

A SOFTWARE RELIABILITY MODEL INCORPORATING FAULT DEPENDENCY CONSIDERING TIME DELAY

Vishal Pradhan¹, Ajay Kumar², JoydipDhar³, Mohini Gupta⁴

¹vishalpradhan999@gmail.com, ²ajay1dma@gmail.com, ³jdhar.iiitmg@gmail.com,

^{1,2,3,4}Department of Applied Sciences

ABV-IIITM Gwalior, MP, India-474015

Abstract

In the digital world, almost every sector of the society is highly dependent on software systems. The tradeoffs between development time and the cost are difficult to balance. In this paper, we introduce fault dependency and various time lags, to predict and quantify the reliability of software in an effective manner. Model parameters are estimated by least square estimation (LSE) method, and the measure of accuracy namely SSE, R-square, Adjusted R-square and RMSE are calculated using a valid dataset. The comparison with previous existing models in literature has also been done.

Keywords: Software reliability, non-homogeneous Poisson process, open source software, fault dependency.

1. Introduction

Software is an important part of numerous basic applications and the size and complexity has been growing dramatically. There has been an exponential increase in the previous couple of decades and the pattern is expected to proceed in future too. This increment in software complexity and size has made it very hard to attain high reliability. One of the real issues today, faced by software industry, is to build reliable software. Software reliability is the likelihood of failure-free programming operation for a predefined period of time in a predetermined situation.

Software reliability growth model (SRGM) is one such part of Software reliability that has gotten the most consideration. According to Software reliability growth models failures are described as random process. SRGM helps in assessing the quantity of initial defects. It also helps the software managers in measuring the level of reliability in testing stage. SRGMs are developed over the time and estimating the reliability of software's [1-8].

Open Source Software (OSS) is a new paradigm to develop software by community, in which group of peoples collaborate each other to develop software. These activities are often called OSS projects. The products of OSS projects include source code, documents of software, and test results. In many OSS projects, their products are published under open-license. Anyone can freely access these OSS products [16]. The usage of open source software has increased rapidly. The role of OSS is becoming great in various domains from research to commercial to educational.

2. Related research

- i. **Optimal version update time for OSS:**

OSS involves much more testers in the testing process and most of these testers are volunteers. More specifically, each release of OSS can attract increasing number of volunteers in the early phase since more and more people know it and use it. After the number of volunteers reaches at the peak, it will decrease since the software is losing its attractiveness over time. Fault-detection rate in OSS follows a hump-shaped curve. So, the Weibull-type fault detection rate function can also capture the property of the hump-shaped curve with fairly good flexibility. The total expected cost including both testing cost and operation cost is a crucial factor for such determination [14].

The cost is no longer an important consideration for the OSS community to decide the release time. For OSS development, there are two important factors for management to determine the optimal version-update time in the testing phase. On one hand, a sufficient number of volunteers are expected to be involved in the development of OSS. On the other hand, reliability is the most important aspect of OSS quality [17]. Since reliability is an increasing function over time, reliable software requires a delay of the release to ensure that there is sufficient time for testing.

ii. Various type of fault assumption in software

Ohba[10] considered that there were two sorts of faults to be specific mutually independent and mutually dependent faults. Mutually independent faults can be identified and remedied instantly. There is no time defer between detection and rectification process. Mutually dependent flaws are not possible to remove immediately. By the bug report information of open source software, it has been inferred that conventional software reliability growth models are not suitable for accessing the reliability growth of open source projects due to the fact that the design and development of open source is not the same as that of closed source [9].

Hung and Lin [18] joined flaw dependencies and time lag while debugging into the exiting SRGM. They studied the problem of suitable release time and update time for software projects based on cost criterion and reliability. The soul of open source programming is the free right of utilizing, replicating, conveying and altering the product, which makes an effective, practical, and profitable. Because of substantial scale platform of Open source software, it is troublesome for a software manager to survey the reliability and quality of software. Keeping up vast variety of software and timely version upgrades is a challenge to many product administrators and managers.

