DEVELOP THE SOFTWARE RELIABILITY USING SOFTWARE RELIABILITY IMPROVING TECHNIQUES

Puli Nageswara Rao¹, D.Jyothirmai², Dr. K. Subba Rao³

¹Asst.Professor, SRKIT-Vijayawada-India, ²Asst.Professor, BVRIT-Narsapur-Medak-India, ³Professor, BVRIT-Narsapur-Medak-India

ABSTRACT

In the Software Engineering, The Software Reliability is an important facet of software quality. Software reliability is the probability of the failure free operation of a computer program for a specified period of time in a specified environment. Software Reliability is dynamic and stochastic. It differs from the hardware reliability in that it reflects design perfection, rather than manufacturing perfection. This Paper talks about the development of Software Reliability using improvement techniques like Process, Software Engineering Tools and software Engineering methods. The Paper will also provide Software Reliability Modelling and then provides various ways to improve software reliability in the life cycle of software development.

Keywords:

I.INTRODUCTION

Now a Days, computers are playing very important role in our daily lives. Dishwashers, TV’s, Microwave Ovens, AC’s are having their analog and mechanical parts replaced by digital devices, CPU’s and software’s. 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 MCCall and ISO9126 standard ;however, the software reliability is the most important and most measurable aspect of software quality. This paper tries to give generalise for software reliability and modelling is used for that. This will also focus on using software engineering principles in the software development and maintenance so that reliability of software will be improved.

2. RELIABILITY

The Software Reliability is defined as the probability of the “failure free software operation for a specified period of time in a specified environment.” Unreliability of any product comes due to the failures or presence of faults in the system. As software does not wear-out” or “age”, as a mechanical or an electronic system does, the unreliability of software is primarily due to bugs or design faults in the software. 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. It is suggested that reliability modeling should be applied to systems larger than 5000LOC.

3.RELIABILITY PROCESS

The reliability processing 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. The artifacts of the software life cycle include documents, reports, manuals, plans, code configuration data and test data. Software reliability is dynamic .In a new or up graded product ,it begins at a low figure with respect to its new intended usage and ultimately reaches a figure near unity immaturity. The exact value of product reliability however is never precisely known at any point in its life time.
4. SOFTWARE RELIABILITY IMPROVEMENT TECHNIQUES:

The Good engineering methods can largely improve software reliability. 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 application of systematic, disciplined, quantifiable approach to the development operation and maintenance of software will produce economically software that is reliable and works efficiently on real machines. The below Figure 1. shows Software Engineering being the layered technology focuses on the quality and reliability of software.

![Software Engineering Diagram](image)

**Figure 1: Engineering approach to high quality software development**

4.1 Process:

The Process defines a framework that must be established for effective delivery of software engineering technology. It forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms etc) are produces, milestones are established, quality is ensured, and change is properly managed. The process itself should be assessed to ensure that it meets the basic process criteria that are necessary for successful software engineering. The possible relationship between the software process and the methods applied for evaluation and improvement is shown in the Figure 2.

![Software Process Evaluation](image)

**Figure 2: Software Process Evaluation**

4.2 Software Engineering Methods:

Software engineering methods provide technical how to "s" for building software. These methods consist of a broad array of tasks that include requirement analysis, design modeling, program construction, testing and support.

<table>
<thead>
<tr>
<th>Omission</th>
<th>Incorrect Fact</th>
<th>Inconsistency</th>
<th>Ambiguity</th>
</tr>
</thead>
<tbody>
<tr>
<td>26%</td>
<td>10%</td>
<td>38%</td>
<td>26%</td>
</tr>
</tbody>
</table>

**Table 1:**

In the early days of software development, emphasis was on coding and testing but researchers have shown that requirement analysis is the most difficult and intractable activity and is very error prone. In this phase software failure rates and hence the reliability can be increased by:

- Properly identifying the requirements.
- Specifying the requirements in the form of software requirement specification (SRS) document. The basic goal of SRS is to describe the complete external behavior of proposed system.
- Requirement reviews (Validating the SRS.)
- Developing the prototypes.
- Performing structured analysis for developing conceptual models using...
Some projects have collected data about requirement errors. In the effectiveness of different methods and tools in detecting requirement errors in specifications for a data processing application is reported in above Table 1. On an average, a total of more than 250 errors were detected, and the percentage of different types of errors was

