ICITKM
Proceedings of the First International Conference on Information Technology and Knowledge Management pp. 53–58
DOI: 10.15439/2018KM48
ISSN 2300-5963 ACSIS, Vol. 14

Reliability Modeling of OSS Systems based on Innovation-Diffusion Theory and Imperfect Debugging

Neha Gandhi[1], Neha Gondwal[2], Abhishek Tandon[3]

[1]*Shaheed Rajguru College of Applied Sciences for Women, University of Delhi, India*
[2]*Department of Operational Research, University of Delhi, India*
[3]*Shaheed Sukhdev College of Business Studies, University of Delhi, India*
[1]*nehagandhi1990@gmail.com,* [2]*neha28gondwal@gmail.com,* [3]*abhishektandon86@gmail.com*

***Abstract*—Open Source Software (OSS) has obtained widespread popularity in last few decades due to the exceptional contribution of some well established ones like Apache, Android, MySQL, LibreOffice, Linux etc. not only in the field of information technology but also in other sectors such as research, business and education. These systems are characterized by a huge shift in development pattern they adopt in comparison to proprietary software. Reliability modeling for such systems therefore is a growing area of research now days. Number of users adopting and working on refinement of such systems post-release play an indispensible role in their reliability growth. In this paper, we have proposed a software reliability growth model (SRGM) based on Non-homogeneous Poisson process (NHPP) based on number of users, under the phenomenon of Imperfect Debugging. The renowned Bass Model from Marketing based on the Theory of Diffusion of Innovation is used to depict the user growth phenomenon. Various fault content functions are considered in proposed models to represent imperfect debugging conditions and their performance is evaluated on fault dataset of GNOME 2.0. Four goodness-of-fit criteria namely Coefficient of Determination, Mean Square Error, Predictive Ratio Risk, and Predictive Power are used to calculate the estimation accuracy of all the proposed models and it has been observed that prediction capabilities of models based on imperfect debugging phenomenon is better than model assuming perfect debugging situation.**

***Index Terms*—Software Reliability Growth Model; Open Source Software; Imperfect debugging; Fault content function; Di usion of innovation.**

I. Introduction

WITH the advancements in field of technology, a visual expansion can be seen in the software industry. Open source software (OSS) has revolutionized the development trend of software in past few decades. Ubiquitous acceptance of OSS has become the sole reason for a huge inclination of developers towards these software systems. OSS can be defined as the software for which the source code is shared to learn, modify, and extend the software under some licensing guidelines [1]. Some of the characteristics of OSS due to which traditional SDLC models cannot be applied to OSS are described as follows:

A. Characteristics of OSS

- *Unclear Requirements:* OSS development doesn't witness a dedicated requirement elicitation phase where requirements are documented as in case of closed software. Here, development just starts with a single developer or a small group with a random idea. Requirements are not properly framed and freezed before development.
- *Unstable Team Size:* On contrary to closed source software a dedicated team of fixed number of people work on a project, OSS doesn't has fixed number of people working on it. It varies with time and attractiveness of software.
- *Minimal Testing Effort:* In OSS development, the main focus is given to implementation of idea as opposed to closed source software where sincere efforts and significant time is spent on testing phase after development completes.
- *No Deadlines:* Since development is totally dependent on voluntary participation of developers across the world, no strict deadlines on deliverables can be imposed.

As observed in above characteristics, during software development life cycle (SDLC) of OSS, it doesn't undergo an exhaustive testing phase as opposed to proprietary software where extensive testing efforts are spent before release of software. Because OSS cannot be tested rigorously for its functionality in such a limited duration of time, it becomes significant to study its reliability growth during its user phase. Software Reliability is the probability of software to perform its operations without any fail for a specified time interval in specified conditions [1]. Reliability growth of software can be modeled by a software reliability growth model (SRGM). An SRGM is a mathematical representation that depicts the software fault detection process as a function of different factors and parameters e.g. CPU time, Testing effort expenditures, Number of test cases, Code coverage etc. In the process of simulating software fault process with an SRGM, it is often assumed that each time a fault is encountered, the responsible fault is removed with certainty.