3. Modification in previous models:

i. Considering failure dependency in software fault modeling:

There are two potential kinds of faults one being the dependent fault and the other being the independent fault. The dependent defect detected may not be uprooted promptly and lags the defect detection procedure by a delay or time-lag factor $\phi(t)$. So, $\phi(t)$ is the time gap between the independent and dependent fault removal. Also it is assumed that no new defects are created in the defect removal process.

When the quantity of recognized independent defects is related to the proportion of number of independent defects left in the system, and then the following differential equation is obtained:

$$\frac{dm_1(t)}{dt} = r[a_1 - m_1(t)]$$

Where a_1 is the total number of initial defects, and r is the defect detection rate.

For the removal of dependent defects, it is necessary to remove the independent faults perfectly first.

ii. Failure dependency model with time lag:

All the above models will be integrated to make a model which is more correct and fits to most software. Additionally the dependency of faults will be considered and the rate of removal will be proportional not only to the remaining errors in the system but also to the debugged error so as to provide a clear picture. When a failure (i.e. an unexpected and incorrect behavior of the system) occurs during testing, it is counted with a time tag. Cumulative failures are counted with corresponding cumulative time. SRM is fitted to the collected data and the fitted models are then used to predict the total number of expected defects (i.e. fault on the execution of which failure occur) in the software [18].

On the basis of the above assumptions when we include time delay $\phi(t)$. The equation can be modified as:

$$m_{original}(t) = a(1 - e^{-rt})$$

$$m(t) = m_{original}(t - \phi(t)) = a(1 - e^{(-rt)}e^{r\phi(t)})$$

On the basis of taking three type of $\phi(t)$, $m(t)$ will be

- $\phi(t) = 0$.

$$m(t) = a(1 - e^{-rt})$$

- $\phi(t) = \frac{\ln(1+rt)}{r}$

$$m(t) = a(1 - (1 + rt)e^{-rt})$$

- $\phi(t) = t + \alpha e^{-\beta t} - \alpha$

$$m(t) = a[1 - e^{-r\alpha(1 - e^{-\beta t})}]$$

Instinctively, the rectification procedure can be seen as a learning methodology since the software testing groups will become acquainted with the debugging environment. These groups' skills can be continuously enhanced and accordingly the measure of time slack will be lesser. Now consider the dependency between the dependent faults and independent faults. So $m(t)$ can be written as summation of $m(t)$ of dependent and independent faults [15]. So,

$$m(t) = m_1(t) + m_2(t)$$

The recognized defect flaw may not be uprooted promptly and it slacks the defect detection procedure by a time delay factor $\phi(t)$. Thus $\phi(t)$ is the time postpone between the evacuation of the main flaw and the evacuation of the ward fault(s).

$$\frac{dm_2(t)}{dt} = \theta[a_2 - m_2(t)] \times \frac{m_1(t - \phi(t))}{a}$$

Where, θ is the dependent fault removal rate.

Now including delay with dependency, the equation can be modified according to $\phi(t)$.

So when

- $\phi(t)=0$

$$\frac{dm_2(t)}{dt} = \theta[a_2 - m_2(t)] \times \frac{a_1(1 - e^{-rt})}{a}$$

$$m_2(t) = a_2 \left[1 - e^{\left[\frac{a_1\theta(1-e^{-rt}) - ra_1\theta t}{ar} \right]} \right]$$

Let P is the percentage of leading faults. So $a_1 = Pa$ (where a_1 is independent faults) & $a_2 = (1 - P)a$ (where a_2 is dependent faults).

$$m(t) = a \left[1 - Pe^{-rt} - (1 - P)e^{\left[\frac{P\theta}{r}(1-e^{-rt}) - Pt\theta \right]} \right]$$

