# Approaches for Enhancing Reliability of Software Product

Yojna Arora
Department of Computer Science & Engineering,
Amity School of Engineering & Technology Amity University,
Haryana, India

## ABSTRACT

In modern world, we are highly dependent upon computer for most of our works. As we know, all computers are controlled by software. So, to operate a computer in a proper manner, software reliability is very necessary. Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. The high complexity of software is the major contributing factor of Software Reliability problems. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability. For good reliability, two approaches have to be used, namely, reactive and proactive approach. This paper provides an overview of Software reliability, hardware reliability, reactive and proactive approaches.

## Keywords

Software, Reliability, Hardware, Product

## 1. INTRODUCTION

Computers and computer systems have become a significant part of our modern society. It is virtually impossible to conduct many day-to-day activities without the aid of computer systems controlled by software. As more reliance is placed on these software systems it is essential that they operate in a reliable manner. Failure to do so can result in high monetary, property or human loss. NASA Software Assurance Standard, NASA-STD-8739.8, defines software reliability as a discipline of software assurance that:

i. Defines the requirements for software controlled system fault/failure detection, isolation, and recovery.

ii. Reviews the software development processes and products for software error prevention and/or reduced functionality states.

iii. Defines the process for measuring and analyzing defects and defines/derives the reliability and maintainability factors.

## 2. RELIABILITY

Reliability means the extent to which a particular test or experiment give same result when it is repeated again and again. In the field of Commuter Science, reliability can be classified as Hardware and Software :

### 2.1 Hardware Reliability

Hardware Reliability has a curve known as "bath tub" curve. The bath tub is given below in fig 1.1.As shown in fig, there are three phases in the life of any hardware component Burn in, useful life and wear out phase. In burn-in phase, failure rate is often quite high initially and it goes decreasing gradually with time. It may be due to initially testing in the premises of the organization. During useful life period, failure rate is approx constant. Failure rate increases in wear out phase due to aging of the components. The best period is useful life period. The shape of this curve is like a "bath tub" and that is why it is known bathtub curve.

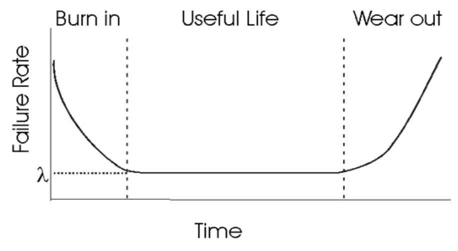

**Fig1.1 Bath Tub Curve Of Hardware Reliability**

### 2.2 Software Reliability

We do not have wear out phase in software .The expected curve for software is given below fig 1.2 .Software may be retired only if it becomes obsolete. Some of the contributing factors are given below:

i. Change in environment
ii. Change in technology
iii. Major change in requirements
iv. Increases in complexity
v. Extremely difficult to maintain
vi. Deterioration in structure of the code
vii. Slow execution of speed
viii. Poor graphical user interfaces

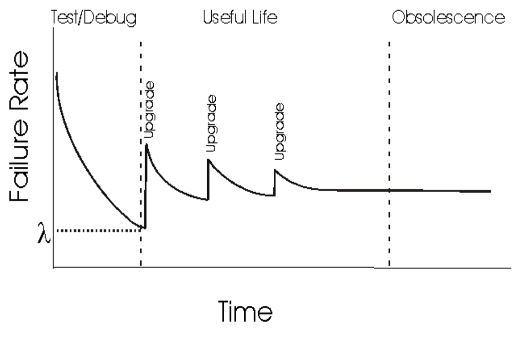

**Fig1.2 Curve of Software Reliability (Failure Rate versus Time)**

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of software is the major problem in software reliability. Software reliability is not a function of time, although researchers have come up with models relating the two. Measurement in software is still in its infancy. No good quantitative methods have been developed to represent software reliability without excessive limitations. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability.

Software reliability is an important attribute of software quality, together with functionality, usability, performance, serviceability, capability, maintainability, and documentation. Software reliability is hard to achieve, because the software may be highly complex. While any system with a high degree of complexity, including software, will be hard 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 ease of doing so by upgrading the software.

A partial list of distinct characteristics of software compared to hardware is listed below:

i. **Failure cause**: Software defects are mainly design defects.

ii. **Wear-out**: Software does not have energy related wear-out phase. Errors can occur without warning.

iii. **Repairable system concept**: Periodic restarts can help fix software problems.

iv. **Time dependency and life cycle**: Software reliability is not a function of operational time.

v. **Environmental factors**: Do not affect Software reliability, except it might affect program inputs.

vi. **Reliability prediction**: Software reliability can't be predicted from any physical basis, since it depends entirely on human factors in design.

vii. **Redundancy**: Cannot improve software reliability if the same software components are used.

viii. **Interfaces**: Software interfaces are purely conceptual other than visual.

ix. **Failure rate motivators**: Usually not predictable from analysis of separate statements.

## 3. APPROACHES FOR SOFTWARE REIABILITY

Software industry is now in the recession period. But it is growing all over the world slowly. The main objectives of the software industry are towards the optimization of the software reliability. The combination of the reactive approach with the proactive approach makes the software quite more reliable. These two approaches are the great achievement in making the software reliable.

### 3.1 Reactive Approach

In software industry reactive approach uses the checkpoint use to mask the error and avoids the failure. There can be considerable gain in the software reliability by using only the reactive approach. The Performance Testing activity is often considered as a reactive way of performance management. In most of the cases, the system performance is never thought of during the early phases of Software Development Life Cycle phases (SDLC). Performance is often thought of only as a last activity after the System Testing phase.

And most of the time , the performance bottlenecks evolves around the system architecture or the system design which is a very high cost activity to think about & in certain cases , the system is put into trash because of the huge deviations in the performance constraints.

Basically waiting for the performance problems to appear & then dealing with it at the end is always not a better approach. Mostly Performance Testing is considered as a reactive approach as there is not much importance is given to the system during early life cycle phases. It is more a 'fix-it-later' approach which is not that effective.

### 3.2 Proactive Approach

The proactive approach uses the dynamic medication of the program for avoiding the future failure. The Proactive approach anticipates the performance problems well in advance & adopts techniques for mitigating them. The importance of the performance is thought about from the Initial phase of Software Development Life Cycle (SDLC) & various performance engineering activities are identified for the system.

The disadvantages of 'fix-it-later' approach are well understood & engineering practices are adopted to analyze the system design in performance angle. The integration of performance engineering activities with the SDLC phases is provided in the below table 1.1

| Phases | Performance Engineering Activities |
|---|---|
| Planning | Check whether performance fall under CTO |
| Requirement Analysis | Performance Requirement Analysis<br>Workload Modeling |
| Architecture & Deign | Performance Modeling<br>Review |
| Coding/ Implementation | Code Profiling |
| Testing | Performance Testing<br>Capacity Planning |
| Maintainenance | Deployment Architecture Review |

## 4. CONCLUSION

The normal concept of combing the reactive and proactive approaches can produce the more dividends for enhancing the software reliability. There are many way to enhance the reliability of the software product. The reactive approach is to deal the breakpoint in the software. And it has been advocated the combination of check pointing and rollback with on-line software version change as a practical technology for making software more reliable. These techniques gives considerable gain in reliability this gain increases as the probability of a failure being transient increases and as the decrease in failure rate obtained by removing a fault increases. The general concept of combining reactive and proactive approaches can produce rich dividends for enhancing software dependability.

## 5. REFERENCES

[1] Robert Graves,“Cost time reliability optimization in product development,” July, 2005.

[2] Walter J. Gutjahr, “Reliability optimization of Redundant Software with Correlated Failure,”

[3] Jeffrey Thomas Oplinger, “Enhancing Software Reliability with Speculative Threads,” Aug, 2004.

[4] MusaJ.D, A. Lannino, K. Okumoto, “Software Reliability Measurement, Prediction & Application”, McGraw Hill Book company NY, PP 183-185, 1987.

[5] Jeff Tian, “Better Reliability assessment And Prediction through Data Clustering”,

[6] D. R. Prince Williams, “Study of the Warranty Cost Model for Software Reliability with an Imperfect Debugging Phenomenon,” Turk J Elec Engines, Volume.15, Number 3, 2007,

[7] Natasha Sharygina, James C. Browne, and Robert P. Kurshan, “A Formal Object-Oriented Analysis for Software Reliability: Design for Verification,”

[8] Booch G., “Object-Oriented Analysis and Design with Applications, “Benjamin/Cummings, Redwood City, CA (1994).

[9] Zeng Wen-hua1, Yiannis Papadopoulos, David Parker, “ReliabilityOptimization of Series-Parallel Systems Using Asynchronous Heterogeneous Hierarchical Parallel Genetic Algorithm,” volume 1, Number 4, 2007.

[10] Jayant Rajgopal, Mainak Mazumdar, “Modular Operational Test Plans for Inferences on Software Reliability Based on a Markov Model”.

[11] Tanvir Khan, “Optimization for Software Release and Crash”, B.S., Louisiana State University – Baton Rouge, May, 2007

[12] Harish Agrawal, “Reliability Based Design Optimization: Formulation And Methodologies”, Dec 2004.

[13] N. Kuschel and R. Rackwitz, “A new approach for structural optimization of series systems”. Applications of Statistics and Probability, 2(8): 987{994 (2000).

[14] Jianwen Xiang, Kokichi Futatsugi, “Fault Tree Analysis of Software Reliability Allocation” School of Information Science, Japan Advanced Institute of Science and Technology 1-1 Asahidai, Tatsunokuchi, Ishikawa, 923-1292 Japan.

[15] M.E. Segal and O. Frieder, “On-the-fly program modification: system for dynamic updating” IEEE software March,1993.

[16] I. Lee ,“DYMOS: A Dynamic Modification System” PhD thesis ,University of Wisconsin,1983.

[17] D. Gupta and P. Jalote, “Online software version change using state transfer between processes” Software-practice and experience,Sept 1993.

[18] O. Frieder and M.E. Segal, “On dynamically updating a computer program: from concept to prototype” J. System software, Sept 1991.

[19] R.S. Fabry ,“How to design systems in which module can be changed on the fly” In Proc. 2nd Int. Conf Software Engg,1976.