International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Software Reliability Engineering: An Overview

Pallavi Priya[1], Shubhra Gautam Sharma[2]

[1]Amity Institute of Information Technology, Amity University, Sec-125, Noida-201303, India

[2] Amity Institute of Information Technology, Amity University, Sec-125, Noida-201303, India

Abstract: ***Software reliability is based on the methods of engineering which involves the development and maintenance of the software systems whose reliability is measurable. Software reliability have been a major subject of research over last many years, still researches are going on. Software reliability is considered a major factor for software quality. It is a system dependability concept. System dependency is increasing day by day due to which Software reliability has become a major concern of users. SRE can be defined as the study of the processes and outcomes of a software system which is the basic requirement of all the users. This paper gives an overview of characters of Software reliability, Software reliability activities, metrics, modeling, improvement techniques.***

Keywords: Reliability, Metrics, Model, Software Quality, Engineering

1. Introduction

Software Reliability Engineering plays a vital role in day to day life. It can be defined as the behavior of error free operation of any software in a specified environment for limited time span. Software Reliability Engineering helps improving the quality of software in real life systems. As the system dependability active Researches have been done in the field of Software Reliability Engineering over the past few years, open questions still exist as the complexity of software systems have grown with size, much than is expected and will continue in the future. Because of the increasing system dependency, failures of software may lead to serious consequences in day to day business. There are a number of weaknesses in the existing SRE methods. Firstly, the data that have failed are combined during the later phases of testing such as the phases after unit and module testing may be too late for doing any changes in the initial design. Also, the data that have failed are collected during the in house testing is not enough, and generally fails to cover the actual operational environment. As software testing is considered one of the important parts of improving the software reliability, these demerits can become serious when it comes to delivering a quality product. This main focus of this paper will be to improve the Software Reliability by using engineering techniques for the software design and development

2. Software Reliability

Product Unreliability is basically found due to the presence of system's error. Software does not get old, the software unreliability is primarily due to the fault in the design or errors in the software. Reliability is considered the most important characteristics inherent in the concept "software quality". Software reliability's concern is that how well the software functions to meet the requirements of the customer. The life cycle of software includes many test items such as documents, manuals, reports, plans, code configuration data and test data which help in measuring its reliability."Figure-1" shows the relationship between Failure rate and Time as shown below:

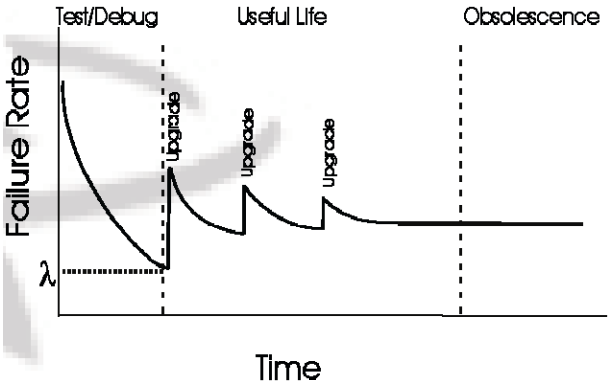

Figure 1: Time Vs Failure Rate Graph

3. Characters of Software Reliability

3.1 Failure occurs primarily due to design faults:

Design is modified for repairs to make it robust against conditions that can detect a failure.

3.2 There is no wear-out phenomenon:

1) Software errors occur without any warning.
2) While doing upgrades, "Old" code can result in more number of failure rate because of errors.
3) External environment conditions generally not affect the reliability of the software.
4) Internal environment related conditions, such as inappropriate clock speeds or insufficient memory affect software reliability.

3.3 Reliability is not time dependent

1) Failure happens due to the error prone execution
2) The growth of the reliability is observed as errors are detected and corrected.

4. Software Reliability Activities

The software reliability process includes software development, operations, and maintenance. A software reliability process includes faults, defects, corrections,

errors, updating, and expenses on the resource, such as manpower effort. Some of the Reliability activities are as follows:

4.1 Construction: Generation of new documentation and code artifacts
4.2 Combination: It emphasis on reusability of old documents and code components with the new one.
4.3 Correction: Analyzing and removing document and code related defects by analyzing the test items.
4.4 Preparation: Generating of different test items.
4.5 Testing: Execution of the test cases, to know the trigger points where failure occurs frequently.
4.6 Identification: Each error or bug whether new or previously encountered, is categorized.
4.7 Repair: Faults are removed which possibly introduces new faults for which regression testing is done.
4.8 Validation: Perform checks to make sure that repairs are effective and have not affected other parts of the software.
4.9 Retest: Execution of the cases to check for specified repair's completion. If it is incomplete, new test cases may be needed to repair them further.

