ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

Reliability Estimation of Open Source Software based Computational Systems

Dr. Shelbi Joseph[1], Akhil P V[2], Seetha Parameswaran[3]

Assistant Professor, Dept. of I.T., School of Engineering, CUSAT, Kerala, India[1,2,3]

ABSTRACT: Software systems are progressively being deployed in many facets of human life. The implication of failure of such systems has assorted impact in its customers. The fundamental aspect that supports a software system is focus on quality. Reliability describes the ability of 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 reliability and software reliability. Most of the earlier works were merely focused on software reliability with no consideration for hardware part or vice versa. However, a complete estimation of reliability of a computing system requires that these two elements (hardware and software) be considered together and thus demands a combined approach. The present work focuses on this and presents a model for evaluating reliability of a computing system. The method involves identifying the failure data of hardware components, software components and building a model based on it to predict the reliability. To develop reliability model, focus is given to Open Source Software since there is an increasing trend towards the use of it and only a few studies were done for the modeling and measurement of the reliability of such products.

KEYWORDS: Failure rate, hardware reliability, mean time between failures, open source software, software reliability

I. INTRODUCTION

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. Many successful software products were developed following Free and Open Source Software Development (FOSS) methodology. The development model used in this scenario is entirely different from traditional software development (closed source). So, the reliability models developed for closed source software cannot be used for FOSS, moreover, the dearth of valuable models call for studies in this area. This study tries to put forward a model for the estimation of the reliability of a computer system, by integrating hardware and software components, especially FOSS.

Considering software based product its reliability is an aggregate of the reliability of software and the underlying hardware. Research in this area considers hardware and software reliabilities separately. This study tries to integrate the hardware and software reliability models to develop a total reliability model of software based system.

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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

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.

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.

D. OPEN SOURCE SOFTWARE

Open source development is an area where people develop and distribute their products by downloading free source code available under a license. Free and Open Source Software (FOSS) refers to those categories of software products that allow users to use, modify and redistribute the software without the need to pay a royalty fee to the author of the product[www.gnu.org, www.opensource.org]. FOSS product includes both system and application software like GNU/Linux, Apache Web Server, Postgresql, OpenOffice, Gimp, OrangeHRM etc. [4]. Universities and colleges spend a huge sum on laboratories and software. The huge software installation costs can be cut down by using FOSS alternatives, instead of proprietary software [www.osalt.com].

II. RELATED WORK

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]). The basic approach is to model past failure data to predict future behavior. 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 [10]. 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 [11], [12]). 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

A systematic frame work designed to predict software reliability from software engineering measures was summarized as follows [13], [14]. Research activities in software reliability engineering have been conducted over the past two decades and many Software Reliability Growth Models (SRGMs) have been proposed for the estimation of software reliability and number of faults remaining in the software [15], [16], [17], [18], [19], [20]. Most of the SRGMs assume that each time a failure occurs, the error which caused it, is immediately removed and no new errors are introduced [21]. Most models make some assumptions about the software failure process so that the model becomes mathematically tractable [22], [23] has given the typical assumptions made in Software Reliability Model with its limitations. Reliability models have been proposed by Goel [15], [24], [25], [26], and [7]. To employ a model for reliability prediction, value of some of the parameters need to be specified. These are typically determined by analyzing the past failure data of the software.

The J-M model proposed by Jelinski and Moranda [24] is one of the simplest and earliest of the software reliability models. The J-M model assumes that times between failures are independent random variables following exponential distributions, there are finite number of faults at the beginning of the test phase, and that the failure rate is uniform between successive failures and is proportional to the current error content(number of faults remaining) of the program being tested. This model is very simple to use. It is also fairly accurate for some data sets, but sometimes leads to inaccurate predictions. [22]

The basic execution model proposed by Musa et. al.[27] make assumptions similar to the above model except that the process modeled is the number of failures in specified execution time intervals. There are a finite number of faults in the beginning of the test phase, and the times between failures are exponential, the failure rate being uniform between successive failures. He also provides a systematic approach for converting the model so that it can be applicable for the calendar time as well.

