Computing, Information Systems, Development Informatics & Allied Research Vol. 4 No. 4 December, 2013

APPROACH FOR ENHANCING THE RELIABILITY OF SOFTWARE

Raghvendra Kumar
L.N.C.T. Group of College
Jabalpur, M.P., INDIA
Raghvendraagrawal7@gmail.com

ABSTRACT
Reliability is always important in all systems but sometimes it is more important than other quality attributes, especially in mission critical systems where the severity of consequence resulting from failure is very high. Software reliability engineering is focused on comprehensive techniques for developing reliable software and for proper assessment and improvement of reliability. Reliability metrics, models and measurements form an essential part of software reliability engineering process. Appropriate metrics, models and measurement techniques should be applied to produce reliable software. Hence, it is the intention to develop some approaches to enhance the reliability of software by the analysis of the structure of the software, execution scenario for various inputs and operational profile.

Keywords: Software Reliability, Rate of occurrence of failure (ROCOF), Mean Time to Failure (MTTF), Hardware Reliability, Mean Time between Failure (MTBR).

INTRODUCTION

1.1 Software Reliability is a field of computer engineering whose approach is focused on comprehensive techniques for developing reliable software and for proper assessment and improvement of the reliability. Reliability metrics, models, and measurements form an essential part of software engineering process. So we should apply appropriate metrics, models, and measurement techniques in software reliability engineering to produce reliable software. Reliability of a software product essentially denotes its trustworthiness or dependability. It is known that a software product having a large number of defects is unreliable. Hence the reliability of a system improves, if the number of defect is reduced. According to IEEE "Software reliability is the probability of a failure free operation of a computer program for a specified period of time in a specified environment".

Reliability is a user-oriented factor relating to quality of software. Intuitively, if the users of the system rarely experience failure it is considered to be more reliable than that one that fails more often. A system without fault is considered to be highly reliable. Software reliability is an important aspect of software quality. Software reliability concerns itself with how well the software functions to meet the requirements of user. In June 4 1996, the Airane 5 Flight 501 veered off the flight path, broke off and exploded resulting in a financial loss of the cargo and rocket of $500 million. Although the software that was used in Airane 5 was same as in Airane 4. In Airane 4 it was working perfectly fine but it resulted in failure in Airane 5. Disintegration of Airane 5 rocket 37 seconds after launch is perhaps commonly referred to as one of the most expensive software bugs in history
.

1.2 Software Reliability and Hardware Reliability:-The software system reliability is classified into software reliability and hardware reliability as the software and hardware behave differently in any system. Hardware part may fail due to its design and manufacturing defects. Sometimes also a perfectly functioning hardware device also fails despite of its well treatment. But software will not change over time unless the software is changed or modified intentionally. Hardware reliability is the probability that the hardware perform its function for specified period of time without any failure. For example, if the resistor get short circuited we can either replace or repair the fault part. For hardware products, it can be observed that failure rate is high initially but decreases as the faulty components are identified and removed. The system then enters its useful life. After some time (called product life time) the components wear out, and the failure rate increases. So by repairing the hardware part its reliability is maintained at the level it existed before the failure occurred. Software Reliability can be increased by applying metrics at different stages of software development life cycle. For software the failure rate is at its highest during integration and test. As the system is tested, more and more errors are identified and removed resulting in reduced failure rate.

1.3 Techniques to Improve Software Reliability A fault is a defect in a program that arises when programmer makes an error and causes failure when executed under particular conditions. Reliability can be increased by preventing the errors and developing quality software through all of the stages of software life cycle. To do this, there are also metrics which helps in improving reliability of the system by identifying the areas of requirements (*for specification),* Design (for verification and validation), Coding (*for errors),* Testing (*for verifying)* phases. **a)**

1.4 Requirements Reliability Metrics:-Requirements indicate what features the software must contain. So for this requirement document, a clear understanding between client and developer should exist. Otherwise it is critical to write these requirements. The requirements must contain valid structure to avoid the loss of valuable information. Next, the requirements should be thorough and in a detailed manner so that it is easy for the design phase.

