iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** II **Month of publication:** February 2018

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue II, February 2018- Available at www.ijraset.com

A Reliable Model Development for Estimation of Computational Reliability

Dr. Shelbi Joseph[1], Akhil P V[2]
[1,2]*Information Technology, School of Engineering, Cochin University of Science and Technology*

Abstract: Softwares and the systems that employ them have become an integral part of human life now. Softwares are now widely used in almost all facets and the reliability of such systems is considered to be very important. There are many systems which are considered to be critical and cannot afford to undergo failures. Failure of such a system will very seriously affect the set of people as well as customers using them. So achieving quality and reliability on a software system is considered to be a crucial affair. Reliability describes the ability of the system to function under specified environment for a specified period of time and is used to objectively measure the quality. Evaluation of reliability of a computing system involves computation of hardware and software reliability. Most of the earlier works were given focus on software reliability with no consideration for hardware parts or vice versa. However, a complete estimation of reliability of a computing system requires these two elements to be considered together, and thus demands a combined approach. The present work focuses on this and presents a model for evaluating the reliability of a computing system. The method involves identifying the failure data for hardware components, software components and building a model based on it, to predict the reliability.
Keywords: Software reliability, hardware, software, open source software, computation model

I. INTRODUCTION

Softwares and the systems that employ them have become an integral part of human life now. Softwares are now widely used in almost all facets and the reliability of such systems is considered to be very important. There are many systems which are considered to be critical and cannot afford to undergo failures. Failure of such a system will very seriously affect the set of people as well as customers using them. So achieving quality and reliability on a software system is considered to be a crucial affair. The quality of a software product decides its acceptance or fate in the software development life cycle. High developmental costs and increasing global competition have intensified the pressures to quantify software systems quality, and the need to measure and control the level of quality delivered. Reliability is the most important and most measurable aspect of software systems quality, and is customer- oriented. It is the capability of a system to deliver results accurately every time the user requests it. The performance of a system is largely affected by its failure to deliver or downtime. Therefore, a system is considered to be reliable if it can rectify its failure at minimum time, thereby ensuring guaranteed results to the user. It is a measure of how well the product functions, to meet its operational requirements. In other words, it decides the software product's acceptance or fate in the life cycle. It ensures the products capability to rectify its failure. Many models where put forward to address the reliability of the computer system, considering software and hardware components independently. The total performance of the system can be modelled only by considering these components together. The basic notions of reliability is introduced, reliability approaches during software life cycle phases, along with discussions on consequences of reliability and its impact on software industry.

A. Reliability

Reliability is the probability of success or the probability that the system will perform its intended function under specified design limits. More specifically, reliability is the probability that a product or part will operate properly for a specified period of time (design life) under the design operating conditions such as temperature, voltage etc., without failure. In other words, reliability may be used as a measure of the system's success in providing its function properly. Reliability is one of the quality characteristics that consumers require from the manufacturer of products.

Product failures cost money has an impact on the development schedules and system performance through the increased use of computer resources such as memory, CPU time, and peripheral requirements. Consequently, there can be too much as well as too little effort spent dealing with faults. The system engineer along with management can use reliability estimation and assessment to understand the current status of the system and make trade-off decisions. The basic objective is to identify required elements for an understandable, credible reliability prediction, which will provide sufficient information to the users to evaluate the effective use of the prediction results. A reliability prediction should have sufficient information concerning inputs, assumptions, and uncertainty; so

that the risk associated with using the prediction results would be understood. To analyse the reliability characteristics further, it is necessary to look at the hardware and software reliabilities separately.

B. Hardware Reliability

Hardware reliability is nothing but the ability of hardware to perform its functions for some specific duration of time and is expressed as mean time between failures (MTBF). Computer systems whether hardware or software are subject to failure. A failure may be produced in a system or product when a fault is encountered resulting in the non operation or disability of the required function and a loss of the expected service to the user [1].The field of hardware reliability has been established for some time, which is related to software reliability and the division between hardware reliability and software reliability is somewhat artificial and both may be defined in the same way. Therefore, it is possible to combine both hardware and software component reliabilities to get system reliability [2].

C. Software Reliability

The IEEE defines software reliability as the probability that software will not cause the failure of a system for a specified time under specified conditions [3]. Software reliability denotes the probability that a software product in a pre-defined condition performs its tasks without malfunctioning for a specified period of time.