The Goel and Okumoto (G-O) model [15] considers the software failure process as a Non Homogeneous Poisson Process (NHPP) with a mean function μ(t). This model treats initial error contents as a random variable.

The OSS development mainly depends on the practice of welcoming every enthusiastic individual who would like to contribute to the project. On top of this, the freedom of using, modifying and distributing OSS leads to more robust software and more diverse business models [28]. Software reliability models are useful to assess the reliability for quality management and testing progress control of software development. Although open source practices have been remarkably successful in recent years, the open source development model faces a number of product quality challenges. Rare open source projects have been archived successfully as a high level quality end product. However, these mature and successful projects face quality problems too. Even though lots of models and tools have been suggested for reliability checking, very few models are applied and tested in this case.

Luyin and Sebastian [29] discussed how quality assurance activities are performed within the OSS development. They pointed out that OSS development is very different from the traditional software development used in most of the software industry. Moreover, the quality assurance activities are also performed in a different fashion. Martin et.al. [30] have done exploratory interviews with free and open source developers to study the common quality practices among the developers to implement a quality process improvement strategy. They found that even though development of OSS projects share common practices the quality of the resulting products needs further empirical evaluation. This implies that we have to look into reliability models for open source software development.

III. SCOPE OF THE RESEARCH

The existing reliability models have been studied and an attempt has been made to develop a new methodology towards measuring and improving software reliability. Existing reliability models do not address reliability aspects in all stages of software development. It is difficult to predict the reliability of the end product. The proposed work's main objective is to develop a model to represent reliability of a computing system by considering both hardware and software failure impacts. More specifically, the objectives of the research are as follows:

1) To study the existing reliability models
2) To study and analyse the role of Free and Open Source Software (FOSS) in different communities.
3) To study and evaluate existing open source software and to arrive at a reliability growth model.
4) To develop a model for estimation of computational reliability by incorporating both software and hardware components.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

5) To evaluate and compare the actual reliability with the developed model and other existing models.

IV. PROPOSED METHOD

A. METHODOLOGY FOR MODEL DEVELOPMENT

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 preprocessing and analysis. The formulation of the reliability model involves algorithm development, model development, and comparison of theoretical products with the formulated model. 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 analyzed 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.

B. ANALYSIS PHASE

For reliability analysis of hardware and software package, information regarding its failure has to be studied. So, this phase can be further broken down into data collection and analysis. The former will deal with collecting parameters essential to evaluate its reliability, and the latter will deal with the evaluation and analysis.

C. DATA COLLECTION

The reliability of software is adversely affected by failures or bugs in computer programs. As a first step towards building a reliability model, failure reports were collected to evaluate the reliability of a software package due to occurrences of bugs. The data required was the time of identification of a bug and time of repairing the same, so that the total failure duration could be obtained. From the failure duration the rate of failure could be calculated. In the case of hardware, failure data are collected from the production system of Cochin University of Science and Technology (CUSAT) where Debian based server system were used to run their website, mail server etc.

The bug reports were collected mainly from online open source development site Debian.org. Debian GNU /Linux is a free operating system that comprises of 25000 packages, precompiled in a user-friendly format. It is developed through distributed development all around the world. The Debian GNU/Linux distribution has a bug tracking system which consists of bugs reported by users and developers. This facility was used as the primary data source.

D. DATA PREPROCESSING

Data preprocessing is an important step to refine the collected data. Generally the real world data is incomplete, lacking attribute values, lacking certain attributes of interest, or containing only aggregate data or may be Noisy: containing errors or outliers or can be Inconsistent: containing discrepancies in codes or names. Data preprocessing includes data cleaning, data integration, data transformation, data reduction and data discretization. Data cleaning usually includes fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies, whereas, data integration is carried out by using multiple databases, data cubes, or files. Data transformation is the normalization and aggregation process. Data reduction is reducing the volume but producing the same or similar analytical results and Data discretization is part of data reduction by replacing numerical attributes with nominal ones.

E. DATA ANALYSIS AND INTERPRETATION

The collected data is analyzed in order to arrive at reliability. The software bug arrival is plotted against time and the failure function is derived. This function is used for arriving at the software reliability. In the case of hardware the mean time to failure for each of the components is sufficient to arrive at the reliability based on the constant hazard model.

In the case of the software a total of 1880 packages were available at the start of the analysis, as per the details available from the official website of Debain. This is taken as the initial population. A time interval of one month is

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

fixed and the bug arrival rate during this interval is noted. The observations are taken for 1 year after which the bug arrival is negligible indicating that the software has more or less stabilized.

F. *ALGORITHM DEVELOPMENT*

A systematic procedure for evaluating reliability of open source software is developed by considering the prevailing trends in industry. The methodology involves defining an equation for the pattern of failure based on the available bug arrival rate and developing a generalized model for the reliability of the software.

Algorithm. Reliability analysis
Input: **Total Initial Population**
Output: **Expression for Reliability**

Initialize the total population, ***P***
Define a time interval **T<t1, t2>**
for each time interval $\mathbf{T_i}$
Obtain the bugs $b_i \in \{t_1, t_2\}$

Calculate Total failure $T_f = \sum_{i=1}^{n} b_i$

Cumulative failure $C_f = number\{k \mid x_k \le t_2\} = \sum_{k=1}^{n} I(x_k \le t_2)$

where I is the indicator function and $x_1, \ldots\ldots x_n$ are observed data

Survivor $S_i = T_f - C_f^{\,i}$ where $i = 1 to 'n' \in T$

Failure rate $F_{r_i} = \frac{b_i}{Avg\{C_f(S_i)\}}$

End

Obtain the equation of relation between failure rate and time using regression analysis

Obtain the expression of reliability of the software using $R(t) = e^{-\int_0^t z(t)\,dt}$

G. *MODEL DEVELOPMENT*

The reliability estimation involves considering the computing system as two subsystems, one comprising of hardware components and the other, the various software modules or packages. These various packages are considered as various components of the software part of the system [http://www.debian.org]. The software and hardware components are assumed to be connected in series and based on the reliability block diagram, the overall system reliability is obtained.

H. *COMPARISON OF DEVELOPED MODEL WITH OTHER EXISTING MODELS*

The developed model arrives at the reliability by combining software and hardware parameters. A comparison is made with the other existing reliability models so as to arrive at the error involved in reliability analysis when computation of reliability is calculated without considering the software and hardware elements together.

V. EXPERIMENTAL RESULTS

The open source software data is made use of and the methodology for evaluating the software reliability involves identifying a fixed number of packages at the start of the time and defining the failure rate based on the failure data for these preset number of packages. The defined function of the failure rate is used to arrive at the software reliability model. The hardware reliability is obtained using constant hazard model.

A total of 1880 packages were available at the start of the analysis as per the details available from the official website of Debain. This is taken as the initial population. A time interval of 1 month is fixed and the bug arrival rate during this interval is noted. The observations are taken for 1 year after which the bug arrival is negligible indicating that the software has more or less stabilized.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

The failure model corresponding to the failure rate can be expressed as:

$$Z(t) = -at + b \qquad \text{eq.(1)}$$

where a=0.0004 and b = 0.078.

The corresponding reliability can be expressed as:

$$R_{Software} = e^{-\int_0^t (-0.0004t + 0.078)dt}$$

$$= e^{\frac{0.0004t^2}{2} - 0.078t} \qquad \text{eq. (2)}$$

Table 1. Hardware component failure rate

Hardware Component No.	Failure Rate (per month)
H_1	0.027778
H_2	0.025
H_3	0.028571
H_4	0.02381
H_5	0.016667
H_6	0.041667
H_7	0.034483
H_8	0.027778

