# A SURVEY ON SOFTWARE RELIABILITY AND MEASUREMENT TECHNIQUES

[1]Mekhala Sridevi Sameera, [2]Dr. T. Anuradha, [3]Dr. R. Satya Prasad
[1]Research Scholar, Dept of CSE, Acharya Nagarjuna University, Guntur
[2]Professor & Head, Dept of CS&E, JKC College, Guntur
[3]Professor & Head, Dept of CSE, Acharya Nagarjuna University, Guntur

***Abstract:*** Software reliability is a unique feature of reliability engineering. Software System reliability, by means of characterization, comprises all components of the system. The software reliability consist hardware, software, sustaining infrastructure, operators and measures. Software reliability is an important concept in software quality assurance. The process of software reliability can be classified into 3 major components: modeling, measurement and enhancement. Software reliability modeling has expanded to the position that significant outcomes can be acquired by placing appropriate prototypes to the problem. We have various numerous models be present to measure software reliability, but no particular model can confine an important magnitude of the software characteristics. The problem can be simplified and ease using hypothesis and concepts. This paper presents summary of Software Reliability measurement and improvement techniques and also observe various kinds of improvement strategies for software reliability, nevertheless, we have no particular prototype that is common to all the circumstances. We also studied various methods for the evaluation of software reliability like fuzzy method, neuro-fuzzy method, artificial neural network method, genetic algorithm method, Bayesian classification method, support vector machine (SVM) method, Self-organizing map method.

***IndexTerms*** **- Software reliability; software quality; reliability measurement; Hypothesis;**

## I. INTRODUCTION

Software does not become old, tarnish, exhaust, collapse or chink. In addition software has no form, color, fabric, and gathering. Software is unable to be glimpsed or fingered, however it has a substantial reality as well as is important to system working behavior. Public consider that previously after the software can execute properly, it will be accurate perpetually. But this is demonstrated to be incorrect by the sequence of disasters rooted by software. Software has ability to take decisions, but not reliable as similar to human beings.

In composite software applications, the significant feature of software quality assurance is reliability, but achieving reliability is complex. Unreliability of any invention arrives because of the collapse or existence of errors in the system produced. Because software does not "age", just like any other engineering products such a mechanical or an electronic system, the untrustworthiness of software is primarily due to defects and design errors in the software. In most cases the central issue that affects system reliability is Software reliability. It is different from hardware reliability as it replicates the design excellence, not mechanized excellence. We can define reliability as a probabilistic quantify that predicts the incidence of collapse of reliability. Randomness indicates that the collapse cannot be guessed exactly.

We can often define software reliability as the probability of failure free operation of a software program for a particular time in a precise environment. Various sort of methods can be applied to progress the software reliability, nevertheless, it is tough to equilibrium software reliability with time for developing system and budget. Software Reliability is significant to characteristic of software quality assurance process, collectively with characters like performance, capability, functionality, usability, maintainability, serviceability, installability and documentation. Software Reliability is tough to realize, since convolution of software lean to be high-ceilinged. Though any product with a high degree of convolution, together with software, will be tough to arrive at a convinced stage of reliability, software developers lean to press on convolution into the software layer, amid the speedy expansion of application size and effortlessness of doing so by raising software.

## II. MEASURING SOFTWARE RELIABILITY

Let us now present various important factors and key issues while measuring the reliability of a software product.

1. Classification of Failures
   Since reliability is apprehensive through the occurrence of diverse categories of failures, it is required to understand comprehensible and definite categorization of failures. The failure categorization method must be broad, complete and must authorize a distinctive categorization of each failure. This failure categorization will be from the users' viewpoint, as developers attempting to incarcerate the user knowledge of reliability. A lot of failure categorizations have been projected in literature or text [2] but at present there exist modest classifications of failures. One categorization was anticipated by Cristian that categorized failures as three parameters timing, omission, and response [4]. In case of contemporary software application, it is proposed that failures be screened at the highest level as unexpected events, scheduled events and configuration faults. This failure categorization provides a structure for identifying failures.

The following is sample of Failure Classification.

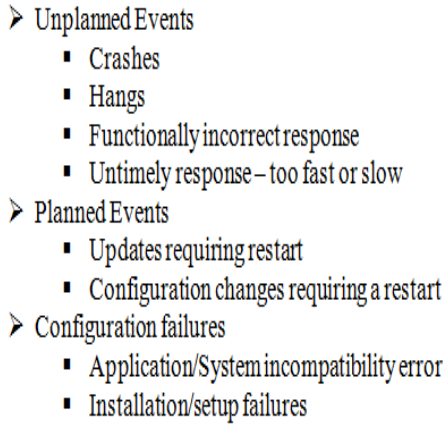


figure 1: a sample of failure classification

2. Population Size

The population size is the important information one require to set-up reliability in other words, Number of entities of the product used in process. It is not simple that receiving precise information concerning the definite population of entities in use. In addition, with the complete customer population support to achieve reliability, it will need acquiring failure information from this support that will be a lot difficult for a large range sponsored product. It is identified in our study that for finding reliability, we need a random sample, called the experiential group, of the population size. With this acknowledged experiential group, fault data will be observed simply for this cluster of users. Usual statistical methods can be employed to find the sample size such that the final result is precise to the quantity preferred.

