RESEARCHERID
THOMSON REUTERS

[Rao * *et al.*, 7(3): March, 2018]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

IMPROVEMENT IN THE SOFTWARE RELIABILITY BY USING THE SOFTWARE RELIABILITY CHARACTERISTIC MODEL AND MEASURES OF DEFECT CONTROL

Puli Nageswara Rao[*1], D.Jyothirmai[2] & Dr.K.Subba Rao[3]

[*1&2]Asst.Professor, SRKIT-Vijayawada-India, BVRIT-Narsapur-Medak,-India,BVRIT-Narsapur-Medak-India

[3]Professor, SRKIT-Vijayawada-India, BVRIT-Narsapur-Medak,-India,BVRIT-Narsapur-Medak-India

DOI: 10.5281/zenodo.1199222

ABSTRACT

In the Software Industry, For high reliability software, mission critical system software and embedded system software, to demonstrate convincingly and validly that the software satisfies the reliability requirements is one of the most challenging issues. In this paper, we present a method for developing software reliability case based on software reliability characteristic model and measures of defect control. Three software reliability argument patterns are proposed. As a case study, we take the load control software to demonstrate how a software reliability case can be generated using the proposed method and corresponding argument patterns.

Keywords: reliability case, argument, pattern, reliability characteristic model, defect.

I. INTRODUCTION

Today, as software applications have been widely used, the quality of software has become more and more important. Software reliability, as a key attribute of software quality, is playing an important role in improving software quality. Because problems brought by software failures could lead to serious consequences, especially for high reliability software, mission critical software and embedded system software. Therefore the assurance of software reliability needs to be demonstrated and supported by evidences before the software is put into application. Relations between software reliability requirements and evidences also need to be well argued. A software reliability case is "a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims regarding software's reliability is adequately justified for a given application in a given environment. It is used to demonstrate how someone can reasonably conclude that software is acceptably reliable

II. RELATED CONCEPT

A system is said to be reliable if it works correctly at all times without failures. IEEE defines: "Software reliability is the probability of failure-free operation of software over a given time interval and under given conditions." Reliability of software depends on the presence or absence of defects in the system. As the system consists of hardware and software, its reliability depends on the reliability of the hardware and the reliability of software.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

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. While it is tempting to draw an analogy between Software Reliability and Hardware Reliability, software and hardware have basic differences that make them different in failure mechanisms. Hardware faults are mostly *physical faults*, while software faults are *design faults*, which are harder to visualize, classify, detect, and correct. Design faults are closely related to fuzzy human factors and the design process, which we don't have a solid understanding. In hardware, design faults may also exist, but physical faults usually dominate. In software, we can hardly find a strict corresponding counterpart for "manufacturing" as hardware manufacturing process, if the simple action of uploading software modules into place does not count. Therefore, the quality of software will not change once it is uploaded into the storage and start running. Trying to achieve higher reliability by simply duplicating the same software modules will not work, because design faults can not be masked off by voting.

The bathtub curve for Software Reliability

Over time, hardware exhibits the failure characteristics shown in Figure 1, known as the bathtub curve. Period A, B and C stands for burn-in phase, useful life phase and end-of-life phase. A detailed discussion about the curve can be found in the topic Traditional Reliability

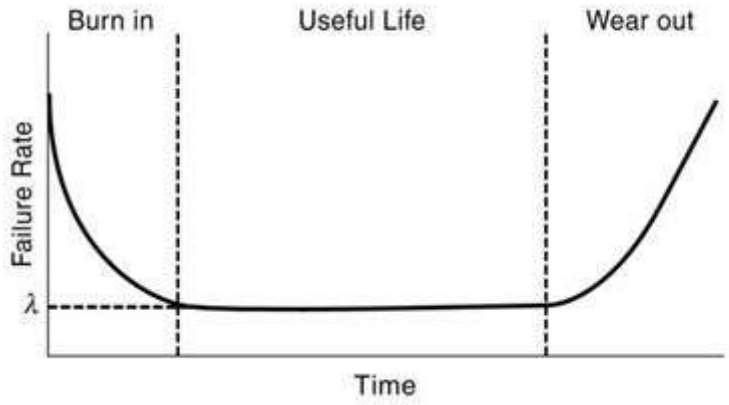

Figure 1. Bathtub curve for hardware reliability

Software reliability, however, does not show the same characteristics similar as hardware. A possible curve is shown in Figure 2 if we projected software reliability on the same axes. There are two major differences between hardware and software curves. One difference is that in the last phase, software does not have an increasing failure rate as hardware does. In this phase, software is approaching obsolescence; there are no motivation for any upgrades or changes to the software. Therefore, the failure rate will not change. The second difference is that in the useful-life phase, software will experience a drastic increase in failure rate each time an upgrade is made. The failure rate levels off gradually, partly because of the defects found and fixed after the upgrades.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

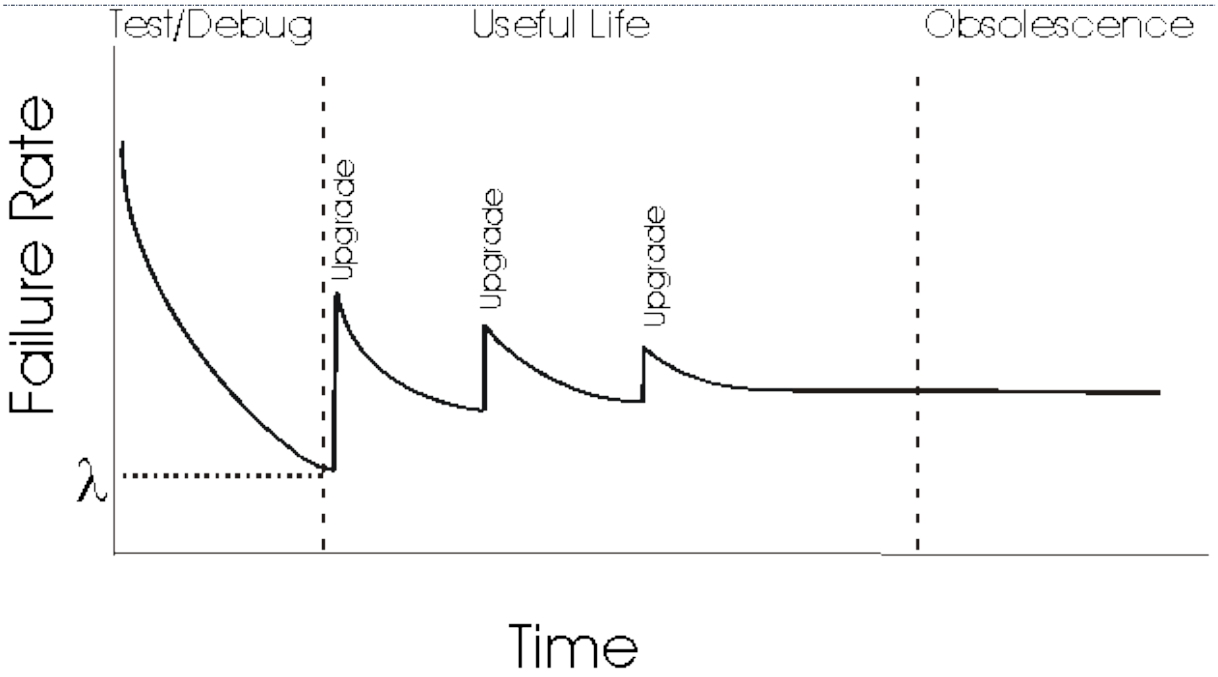

Figure 2. Revised bathtub curve for software reliability

The upgrades in Figure 2 imply feature upgrades, not upgrades for reliability. For feature upgrades, the complexity of software is likely to be increased, since the functionality of software is enhanced. Even bug fixes may be a reason for more software failures, if the bug fix induces other defects into software. For reliability upgrades, it is possible to incur a drop in software failure rate, if the goal of the upgrade is enhancing software reliability, such as a redesign or reimplementation of some modules using better engineering approaches, such as clean-room method.