5. Software Reliability Metrics

Measurement of Software reliability is an unsolved issue because we have not enough understanding of the software's nature. We use some of the metrics to measure the software reliability. They are categorized as:

5.1 Project management metrics

A good project can be achieved by maintaining good management which results in completion of projects and with the objectives of the quality. If developers have not enough processes, increase in cost occurs. Improved and efficient reliability is targeted by improving procedure management, strategy management, development processes, risk management process etc.

5.2 Product metrics

LOC (i.e. Lines of Code), or KLOC (i.e. LOC in thousands), is a way to measure the size of the software. Typically, comments and the statements which are non-executable are not counted and source code is used. Function point metric measures the functionality of a proposed software development on the basis of count of inputs, outputs and interfaces. It helps to measures the basic functionalities of the software. Complexity-oriented metrics determines the program structure's complexity, by the code simplification into the graphical representation as complexity is related to the reliability of the software, so complexity representation is important.

5.3 Fault and failure metrics

Fault and failure metrics helps to achieve the software's failure free execution. A lot of errors found at the testing phase before delivering it to the customer and the number of faults encountered and told by the users after software delivery are combined, analyzed and summarized to complete this main goal. Failure metrics are based upon information gathered from the users on the failures found after the software's release. Therefore the collected failure data is used to calculate the density of the failure and many related parameters to measure or predict software reliability.

5.4 Process metrics

Process metrics is used for estimation, monitoring and improvement of the quality and reliability of software

5.5 Efficiency

The computing time required and amount of resources for the software to perform required function is a vital factor in differentiation of low and a high quality software.

5.6 Integrity

The software access by an unauthenticated person or a hacker can be controlled by improving the Integrity techniques.

5.7 Flexibility

The ability of software to be compatible with different hardware defines its flexibility.

5.8 Maintainability

The software requires time to time maintenance and its cost is also very high.

6. Software Reliability Modeling

A number of models have been proposed, still there is no model that can be used for all the software. No model can give the exact result; one model works well for some of the software, but may not be suited for other problem type. Most of the existing methods for obtaining software system's reliability measures are related to the models of Markovian and they depend upon the assumption of distribution of the failure time. These models explain clearly about the problem of large state space. On the other hand, the model of simulation provides an efficient alternative to analytical models because it defines a system characterization for its interdependency, interrelations in such a way that rather than on the system itself, one may perform experiments on the model.

7. Software Reliability Improvement Techniques

7.1 Methods of Software Engineering

Methods of SE provide an overall view of how to build software. It covers the technical aspects as well. Various tasks come under these methods which are as follows:

1. Requirement analysis
2. Design modeling
3. Program construction
4. Testing

"Figure-2" shows the basic steps for software development as shown below:

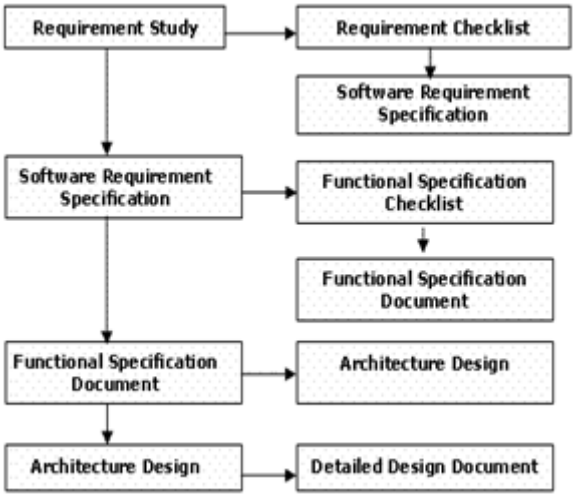

Figure 2: Software Development Process

7.1.1 Requirement Analysis

Only coding and testing is not enough for achieving reliability, Requirement analysis is considered the important phase which can reduce the rate of the failure and hence the improved reliability can be achieved by:

a) Requirement identification
b) Preparing software requirement specification (SRS) document that describe the overall external behavior of system proposed
c) Prototype designing
d) Creating Data flow diagrams (DFDs) to explain the basic and overall flow of the software
e) Task, cost, effort and resource estimation
f) Risk management and control.