The requirements should not contain inadequate information. Next one is to communicate easily .There should not be any ambiguous data in the requirements. If there is ambiguous data, then it is difficult for the developer to implement that specification. Requirement Reliability metrics evaluates the above said quality factors of the requirement document. Software Metrics used in Requirement Phase are:

1. Function Points
   - Count number of inputs and output, user interactions, external interfaces, files used.
   - assess each for complexity and multiply by a weighting factor.
   - Used to predict size or cost and to access project productivity.

2. Number of requirements errors found (to assess quality)
   Change request frequency
   - To access stability of requirements.
   - Frequency should decrease over time. If not, requirements analysis may not have been done properly.

b) Design and Code Reliability Metrics The quality factors that exist in design and coding phase are complexity, size and modularity. If there are more complex modules, then it is difficult to understand and there is a high probability of occurring errors. So complexity of the modules should be less. Next coming to size, it depends upon the factors such as total lines, comments, executable statements etc. The reliability will decrease if modules have a combination of high complexity and large size or high complexity and small size. In the later combination also the reliability decreases because, the smaller size results in a short code which is difficult to alter. These metrics are also applicable to object oriented code, but in this, additional metrics are required to evaluate the quality.

Software Metrics used in Coding Phase are:

1. Common measures
   - Lines of code written per programmer per month.
   - Object instructions produced per programmer month.
   - Pages of documentation written per programmer month.
   - Test cases written and executed per programmer month.

Software Metrics used in Design Phase are 1. Number of Parameters
   - Tries to capture coupling between modules.
   - Understanding modules with large number of parameters will require more time and effort (assumption).
   - Modifying modules with large number of parameters likely to have side effects on other modules.
   - Number of modules
   - Number of modules called (estimating complexity of maintenance).
   - Fan-in: number of modules that call a particular module. Fan-out: how many other modules it calls.
   - High fan-in means many modules depend on this module. High fan-out means module depends on many other modules.
   - Makes understanding harder and maintenance more time-consuming.

3. Data Bindings
 Triplet (p, x, q) where p and q are modules and X is variable within scope of both p and q

– Potential data binding
X declared in both, but does not check to see if accessed. Reflects possibility that p and q might communicate through the shared variable.

– Used data binding:
A potential data binding where p and q use X. Harder to compute than potential data binding and requires more information about internal logic of module.

– Actual data binding:
Used data binding where p assigns value to x and q references it. Hardest to compute but indicates information flow from p to q. 4. Cohesion metric
 Construct flow graph for module.
   - Each vertex is an executable statement.
   - For each node, record variables referenced in statement.
   - Determine how many independent paths of the module go through different statements.
   - If a module has high cohesion, most of variables will be used by statements in most paths.
   - Highest cohesion is when all the independent paths use all the variables in the module.

c) Testing Reliability Metrics Testing Reliability metrics uses two approaches to evaluate the reliability. First, it ensures that the system is fully equipped with the functions that are specified in the requirements. Because of this, the errors due to the lack of functionality decreases. Second approach is nothing but evaluating the code, finding the errors and fixing them.

1.5 Software Reliability Metrics There are some reliability metrics which can be used to quantify the reliability of software products.

Rate of occurrence of failure (ROCOF) ROCOF measures the frequency of occurrence of unexpected behavior (i.e. failures). ROCOF measure of a software product can be obtained by observing the behavior of a software product in operation over a specified time interval and then recording the total number of failures occurring during the interval.

Mean Time To Failure (MTTF) MTTF is the average time between two successive failures, observed over a large number of failures. To measure MTTF, we can record the failure data for n failures. Let the failures occur at the time instants *t1*, *t2*, …, *tn*. Then, MTTF can be calculated as It is important to note that only run time is considered in the time measurements, i.e. the time for which the system is down to fix the error, the boot time, etc are not taken into account in the time measurements and the clock is stopped at these times.

Mean Time to Repair (MTTR) Once failure occurs, some time is required to fix the error. MTTR measures the average time it takes to track the errors causing the failure and to fix them. **Mean Time between Failure (MTBR)** MTTF and MTTR can be combined to get the MTBR metric: MTBF = MTTF + MTTR. Thus, MTBF of 300 hours indicates that once a failure occurs, the next failure is expected after 300 hours. In this case, time measurements are real time and not the execution time as in MTTF. **Probability of Failure on Demand (POFOD)** Unlike the other metrics discussed, this metric does not explicitly involve time measurements. POFOD measures the likelihood of the system failing when a service request is made.