- $\phi(t) = \frac{\ln(1+rt)}{r}$

-

$$\frac{dm_2(t)}{dt} = \theta[a_2 - m_2(t)] \times \frac{a_1(1 - (1 + tr)e^{-rt})}{a}$$

$$m_2(t) = a_2 \left[1 - e^{\left[\frac{-a_1\theta(tr+2e^{-rt}+rte^{-rt}-2)}{ar} \right]} \right]$$

$$m(t) = a \left[1 - Pe^{-rt} - (1 - P)e^{\left[\frac{2P\theta}{r}(1-e^{-rt}) - Pt\theta(1+e^{-rt}) \right]} \right]$$

- $\phi(t) = t + \alpha e^{-\beta t^\gamma} - \alpha$

$$\frac{dm_2(t)}{dt} = \theta[a_2 - m_2(t)] \times \frac{a_1 \left(1 - e^{\left(-r\alpha(1-e^{-\beta t^\gamma}) \right)} \right)}{a}$$

When $\gamma = 1$ for Yamada's Weibull-type testing-effort function model.

$$m_2(t) = a_2 \left[1 - e^{\left[\frac{-a_1\theta e^{-r\alpha}(t\beta e^{r\alpha} - \zeta[r\alpha] + \zeta[e^{-t\beta r\alpha}])}{ar} \right]} \right]$$

$$m(t) = a \left[1 - Pe^{[-r\alpha(1-e^{-\beta t})]} - (1 - P)e^{\left[\frac{P\theta e^{-r\alpha}}{\beta}(e^{r\alpha}\beta t - \zeta[r\alpha e^{-\beta t}]) \right]} \right]$$

When $\gamma = 2$

$$m_2(t) = a_2 \left[1 - e^{\left[-\int_x^0 P\theta(-1+e^{\left(-1+r\alpha e^{-\frac{y^2\beta}{2}} \right)})dy + -\int_x^t P\theta(-1+e^{\left(-1+r\alpha e^{-\frac{y^2\beta}{2}} \right)})dy \right]} \right]$$

$$m(t) = a \left[1 - \text{Pe}^{\left[-r\alpha \left(1 - e^{-\frac{\beta t^2}{2}} \right) \right]} \right. \\ \left. - (1 - P) e^{\left[-\int_x^0 P\theta(-1+e^{\left(-1+r\alpha e^{-\frac{y^2\beta}{2}} \right)}) dy + -\int_x^t P\theta(-1+e^{\left(-1+r\alpha e^{-\frac{y^2\beta}{2}} \right)}) dy \right]} \right]$$

Where ζ is $\zeta[z] = -\int_{-z}^{\infty} \frac{e^{-t}}{t} dt$

Now taking the case of imperfect debugging with TEF

$$\frac{dm(t)}{dt} \times \frac{1}{w(t)} = r[a(t) - m(t)]$$

and

$$\frac{dm(t)}{dt} = \eta \times \frac{da(t)}{dt}$$

$$m(t) = \frac{a}{\eta - 1} [e^{r(\eta-1)W(t)} - 1]$$

4. Parameter Estimation:

Most important part of the software reliability prediction is parameter estimation. After determining the logical solution for $m(t)$ for a given model, it is needed to determine the parameters. There are some techniques, which are used for parameter estimation such as least squared estimation (LSE), maximum likelihood estimation (MSE), and Bayesian methods. The most important and broadly used technique for is MSE. In several cases, as the sample size increases, the maximum likelihood estimators are reliable and asymptotically normally distributed.