The failure rate of the hardware components are indicated in Table 1. The values of the hardware failure rate can be substituted in equation 2 to get the hardware reliability equation and can be expressed as:

$$R_{hardware} = e^{-\sum_{i=1}^{n} \lambda_{h_i} t}$$

$$= e^{-0.225753t} \qquad \text{eq. (3)}$$

Thus the reliability of the computing system R(t) at any given time will be the product of equations 2 and 3 and can be expressed as:

$$R(t) = e^{\frac{0.0004t^2}{2} - 0.078t} \times e^{-0.225753t}$$

$$= e^{\frac{0.0004t^2}{2} - 0.303753t} \qquad \text{eq. (4)}$$

The reliability of the software at different points in time is calculated using the equation (2). The actual values of reliability obtained by dividing the survivors at the given point in time by the initial population are also calculated. The Musa model assumes a constant value for the failure rate and by considering this as the average value of failure rates the reliability values are calculated using the equation

$$R(t) = e^{-\lambda t} \qquad (5)$$

The reliability values for Weibull distribution is calculated using the equation

$$R(t) = e^{-(t/\alpha)^{\beta}} \qquad (6)$$

The reliability values calculated using the four different methods and the failure density values are shown in table 2.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

Table 2. Reliability and failure density

Time	Failure Density	Reliability (Actual)	Reliability (Musa)	Reliability (Weibull)	Reliability (Model)
8-Feb		1.00000	1.00000	1.00000	1.00000
	0.013297872				
8-Mar		0.98670	0.85985	0.98144	0.92515
	0.032446809				
8-Apr		0.95426	0.73934	0.92263	0.85624
	0.180851064				
8-May		0.77340	0.63572	0.82780	0.79279
	0.02606383				
8-Jun		0.74734	0.54662	0.70739	0.73433
	0.029255319				
8-Jul		0.71809	0.47001	0.57487	0.68045
	0.113829787				
8-Aug		0.60426	0.40414	0.44377	0.63078
	0.072340426				
8-Sep		0.53191	0.34750	0.32507	0.58497
	0.019680851				
8-Oct		0.51223	0.29879	0.22577	0.54270
	0.021276596				
8-Nov		0.49096	0.25692	0.14856	0.50369
	0.025531915				
8-Dec		0.46543	0.22091	0.09256	0.46767

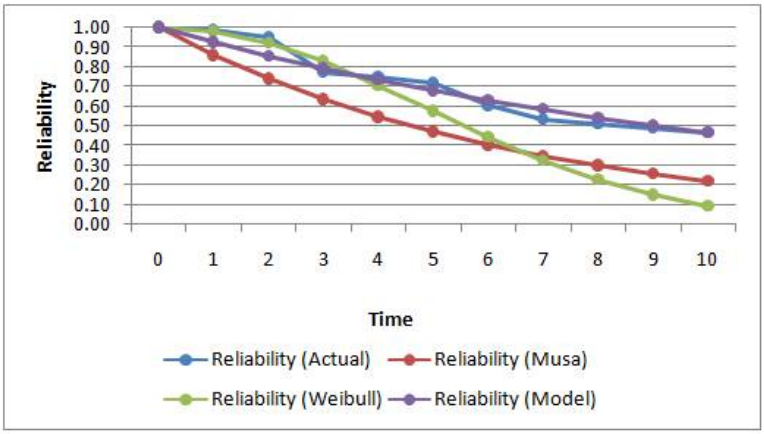

Fig 1. Comparison of reliability obtained using different models

Figure 1 shows a comparison of reliability obtained using the Developed, Simplified model, Weibull and Musa model with the actual reliability values. It can be seen that the simplified model shows a better result compared with other models. Further, these two models very closely approximate the real situation.

VI. CONCLUSION AND FUTURE WORK

A new method for estimation of reliability of computational systems was developed. The methodology is far more realistic in comparison with the traditional methods which focus either on hardware or software alone rather than integrating these elements. The concept is very much in accordance with the systems approach. The developed method could prove to be a very effective tool for reliability analysis of computational systems. An error analysis was also conducted and it can be seen that if the hardware and software components are not integrated in reliability analysis the

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 2, February 2017

calculated values will be an over estimated one. That is, the calculated values would be much higher than the actual reliability values. A comparison of reliability obtained using the developed model, Weibull and Musa model with the actual reliability values are also shown.