For example, a POFOD of 0.001 would mean that 1 out of every 1000 service requests would result in a failure.

Availability Availability of a system is a measure of how likely shall the system be available for use over a given period of time. This metric not only considers the number of failures occurring during a time interval, but also takes into account the repair time (down time) of a system when a failure occurs. This metric is important for systems such as telecommunication systems, and operating systems, which are supposed to be never down and where repair and restart time are significant and loss of service during that time is important.

Efficiency The amount of computing time and resources required by software to perform desired function it is an important factor in differentiating high quality software from a low one.

Integrity The extent to which access to software or data by unauthorized persons can be controlled Integrity has become important in the age of hackers.

Flexibility The effort required to transfer the program from one hardware to another.

Interoperability The effort required to couple one system to another as indicated by the following sub-features: adaptability, insatiability, conformance, replacebility.

Maintainability It is the ease with which repair may be made to the software as indicated by the following sub-feature: analyzability, changeability, stability, testability. If a software needs" less mean time to change (MTTC), it means it needs less maintainability.

2. LITERATURE SURVEY

The survey of various papers concerned with the enhancement of software reliability is summarized as follows.

Technique1
In this era of computer, computer are playing very important role in our daily lives. Mechanical objects now-a-days are replaced by digital devices, CPUs' and software. Increasing competition and high development costs have intensified the pressure to quantify software quality and to measure and control the level of quality delivered. There are various software quality factors as defined by MC CALL and ISO 9126 standard, however software reliability is most important and most measurable aspect of software quality. This paper tries to give general idea for software reliability metrics and model used. This will focus on software engineering principles in the software development and maintenance so that reliability of software will be improved. This paper provides an overview of improving software reliability and provides various ways to improve software reliability in the life cycle of software development.

Reliability is a probabilistic measure that assumes that the occurrence of failure of software is a random phenomenon. Randomness means that the failure can't be predicted accurately. The randomness of the failure occurrence is necessary for reliability modeling. In [MIO87], it is suggested that reliability modeling should be applied to systems larger than 5000 LOC.

Reliability Process in generic terms is a model of the reliability-oriented aspects of software development, operations and maintenance. The set of life cycle activities and artifacts, together with their attributes and interrelationships that are related to reliability comprise the reliability process.

Software Reliability Activities are grouped into classes: *Construction* Generates new documentation and code artifacts *Combination* Integrates reusable documentation and code components with new documentation and code components. *Correction* Analyzes and removes defects in documentation and code using static analysis of artifacts. *Preparation* Generates test plans and test cases, and readies them for execution. *Testing* Executes test cases, whereupon failure occur *Identification* Makes fault category assignment. Each fault may be new or previously encountered. *Repair* Removes faults and possibly introduces new faults. *Validation* Performs inspections and checks to affirm that repairs are effective *Retest* Executes test cases to verify whether specified repairs are complete if not, the defective repair is marked for repair. New test cases may be needed.

Software Reliability Metrics: In this paper, software reliability measurement is divided into 4 categories: [RAC96]

Product Metrics Function point metric is a method of measuring the functionality of a proposed software development based upon a count of inputs, outputs, master files, inquires, and interfaces. It measures the functionality delivered to the user and is independent of the programming language. It is used primarily for business systems; it is not proven in scientific or real-time applications. Complexity is directly related to software reliability, so representing complexity is important. Complexity-oriented metrics is a method of determining the complexity of a program's control structure, by simplifying the code into a graphical representation. Representative metric is McCabe's Cyclomatic Complexity Metric McCabe's Cyclomatic complexity is a software quality metric that quantifies the complexity of a software program. Complexity is inferred by measuring the number of linearly independent paths through the program.

The higher the number the more complex the code. Test coverage metrics are a way of estimating fault and reliability by performing tests on software products, based on the assumption that software reliability is a function of the portion of Software that has been successfully verified or tested.