3. Usage Time

In case of a precise calculation of reliability, any tangible practice time of the application by the customer desires are identified to compute the fault rates. Because of handiness, it is frequently completely understood that the application is used, on an average, for the identical quantity of time each day by different number of users. By means of identified postulation, one can find day count and it is used for finding reliability. On the other hand, the practice period for dissimilar customers may diverge significantly for large sponsored products. Since the faults faced by a customer obviously based on the quantity of usage of the application – the longer the usage period the extra the probability of facing failures to acquire an exact plan of the reliability of the application in use, we need to capture the usage time. Utilizing usage time as an alternative to number of days of rights for reliability calculation, it is analogous to the schedule time and CPU time dialogue in reliability growth models [17]. For these, it is extensively assumed that using CPU time presents improved reliability approximation.

## III. SOFTWARE RELIABILITY METRICS

Software Reliability Measurement is not a precise discipline, though the search of enumerating software reliability has not stopped. So far, we have no better way of measuring software reliability. Measuring software reliability exist as a tricky problem since we don't have a better perception of the character of software. There is no obvious meaning to what features are connected to software reliability. We cannot discover a appropriate way to measure software reliability. The present practices of software reliability measurement can be separated into four groups:

- Product metrics

  Software size is considered to be insightful complexity, development effort and reliability. Lines of Code (LOC), otherwise LOC in thousands (KLOC), is an sensitive early approach to measuring software size. However there is not a customary way of counting. Usually, source code is used as SLOC or KSLOC and comments and other non-executable statements are not reckoned. This process cannot authentically evaluate software not written in the same language. The beginning of new technologies of code reuses and code generation technique also shed uncertainty on this easy method. Function point metric is a technique of measuring the functionality of planned software depends on a no of inputs, outputs, files and interfaces. The technique can be exercised to guess the size of a software system as soon as these functions can be recognized. It is a measure of the functional complexity of the program.

- Project management metrics

  Software developers know that efficient quality management can answer in improved products. Study and Investigations has confirmed the fact that there exists association involving the product development process and the competence to complete projects on time and inside the preferred eminence goals. Expenditure amplifies when the product developers employ insufficient methods. Privileged reliability can be obtained with the help of improved application development method, risk management method, configuration management method.

- Process metrics

  Based on the facts that postulation size and the quality of the application is a direct function of the process, the metrics used to measure process can be used to evaluate, observe and progress the software product efficiency and reliability.

- Fault and failure metrics

The purpose of gathering metrics related to bugs, defects and failures is to fix on when the application is impending defect free execution. Simply this is equal to the total quantity of faults identified throughout testing by clients following release are gathered. Then they are aggregated and explored to attain this objective. Typically, metrics related to defects are established with client information about bugs found after discharge of the software. The data related to defects are gathered and consequently used to compute failure density, Mean Time between Failures..

- Efficiency

The sum of calculated time and resources needed by software to carry out desired task it is a significant issue in distinguishing high quality software from a low one.

- Integrity

The degree to which admission to application or data by illicit people can be restricted. Integrity has become significant in the age of hackers.

- Flexibility

The exertion important to shift the program from one kind of hardware resource to another one.

- Interoperability

The exertion needed to combine one application to other as specified by the characteristics: insatiability, conformance, adaptability, replacebility.

- Maintainability

It is defined as effortlessness through which fix can be applied to the application as specified by features: changeability, stability, analyzability, testability. If application requires very less mean time to change, it indicates requires less maintainability.

## IV. SOFTWARE RELIABILITY IMPROVEMENT TECHNIQUES

Developing high reliability software depends on the relevance of quality attributes at each stage of software development life cycle. These quality characteristics are prominence on fault prevention, removal and measuring methods. Metrics are needed at every software process model phases to measure applicable quality attributes. Best and Modern Engineering techniques, methods and tools largely improve software reliability.

Following are the best techniques used to improve software reliability.

- Verification and Validation
- Trigger, locate and remove software defect
- Measure and Analyze
- Trending reliability
- Predictive reliability

Trending reliability: - Search the fault information created by the software application. It divides into four types.

