Software Reliability Measurement and Improvement Policies

Sargun Toor  
PG student in CSE at PIET  

Kailash Bahl  
Faculty in CSE at PIET

ABSTRACT
This paper discusses about various aspects of software reliability. Software reliability is the probability of the failure free operation of a computer program for a specified period of time in a specified environment. Although Software Reliability is defined as a probabilistic function, and comes with the notion of time, different from traditional Hardware Reliability, Software Reliability is not a direct function of time. Electronic and mechanical parts may become "old" and wear out with time and usage, but software will not rust or wear out during its life cycle. Reliability measures the probability of failure, not the consequences of those failures. Software Reliability is dynamic and stochastic. This article provides an overview of Software Reliability measurement and improvement policies then examines different improvement policies for software reliability, however, there is no single model that is universal to all the situations.

Keywords  
Software reliability, stochastic, Fault, Fault tolerance, Phases of software.

1. INTRODUCTION
Software does not age, rust, wear-out, deform or crack. Also software has no shape, color, material, mass. It can not be seen or touched, but it has a physical existence and is crucial to system functionality. People think that once after the software can run correctly, it will be correct forever. But this proves to be wrong by the series of tragedies caused by software. Software can make decisions, but as unreliable as human beings. In complex software systems, reliability is most important aspect of software quality but it is difficult to achieve. 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.
Software Reliability is an important factor that effects system reliability. It differs from hardware reliability as it reflects the design perfection, not manufacturing perfection. Reliability is a probabilistic measure that assumes that the occurrence of failure of reliability. Randomness means that the failure can not be predicted accurately.

2. SPECIFICATIONS
There are three software reliability specifications:

2.1 Hardware Reliability
It occurs due to physical fault. It occurs when system is not in use and is mostly preceded by warnings. Hardware parts may become old and wear out with time.

2.2 Software Reliability
It occurs only by design faults. It occurs only when system is in use and are not preceded by warnings. Software will not change over time unless software is changed or modified intestinally.

2.3 Operator Reliability
A system involves the operators in improving reliability to identify potential equipment problems and failures early. If the failures are minor then the operator fixes them. If the failures are major then it issues a repair request.

3. RELIABILITY DEVELOPMENT FACTORS
Fixing problems may not necessarily make the software more reliable. Reliability depends upon various Software Product and Development process characteristics. Various approaches are used to improve the Software Reliability but is difficult to balance the Software Reliability with time of development and budget.

Development factors are:
- Structured Design, Code, Test and Maintenance
- Pseudo code and Flow Diagrams
- Designing, Coding, Testing and Maintenance Tools.
- Tracing Code to requirement
- Progress and Status reporting and error tracking
- Design and Code walkthroughs
- Formal Reviews

4. SOFTWARE MEASUREMENT TECHNIQUES
Software Measurement techniques are important for software project planning and software quality assurance purposes. Software reliability measurement is naive and hard. Reliability measurement is a mathematical technique which is used to estimate and predict the reliability behavior of software during its development and operation. Perfectly working software can also break if the running environment will change. Measuring software is done by understanding the nature of software.
The primary goal of software reliability modeling is to answer the following question: “Given a system, what is the
probability that it will fail in a given time interval, or, what is the expected duration between successive failures?"

Fig 1: Software Quality Improvement Factors

Software Reliability is an important attribute of software quality, together with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation, it is hard to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, is hard to reach a certain level of reliability. Software reliability can not be directly measured other factors are measured to estimate software reliability and compare it among products. Development process, faults and failures found are all factors related to software reliability.

The various techniques are:

4.1 Product Metrics
As there is no standard way of counting, initial approach to measure the software size is Lines Of Code (LOC), or LOC in thousands (KLOC). SLOC, KSLOC Source codes are used and comments and other non-executable statements are not counted. This method is used to estimate the size of a software system when these functions are identified. It is a measure of the functional complexity of the program and is independent of the programming language. It is used primarily for business systems.

- Function point metric is a method to measure the functionality of a proposed software development based on the count of inputs, outputs, master files, inquiries, and interfaces.
- Test coverage metric estimate fault and reliability by performing tests on software products, assuming that software reliability is a function of the portion of software that is successfully verified or tested.

4.2 Project Management Metrics
Better products are achieved by good management. Relationship exists between the development process and the ability to complete projects on time and within the desired quality objectives. If inadequate processes are used, the development cost increases. Reliability is increased by using better development process like risk management process, configuration management process etc.