However this assumption is quite impractical as correction of a fault involves modifications in original source code which may lead to addition of new faults. In this paper, we propose an SRGM in which phenomenon of software fault detection is modeled with respect to user growth function on real life failure dataset (GNOME 2.0) under imperfect debugging conditions.

II. Literature Review

Numerous studies have been done in past to model reliability growth phenomenon for OSS and various SRGMs have been proposed. Goel-Okumoto model [3], Yamada S-Shaped model [4], Musa-Okumoto model [5] are some of the traditional NHPP based software reliability growth models for closed source software systems. Paulson et al. [6] performed an empirical study on open and closed source software to quantitatively investigate and validate the perceptions about these software. Rossi et al. [7] discussed the pattern of occurrence of faults in various OSS and thus make reliability predictions for future. Yamada and Yamaguchi [8] discussed a statistical process control method for OSS to determine the stability and thus getting estimation of development time needed to attain desired level of reliability. Li et al. [9] proposed reliability analysis model and used the model to predict optimal version-updating for OSS. Tamura and Yamada [10] combined software growth modeling with neural network approaches to estimate the reliability of OSS. Yang et al. [11] have given an SRGM for reliability estimation of multi-release OSS. Several other studies also have performed analysis on reliability of open source software [12, 13, 15]. Various studies demonstrated the reliability analysis of software under imperfect debugging scenarios. Kapur et al. [16] proposed frameworks to derive SRGMs in the presence of some realistic processes like imperfect debugging and error generation. Pham [17] developed a cost model with imperfect debugging conditions considering penalty cost due to delays in software delivery and random length of software development. Chu-Ti Lin [18] investigated the effects of imperfect debugging in modeling reliability via a simulation based model. In our paper, we model software fault process for OSS in its functional phase relating the usage factor with reliability modeling and comparing its performance under different fault content functions used for imperfect debugging element in model.

III. Model Foundations

The SDLC followed by OSS is unlike normal commercial software. For OSS, the testing effort spent is almost negligible in comparison to commercial software and therefore voluntary participation from developers across the world is crucial in refinement of quality of OSS. Studies done in past in direction of reliability assessment of OSS assumed that fault detection rate for OSS in operational phase will follow a hump-shaped curve. Initially, it grows with the growth in volunteer participation and reaches the highest point. This is due to the craze in developers for the new product in market. And then starts decreasing with the decrease in number of volunteers with time. This may be due their fall in interest for same product over time or due to the introduction of some new product in market. Similar trend is observed in user growth pattern of an Innovation. According to Rogers (1962), An Innovation can be characterized by following key features.

A. Characteristics of Innovation

1. *Relative Advantage:* It is the measure of improvement innovation brings over its competitive option. OSS brings lot of advantages in terms of cost benefit, vast resource knowledge base; lack of delivery deadlines saves the product from quality compromises.
2. *Compatibility:* It is the level of compatibility the innovation has with the current lifestyle of people. OSS doesn't need a huge lifestyle change over its closed source competitors. A user of commercial software can seamlessly adopt OSS.
3. *Complexity:* It refers to the level of difficulty an adopter of innovation will face to learn or use innovation. Various communities are there on internet for discussion of problems and challenges on OSS. They serve perfect to carry any discussion on OSS and pass on the reviews of the product to other potential users.
4. *Trialability:* It refers to the degree to which the innovation can be explored and tested by potential users. OSS is released in beta versions for the trial of users. Moreover after the release source code is also made available so that it can be expanded and customized as per needs.
5. *Observability:* It is the extent to which results of innovation are accessible to group of potential adopters. The important features of OSS like cost effectiveness, easy access and sharing of code etc are easily observable by people using it like educational institute, employees in corporate sector, freelancers etc.

Due to the presence of above characteristics in OSS, it can be considered as an Innovation. The usage growth process for an innovation can be best explicated with the renowned Innovation Diffusion Model of marketing (Bass, 1969). Since OSS is an Innovation it is justified to apply this model to describe user growth for an OSS.

B. Diffusion of Innovation

According to Rogers [19], the process of propagation of innovation i.e. new idea or new product, among the members of the communication system over a time frame is known as diffusion. As OSS is an Innovation, This theory perfectly applies to it. Among the potential volunteers of OSS, initially it is adopted by ones who are opinion leaders i.e. people who adopt it based on their own interest in software. They are also known as Innovators. Another group of people who later start using OSS based on word-of-mouth from innovators are known as Imitators. To represent the usage growth factor in our proposed SRGM we have used The Bass Model [20].