4.2.2 Modelling Design:
Design activity is the first step in moving from problem domain to solution domain. The goal of the design is to produce the model of the system which can be later used to build up the system. In this phase reliability can be improved by:

- a) Using “divide and conquer” principle that is dividing the system into smaller pieces (modules) so that each piece can be conquered separately.
- b) Abstraction of components so that maintenance will become easy.
- c) Performing different levels of factoring.
- d) Controlling and understanding the interdependency among the modules.
- e) Design Reviews to ensure that design satisfies the requirements and is of good quality.
- f) Reducing the coupling between modules and increasing cohesion within a module.
- g) Developing design iteratively.

4.2.3 Program Construction
It includes coding and some testing tasks. In this phase software reliability can be increased by:

- a) Constraining algorithms by following structured programming [BOH00] practice.
- b) Write self-documenting code.
- c) Creating interfaces that are consistent with architecture.
- d) Conducting a code walkthrough.
- e) Performing unit tests.
- f) Refactoring codes.

4.2.4 Testing:
After the code construction of software products, testing, verification and validation are necessary steps. Software testing is heavily used to trigger, locate, and remove software defects. Software testing is still in its infant stage; testing is crafted to suit specific needs in various software development projects in an ad-hoc manner. Various analysis tools such as strand analysis, fault-tree analysis, Orthogonal Defect classification and formal methods, etc, can also be used to minimize the possibility of defect occurrence after release and therefore improve software reliability. A strategy for testing may be viewed as shown in Figure 3.

It starts with testing the individual modules and the progress by moving upward to integration testing where the modules are collectively tested for errors. Invalidation testing customer requirements are validated against the software that has been developed. Finally, in system testing, the entire software is tested as a single unit. Once the above testing strategy will be followed for software testing, software reliability can be highly improved.

Figure 4 shows the effect of identifying and removing errors in the early phases of software development, on the software reliability.
fault occurrence or impact of the fault on the system.

4.3 Software Engineering Tools:
Software engineering provides a collection of tools that help in every step of building a product and is termed as CASE (Computer Aided Software Engineering) tools. Case provides the software engineer with the ability to automate manual activities and analysis, design, coding, and testing work. This leads to high quality and high reliable software.

5. SOFTWARE RELIABILITY MODELING:
In the Modeling, To study a system, it is possible to experiment with the system itself or with the model of the system, but experimenting with the system itself is very expensive and risky. The objective of many system studies, however, is to predict how a system will perform before it is built. Consequently, system studies are generally conducted with a model of a system. A model is not only a substitute of a system; it is also a simplification of the system.

A number of software reliability models have emerged as people try to understand the attributes of how and why software fails, and try to quantify software reliability. Over 200 models have been proposed since 1970s, but how to quantify software reliability still remain unsolved. There is no single model that can be used in all the situations. No model is complete; one model may work well for a set of certain software, but may be completely off track for other kinds of problems.

Most existing analytical methods to obtain reliability measures for software systems are based on the Markovian models and they rely on the assumption on exponential failure time distribution. The Markovian models are subject to the problem of intractably largest state space. Methods have been proposed to model reliability growth of components which cannot be accounted for by the conventional analytical methods but they are also facing the state space explosion problem. A simulation model, on the other hand, offers an attractive alternative to analytical models as it describes a system being characterized in terms of its artifacts, events, interrelationships, and interactions in such a way that one may perform experiments on the model, rather than on the system itself, ideally within distinguishable results.

6. CONCLUSION:
Now a days, The Computers are playing very important role in our day-to-day life and there is always a need for high quality software. Software reliability is the most measurable aspect of software quality. Unlike hardware, software does not age, wear out or rust, and reliability of software mainly due to bugs or design faults in the software. Software reliability is dynamic and stochastic. The exact value of product reliability is never precisely known at any point in its lifetime. The study of software reliability can be categorized into three parts: Modeling, Measurement & improvement. Many Models exist, but no single model can capture a necessary amount of software characteristics. There is no single model that is universal to all the situations. It can't be directly measured, so other related factors are measured to estimate the 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. Finally, to develop the software reliability by using software reliability improvement techniques are very easy to understand.
7. REFERENCES: