Software Reliability - An Overview

[1]E.Sridevi, [2]B.Aruna, [3]P. Sowjanya
[1,2,3]Dept. of Freshman Engineering, KL University, AP, India

Abstract
Usually, software that is deemed reliable does what it shall. Software that is deemed safe, doesn't do what it shall not do, i.e. does nothing that can lead to accident. Example a weapon system should destroy and kill. The property that the system destroys and kills is a reliability property. The property that the software in the weapon system doesn't destroy and kill friendly forces is a safety property. Many systems in these days are much more becoming software intensive and many software intensive systems are safety critical. For this reason, the technique well developed to measure of software reliability is very important for whom to assess such a system. This paper briefly explains what are different models, metrics and techniques available to increase the reliability of safety critical systems.

Keywords
Probabilistic Failure Models, POFOD, ROCOF, MTTF, AVAIL

I. Introduction

Computer and as well as software running on them are playing a vital role in our daily lives .The analog and mechanical parts of some appliances such as TVs ,washing machines are replaced by cpu's and software .Like machinery replaced craftsman's and intelligent parts are quickly pushing their mechanical counter parts out of the market. Software doesn't age, rust, wear-out, deforms or crack, there is no environmental constraint for software to operate as long as the hardware processor it runs on can operate .Without being proven to be wrong, optimistic people would think that once after the software can run correctly, it will be correct forever. A series of tragedies and chaos caused by software proven this to be wrong. These events will always have their place in history. Software can make decision, but can just as unreliable as human beings. Software can also have small unnoticeable errors or drift that can culminate in to a disaster. Once perfectly working software may also break if the running environment changes. Fixing problems may not necessarily make the software more reliable on the contrary, new serious problems may arise [1]. This makes us wondering whether software is reliable at all, whether we should use software in safety-critical embedded applications. Serious problems are not generated when the embedded software in some machines like washing machines, ATM are not working properly. But the software errors in the machines like airplanes, heart pace-makers, and radiation therapy machine can easily claim peoples' lives. In the safety critical embedded world, the reliability of software is simple a matter of life and death [2].

II. Software Reliability

Software reliability is the probability that the software system will function properly without failure over a certain time period. Software reliability is one of the important attribute of the quality. It is hard to reach certain level of reliability in any system with high degree of complexity [5].

III. Why Software Fails

A failure corresponds to an occurrence where the operational behavior of a program deviates from the requirements. Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems. Software and hardware don't behave in the same way. The long software is used and maintained, the higher the degree of confident it obtains as defects are found and weeded out. Hardware on the other hand ages. Heat, oxidation and wear take their tolls on hardware, and even a perfectly function device will eventually fail, despite how well it's treated.

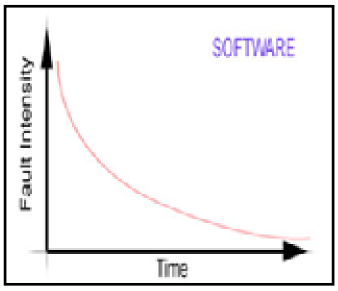

Fig1: Softwar Reliability Curve

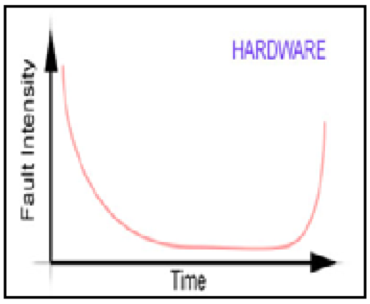

Fig. 2: Hardware Reliability Curve

A software reliability curve flattens out over time. A hardware reliability curve is more U-shaped. Since a system consists of running software on hardware, both need to be taken into account concurrently, and for that we have the system reliability model [5].

IV. What Techniques, Models and Metrics Available

Software reliability compressed of three activities: such as Error prevention, Fault detection and Removal measurements to minimize reliability. There are many quantitative block diagrams, Fault Tree Analysis, Event Tree, Markov model. Failure models and effects analysis, Fault Tree Analysis etc qualitative techniques are used to compare different systems. Reliability can be improved by using redundancy, quality, diversity, number of spare parts, Mean Time of Failure are the other parameters that effect the system/device reliability.

V. What are Software Reliability Models

Software reliability is one of the most important features for a critical system which can affect human's life. Therefore it is necessary to measure and control the reliability of a software system. A number of software reliability growth models are been proposed. Software reliability relies on three basic models [5].

A. Usage Models
Describes how the software is used.

B. Trend Models
Describes how reliability evolve our times as certain bugs are fixed or new bugs are introduced.

C. Probabilistic Failure Models
Captures the fact that failures may happens randomly. Most software models contains assumption, factors and a mathematical function that relates the reliability with the factors. The mathematical function is usually higher order exponent or logarithms. Software modeling techniques can be divided as prediction modeling and estimation modeling. Both kinds of modeling techniques are based on observing and accumulating failure data and analyzing with statistical inference .The term reliability prediction denotes the process of applying mathematical modeling and for the purpose of estimating field reliability of system before the empirical data about the system is available. Reliability prediction provides the quantitative baseline needed to assess progress in reliability design.
In 1962, first version of US Military Handbook "USML-217", was published by US Navy which become the standard for reliability prediction. Prediction models include Musa's Execution Time Model, Putnam's Model and Rome Laboratory Models TR-9251 and TR-92-15 etc. Using prediction models, software reliability and be predicted early in the development phase and enhancements can be initiated to improve the reliability.