Bass Model

The Bass Model is a mathematical representation of process of diffusion. It depicts the concept of adoption of a product (OSS) among the potential users i.e. Innovators and Imitators through a mathematical expression. According to Bass, the diffusion process is defined by the following equation,

$$\frac{dN(t)}{dt} = \left(p + q\frac{N(t)}{\overline{N}}\right)\left(\overline{N} - N(t)\right) \quad (1)$$

Where, $N(t)$ depicts cumulative number of adopters at time t, $p(\overline{N} - N(t))$ represents those adopters (Innovators) who are not influenced by number of users already adopted therefore p denotes the coefficient of innovation. The term $q\frac{N(t)}{\overline{N}}\left(\overline{N} - N(t)\right)$ represents those users (Imitators) who are influenced by number of previous adopters and thus q depicts the coefficient of imitation. Solution of equation (1) under the initial condition of *N(0)=0* results in equation (2) which is given as follows:

$$N(t) = \overline{N}\frac{1-e^{-(p+q)t}}{1+\frac{q}{p}e^{-(p+q)t}} \quad (2)$$

C. Model Development

Notations

t	Calendar time
m, m(t)	Expected number of faults removed in time interval (0, t]
N, N(t)	Cumulative number of users in time interval (0,1]
a	Constant, the number of initial faults in a software
b	Constant, fault removal rate
p	Constant, coe cient of innovation
q	Constant, coe cient of imitation
$\overline{N}$	Constant, total number of potential users of the software
α	Proportion of error generation

Assumptions

- Software fault process is a NHPP phenomenon.
- The number of failures during operational phase is dependent upon the number of faults remaining in the software.
- As soon as any deviation from expected behaviour is encountered, it is considered as a failure and the fault that causes that failure is located. There is a possibility that at the time of fixing of a fault, few additional faults may get introduced.
- The number of faults removed is a function of number of users working on that software.
- The number of users is assumed to be a function of time and they are represented by the diffusion model given by Bass [20].

Based on above assumptions, the failure phenomenon can be illustrated as follows:

$$\frac{\mathrm{dm}}{\mathrm{dt}} = \frac{\mathrm{dm}}{\mathrm{dN}}\frac{\mathrm{dN}}{\mathrm{dt}} \quad (3)$$

The components on the right side of the equation are discussed as follows:

Component-1

This component describes the rate of detection of faults with respect to users. The rate at which failures occur depends upon the number of faults remaining in the software. As per this assumption the differential equation for fault removal can be written as,

$$\frac{dm}{dN} = b(a(N) - m) \quad (4)$$

where, $\alpha(N)$ denotes the fault content function with respect to number of users.

Component-2

The concept of user growth in adoption of OSS is the focus of the paper. The Innovation- Diffusion Model proposed by Bass [20] is used to describe user growth phenomenon in this study. According to Bass, the process of adoption of innovation in the potential population comprises of innovators and imitators and is defined by the equation (1) and is used to represent this component in equation (3). The solution of this equation for initial condition *N(0)=0* is given by equation (2).

D. Fault Content functions

In this paper, we have assumed fault content function as a function of number of users working on OSS in its operational phase and is denoted by a(N). The reason behind this is as the number of users working on an OSS increase, the more are the changes and modifications done in original code and hence the fault content also increases. Different fault content functions for corresponding to the cases of perfect and imperfect debugging are discussed as follows,

Perfect Debugging: The process of detection of faults and their rectification is termed as Debugging. Debugging

process that does not incorporate additional faults in the software is referred to as perfect debugging.

