A Simplified Model for Evaluating Software Reliability at the Developmental Stage

Shelbi Joseph achayanshelbil@gmail.com
Division of Information Technology
School of Engineering
Cochin University of Science and Technology
Cochin, India

Shouri P.V pvshouri@gmail.com
Department of Mechanical Engineering
Model Engineering College
Cochin, India

Jagathy Raj V. P jagathy@cusat.ac.in
School of Management Studies,
Cochin University of Science and Technology
Cochin, India

Abstract

The use of open source software is becoming more and more predominant and it is important that the reliability of this software are evaluated. Even though a lot of researchers have tried to establish the failure pattern of different packages a deterministic model for evaluating reliability is not yet developed. The present work details a simplified model for evaluating the reliability of the open source software based on the available failure data. The methodology 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 reliability model. The reliability values obtained using the developed model are also compared with the exact reliability values.

Key words: Bugs, Failure Density, Failure Rate, Open Source Software, Reliability

1. INTRODUCTION

Open Source Software (OSS) has attracted significant attention in recent years [1]. It is being accepted as a viable alternative to commercial software [2]. OSS in general refers to any software whose source code is freely available for distribution [3]. However the OSS development approach is still not fully understood [4]. Reliability estimation plays a vital role during the developmental phase of the open source software. In fact, once the package has stabilized (or developed) then chances of further failure are relatively low and package will be more or less reliable. However, during the developmental stage failures or bug arrival are more frequent and it is important that a model has to be developed to evaluate the reliability during this period. The bug arrivals usually peak at the code inspection phase and get rather stabilized in the system test phase [5]. Software reliability evaluation is an increasingly important aspect of software development process [6].

Reliability can be defined as the probability of failure free operation of a computer program in a specified environment for a specified period of time [4,5]. It is evident from the definition that there are four key elements associated with the reliability namely element of probability, function of the product, environmental conditions, and time.

Reliability is nothing but the probability of success. As success and failure are complementary, a measure of the failure is essential to arrive at the reliability. That is,

$$Reliability = Probability\ of\ success = 1 - Probability\ of\ failure \quad (1)$$

From equation (1), it is evident that the first step in reliability analysis is failure data analysis. This involves fixing up a time interval and noting down the failures at different time intervals. The number of packages at the start of the analysis is defined as the initial population and the survivors at any point of time is the difference of initial population and the failures that have occurred till this point. Failure rate associated with a time interval can be defined as the ratio of number of bugs reported during the

NOMENCLATURE

$f_d(t)$	failure density
N	initial population
R(t)	reliability
t	time
Z(t)	failure rate
λ	constant failure rate

given time interval to the average population associated with the time interval. Once the variation of failure rate with respect to time can be established an equation can be used to fit the variation which will be the failure model for reliability estimation. Typical reliability models include Jelinski-Moranda [6], Littlewood [7] , Goel-Okumoto [8], Nelson model [9], Mills model [10],Basin model [10],Halstead model [11] and Musa-Okumoto [4]. For software projects that have not been in operation long enough, the failure data collected may not be sufficient to provide a decent picture of software quality, which may lead to anomalous reliability estimates [12, 13] Weibull function is also used for reliability analysis and the function has been particularly valuable for situations for which the data samples are relatively small [14].

Concern about software reliability has been around for a long time [15,16] and as open source is a relatively novel software development approach differing significantly from proprietary software waterfall model, we do not yet have any mature or stable technique to assess open source software reliability [17].

It is clear from the above discussions that even though a variety of models are available for reliability prediction, a deterministic model is presently not available. Or in other words, none of these models quantifies reliability. The present work focuses on development of an algorithm and there by a simplified method of quantifying reliability of a software.

2. MODEL DEVELOPMENT AND ALGORITHM

An open source program typically consists of multiple modules [18]. Attributes of the reliability models have been usually defined with respect to time with four general ways to characterize [19, 20] reliability, time of failure, time interval between failures, cumulative number of faults upto a period of time and failure found in a time interval. The present 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. The following are the assumptions involved in the analysis.

1. The software analyzed is an open source.
2. As the open source software is made up of a very large community the environmental changes are not considered.
3. The total number of packages at the beginning of the analysis is assumed to remain constant and is taken as the initial population.
4. The failures of various packages are assumed to be independent of each other.
5. The model is developed for evaluation of the software reliability at the developmental stage and the packages that fail during this period are not further considered. It is further assumed that by the end of

developmental stage the bug associated with the failed packages would be eliminated and will be stable further.

6. The reliability of the software is inversely proportional to the number of bugs reported at any point of time.
7. The beginning of the time period after which the bug arrival or failure rate remains constant marks the culmination of the developmental stage and the software will be stabilize.