III. SOFTWARE RELIABILITY METRICS

The current practices of software reliability measurement can be divided into four categories

1. Product metrics

Software size is thought to be reflective of complexity, development effort and reliability. Lines Of Code (LOC), or LOC in thousands(KLOC), is an intuitive initial approach to measuring software size. But there is not a standard way of counting. Typically, source code is used(SLOC, KSLOC) and comments and other non-executable statements are not counted. This method can not faithfully compare software not written in the same language. The advent of new technologies of code reuse and code generation technique also cast doubt on this simple method. Function point metric is a method of measuring the functionality of a proposed software development based upon a count of inputs, outputs, master files, inquires, and interfaces. It is a measure of the functional complexity of the program. It measures the functionality delivered to the user and is independent of the programming language. It is used primarily for business systems; it is not proven in scientific or real-time applications. Complexity is directly related to software reliability, so representing complexity is important. Complexity-oriented metrics is a method of determining the complexity of a program's control structure, by simplify the code into a graphical representation. Representative metric is McCabe's Complexity Metric. Testcoverage metrics are a way of estimating fault and reliability by performing tests on software products, based on the assumption that software reliability is a function of the portion of software that has been successfully verified or tested. Detailed discussion about various software testing methods can be found in topic Software Testing

2. Project management metrics

Researchers have realized that good 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 objectives. Costs increase when developers use inadequate processes. Higher reliability can be

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

achieved by using better development process, risk management process, configuration management process, etc.

3. Process metrics

Based on the assumption that the quality of the product is a direct function of the process, process metrics can be used to estimate, monitor and improve the reliability and quality of software. ISO-9000 certification, or "quality management standards", is the generic reference for a family of standards developed by the International Standards Organization(ISO).

4. Fault and failure metrics

The goal of collecting fault and failure metrics is to be able to determine when the software is approaching failure-free execution. Minimally, both the number of faults found during testing (i.e., before delivery) and the failures (or other problems) reported by users after delivery are collected, summarized and analyzed to achieve this goal. Test strategy is highly relative to the effectiveness of fault metrics, because if the testing scenario does not 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.

IV. CONCLUSION

Software reliability is a key part in software quality. The study of software reliability can be categorized into three parts: modelling, measurement and improvement. Software reliability modelling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, but no single model can capture a necessary amount of the software characteristics. Assumptions and abstractions must be made to simplify the problem. There is no single model that is universal to all the situations.Software reliability measurement is naive. Measurement is far from commonplace in software, as in other engineering field. "How good is the software, quantitatively?" As simple as the question is, there is still no good answer. Software reliability cannot be directly measured, so other related 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. Software reliability improvement is hard. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software. Until now there is no good way to conquer the complexity problem of software. Complete testing of a moderately complex software module is infeasible. Defect-free software product can not be assured. Realistic constraints of time and budget severely limit the effort put into software reliability improvement.

V. REFERENCES

[1] C.T.Lin,C.Y.Huang,andC.C.Sue,"MeasuringandAssessingSoftwareReliabilityGrowthThroughSimulationBasedApproaches,"Proceedingsofthe31stIEEEAnnualInternationalComputerSoftwareandApplicationsConference(COMPSAC2007),pp.439-446,Beijing,China,July 2007.
[2] J.Lo,S.Kuo,M.R.Lyu,andC.Huang,"OptimalResourceAllocationandReliabilityAnalysisforComponentBasedSoftwareApplications,"Proc.26thAnn.Int'lComputerSoftwareandApplicationsConf.(COMPSAC),pp.7-12,Aug.2002.
[3] JohnD.Musa,"OperationalProfilesinSoftwareReliabilityEngineering,"IEEESoftware,v.10n.2,p.14-32,March1993
[4] KishorS.Trivedi,"Probabilityandstatisticswithreliability,queuingandcomputerscienceapplications,"JohnWileyandSonsLtd.,Chichester,UK,2001
[5] K.KanounM.KaanicheC.BeounesJ.C.LaprieandJ.Arlat,"ReliabilityGrowthofFault-TolerantSoftware,"IEEETrans.Reliability,vol.42,no.2,pp.205-219,June1993.
[6] Kumar,M.,Ahmad,N.,Quadri,S.M.K.(2005),"SoftwarereliabilitygrowthmodelsanddataanalysiswithaParetotest-effort",RAUJournalofResearch,Vol.15,No.1-2,pp124-8
[7] NormanF.Schneidewind,"FaultCorrectionProfiles,"Proceedingsofthe 14th InternationalSymposiumon SoftwareReliabilityEngineering,p.257, November17-21,2003
[8] Quadri,S.M.K.,Ahmad,N.,Peer,M.A.(2008),"Softwareoptimalreleasepolicyandreliabilitygrowthmodeling",Proceedingsof2ndNationalConferenceonComputingforNationDevelopment,INDIACom-2008,NewDelhi, India,pp423-31

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

[9] Improving Software Reliabilityusing Software EngineeringApproach- AReview International Journal of Computer Applications (0975–8887) Volume10–No.5,November2010
[10] Bittanti S, Bolzern P, Pedrotti E, Scattolini R "A flexible modeling approach for software reliability growth". In: Goos G,Harmanis J (eds) Software reliability modelling and identification. Springer, Berlin, pp 101-140, 1988.
[11] Brooks WD and Motley RW, "Analysis of Discrete Software Reliability Models", Technical Report (RADC-TR-80-84), Rome Air Development Center, New York, 1980.
[12] Chakravart, Laha and Roy, Handbook of Methods of Applied Statistics, Volume I, John Wiley and Sons, pp. 392-394, 1967.
[13] Chen Y and Singpurwalla N D "Unification of software reliability models by self-exciting point processes", Adv. Appl. Probab., 29, pp. 337–352, 1997.
[14] Cheung RC. "A User Oriented Software Reliability Growth Model" IEEE Transactions on Software Engineering; SE-6: pp 118-1251980.
[15] Chang, I.P, " An analysis os software reliability with change-point models". NSC 85-2121-M031-003; National Science Council, Taiwan., 1997
[16] Chang, Y.P., "Estimation of Parameters for Non Homogenous Poisson Process Software Reliability with Change-Point Model" Communications in Statistics-Simulation and Computation, 30, pp 623-635, 2001.
[17] C.-T. Lin, C.-Y. Huang, "Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models", The Journal of Systems and Software 81 (2008) 1025–1038.
[18] Downs T and Scott A., Evaluating The Performance Of Software Reliability Models, IEEE transactions on reliability, 41(4): 532-538, 1992.
[19] Dohi, T., Osaki, S. and Trivedi, K.S, " An infinite server queuing approach for describing software reliability growth ~ Unified Modeling and Estimation Framework ~, Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), pp. 110.119, 2004.
[20] Farr W., "Handbook of Software Reliability Engineering", M.R. Lyu, Editor, Chapter "Software Reliability Modeling Survey", pp 71-117, McGraw Hill, New York, 1996.
[21] Garrison K, Estimating Defects in Commercial Software during Operational Use. IEEE Tr. on Reliability 42(1): 107–115, 1993.
[22] A.Leela Krishna Reddy, Dr.K.SubbaRao Published a international Journal on"Awareness the Analysis on Software Design and Its Partners Relations" International Journal of Engineering Science AND Research Technology(IJESRT), Vol: 7 Issue: 2ISSN No: 2277-9655, Feb 2018(UGC Approved Journal).
[23] A.Leela Krishna Reddy, Dr.K.Subba Rao Published a international Journal on"An Analysis of Software Industry and Its Partners Relations" International Journal of Current Engineering Scientific Research (IJCESR), Vol:5 Issue: 2 ISSN No:2393-8374, Feb 2018(UGC Approved Journal).

CITE AN ARTICLE

Rao, P., Jyothirmai, D., & Rao, K. (n.d.). IMPROVEMENT IN THE SOFTWARE RELIABILITY BY USING THE SOFTWARE RELIABILITY CHARACTERISTIC MODEL AND MEASURES OF DEFECT CONTROL. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY*, *7*(3), 374-378.