Case 1: In this case the fault content function represented by α(*N*) is assumed to be a constant. i.e. no new faults are introduced during debugging process. Therefore here,

$$a(N)=a \tag{5}$$

Imperfect Debugging: Imperfect debugging is the phenomenon where new faults get introduced while correcting the previous ones. Various fault content functions pertaining to this phenomenon are discussed hereby,

Case 2: In this case, we considered number of faults to be a linear function of number of users.

$$a(N)=a\left(1+\alpha N\right) \tag{6}$$

Case 3: In this case, we consider an exponential fault content function using Yamada et al. [21], it means faults are introduced exponentially with respect to number of users.

$$a(N)=a\,e^{\alpha N} \tag{7}$$

Case 4: Here, we have adopted rate of introduction of new faults as a function of the number of faults already removed in the software.

$$a(N)=a+\alpha m\left(N\right) \tag{8}$$

Case 5: Under this case of imperfect debugging, we assume that the new faults can be introduced exponentially per detected fault (Pham & Zhang [22]). Here c is assumed as constant.

$$a(N)=c+a\left(1-e^{-\alpha N}\right) \tag{9}$$

Here, $N=N\left(t\right)$ represents the number of users of OSS up till time t.

(5) Proposed Models

The proposed models obtained by combining both components of Equation (3) are explained as follows:

SRGM 1: We obtained following model under above stated case 1.

$$m_1(t)=a\left(1-e^{-bN}\right) \tag{10}$$

SRGM 2: This model is obtained under case 2 which represents imperfect debugging.

$$m_2(t)=a\left(\alpha N+\left(1-\frac{\alpha}{b}\right)\left(1-e^{-bN}\right)\right) \tag{11}$$

SRGM 3: Under case 3, following expression is obtained:

$$m_3(t)=\frac{ab}{\alpha+b}\left(e^{\alpha N}-e^{-bN}\right) \tag{12}$$

SRGM 4: For case 4, we got the following model:

$$m_4(t)=\frac{a}{1-\alpha}\left(1-e^{-b(1-\alpha)N}\right) \tag{13}$$

SRGM 5: The following model is based on above discussed case 5:

$$m_5(t)=(a+c)\left(1-e^{-bN}\right)-\frac{ab}{b-\alpha}\left(e^{-\alpha N}-e^{-bN}\right) \tag{14}$$

For SRGM 1- SRGM 5, $N\left(t\right)$ is given by equation (2).

IV. Numerical Study

For the purpose of parameter estimation, we have used the real time failures data set of a very renowned OSS i.e. GNOME 2.0 provided by Li et al. (2011) as given in Table 1.

Table 1: Detected faults in GNOME 2.0 release

Day	Detected Bugs	Day	Detected Bugs	Day	Detected Bugs	Day	Detected Bugs
1	6	7	8	13	6	19	1
2	5	8	4	14	8	20	1
3	3	9	8	15	6	21	1
4	2	10	3	16	2	22	2
5	5	11	2	17	2	24	3
6	5	12	1	18	1		

Estimation of proposed models is performed with the Least Squares Principle. The estimation results of the models proposed in this study as described in Equation 10-14 on the GNOME 2.0 dataset are presented in Table 2 .

Table 2: Estimated Parameters

Proposed model	a	b	$\overline{N}$	p	q	α	c
SRGM 1	94.445	0.053	50.922	0.014	0.143	-	-
SRGM 2	81.349	0.420	37.104	0.003	0.117	0.004	-
SRGM 3	80.736	0.827	58.038	0.01	0.112	0.008	-
SRGM 4	84.537	0.259	57.586	0.003	0.112	0.001	-
SRGM 5	87.243	0.721	30.237	0.002	0.117	0.009	80.068

Also, we have performed the comparison of our proposed SRGMs using four goodness-of-fit criteria namely Coefficient of Determination (R^2), Mean Square Error (MSE), Predictive Ratio Risk (PRR), and Predictive Power (PP). The expression and interpretation of these is discussed may be found in Ref. [23].

The goodness-of-fit values obtained for the above mentioned criteria for the five proposed SRGM are summarised in Table 3.

Table 3: Goodness of fit Comparison Table

Proposed Model	R^2	MSE	PRR	PP
SRGM 1	0.994	4.336	4.050	3.395
SRGM 2	0.994	4.081	3.635	3.132
SRGM 3	0.994	4.024	3.543	3.080
SRGM 4	0.994	4.051	3.521	3.072
SRGM 5	0.994	4.077	3.648	3.137

The goodness-of-fit curves obtained corresponding to SRGM 1, SRGM 2, SRGM 3, SRGM 4, and SRGM 5 are shown in Figure 1-5.