VI. What are Software Reliability Metrics
Metrics are used to pinpoint problem areas so that remedies can be developed and the software process can be improved. Reliability metrics are derived from failure occurrence expressions and data.

Table 1: Reliablity Metrics

Metrics	Reliability Specification
Probability of failure on demand(POFOD)	For systems where services request happens in an unpredictable way or when there is a long time interval between consecutive requests
Rate of occurrence of failures(ROCOF)	For systems when services are demand in more regular way
Mean time to failure(MTTF)	For systems involving long transactions, during which a guarantee of service continuity and delivery should be expected.
Availability(AVAIL)	For systems where continuous services delivery is a major concern

The software reliability measurement can be divided into four categories.
- Product metrics
- Project management metrics.
- Process metrics
- Fault and failure metrics.

Software process and product metrics are quantitative measures that enable software people to gain insight into the efficiency of the software process and the project that are conducted using the process as a framework. Basic quality and productivity data are collected. These data are then analyzed, compared against post averages, and assessed to determine whether quality and productivity improvements have occurred. Good project management can result in better products. Research has demonstrated that a relationship exists between the development process and the ability to complete projects on time and within the desired quality objects. Higher reliability can be achieved by using better development process, risk management process, configuration management process, etc.
A fault is a software defects that causes a failure and a failure is the unacceptable departure of a program operation from program requirements. When measuring reliability, we are usually measuring only defects found and defects fixed. If the objective is to fully measure reliability we need to address the prevention as well as investigate the development starting in the requirement phase-what the programs are developed to. Usually, failure metrics are based upon customer information regarding failures found after release of the software. The failure data collection is therefore used to calculate failure density. Mean Time Between Failures (MTBF) or other parameters to measure or predict software reliability.
Metrics are identified and data collection plans are then developed for satisfying reliability goals. Criteria are identified for measuring and interpreting conformance with the reliability requirements during inspection and testing.

V. What are Software Reliability Improvement Techniq-ues
Measuring software reliability does not directly make software reliable even if there is a proper answer for estimation of software reliability. Software fault should be carefully handled to make software more reliable with as many reliability improvement techniques as possible. The software reliability improvement techniques divided into three categories [4].
- Fault avoidance/prevention that includes design methodologies to make software provably fault-free.
- Fault removal that aims to remove faults after the development stage is completed.
- Fault tolerance that assumes a system has unavoidable and undectable faults and aims to make provisions for the system to operate correctly, even in the presence of faults.

Good engineering methods can largely improve software reliability. Before deployment of software products, testing, verification and validation are necessary steps. Software testing is heavily used to locate and remove the software defects. Software testing is still in its infant stage, testing is crafted to suit specification needs in various software development projects in an Ad-Hoc manner. Various analysis , orthogonal defect classification and formal methods, etc. Can also be used to minimize the possibility of defect occurrence aften release and therefore improve software reliability [3].
After deployment of the software product, field data can be gathered and analyzed to study the behavior of software defects. Fault tolerance or fault/failure of forecasting techniques will be helpful techniques and guide rules minimize fault occurrence or impact of the fault on the system [3].

VI. Conclusion

Over 200 models have been developed since the early 1970s, but how to quantify software reliability still remains largely unsolved. As many models as there are many more emerging, none of the models can capture a satisfying amount of the complexity of software, constraints and assumptions have to be made for the quantifying process. Therefore, there is no single model that can be used in all situations. No model is complete or even representative. One model nay work well for a set of certain software, but may be completely off track for other kinds of problems. Good metrics should facilitate the development of models that are capable of predicting process or product parameters, not just describing them. Thus, ideal metrics should be: simple, objective, easily obtainable, valid, robust. It is possible that, in future the scope of software metrics may be expanded to include performance evaluation, or that both activities may be considered part of a large area that might be called software measurement.

References

[1] Chin-Yu Huang, Jung-Hua Lo, Sy-Yen Kuo, Michal R.Lyu,"Software reliability modelling and cost estimation incorporating testing-effort and efficiency", Journal of Software Reliability Engineering,1999. Proceedings. 10th International Symposium on pp. 62-72, 1999.

[2] Peter Liggesmeyer, Mario Trapp,"Trends in Embedded software engineering", Jornal of IEEE software ,Vol. 26, Issue 3, 2009.

[3] Han Seong Son, Seo Ryong Koo,"Software reliability improvement techniques", Springer Series On Reliability Engineering, Vol. 2 , pp. 105-120, 2009.

[4] M.Xie,"Software reliability modelling".

[5] R.S Pressman, Associates,"Software Engineering Resources".

E. Sridevi received her B.Sc. degree in Electronics from St. Theresa's Women's Degree College, Eluru, Andhra Pradesh, India, in 1998, the M. Tech. degree in Computer Science from Acharya Nagarjuna University, Guntur, Andhra Pradesh, India, in 2010. She was a Assistant professor, with Department of Freshman Engineering Department in KL University, Andhra Pradesh, India, from 2007 to till date. Her research interests include Software Engineering and Data Mining.