Project Management Metrics
It is known that good management can result in better products. It has been demonstrated that a relationship exists between the development process and the ability to complete projects on time and within the desired quality objectives. Costs increase when developers use inadequate processes. Higher reliability can be achieved by using better development process, risk management process, configuration management process, etc.

Process Metrics
Based on the assumption that the quality of the product is a direct function of the process, process metrics can be used to estimate, monitor and improve the reliability and quality of software. ISO- 9000 certification, or "quality management standards", is the generic reference for a family of standards developed by the International Standards Organization (ISO).

Fault and Failure Metrics
The goal of collecting fault and failure metrics is to be able to determine when the software is approaching failure-free execution. Test strategy is highly relative to the effectiveness of fault metrics, because if the testing scenario does not cover the full functionality of the software, the software may pass all tests and yet be prone to failure once delivered. Usually, failure metrics are based upon customer information regarding failures found after release of the software. The failure data collected is therefore used to calculate failure density, Mean Time between Failures (MTBF) or other parameters to measure or predict software reliability. Besides the above metrics, other possible metrics are: *Efficiency* The amount of computing time and resources required by software to perform desired function it is an important factor in differentiating high quality software from a low one. *Integrity* The extent to which access to software or data by unauthorized persons can be controlled Integrity has become important in the age of hackers. *Flexibility* The effort required to transfer the program from one hardware to another. *Interoperability* The effort required to couple one system to another as indicated by the following sub-features: adaptability, insatiability, conformance, replacebility. *Maintainability* It is the ease with which repair may be made to the software as indicated by the following sub-feature: analyzability, changeability, stability, testability. If a software needs" less mean time to change (MTTC), it means it needs less maintainability. **Software Reliability Improvement Techniques:** In real situations, it is not possible to eliminate all the bugs in the software; however, by applying sound software engineering principles software reliability can be improved to a great extent. The layer technology in software engineering focuses on reliability and quality of software.

In this paper, study of software reliability can be categorized into three parts: Modeling, Measurement & improvement. Software reliability measurement is naive. It can't be directly measured, so other related factors are measured to estimate software reliability. Software reliability improvement is necessary & hard to achieve. It can be improved by sufficient understanding of software reliability, characteristics of software & sound software design. Complete testing of the software is not possible; however sufficient testing & proper Maintenance will improve software reliability to great extent.

Technique2
Software reliability relies on 3 basic models such as Usage modes describe how software is used. Trend model describe how reliability evolve our times as certain bugs are fixed or new bugs are introduced. Probabilistic failure models capture the fact that failures may happen randomly. **Software Reliability Metrics**: Reliability metrics are derived from failure occurrence expressions and data. According to this paper, reliability metrics is categorized into 4 types. They are Probabilistic of failure on demand (POFOD): Its main specification is for the systems where service requests happen in an unpredictable way or when there is a long time interval between the requests. Rate of occurrence of failure (ROCOF): It is specified for system where services are demand in more regularly. Mean time to failure (MTTF): It is specifically for the systems involving long transactions, during which a guarantee of service continuity and delivery should be expected. Availability (AVAIL): It is used for system where continues service delivery is a major concern. **Software Reliability Measurement**: Reliability measurement is divided into 4 categories: Product Metrics, Project Management Metrics, Process Metrics, Fault and Failure Metrics. Software Process and Product Metrics are quantitative measures that enable people to gain insight into the efficiency of software process and the project that are conducted using the process as a framework. Higher reliability can be achieved by using better development process, risk management process, configuration management process etc. Fault metrics are used to find defects and fix those defects. Failure metrics are based upon customer information regarding failures found after release of software. The failure data collection is therefore used to calculate failure density. Mean Time Between Failure(MTBF) or other parameters are used to measure or predict software reliability.

Software Improvement Techniques: Software fault should be carefully handled to make software more reliable with as many reliable improvement techniques as possible. Software reliability improvement techniques are divided into 3 categories. Fault avoidance/prevention that includes design methodologies to make software probably fault-free. Fault removal that aims to remove faults after development stage is completed. Fault tolerance that assumes a system has unavoidable and undetectable faults and aims to make provisions for the systems to operate correctly, even in the presence of faults.