V. Discussion on Results

From Table 2, It can be observed that the value of q which represents the coefficient of imitation is always greater than p which is the coefficient of innovation. This observed relation between p and q is consistent with the Bass Diffusion Model and illustrates the importance of word of mouth on the decision making of used by the potential users. From the Table 3, it can also be concluded that the all the SRGMs based on bass growth function fits data well. Moreover, from the values of goodness-of-fit criteria obtained it can be inferred that models biult over imperfect debugging assumption (SRGM 2-SRGM 5) outperform the model depicting perfect debugging situation (SRGM 1) which explains the fact that imperfect debugging models are close to realistic situation.

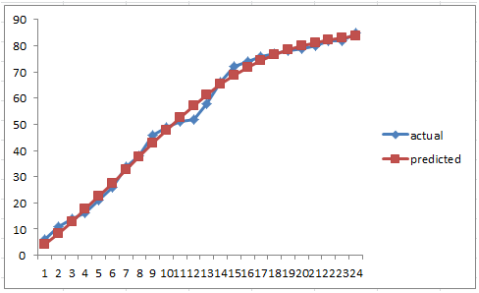

Figure 1: goodness-of-fit curve for SRGM 1

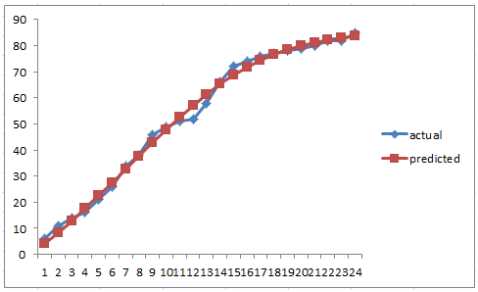

Figure 2: goodness-of-fit curve for SRGM 2

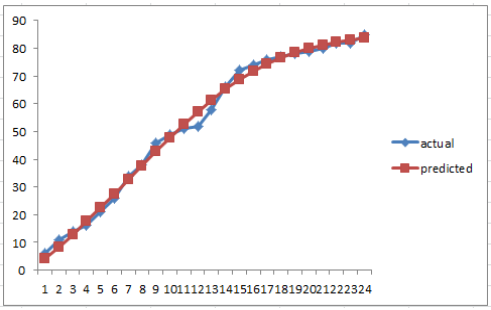

Figure 3: goodness-of-fit curve for SRGM 3

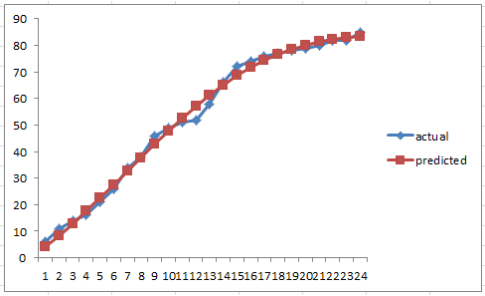

Figure 4: goodness-of-fit curve for SRGM 4

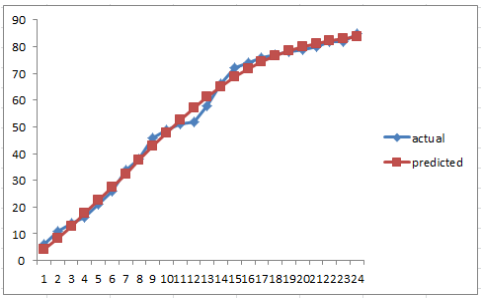

Figure 5: goodness-of-fit curve for SRGM 5

VI. Conclusion and Future Scope

In this study, we have related user growth with reliability growth phenomenon for OSS. Bass Innovation Diffusion model has been used to represent user growth function. Five SRGMs have been proposed using various fault content functions to model perfect and imperfect debugging phenomenon. The mean value functions for the proposed SRGMs are used and parameters' estimates are calculated using least square estimation technique. Performance of all the proposed models is compared using four goodness-of-fit methods: R^2, MSE, PRR, and PR and it is found that imperfect debugging models gave better estimation results as compared to model with assumption of perfect debugging. In future studies the functions to model imperfect debugging can be extended to include the concept of change point or randomness in user growth. For user growth we have used Bass Innovation-Imitation model. In future, we may extend our work for multi release OSS.