Based on the above assumptions a 6- step algorithm is developed for the analysis as detailed below.

1. Indentify the total initial population. This corresponds to the total number of packages existing at the beginning of the time period. That is, at the start of analysis.
2. Define a time period and find out the bugs reported during this time interval. As the failure would have occurred anywhere between the time interval, the reported failures are indicated in between the time interval.
3. Calculate the cumulative failures and thereby the survivors and different points in time.
4. Estimate the failure rate associated with the time intervals by dividing the number of failures associated with the given unit time interval by average population associated with the time interval. Average population associated with a given time interval is the average of survivors at the beginning and end of the time period.
5. Plot the graphs defining the relation between failure rate and time and obtain the equation defining the relation between failure rate and time.
6. Obtain the expression for reliability of the software by substituting the equation of failure rate in equation(1) given as

$$R(t) = e^{-\int_0^t Z(t)dt} \quad (1)$$

3. RESULTS AND DISCUSSION

A total of 1880 packages were available at the start of the analysis as per the details available from the official website of Debian [21]. 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 reported errors at different time intervals are given in the Table 1. The observations are taken for 1 year after which the bug arrival is negligible indicating that the software has more or less stabilized.

Time	No. of Failures	Cumulative Failures	Survivors	Failure Rate
Feb-08		0	1880	
	25			0.013386881
Mar-08		25	1855	
	61			0.033433817
April-08		86	1794	
	340			0.209359606
May-08		426	1454	
	49			0.034277719
Jun-08		475	1405	
	55			0.039927405
Jul-08		530	1350	
	214			0.172164119
Aug-08		744	1136	
	136			0.127340824
Sept-08		880	1000	
	37			0.037697402
Oct-08		917	963	
	40			0.042417815
Nov-08		957	923	
	48			0.048929664
Dec-08		1005	875	
			Average	0.075893525

TABLE 1: Failure Data Analysis

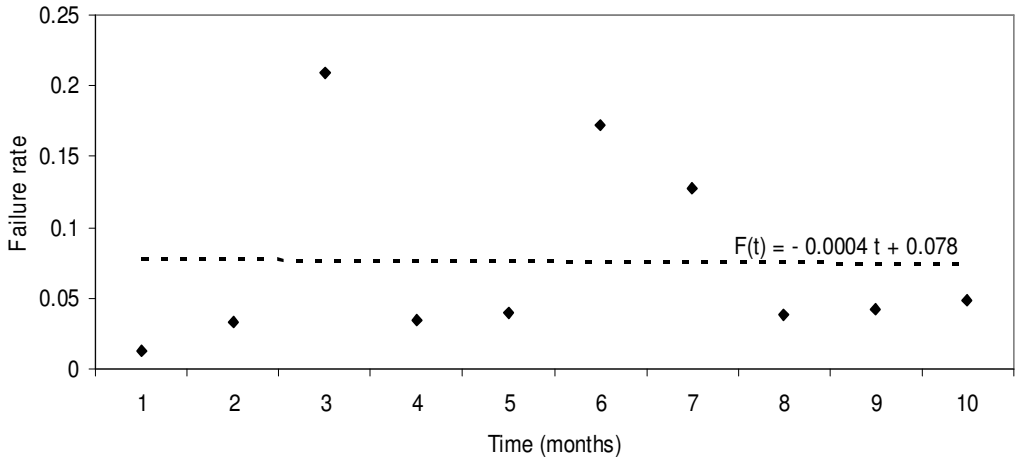

FIGURE 1. Variation of failure rate with time

The variation of failure rate with respect to time is shown in Fig. 1. It can be seen that after the 8^{th} month onwards the software has somewhat stabilized indicating the completion of developmental phase. The failure model corresponding to the failure rate can be expressed by the equation (2)

$$Z(t) = -0.0004t + 0.078 \tag{2}$$

The corresponding reliability can be expressed by the equation (3) as

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

$$\text{That is, } R(t) = e^{\frac{0.0004t^2}{2} - 0.078t} \tag{3}$$

Failure density associated with the time intervals is the ratio of number of failures associated with the given unit time interval to the initial population. Failure density can be related with Reliability and failure rate using the equation (4) as

$$f_d(t) = R(t) \times Z(t) \tag{4}$$

Therefore, based on the developed model failure density can be expressed as

$$f_d(t) = e^{\frac{0.0004t^2}{2} - 0.078t} \times (-0.0004t + 0.078) \tag{5}$$

The reliability of the software at different points in time is calculated using the equation (3). 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

$$R(t) = e^{-\lambda t} \tag{6}$$

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

Time	Failure Density	Reliability (Actual)	Reliability (Musa)	Reliability (Simplified Model))
Feb-08		1	1	1
	0.013297872			
Mar-08		0.98670	0.92691	0.92496
	0.032446809			
April-08		0.95426	0.85917	0.85556
	0.180851064			
May-08		0.77340	0.79638	0.79136
	0.02606383			
Jun-08		0.74734	0.73818	0.73198
	0.029255319			
Jul-08		0.71809	0.68423	0.67706
	0.113829787			
Aug-08		0.60427	0.63422	0.62625
	0.072340426			
Sept-08		0.53191	0.58787	0.57926
	0.019680851			
Oct-08		0.51223	0.54490	0.5358
	0.021276596			
Nov-08		0.49096	0.50508	0.49559
	0.025531915			
Dec - 08		0.46543	0.46816	0.45841

TABLE2. Reliability and failure density

Fig. 2 shows a comparison of reliability obtained using the developed simplified model and Musa model with the actual reliability values. It can be seen that the simplified model and the Musa model nearly provides the same results. Further, these two models very closely approximate the real situation. The variation of failure density with time is also shown in Fig. 3

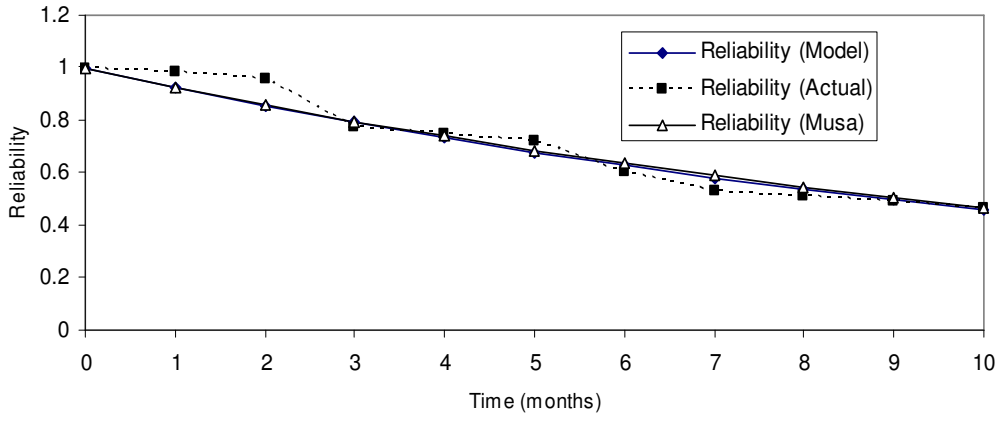

FIGURE2. Comparison of reliability obtained using different models

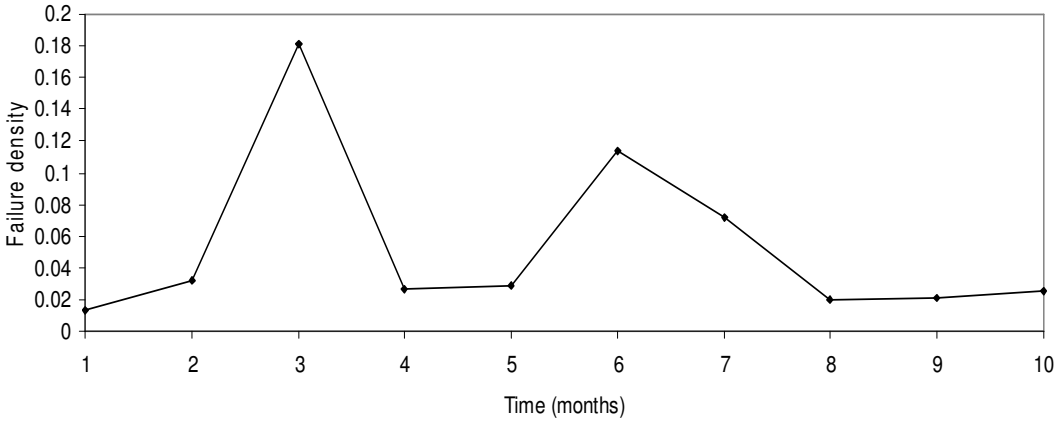

FIGURE 3. Variation of Failure density with time

Fig 4 compares the reliability value obtained using the model with the theoretical value. It can be seen that the percentage error is always within 10% of the actual value which is a reasonably good result for all engineering problems.