Software Reliability is important for many sectors of the software industry. Besides knowing how to achieve reliability, the most important thing is to know the actual reliability achieved in a specific software product. Assessing the reliability of software- based systems is increasingly necessary because of the survival of companies and at times the lives and limbs of people on the service they expect from the software. Sound decision-making requires some understanding of the uncertainties thus incurred. Meanwhile, software complexity increases and progress in development tools enables lesser- trained people to build software based systems. The short term economic incentive to use off-the-shelf software, even in sensitive applications, imposes new requirements to evaluate the risk thus assumed. The pressure on vendors to guarantee some level of quality of service will thus also increase, extending from bespoke software to off-the-shelf software and from mission-critical to productivity critical software.

A Software reliability model specifies the general form of the dependence of the failure process on the principal factors that affect it; fault introduction, fault removal, and the operational environment. The failure rate of a software system generally decreases due to fault identification and removal. It is possible to observe the history of failure rate by statistically estimating the parameters associated with the selected model. The purpose of the model is twofold. Firstly, to estimate the extra execution time during the test required meeting a specified reliability objective and secondly to identify the expected reliability of the software when the product is released [4].

This paper reviews some significant works in reliability estimation model and then proposes a more reliable method to estimate reliability. The chapters are arranged as follows. Chapter II reviews some of the exiting works, followed by chapter III which proposes a reliability estimation model. And finally chapter IV concludes the work and gives an insight to possible future works on the same.

II. BACKGROUND STUDY

The background study performed in this paper will be organised to three different areas namely, Software Reliability Growth Models, Hardware Reliability Growth Models and Computational System Reliability.

A. Software Reliability Growth Models

Software Reliability is considered as part of software quality assurance and have many attributes including usability, capability, performance, functionality, documentation, maintainability and reliability. It is essentially being able to deliver usability of the services while assuring the constraints of the system. Software reliability modeling surprisingly to many, has been around since the early 1970s, with pioneering works by [5] [6] [7] [8] [9] [10] [11]. The basic approach is to model past failure data to predict future behaviour. The models fall into two basic classes namely failures per time period and time between failures.

A software reliability growth model provides a systematic way of assessing and predicting software reliability based on certain assumptions about the fault in the software and fault exposure in a given usage environment [12]. The reliability growth for software is the positive improvement of software reliability over time, accomplished through the systematic removal of software faults. The rate at which the reliability grows depends on how fast faults can be uncovered and removed. A software reliability growth model allows project management to track the progress of the software's reliability through statistical inference, and to make projections of

future milestones [13] [14]. Models are classified in terms of five different attributes. Time domain: Wall clock versus Execution time. Category: Total number of failures that can be experienced in finite or infinite time. Class/Finite failure category: Functional form of the failure intensity expressed in terms of time. Family/Infinite failure category: Functional form of the failure intensity function expressed in terms of the expected number of failures experienced. Type: The distribution of the number of the failures experienced by time t. Poisson and Binomial are the two important types.

B. Hardware Reliability Growth Models

A hardware reliability growth model is used to mention product reliability in the period during which, the observed reliability advances towards the inherent reliability of the product. Hardware and software reliability predictions adjusted by their respective growth models to coincide with the same point in time can be combined to obtain a prediction of the overall system reliability. There was a number of reliability growth models suggested for hardware reliability in the literature. In this section, we outline important models that discuss hardware reliability. Understanding the dynamic behaviour of system reliability becomes an important issue in either scheduling the maintenance activities or dealing with the improvement in the revised system design. In doing so, the failure or hazard rate function should be addressed. Bathtub curve is usually adopted to represent the general trend of hazard rate function. This curve exhibits three distinct zones. The first is, the short initial period called variously the early failure, infant mortality, or the burn in period. The decreasing but greater failure rate early in the life of the system is due to one or more of several potential causes. The causes include inadequate testing or screening of components during selection or acceptance, damage to components during production, assembly, or testing, and choice of components which have too great a failure variability. It shall be a specific goal of the supplier to ensure that the early failure period is rigorously controlled and covered by a suitable warranty [15] [16].

C. Computational System Reliability

Computational system reliability is concerned with hardware reliability, software reliability, reliability of interaction between hardware and software and reliability of interaction between the system and the operator. In general, a system may be required to perform various functions, each of which may have a different reliability. In addition, at different times, the system may have a different probability of successfully performing the required function under stated conditions. The analysis of the reliability of a system must be based on precisely defined concepts. Software intensive systems are increasingly used to support critical business and industrial processes, such as in business information systems, e-business applications, or industrial control systems. Reliability engineering gains its importance in the development process. Reliability is compromised by faults in the system and its execution environment, which can lead to different kinds of failures during service execution: Software failures occur due to faults in the implementation of software components, hardware failures result from unreliable hardware resources, and network failures are caused by message loss or problems during inter component communication [17].