References

[1] W. S. Jawadekar, "Software Engg", Tata McGraw-Hill Education, 2004.

[2] P. Kapur, H. Pham, A. Gupta, P. Jha, "Software reliability assessment with OR applications", Springer, 2011.

[3] A. L. Goel, K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures", IEEE transactions on Reliability, Vol. 28, Issue 3, 1979, 206–211.

[4] S. Yamada, M. Ohba, S. Osaki, "S-shaped reliability growth modeling for software error detection", IEEE Transactions on reliability Vol. 32, Issue 5, 1983, 475–484.

[5] J. D. Musa, K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement". In Proceedings of the 7th international conference on Software engineering, IEEE Press, 1984, 230-238.

[6] J. W. Paulson, G. Succi,A. Eberlein, “An empirical study of open-source and closed-source software products”, IEEE Transactions on Software Engineering, Vol. 30, Issue 4,2004,246-256.
[7] B. Rossi, B. Russo, G. Succi, “Modelling failures occurrences of open source software with reliability growth”, Open Source Software: New Horizons, 2010, 268–280.
[8] S. Yamada, M. Yamaguchi, “A method of statistical process control for successful open source software projects and its application to determining the development period , International Journal of Reliability, Quality and Engineering, Vol. 23, Issue 5, 2016.
[9] X. Li, Y. F. Li, M. Xie, S. H. Ng, “Reliability analysis and optimal version-updating for open source software”, Information and Software Technology, Vol. 53, Issue 9, 2011, 929–936.
[10] Y. Tamura, S. Yamada, “A component-oriented reliability assessment method for open source software”, International Journal of Reliability, Quality and Safety Engineering, Vol. 15, Issue 1 ,2008, 33-53.
[11] J. Yang, Y. Liu, M. Xie, M. Zhao, “Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes”, Journal of Systems and Software, Vol. 115, 2016, 102–110.
[12] C. Rahmani, A.H. Azadmanesh, L. Najjar, “A Comparative Analysis of Open Source Software Reliability”, JSW, Vol. 5, Issue 12, 2010, 1384-1394.
[13] C. Rahmani, H. Siy, A. Azadmanesh, “An experimental analysis of open source software reliability”, Department of Defense/Air Force Office of Scientific Research, 2009.
[14] Y. Tamura, S. Yamada, “Comparison of software reliability assessment methods for open source software, in: Parallel and Distributed Systems”, 2005. Proceedings. 11th International Conference on, Vol. 2, IEEE, 2005, 488–492.
[15] Y. Zhou, J. Davis, “Open source software reliability model: an empirical approach”, ACM SIGSOFT Software Engineering Notes, Vol. 30, 2005, 1–6.
[16] P. Kapur, H. Pham, S. Anand, K. Yadav, “A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation”, IEEE Transactions on Reliability Vol. 60, Issue 1, 2011, 331–340.
[17] H. Pham, “A software cost model with imperfect debugging, random life cycle and penalty cost”, International Journal of Systems Science Vol. 27 , Issue 5, 1996, 455–463.
[18] C. T. Lin, “Analyzing the effect of imperfect debugging on software fault detection and correction processes via a simulation framework”, Mathematical and Computer Modeling, Vol. 54, Issue 11, 2011, 3046–3064.
[19] E. M. Rogers, “Diffusion of innovations”, Simon and Schuster, 2010.
[20] F. M. Bass, “A new product growth for model consumer durables”, Management science, Vol. 15, Issue 5, 1969, 215-227.
[21] S. Yamada, K. Tokuno, S. Osaki, “Imperfect debugging models with fault introduction rate for software reliability assessment”, International Journal of Systems Science, Vol. 23, Issue 12, 1992, 2241-2252.
[22] H. Pham, X. Zhang, X,”NHPP software reliability and cost models with testing coverage”, European Journal of Operational Research, Vol. 145, Issue 2, 2003.
[23] N. Gandhi, N. Gondwal, A.G. Aggarwal, A. Tandon, “Estimating Reliability for OSS: An approach with change-point in operational phase”, In proceedings of 6th International Conference on Reliability, Infocom Technologies and Optimization, IEEE, 2017, 251-255.