4.3 Process Metrics
The quality of the product is a direct function of the process. So process metrics can be used to estimate, monitor and improve the reliability and quality of software. "quality management standards” like ISO-9000 certification is the generic reference for the standards developed by the International Standards Organization (ISO).

4.4 Fault and Failure Metrics
Fault and failure metrics are collected to determine the failure-free execution software. To achieve this goal, number of faults found during testing and the failures or other problems which are reported by the user after delivery are collected, summarized and analyzed. If the testing don’t 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.

5. FAULT
The reliability of any software depends upon the faults of that software. There are various methods to detect the nature of fault. Four ways to deal with Software Faults are: prevention, removal, fault tolerance and input sequence workarounds. The process can be controlled in better way, if we understand the nature of fault by which it can be minimized and its performance can be improved. Best way to produce a reliable software is to develop a high quality software through all the stages of software life cycle.

The various stages where fault can be introduced:

Fig 2: Various Stages of SDLC

5.1 During the Requirements Specification
Requirement Specification phase has the advantage of a language without ambiguities but on the other hand, reduces the freedom to specify requirements.

5.2 During the Design
In Design phase, determine the feasibility of proposed solution before its implementation.

5.3 During the Coding Phase
During the Coding phase, compiler checks various lexical and syntactical errors. It is the static validation of properties that are exhibited during the run time.

5.4 During Maintenance
In Maintenance phase, it can be introduced by re-engineering i.e. new requirements, designing, coding etc.

6. TYPES OF FAULT
Hardware faults are mostly physical faults, while software faults are design faults which are difficult to visualize, classify, detect, and correct. Design faults are based on fuzzy human factors and the design process. In hardware, design faults exist, but physical faults usually dominate. There are mainly three types of faults:

6.1 Transient faults
Transient faults are of short period of time within the program software. They occur due to processor failure or network problem.

6.2 Permanent faults
Permanent faults exist within the system until they are removed. They occur due to design defects.

6.3 Intermittent faults
Intermittent faults appear or disappear time to time.

8. TYPES OF RELIABILITY IMPROVEMENT POLICIES

8.1 Fault Avoidance
Its aim is to prevent introduction of faults during the software development process by standards and various methodologies e.g. ISO 9000. It is achieved at every phase of software life cycle. It can be classify on the basis of orthogonal defects which helps to better understand due to which problems can be introduced in every phase. It can also be detected by software hardening which tries to limit the complexity of the software by establishing reference values for a set of measurements that provide a quantification of software complexity. The formal methods are design techniques that use rigorous mathematical models to build software, hardware systems and help to reduce the errors introduced into a system which is suitable for critical parts of software and embedded applications of limited size. It suffers from state space explosion that limits its applicability. It is done by construction and fault occurrences.

8.2 Fault Detection
Mainly the faults are detected from the developed code on the product.

Fault detection techniques are:
- Formal method
- Dynamic analysis
- Semantic analysis

Formal methods are used to specify the system. It can also be detected by dynamic analysis which is usually done by instrumenting the code at run-time. and semantic analysis of software by static validation of properties exhibited during the run-time, concurrent accesses to shared data. It is done by verification and validation, the existence of faults and eliminate them.

8.3 Fault Tolerance
It provides the controlled response for the uncovered faults at degraded level. Software fault tolerance is a necessary part of a system with high reliability. It is a way of handling unknown and unpredictable software or hardware failures. It can be achieved by various software techniques such as SV, ESV, NVP and ESVBNVP. Several levels of fault tolerance can be implemented:

8.3.1 Full fault tolerance
Full Fault Tolerance where the system continues normal operation even if failure occurs

8.3.2 Graceful Degradation
Graceful Degradation where system operates with reduced functionality or performance. Technique can be classify into two groups like single version and multi version software technique.
8.3.3 Safe mode
It is done by service complying with the specification in-spite of faults having occurred or occurring.

7. CONCLUSION
In this paper we conclude that the reliability of any software depends upon its quality, design and maintenance. A software can be reliable only if it is fault free.

We have discussed various types of faults and the phases where it can be introduced. There are 3 types of Reliability improvement policies. We have also discussed in this paper the 4 types of metrics as Reliability measurement policies. If not considered carefully, software reliability can be the reliability bottleneck of the whole system. Software reliability is not an easy task. As hard as the problem is, progresses are still being made towards more reliable software. More standard components, and better process are introduced in software engineering field.

Fig 4: ISO/IEC 9126 Software engineering

9. REFERENCES