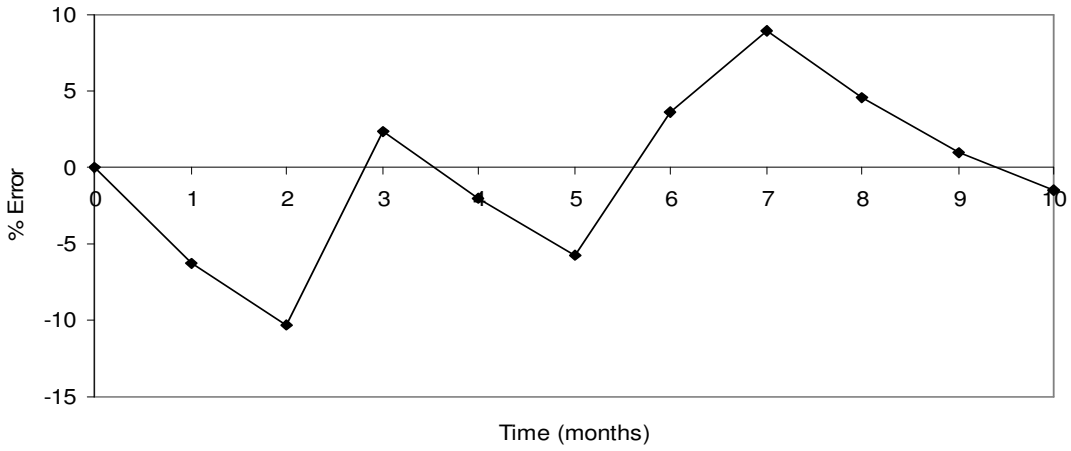

FIGURE 4. Error Analysis

4. CONCLUSION

A simplified model for evaluation of software reliability was presented. This is a relatively simplified and a totally new method of analysis of software reliability as the initial population is assumed to remain constant. The method provides fairly good results and the related errors are negligible. It is hoped that this model will prove to be a powerful tool for software reliability analysis.

5. REFERENCES

1. Ying ZHOU,Joseph Davis. Open Source Software reliability model: an empirical approach, ACM 2005

2. Sharifah Mashita Syed-Mohamad,Tom McBride. A comparison of the Reliability Growth of Open Source and In-House Software. 2008 IEEE 15th Asia-Pacific Software Engineering Conference

3. Cobra Rahmani,Harvey Siy, Azad Azadmanesh. An Experimental Analysis of Open Source Software Reliability. Department of Defense/Air Force Office of Scientific Research

4. Lars M.Karg, Michael Grottke, Arne Beckhausa . Conformance Quality and Failure Costs in the Software Industry: An Empirical Analysis of Open Source Software. 2009 IEEE

5. Kan H.S. Metrics and models in software quality engineering, 2nd edition, Addison-Wesley(2003)

6. S.P. Leblanc, P.A.Roman Reliability Estimation of Hierarchical Software Systems. 2002 Proceedings annual reliability and maintainability symposium

7. J D Musa and K. Okumoto. A logarithamic Poisson execution time model for software reliability measurement 7th international conference on Software Engineering(ICSE),1984, pp. 230-238

8. H. Pham, Software Reliability Spriner-Veriag , 2000

9. E.C. Nelson, A Stastistical Basis for Software Reliability Assessment, TRW-SS-73-03. 1973.

10. ShaoPing Wang, Software Engineering. BEIJING BUAA PRESS.

11. M.H.Halstead, Elements of Software Science, North Holand, 1977.

12. Z. Jelinski and P.B. Moranda, Software Reliability research, in Stastistical Computer Performance Evaluation, W. Freiberger, Ed. New York: Academic Press, 1972, pp.465-484

13. B. Littlewood and J.L. Verrall, A Bayesian reliability growth model for computer software, Applied Stastistics, Vol 22, 1973, pp 332-346

14. A. L. Goel and K. Okumoto, A time-dependent error-detection rate model for software reliability and other performance measure, IEEE Transactions on Reliability, vol. R-28, 1979, pp. 206-211.

15. Adalberto Nobiato Crespo, Alberto Pasquini ,"Applying Code Coverage Approach to an Infinite Failure Software Reliabilit Model" , 2009 XXIII Brazilian Symposium on Software Engineering, 2009 IEEE.

16. Hudson, A, "Program Error as a British and Death Process", Technical Report SP- 3011, Santa Monica, Cal,:Systems development Corporation, 1967.

17. Fenghong Zou , Joseph Davis "Analysing and Modeling Open Source Software Bug Report Data', 19th Australian Confeence on Software Engineering.,2008 IEEE.

18. Fenghong Zou , Joseph Davis " A Model of Bug Dynamics for Open Source ", The second International Conference on Secure System Integration and Reliability Improvement., 2008 IEEE.

19. Sharifah Mashita Syed-Mohamad, Tom McBride, 'Reliability Growth of Open Source Software using Defect Analysis', 2008 International conference on Computer Science and Software Engineering, 2008 IEEE.

20. Musa, J.D., Iannino, A. and Okumoto, K. (1987),"Software Reliability: Measurement, Prediction, Application', pp. 621.

21. http://www.debian.orgs