The analysis of the reliability of a system must be based on precisely defined concepts. Since it is readily accepted that a population of supposedly identical systems, operating under similar conditions, fall at different points in time, then a failure phenomenon can only be described in probabilistic terms. Thus, the fundamental definitions of reliability must depend on concepts from probability theory [18]. System-level reliability and availability requirements set forth by U.S. Government agencies procuring large software intensive systems encompass both hardware and software. However, specifications, statement of work requirements, and compliance documents (standards) usually implicitly or explicitly focus on hardware and are largely silent about software reliability, maintainability, availability and dependability.

Consequently, contractor system reliability analyses and design reviews usually ignore quantitative software reliability, maintainability, availability, and dependability requirements. During system testing and evaluation, data on software operating times, failure rates, and recovery times are not collected. Finally, logistics and support specialists devote significant attention to sparing and maintenance concept development, but often do not adequately consider the software-related sustainment issues of large computer systems. These problems can be solved, by an appropriate definition of requirements for software-intensive system reliability in specifications, and in the definition of programmatic requirements in contractual documentation [19].

In general, a system may be required to perform various functions, each of which may have a different reliability. In addition, at different times, the system may have a different probability of successfully performing the required function under stated conditions. The term failure means that the system is not capable of performing a function when required. The term capable used here is to define if the system is capable of performing the required function. However, the term capable is unclear and only various degrees of capability can be defined [20] [18].

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue II, February 2018- Available at www.ijraset.com

III.PROPOSED WORK

The methodology involves defining a system configuration consisting of software and hardware elements. The software and hardware components are considered independently consisting of two subsystems. The failure related data of hardware components are based on the field data. On the other hand, the software failure is based on the bug arrival rate as published in the Debian site. The hardware modelling is based on the constant Hazard model whereas, the software model is based on the distribution as obtained from the bug arrival. A relatively simplified model is one, where both hardware and software exhibit a constant failure.

The objective of the research is to develop a model to represent reliability of a computing system by considering both hardware and software failure impacts. The methodology involves studying the effects of failure of an actual software package and working towards formulating a reliability model, taking into consideration the hardware issues.

Studying the effects of failure of actual software package is comprised of three stages namely data collection, data pre-processing and analysis. The formulation of the reliability model involves algorithm development, model development, and comparison of theoretical products with the formulated model as shown in Fig. 1. The first phase of the work is the data collection. In this phase, failure data for software and hardware are to be collected. The collected data is pre-processed to get the valid data set in the second phase of the work. In the third phase the valid data set is analysed to get the reliability equation and thus the model generation. The generated model is then compared with the theoretical and existing models and concluded that the model developed is a reliable one as the final phase.