1. Error Seeding: Estimate the quantity of faults in a program and divide them in indigenous (original) and induced (simulated)
2. Failure Rate: rate of failure per fault is the failure period
3. Curve Fitting: identifying the association between complexity of software and #No of faults in a program
4. Reliability Growth: Measure and forecast the advancement of reliability applications all the way through the testing process

Predictive reliability: Assign probabilities to the operational profile Software reliability improvement methods handle the existence and demonstration of failures in application. It is divided into three categories

1. Failure avoidance and failure prevention
2. Failure removal
3. Failure acceptance that guesses a application has inevitable and untraceable faults and aims to make provisions for the system to operate correctly, even in the presence of faults

These activities can be exercised to reduce the option of finding incidence of bug subsequent to its release and as a result get better software reliability.

## V. SOFTWARE RELIABILITY GROWTH MODELS

Software reliability models are statistical models which can be used to make predictions about a software system's failure rate, given the failure history of the system. The models make assumptions about the fault discovery and removal process. These assumptions determine the form of the model and the meaning of the model's parameters.

In contrast to Rayleigh, these models the defect pattern of the entire development process, reliability growth models are usually based on data from the formal testing phases. In practice, such models are applied during the final testing phase when the development is virtually completed. The rationale is that defect arrival or failure patterns during such testing are good indicators of the product's reliability when it is used by customers. During such post-development testing, when failures occur and defects are identified and fixed, the software becomes more stable, and reliability grows over time. Hence models that address such a process are called reliability growth models.

There are two types of models:

- Those that predict times between failures, and
- Those that predict the number of failures.

1. Time between failures models
   Models that predict times between failures can be expressed as a probability density function, fi (t) whose parameters are estimated based on the values of previously observed times between failures t1, t2,…, ti-1. This probability density function is used to predict the time to the next failure as well as the reliability of the software system.
2. Failure Count Models
   For models that predict the number of failures in future test intervals, we also have a probability density function fi(t). The parameters of fi(t) are computed based on the failure counts in the previous (i-1) test intervals. Suppose that we've observed failure counts in test intervals f1, f2, …, fi, and we want to predict what the number of failures will be in interval i+1.

**REFERENCES**

[1] Garima Chawla et. al. , A Fault Analysis based Model for Software Reliability Estimation, International Journal of Recent Technology and Engineering (IJRTE ),ISSN: 2277 -3878, Volume-2, Issue-3, July 2013.

[2] Rita G. Al gargoor et. al. , Software Reliability Prediction Using Artificial Techniques, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2, July 2013,ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784,www.IJCSI.org.

[3] Dr. R Satya Prasad ,K Ramchand H Rao and Dr. R.R. L Kantham (2011),” Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC” IJCA Journal, Number 7 – Article1

[4] R.satyaprasad, Half Logistic Software Reliability Growth Model,Ph.D. Thesis,2007

[5] Mamta Arora et. al. , Software Reliability Prediction Using Neural Network, International Journal of Software and Web Sciences 5(2), June-August, 2013, pp.88-92.

[6] Inoue S., and Yamada S ,Two-Dimensional Software Reliability Measurement Technologies, IEEE , 2009, pp. 223 – 227.

[7] Quadri S. M., Ahmad N. and Farooq S. U. ,Software Reliability Growth Modeling With Generalized Exponential Testing – Effort And Optimal Software Release Policy , Global Journal of Computer Science and Technology , 2011, pp.27 – 42.

[8] Prof. C. J. van Duijn et. al. ,Statistical Procedures for Certification of Software Systems, © Corro Ramos, Isaac 2009.

[9] Latha Shanmugam et. al., A Comparison Of Parameter Best Estimation Method For Software Reliability Models, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.5, September 2012.

[10] Mohd. Anjum et. al. , Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value, I.J. Information Technology and Computer Science, 2013, 02, 1-14.

[11] Al-Rahamneh Z., Reyalat M. Sheta A. F., Bani-Ahmad S., and Al-Oqeili S., A New Software Reliability Growth Model: Genetic-Programming-Based Approach , Journal of Software Engineering and Applications, 2011, pp. 476-481.

[12] Aasia Quyoum, Mehraj – Ud - Din Dar, M. K. Quadri Di. “Improving Software Reliability using Software Engineering Approach- A Review”, International Journal of Computer Applications (0975 – 8887)Volume 10– No.5, November 2010.

[13] Goutam Kumar Saha, “Software Reliability Issues: Concept Map”, IEEE Reliability Society 2009 Annual Technology Report.

[14] E Eduardo Valido-Cabrera, “Software reliability methods”, Technical University of Madrid August, 2006.

[15] E. Valido-Cabrera, "Software Reliability Methods," Technical Report, Technical University of Madrid, 2006.

[16] Yang Gu, “Adopting ODC to improve Software Quality: A case study.” Technical report, IBM, August, 2006.