$\phi(t)$	P	a	r	θ
$\phi(t) = 0$	0.961	472.6	0.0183	2.494
$\Phi(t) = \ln(1 + rt)/r$	0.9425	428.5	0.03265	3.558
$\phi(t) = t + \alpha e^{-\beta t^\gamma} - \alpha$	0.8332	231.5	0.3246	2.478

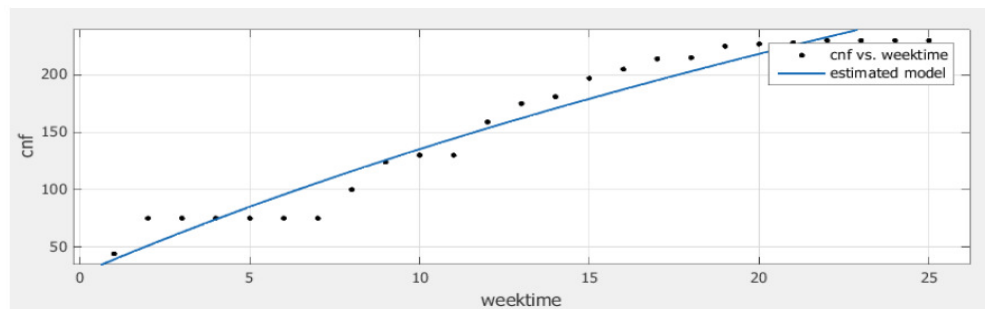


Fig 1: Proposed model 1.

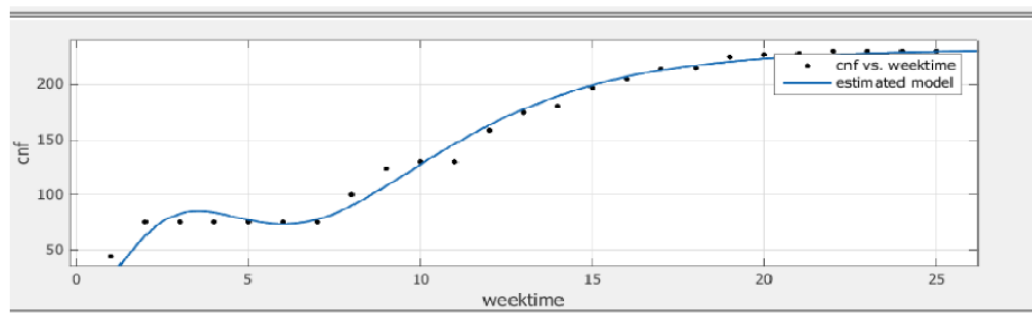


Fig 2: Proposed model 2.

5. Evaluation Criteria:

The result obtained after the parameter estimation are

- When $\phi(t) = 0$.
SSE: 4.086e+04
R-square: 0.6153
Adjusted R- square: 0.5603
RMSE: 44.11
- When $\phi(t) = \frac{\ln(1+rt)}{r}$
SSE: 5121
R-square: 0.9518
Adjusted R- square: 0.94449
RMSE: 15.62
- When $\phi(t) = t + \alpha e^{-\beta t^\gamma} - \alpha$
SSE: 1483
R-square: 0.986
Adjusted R- square: 0.984
RMSE: 8.405

6. Optimal software update policy:

- **Software release-time based on reliability criteria**

When reliability of the software is at its peak for a specified time then

Data-set

Weak time	Cumulative faults
1	44
2	75
3	75
4	75
5	75
6	75
7	75
8	100
9	124
10	130
11	130
12	159
13	175
14	181
15	197
16	205
17	214
18	215
19	225
20	227
21	228
22	230
23	230
24	230
25	230

that moment of time software to be delivered. Let $w(t)$ is defined as the current testing effort and $W(t)$ describes the cumulative testing effort. Relation between $w(t)$ and $W(t)$ is:

$$W(t) = \int_0^1 w(t) dt$$

$$W(t) = \alpha \left[1 - e^{-\beta t^m e^{\delta} - 1} \right]$$

On the basis of reliability criteria, the optimal update time of the open source software is given below:

S.No.	1	2	3	4	5	6	7
Reliability	80	85	88	90	91	92	93
Time	11	15	20	24	28	33	43

For $\Delta t = 0.1$ we get $R = 0.93$ at $T = 43.1$ weeks.