Till now many models have been developed but how to quantify software reliability still remains largely unsolved. No single model can be used in all situations. One model may work well for a set of certain software but May completely off track for other kind of problems. Good metrics should be used to develop model that are capable of predicting process and product parameters. Ideal metric should be simple, objective, easily obtainable, valid and robust. From this paper it can be concluded that the scope of software metrics can be expanded to include performance evaluation and software measurement.

Technique3

Reliability is always important in all systems but sometimes it is more important than other quality attributes. Software reliability engineering approach is focused on comprehensive techniques for developing reliable software and for proper assessment and improvement of the reliability. Reliability metrics, models and measurements form an essential part of software reliability engineering process. We should apply appropriate metrics, models and measurement techniques in SRE to produce reliable software, as no metric or model can be used in all situations. So, we should have profound knowledge of metrics, models and measurement process before applying them in SRE. In this paper, the author represents in-depth analysis of all metrics, models and measurements used in software reliability. According to this paper, reliability is the probability of a system or component to perform its required functions (output that agrees with specifications) without failure under stated conditions (operational profile) for a specified period of time. Informally reliability denotes a product's trustworthiness or dependability.

Mathematically, reliability R(t) is the probability that a system will be successful in the interval from time 0 to time t. i.e. $\mathbf{R(t) = P(T > t), t \geq 0}$ Where T is a random variable denoting the time to failure or failure time. Unreliability F(t), a measure of failure, is defined as the probability that the system will fail by time t" i.e. $\mathbf{F(t) = P(T \leq t), t \geq 0.}$ In other words, F (t) is the failure distribution function. The following relationship applies to reliability in general. The Reliability R (t) is related to failure probability F (t) by: $\mathbf{R(t) = 1 - F(t).}$ This probability is, however, the object of interest mainly when the probability of failure-free survival of system/mission is of concern. Other measures are more appropriate in other situations, and include various reliability metrics like MTTF, ROCOF, and POFOD which are discussed in this paper. **Key concepts in Reliability** It is imperative to define key elements of reliability before discussing finer details of software reliability. The key elements of the reliability are as follows: 1. Probability of failure-free operation. 2. Duration of time of failure-free operation. 3. A given execution environment. Their work is mainly focused on software reliability so key elements of software reliability are mainly defined in the context. In this paper software reliability metrics is categorized into five types

. **1. MTTF (Mean Time to Failure)** The MTTF is the mean time for which a component is expected to be operational. MTTF is the average time between two successive failures, observed over a large number of failures. To measure MTTF, we can record the failure data for n failures. Let the failures occur at the time instants t1, t2… tn. Then, MTTF can be calculated as It is important to note that only run time is considered in the time measurements, i.e. the time for which the system is down to fix the error, the boot time, etc are not taken into account in the time measurements. 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, as is the case of database query systems. MTTF is relevant for systems with long transactions, i.e., where system processing takes a long time. We expect MTTF to be longer than average transaction length.

2. MTTR (Mean Time to Repair) MTTR is a factor expressing the mean active corrective maintenance time required to restore an item to an expected performance level. This includes activities like troubleshooting, dismantling, replacement, restoration, functional testing, but shall not include waiting times for resources. In software, MTTR (Mean time to Repair) measures the average time it takes to track the errors causing the failure and then to fix them. Informally it also measures the down time of a particular system.

3. MTBF (Mean Time between Failures) MTTF and MTTR can be combined to get the MTBF metric: MTBF = MTTF + MTTR. In this case, time measurements are real time and not the execution time as in MTTF. Thus, MTBF of 300 hours indicates that once a failure occurs, the next failure is expected after 300 hours

4. POFOD (Probability of Failure on Demand) POFOD measures the likelihood of the system failing when a service request is made. Unlike the other metrics discussed, this metric does not explicitly involve time measurements. A POFOD of 0.005 means that five out of a thousand service requests may result in failure. POFOD is an important measure and should be kept as low as possible. It is appropriate for systems demanding services at unpredictable or relatively long time intervals. Reliability for these systems would mean the likelihood the system will fail when a service request is made.

