test which they proposed. They challenged these models to take data which comes from a process which they have correctly modeled and to make predictions about the reliability of that process. The either model, given data precisely from a process it correctly models, will usually fail to make good predictions. They attribute these problems to randomness in the data used as input to the models and indicate the remedy for this lack of robustness, namely replication of data.[17]

c. Geol and Okumoto imperfect debugging Model: Goel and Okumoto in 1979 extended the J-M model by assuming that a fault is removed with probability whenever a failure occurs.

3. *Curve Fitting Models*

This group of models uses statistical regression analysis to investigate the relationship between program complexity and the number of faults, the failure rate and also the number of change. These models use linear regression, non-linear regression or time series analysis to find a functional relationship between dependent and independent variables.Models which are presented in this group are:

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

a. Estimation of errors Model: Using regression models we can estimate number of errors in a program. By which calculate the software reliability.

But most of the Curve fitting models involve only one error factor.[8]

b. *Estimation of complexity Model:* This model is proposed as a model for estimation of the software complexity. This model uses time series approach to estimate software complexity.

This model is usable for some software which have various release versions and evolving in long period of time.

c. *Estimation of Failure Rate Model*: This model tries to estimate the failure rate of a software product using previous failure times. This model assumes that the failure rate is monotonically non-increasing and can be earned using the least squared method.

4. *Non-Homogenous Poisson Process Models (NHPP)*

These models are based on non-homogeneous Poisson process statistical model. A non-homogeneous Poisson process is a Poisson process with rate parameter such that the rate parameter of the process is a function of time.

The major issue in these models is to determine a proper mean value function to designate the expected number of defects happened up to certain point in time. There are different models in this group which are:

- Goel and Okumoto NHPP (Goel, 1979)
- Musa exponential (Musa, 1985)
- S-shaped growth models (Ohba, 1984)

a. *Goel and Okumoto NHPP (Goel, 1979):* Non Homogeneous Poisson process models first are proposed by Goel and Okumoto in 1979 which has formed the basis for the failure count models. The idea is that the number of observed faults at time t is nonlinear because the failure intensity decreases when faults are found and corrected. In general there are different types of NHPP process. Some of these categorizations are I–NHPP exponential model II– NHPP S-shaped model III– NHPP imperfect debugging model and IV-NHPP Sshaped imperfect debugging model.

b. *Musa exponential (Musa, 1985)*: Musa in 1985 presented a model which is similar to Goel-Okumoto model. In this model m(t) is the number of failure discovered as a result of test case runs up to the time of observation.

c. *S-shaped growth models:* The NHPP S-shaped model is based on the following assumptions:

i. The error detection rate differs among failure.

ii. Each time a program failure occurs, the software error which caused it is immediately removed, So new errors can't be introduced.

5. *Markov Structure Models*

A Markov model has a special property that the future behavior of the process depends only on the current state and it is independent of its past history. This type of models is a general way of representing the failure process of software. In these models it is supposed that failures of the modules are independent of each other. This assumption seems reasonable at the module level, because they can be designed, implemented, and tested independently, but may not be true at the system level.

6. *Reliability Growth Models*

Reliability growth models estimates the improvement of reliability measure through the testing process. These models represent the failure rate of a system as a function of time or the number of test cases.

Two famous models in this group are:

- Coutinho model (Coutinho, 1973)
- Wall and Ferguson model (Wall, 1977)[17]

a. Coutinho model: This model plots the cumulative number of failures discovered in the program and the number of correction actions made versus the cumulative testing weeks on log papers.[17]

b. Wall and Ferguson model: This model is presented in 1977 and it is similar to Weinbull growth model. This model also predicts the failure rate of software during the software process.Wall and Ferguson tested this model using several software failure data and observed that failure data correlate well with the model.[17]

III CONCLUSION

In this paper we have discuse about the basics of software reliability, reliability metrics and different types of software reliability model. Which help for the deep study of software reliability modeling. The future work can be to study a company project and calculate software reliability factor for that project.

References

[1]. Philip J. Bolan ,"Challenges in Software Reliability and Testing".
[2]. Singpurwalla, N. D. and Wilson, S. P, "Software reliability modeling. International Statistical Review".
(1994), 289-317.
[3]. Israel Koren, C. Mani Krishna, "Fault Tolerant Systems", March 2007, Published by Morgan Kaufmann.
[4]. GlennBooker ,"Evaluation of Information Systems Reliability Models and Customer Satisfaction",users.snip.net/~gbooker/INFO630/lect6.ppt
[5]. A. Wood, Tandem Tech, "Software Reliability Growth Models", Report 96-1, Tandem Computers Inc., Corporate Information Center, Cupertino, Calif., 1996
[6]. Stephen H. Kan, "Metrics and Models in Software Quality Engineering", Published by Addison-Wesley, 2003.
[7]. "Overview of Software Reliability", http://swassurance.gsfc.nasa.gov/disciplines/reliability/index.php#o
[8]. Hoang Pham,"Software Reliability", Springer-Verlag Singapore 2000.
[9]. McCabe, T.J., "A Complexity Measure", IEEE transaction, Software Engineering, Vol. SE-2(4), 1976.
[10]. Halstead, M.H., Elsevier, "Elements of Software Science", New York, 1977.
[11]. Fitzsimmons, A. and Love, T., "A review and evaluation of software science", ACM Computing Surveys,Vol. 10(1), March 1978.
[12]. "Multistage sampling", http://en.wikipedia.org/wiki/Multistage_sampling.
[13]. D. Mills, "On the statistical validation of computer programs", IBM FSD, Report FSC-72-6015, 1972.
[14]. Cai, K.Y. ,"On estimating the number of defects remaining in software", Journal of Systems and Software,Volume 40 , Issue 2, Pages: 93 - 114 ,Year of Publication: 1998
[15]. Yoshihiro Tohma, et al., "The Estimation of Parameters of the Hypergeometric Distribution and its Application to the Software Reliability Growth Model", IEEE Transactions on Software Engineering, Volume 17, Issue 5,Pages: 483 – 489, 1991.
[16]. Wendy W. Peng, Dolores R. Wallace, "Software Error Analysis", Published by Silicon Press, 1995.
[17]. Wetiliui Slim, Larry Wilsoti, "Software Reliability Growth Models Dominated By Randomness", Report No.NASA CR-181889, September 1989.
[18]. Jelinski, Z. and P. B. Moranda, "Software Reliability Research", in Statistical Computer Performance Evaluation, ed. W. Freiberger, Academic Press, New York. 1972