7.1.2 Modeling Design

In this phase, system's model is developed which may further be used to develop the real system. Here, the reliability is achieved by:

a) Dividing the system into small and recognizable modules which can be independently developed and handled
b) Interrelating each module and understanding the inter-dependency between them

7.1.3 Program Construction

Coding and testing is combined in Program Construction. Here, reliability of the software can be made better by:

a) Development of structured programming
b) Interface creation that are consistent with architecture,
c) Performing unit testing

7.1.4 Testing

Software testing is basically performed to detect and remove the defects. However it is being assumed that "no software can be 100% bug free", testing minimizes the bug to the maximum. First of all the system is divided into modules and is tested independently. Functionality of each module is checked and then all the modules are combined together and is tested called integration testing. Validation testing is also done which validates the expected requirements of the customer for the software which has been developed. At last, system testing provides the components of the software testing together as one unit.

8. Reliability Simulation Process

In the field of software reliability, simulation can explain the key features of the processes that validate the documents and code. The software test items such as errors in the programs such as fault removal and failure generally violate the expected behavior of software reliability models, Simulation processes can be considered a good way to understand these assumptions and it might result in a better understanding of why some of the models work well irrespective of such violations. "Figure-3" explains the steps included in the simulation process study illustrated by the flowchart as shown below:

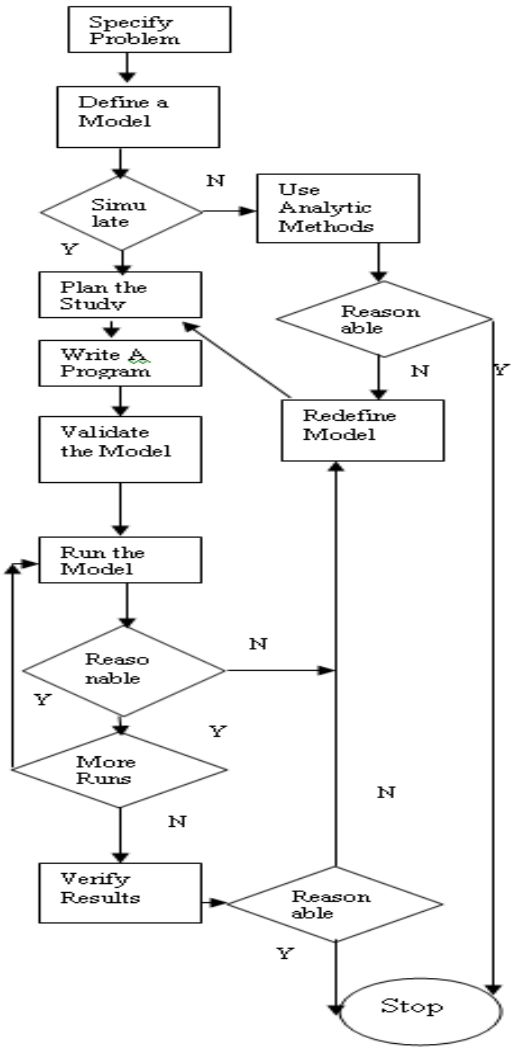

Figure 3: Process of Simulating

9. Conclusion

Reliability of the software is considered as the most efficient aspect for the quality of software. Hardware can age or rust, but software do not. Software's unreliable behavior is basically due to errors or faults in the design of the developed software. Many of the Models exist, but not single model can define the expected or required behavior of the software under different environment. None of the model has been universally accepted for all types of software. The

main aim is to build and deliver reliable software to the customer. If the resources are not enough to test until the reliability goals are met, we will at least know where we are and what should be the next step. This paper provides an overview of SRE noting the key factors of each topic.

References

[1] Aasia Quyoum, Mehraj–Ud - Din Dar, S. M. K. Quadri, "Improving Software Reliability using Software Engineering Approach" Volume-5 Number-3
[2] Ellen Walker, "Applying Software Reliability Engineering to build Reliable software"
[3] John D. Musa**, "**More Reliable Software Faster and Cheaper: An Overview of Software Reliability Engineering"
[4] John D Musa, "Software Reliability Engineering in Industry"
[5] KaramaKanoun, "Software Reliability Engineering"
[6] Michael R. Lyu, "Software Reliability Engineering: A Roadmap"
[7] N K Srinivasan, "Reliability- How to Quantify and Improve? Improving the health of products"

Author Profile

Pallavi Priya received the BCA degree in Computer Science and is pursuing MCA from Amity Institute of Information Technology, Amity University, Sector – 125, Noida-201303(UP), India