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. Smrithy V. , Rekha, V.Adinarayanan,Anurag Maherchandani,Sneha Aswani, "Bridging the Computer Science Skill Gap with Free and Open Source Software" International Conference on Engineering Education (ICEED 2009),Kuala Lumpur, Malaysia,2009 IEEE.
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. Shooman M.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. 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
11. 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/
12. Musa J.D. and Okumoto K. "A Logarithamic Poisson Execution Time Model for Software Relibility Meassurement,"Proceedings Seventh International Conference on Software Engineering, Orlando, Florida,1983, PP.230-238.
13. Smidts C and M.Li, "Software Engineering Measures for Predicting Software Reliability in Safety Critical Digital Systems, "University of Maryland, Washington D.C.NUREG/GR-0019, November 2000.
14. Li M. and C.Smidts,' A Ranking of Software Engineering Measures based on Expert Opinion," IEEE Transactions on Software Engineering, vol. 29,pp, 811-24, 2003.
15. Goel A.L. and K.Okumoto, "Time-Dependent Error-Detection Rate Model for software and other performance measures," IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979.
16. Hossain S.A and R.C.Dhahiya, "Estimating the parameters of a Non-Homogeneous Poisson Process Model for software Reliability Model for Software Reliability," IEEE Trans. Reliability, vol.42, pp. 604-612, 1993.
17. Leung Y.W., "Optimal Software Release Time with a given Cost Budget," J. Systems and Software, vol. 17. Pp. 23-242, 1992.
18. Ohba M., "Software Reliability Analysis Models," IBM J. Research and Development, vol 28, pp. 428-443, 1984.
19. Pham H, "Software Reliability Assessment: Imperfect Debugging and Multiple Failure Types in Software Development,"EG&GRAAM-10737, Idaho Nat'1 Eng. Laboratory,1993.
20. Yamada S. and S.Osaki, "Software Reliability Growth Modeling: Models and Applications," IEEE Trans. Software Eng., vol. 11, pp.1,431-1,437,1985.
21. Hoang Pham, Xuemei Zhang, "A Software Cost Model with Warranty and Risk Costs,"IEEE Transactions on Computers, vol. 48, No.1, January 1999.
22. Pankaj Jalote, and Rajib Ghosh, "An Approach for Cost Effectiveness Analysis of Multiversion Software Using Software Reliability Models" http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.3640.,2011
23. Goel .A.L, "Software reliability models: Assumptions,limitations, and applicability", IEEE Trans. Software Engineering,vol SE-11, num 12, 1985, pp 1411 - 1423.
24. Jelinski Z. and P. Moranda, Software reliability research", In Statistical Computer performance evaluation , W. Frieberger, Ed., New York: Academic, 1972, pp. 465-484.
25. Littlewood B. and J.L. Verrall, A Bayesian Reliability Growth Model for Computer Software, J. Royal Statist. Soc., C (Applied Statistics), Vol. 2, pp 332-346, 1973.
26. Musa J.D. and K. Okumoto, A Logarithamic Execution time model for software reliability measurement, Proc. 7th International conference on Software Engg., Orlando, Florida, march 26-29, pp 230-238, 1984.
27. Musa J.D, A. Iannino, K. Okumoto, Software Reliability,1987; McGraw-Hill.
28. Ming-Wei Wu, Ying Dar – Lin, Open Source Software Development: An Overview. Computer, 34(6) 33 - 38, June 2001.
29. Luyin Zhao and Sebastian Elbaum A Survey On Quality Related Activities in Open Source; Software Engineering Notes Vol 25 no 3: 54-57 May 2000.
30. MartinMichlmayr, FrancisHunt, DavidProbert Quality Practices and Problems in Free Software Projects, Proceedings of the First International Conference on Open Source Systems Genova, pp 24-28, July 2005
31. Leblanc S. P. , P.A.Roman, "Reliability Estimation of Hierarchical Software Systems.", 2002 Proceedings annual reliability and maintainability symposium.