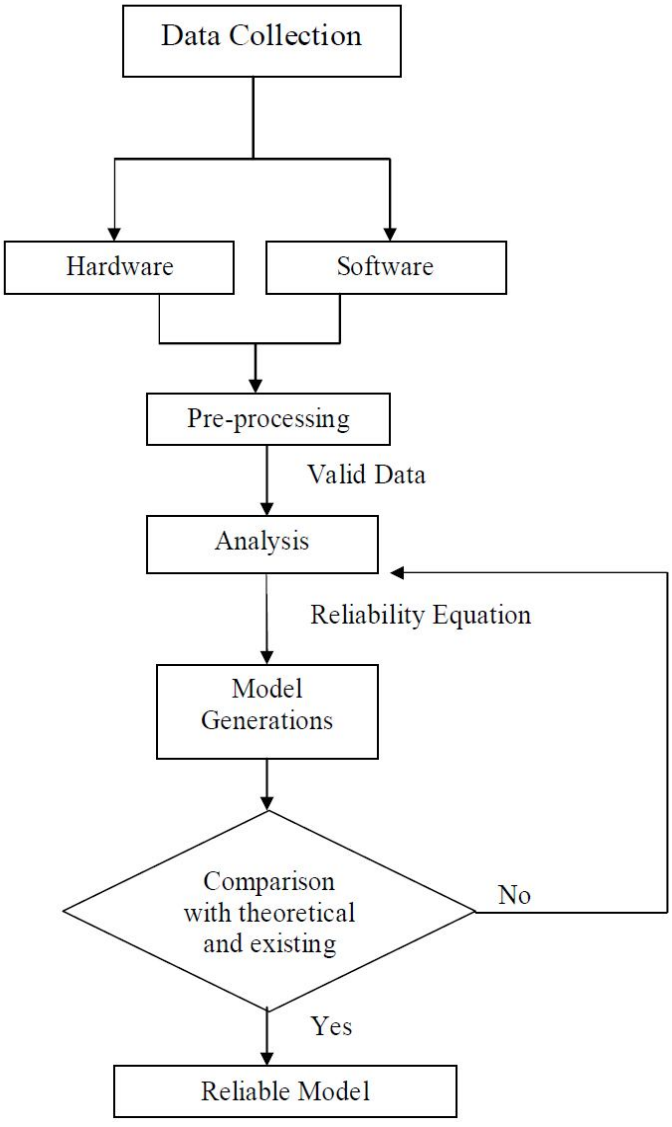


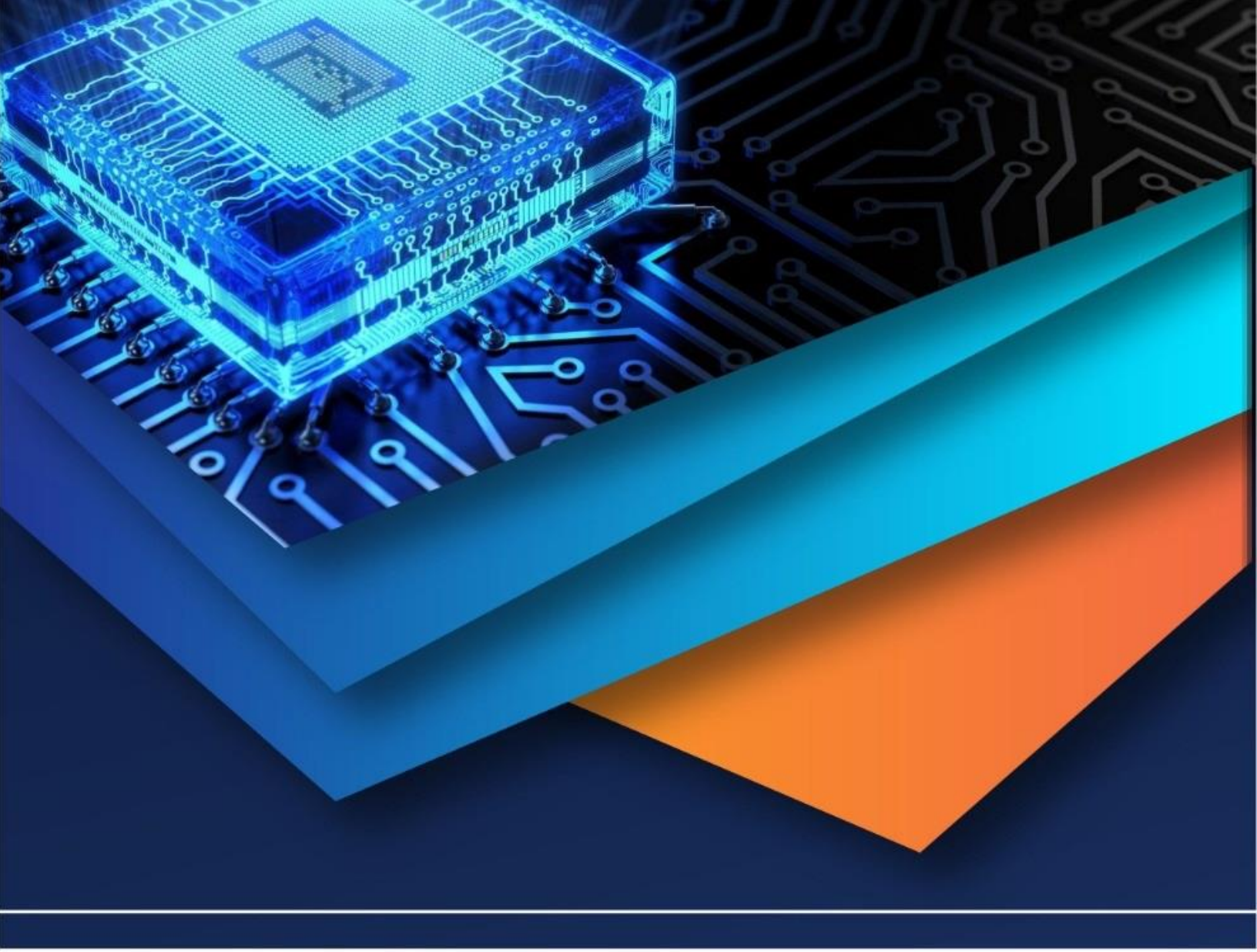
Figure 1: Proposed Method

IV. CONCLUSION AND FUTURE WORK

The paper has carefully reviewed some of the significant works in reliability estimation of softwares. The survey inturn was helpful in proposing a system which is aimed at arriving at a model for reliability by combining hardware and software reliabilities. The methodology involves the analysis of software reliability by collecting bug reports of software packages and analyzing the failure rate thereby evaluating its reliability. The reliability of the hardware part is obtained using the collected component failure data by employing a constant hazard model. Finally, the hardware and software reliabilities are integrated to arrive at the overall system reliability.The proposed method is expected to shine over the existing methods due to the clear advantage of combining both hardware and software inputs. A future extension of this work is by incorporating the possibilities of open source softwares in determining the reliability of the system over the existing ones. The required data necessary for conducting the experiments also will be readily available for open source softwares. The help from online communities can also be a crucial motivating factor on pursuing a work on the same.

REFERENCES

[1] Norman Schneidewind,'Tutorial on Hardware and Software Reliability, Maintainability, and Availability" 2008,IEEE.
[2] Musa J.D "Meassurement and Management of Software Reliability" proceedings of the IEEE, VOL. 68, NO. 9, September 1980.
[3] "IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software", 1998
[4] IEEE Std 1413-2010, IEEE Standard Framework for Reliability Prediction of Hardware.
[5] Moranda.P.L. and Jelinski. Z.,Final Report on Software Reliability Study, McDonnell Douglas Astronautics Company, MADC Report Number 63921,1972.
[6] Moranda P.B, "Software Reliability Predictions" Proceedings of the Sixth Triennia World Congress of the International Federation of Automatic Control, 1975, pp 342-347.
[7] M.L. Shooman, "Probabilistic models for software reliability prediction", in Statistical Computer Performance Evaluation, W. Freidberger, Ed., New York: Academic Press, 1972, pp. 485-502.