7. Conclusion

This work would help in correctly predicting the reliability of various softwares. It will help in the testing process and would make the process more efficient. Bug severity and imperfect debugging would help in correctly analyzing the likelihood of system failure. Time between the bug detection and removal would be considered and the total testing effort would be minimized. Also the fault dependency which is usually ignored will help in achieving a more practical estimation of product failure. The work would see the analysis of old models and on how can the existing models could be made more efficient and appropriate.

References:

- [1] M. Xie and M. Zhao, The Schneidewind Software Reliability Model Revisited, Proceedings of the 3rd International Symposium on Software Reliability Engineering, pp. 184-192, 1992.
- [2] N.Ullah, M. Morisio, "An Empirical analysis of Open Source Software Defects data through Software Reliability Growth Models", Control and Computer Engineering Department, EuroCon 2013.
- [3] M. Xie, B. Yang, "A study of the effect of imperfect debugging on software development cost", IEEE Transactions on Software Engineering, 29 471473, 2003.
- [4] Huang Yu Chin, "Performance Analysis of Software Reliability Growth Models with Testing Effort and Change-Point", Journal of Systems and Software Vol. 76, Issue 2, pp 181-194, 2005.
- [5] Yu-Shen Su, Chin-Yu Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models", The Journal of Systems and Software 80,606-615, 2007.

- [6] McLaughlin, L., "Automated bug tracking: the promise and the pitfalls", Software, IEEE, Vol.21, no 1, pp. 100-103, 2004.
- [7] Syed-Mohamad et al, "Reliability Growth of Open Source Software Using Defect Analysis", International Conference on Computer Science and Software Engineering, 2008.
- [8] P.K.Kapur, D.N.Goswami, Amit Bardhan, Ompal Singh, "Flexible software reliability growth model with testing effort dependent learning process", Applied Mathematical Modelling, 32,1298-1307, 2007.
- [9] Fengzhong Zou, Joseph Davis, "Analyzing and Modeling Open Source Software Bug Report Data" 19th Australian Conference on Software, IEEE computer Society, 2008.
- [10] S. Yamada, Ohba, M., and Osaki, S., "Sshaped software reliability growth models and their applications", IEEE Transactions on Reliability, 33(4), 289-292, 1984.
- [11] Kapur P.K., H. Pham, Anand Sameer, and Yadav Kalpana, "A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation", IEEE Transaction on Reliability, Volume: 60 Issue: 1, 2011.
- [12] Y.P. Wu, Hu, Q. P., Xie, M. and Ng, S. H., "Modeling and analysis of software fault detection and correction process by considering time dependency, IEEE Transactions on Reliability, vol. 56, no. 4, pp.629-642, 2007.
- [13] M. Xie, Hu, Q. P., Wu, Y. P., Ng, S. H. , "A study of the modeling and analysis of software fault-detection and fault correction processes", Quality and Reliability Engineering International, vol. 23, no. 4, pp. 459-470, 2007.
- [14] Eric S. Raymond , "The Cathedral and the Bazaar: Musings on Open Source by an Accidental Revolutionary". O'Reilly and Associates, 1999.
- [15] C. Y. Huang and C. D. Lin, "Software reliability analysis by considering fault dependency and debugging time lag", IEEE Trans. Reliability, vol. 55, no. 3, pp. 436-450, 2006.
- [16] Kuwata Y, Takeda K, Miura H, "A study on maturity model of open source software community to estimate the quality of products", Proc Comput Sci 35:1711-1717, 2014.
- [17] Li X, Li Y, Xie M, Ng S, "Reliability analysis and optimal version-updating for open source software", Inf Softw Technol 53:929-936, 2011.
- [18] IEEE Std 1633-2008 IEEE Recommended Practice in Software Reliability.
- [19] A.L Goel, and Okumoto, K., "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", IEEE Transactions on Reliability, vol. r-28, 206-2110, 1979.