5. ROCOF (Rate of Occurrences of Failure) ROCOF measures the frequency of occurrence of unexpected behavior (i.e. failures). It is measured by observing the behavior of a software product in operation over a specified time interval and then recording the total number of failures occurring during the interval. It is relevant for systems for which services are executed under regular demands and where the focus is on the correct delivery of service like operating systems and transaction processing systems. Reliability of such systems represents the frequency of occurrence with which unexpected behavior is likely to occur. A ROCOF of 5/100 means that five failures are likely to occur in each 100 operational time units. This metric is sometimes called the failure intensity. **Conclusion** In this paper they presented an in-depth study of software reliability metrics, models and Software reliability measurement.

We observed that these three things are essential part of software reliability engineering process and can yield excellent results if right metrics and models are selected during software measurement process. The key to success with SRE is to acquaint ourselves well with details of basic elements of SRE i.e. software reliability metrics, models and measurement process before using them in practice.

Technique4

Assessment of software reliability is an area of the utmost importance for software-based systems employed in safety-critical applications such as computer relaying of power transmission lines. Today it is hard to think of any area in modern society in which computer systems do not play a dominant role In space and air navigation defense telecommunication and healthcare to name a few computers have taken over the most life critical tasks Unlike most human beings computers seem to do their job perfectly at all times and under all conditions But do they really well most of the time they do Sometimes however a zillion dollar satellite goes of course rocket misses its target or a large telephone exchange gives up Possible sources for such dissatisfactory behavior are physical deterioration or design faults in hardware components In a emblematic software development process, software is deliberate as a combination of apparatuses and modules instead of a gigantic block. Different components contribute to erratic degrees to the overall operative of software and thus knowledge of component prominence can reveal vital information for software designers and test engineers. Moreover, software reliability is estimated on the basis of collected antique data followed by an assumed distribution curve and is thus inherently vulnerable to uncertainties. The fault tree is an important aid that is used extensively for safety and reliability consideration of component-based systems and can also be applied to software. In this paper it is stated that Software Reliability is an imperative to trait of software quality, together with functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation. Software Reliability is rigid to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, will be inflexible to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the rapid growth of system size and affluence of doing so by upgrading the software.

3. CONCLUSION

Software reliability depth is ingenuous. Dimension is far from communal place in software, as in other engineering field. "How good is the software, quantitatively?" Software consistency cannot be directly measured, so other related factors are measured to estimate software reliability and equate it among products.

4. FUTURE WORK

The proposed approach will be applied to some software and results will be obtained. Further the evaluation of proposed approach will be made.

REFERENCES

[1] Improving Software Reliability Using Software Engineering Approach-A Review Aasia Quyuom(Research Scholar, University of Kashmir, India), Mehraj-Ud – Din Dar(Director, IT & SS, University of Kashmir, India), S. M. K. Quadri(Director Computer Sciences, University Of Kashmir, India)

[2] Software Reliability-An Overview by E.Sridevi,B.Aruna,P.Sowjanya(Department of Freshman Engineering,KL University,Andhra Pradesh,India) IJCST Vol.3,Issue 1,Jan-March 2012

[3] Metrics, Models and Measurements in Software Reliability- Sheikh Umar Farooq (Research Scholar, P.G Department of Computer Sciences, University of Kashmir, Srinagar, India), SMK Quadri (Director, P.G Department of Computer Sciences, University of Kashmir, Srinagar, India)and Nesar Ahmad(Department of Statistics and Computer Applications, T. M. Bhagalpur University, Bhagalpur, India) 10th IEEE Jubilee International Symposium on Applied Machine Intelligence and Informatics • January 26-28, 2012

[4] A Critical Review of Software Reliability Tariq Hussain Sheakh(Lecturer in computer sciences at Govt. Degree College Poonch, J&K,INDIA) Dr S.M.K.Quadri(Director, P. G. Department of Computer Sciences, University of Kashmir, Srinagar-190006 Jammu and Kashmir) Vijay Pal Singh (Assistant Professor in Computer Science at JJT University Rajasthan) (International Journal of Emerging Technology and Advanced Engineering Volume 2, Issue 4, April 2012)

[5] "IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software", 1998

[6] [L. Rosenberg, T. Hammer, J. Shaw, "Software Metrics and Reliability",http://satc.gsfc.nasa.gov/support/ISSRE_NOV98/software_metrics_and_reliability.html Last Date Accessed: 25.08.2005