[8] ShoomanM.L,"Operational Testing and Software Reliability Estimation During Program Developments," Record of 1973 IEEE Symposium on Computer Software Reliability, IEEE Computer Society, New York, 1973, pp. 51-57.
[9] Shooman M.L "Structural Models for Software reliability Prediction," Proceedings of the 2nd International Conference on Software Engineering, IEEE, Computer Society, New York, October 1976.
[10] Shooman M.L. "Spectre of Software Reliability and its Exorcism," Proceedings of the 1977 Joint Automatic Control Conference , IEEE, New York, 1977, pp-225-231.
[11] Coutinho J. de S, "Software Reliability Growth ,"IEEE Symposium on Computer Software Reliability, 1973.
[12] Joe Palma, Jeff Tian and Peng L u. Collecting Data for Software Reliability Analysis and Modeling. Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering, Vol1, 1993, pp 483-494
[13] Lakey, Peter and Neufelder, Ann Marie, "System and Software Reliability Assurance Notebook," Rome Laboratory Report, Griffiss Air Force Base, Rome NY, 1997. http://www.cs.colostate.edu/~cs530/rh/
[14] Musa J.D. and Okumoto K. "A Logarithamic Poisson Execution Time Model for Software Reliability Measurement,"Proceedings Seventh International Conference on Software Engineering, Orlando, Florida,1983, PP.230-238.
[15] Shooman, M. I. (1968) Probabilistic reliability: an engineering approach. McGraw Hill, New York.
[16] Thomas, W. C. (1973) Modeling the bathtub curve. Proceedings of the Annual Reliability and Maintainability Symposium.
[17] Franz Brosch, HeikoKoziolek "Architecture-Based Reliability Prediction with the Palladio Component Model" IEEE Transactions on Software Engineering, Vol. 38, No. 6, November/December 2012.
[18] Pham H. "System Software Reliability" http://www.springer.com, , XIV,440 p.63 illus., Hardcover ISBN, 2007.
[19] Myron Hecht, Karen Owens, Joanne Tagami "Reliability-Related Requirements in Software-Intensive Systems" 2007 IEEE.
[20] Musa J.D "Meassurement and Management of Software Reliability"proceedings of the IEEE, VOL. 68, NO. 9, September 1980.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY