International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

Modeling Software Multi Up-Gradations with Error Generation and Fault Severity

Raksha Verma[*], R. S. Parihar[#]
Department of Mathematics
Sunrise University, Alwar, Rajasthan, India
E-mails: [*]rakshajonwal@gmail.com, [#]roopsinghp@gmail.com

Subhrata Das
Department of Operational Research
University of Delhi, Delhi, India
E-mail: shus.das@gmail.com
Corresponding author

(Received October 28, 2017; Accepted December 4, 2017)

Abstract
World is moving very fast and with each new step 'our desire to want something more' is increasing in all the aspects of lives. Keeping in mind the growing demand of society for highly equipped software; the suppliers tend to come with up-gradations very frequently with the desire to make an impression in the market. Each organization is battling to showcase that their products is an enhanced version. As a result, companies at proper intervals are coming up with innovative functionalities. In order to have an upper hand amongst the competitors; the judgement on apt time of release of a software is utmost sensitive. This decision is majorly governed by company's expertise in being able to remove all kinds of faults in the software. Development of a multi-release upgraded software reliability growth model is done with regard to trace the upshots of faults in line for the existing software and the one's remaining in the software at different intervals. This model ascertains the remaining bugs occurred in the software during its operational phase while testing of the new code in other words which occurs in course of addition of new features to the current software. Further, the case of three types of faults existing in the software namely: simple, hard and complex has been incorporated to model the fault removal phenomenon. Data Analysis has been given to support the results.

Keywords- Error generation, Fault severity, Multi up-gradation, Software reliability growth model (SRGM).

1. Introduction

Global economy of present world has much interconnectedness and to ensure efficacy and advancement companies are majorly dependent on their IT department which further relies on numerous software commodities. In this case to serve its designated purpose in the business the reliability of software product is desired. Lately, software tends to have a great impression on our day to day lives, thus the software developers are focusing more on developing technologies which could in turn build high-quality computer programs. As a result, software companies are straining to bring up an upgraded version from time to time. The operation of a new technology in its lifetime is demonstrated by the S-shaped curve or sigmoid curve, signifying that more efforts are required in the initial phase. After achieving the anticipated level of operational reliability set by the firm, up-gradation of software takes place replacing the prevailing product in the market with an upgraded version consisting of enhanced functionalities.

In order to survive the strong market pressure, firms tend to keep upgrading their software and re-release it with new features. The up-graded version is further able to improve the software's performance and functionality. Incorporating this modern concept of multi- up-gradations, many

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

software reliability models have been proposed. Kapur et al. (2010a) have discussed a multi release software model which identifies the faults when the software is in its operational phase and is being tested for additional features. The additional features can make the software highly complex and thus lead to faults being generated. In the work of Singh et al. (2012a), they have proposed two multi up-gradation software reliability growth models for multi release. The SRGMs have been modeled using Normal and Logistic Distribution to identify the faults of the previous as well as the next release. Above discussed models were based on the perfectly debugging environment. There can be many instances when the testing team may not be able to debug bugs perfectly instead it affects the eventual fault count by either not being able to debug the faults completely (i.e. imperfect fault removal phenomenon) or leading to addition of some of new flaws (Aggarwal et al., 2011; Kapur et al., 2010b; Singh et al., 2014c).

There is another stream of research which focused on the impact of fault severity within a fault removal process for multi upgraded software. This categorization is based upon the time taken for fault isolation and its removal after its observation. On the basis of which faults can be categorized as 'Simple fault' if the time between their observation, isolation and removal is negligible, another class of 'Hard faults' in which more efforts and time is required to observe and isolate the faults, third category of faults called as 'Complex faults' in which there is reasonable amount of both efforts and time is required for faults to observed, identified and debugged (Kapur et al., 2011a).

Several researchers have incorporated this concept in fault removal phenomenon for software with multiple releases viz. Researchers have considered the fault severity based multi upgradation software reliability growth model. Further a trend line shows the reliability growth the various releases of the software (Singh et al., 2014a). Another stream has focused on the inclusion of testing effort in fault removal process; Singh et al. (2012b) have developed a multi up-gradation software reliability model with testing effort. They have described the procedure to incorporate the Weibull testing-effort function into software multi up-gradation reliability models. They have further described an Exponential Power Distribution to identify the faults in software and predict the faults in its successive release. As an extension of the previous work; Singh et al. (2015a) have considered the severity of the faults induced along with the testing effort consumed when multi release of the software takes place. The faults of the previous release as well as the current release are considered to understand the effect of faults on the software. The fault removal rate has been considered to be dependent on the testing effort put in throughout the testing process. Singh et al. (2014b) have used different fault removal process based on the generalized Erlang Model for different release of multi upgraded software. Singh et al. (2014c) have discussed imperfect debugging environment in the multi release of the software. Their model describes the fault detection and correction as a stop step process and uses it in the two types of imperfect debugging i.e. incomplete fault removal and error generation while removing a fault. Kapur et al. (2014) have considered a Distributed Development Environment (DDE) during software development multi-up gradations modeling for removal of fault for newly developed and reused components in distributed environment using probability distribution functions. Singh et al. (2015b) have modeled the fault detection as a stochastic process with continuous state space in a multi-up-gradation model. They have considered the fault severity and the effect of learning. The simple faults are removed at an exponential rate whereas hard faults are removed by Yamada with learning effect function (Kapur et al., 2011a).

Enormous work has been done in the field of software engineering to model the fault removal process under the concept of multi upgradation with the inclusion of fault severity, imperfect debugging, testing effort and many more (Anand et al., 2015; Garmabaki et al., 2014; Singh et al.,

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

2011). Moving with the same trend in this paper emphasis has been laid to model the mean value function for multi upgraded software under the impact of fault severity with the assumption that not all faults are removed perfectly i.e. addition of new faults might happen. In order to differentiate between different types of faults based on their severity, three different forms of distribution functions have been considered. Rest of the paper organized as follows: Different set of notations are explained in section 2. Software reliability modeling and multi up-gradation modeling framework are demonstrated in section 3. Model validation and conclusion stated in section 4 & 5.

2. Notations

$m_i(t)$: Cumulative number of bugs removed in the time interval $(0, t]$.

a_i : Total number of bugs present in the software $(i=1,2)$.

b_{i1} : Fault (simple) removal rate for i^{th} releases $(i=1,2)$.

b_{i2} : Fault (hard) removal rate for i^{th} releases $(i=1,2)$.

b_{i3} : Fault (complex) removal rate for i^{th} releases $(i=1,2)$.

α_{i1} : Rate of generating simple faults for i^{th} releases $(i=1,2)$.

α_{i2} : Rate of generating hard faults for i^{th} releases $(i=1,2)$.

α_{i3} : Rate of generating complex faults for i^{th} releases $(i=1,2)$.

λ_{i1} : Proportion of simple faults of i^{th} releases $(i=1,2)$.

λ_{i2} : Proportion of hard faults of i^{th} releases $(i=1,2)$.

$(1-\lambda_{i1}-\lambda_{i2})$: Proportion of complex faults of i^{th} releases $(i=1,2)$.

3. Software Reliability Modeling

Vast literature exists in the field of multi up-gradation to model the Fault Removal Phenomenon (FRP) for any software system using the Non-Homogeneous Poison Process (NHPP). Before we begin with FRP, it is important to have insight regarding the unified approach to model FRP (Kapur et al., 2011a).

Consider $\{N(t), t \geq 0\}$ as a counting process representing the cumulative number of faults at any instance of time 't'. It can be easily deduced that $N(t)$ follows a NHPP with a mean value function $m(t)$ (representing the number of bugs debugged till time 't') (Kapur et al., 2011a).

$$\Pr(N(t)=n)=\frac{(m(t))^n}{n!}\exp((-m(t)) \quad (1)$$

where $m(t)=\int_0^t \lambda(x)dx$.

According to the assumption of NHPP model, a constant 'a' implies the perfect fault removal process, i.e., no new faults are introduced. Pham (1993) introduced the concept of SRGM incorporating imperfect debugging wherein it has been assumed that there are certain cases when the removal of faults leads to introduction of some new faults with the rate 'α'. Let $a(t)$ represents

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

the number of faults that has to be detected plus some new faults that might get introduced. Moving with above assumptions and with the consideration that the rate by which fault are removed is directly proportion to remaining number of faults in the system. The differential equation for mean value function by incorporation of error generation and hazard rate approach we have:

$$\frac{dm(t)}{dt} = \frac{f(t)}{1-F(t)}\left(\left(a+\alpha m(t)\right)-m(t)\right) \tag{2}$$

Solving Eq. (2) under the initial condition that at the start of FRP there will be zero faults detection or correction i.e. $m(t=0)=0$ we have

$$m(t) = a'F(t) \tag{3}$$

where $a' = \frac{a}{1-\alpha}$.

Above modeled equation represents a general equation for a FRP. In order to include the impact of fault severity equation (3) can be modified as:

$$m(t) = \lambda_1 a' F_1(t) + \lambda_2 a' F_2(t) + \lambda_3 a' F_3(t) \tag{4}$$

where $\lambda_1; \lambda_2$ and λ_3 represents the proportion of simple, hard and complex faults with the condition $(\lambda_1+\lambda_2+\lambda_3=1)$; $F_1(t)$; $F_2(t)$ and $F_3(t)$ represents the distribution functions pertaining to simple, hard and complex faults.

Further, the equation (4) under the case of error generation can be represented as follows:

$$m\ (t) = a_1^* F_1(t) + a_2^* F_2 + a_3^* F_3 \tag{5}$$

where $a_1^* = \frac{\lambda_1 a}{1-\alpha_1}$, $a_2^* = \frac{\lambda_2 a}{1-\alpha_2}$, $a_3^* = \frac{(1-\lambda_1-\lambda_2).a}{1-\alpha_3}$.

Here the emphasis is given to model the case of successive releases of software discussed below.

3.1 Multi-Upgradation Modeling Framework

Release I

The competitive environment present in the market has pushed the software firm to release multiple version of their software after a certain time point. The software based on a product family is upgraded after adding new and tested features to sustain in the market. The foundation of a new software product is more prone to fail under its usage. Therefore, the software testing team has to pay more attention during the test case execution process. The test cases executed under this release should capture as many faults and provide a positive assurance about the quality of the software. Due to complexity in the software code, there exist some cases while debugging the faults; faults underlying in the system might introduce some new faults. In this modeling framework, we have considered that the fault removal process behaves according to the existence of various types of faults and newly generated faults. Here we have considered three types of faults such as simple,

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

hard and complex faults. The total numbers of bugs are then resolved by considering exponential rate (G-O Model) for simple faults, Yamada (two-stage) Model for hard faults and Erlang 3-stage model for the complex faults (Goel and Okumoto, 1979; Kapur et al., 2011a; Yamada et al., 1983). The mathematical expression for release I is given as follows:

$$m_1(t) = a_{11}^* F_{11}(t) + a_{12}^* F_{12} + a_{13}^* F_{13} \qquad ; 0 < t < t_1 \tag{6}$$

where $a_{11}^* = \frac{\lambda_{11} a_1}{1-\alpha_{11}}$, $a_{12}^* = \frac{\lambda_{12} a_1}{1-\alpha_{12}}$, $a_{13}^* = \frac{(1-\lambda_{11}-\lambda_{12}).a_1}{1-\alpha_{13}}$

$$F_{11}(t) = \left(1 - e^{-b_{11}(1-\alpha_{11})t}\right)$$

$$F_{12}(t) = \left(1 - \left((1+b_{12}.t)e^{-b_{12}t}\right)^{(1-\alpha_{12})}\right) \quad .$$

$$F_{13}(t) = \left(1 - \left(\left(1 + b_{13}.t + \frac{b_{13}^2 t^2}{2}\right)e^{-b_{13}t}\right)^{(1-\alpha_{13})}\right)$$

Release II

The existence of the second release is mainly due to the enhancement of user's requirement gained from the feedback of its preceding release, the competitive environment and to provide new functionality based on the market scenario. The accumulation of new code will result in an increase in the fault content and the testing team has to design respective test cases in order to deal with the new functionality and its effect. The new testing team for the second release will monitor or debug the upgraded version with a different rate. The test cases executed will be designed in order to gather the discovery or removal of faults due to the added functionality or due to the remaining faults of previous release. During the debugging process under the impact of error generation, the various types of faults based on their severity level either of current release or left over from the previous release would be removed with the newly allocated detection rate of simple, hard and complex faults respectively i.e. $F_{21}(t-t_1)$ for simple faults, $F_{22}(t-t_1)$ for hard faults and $F_{23}(t-t_1)$ for complex faults.. The mathematical framework for this release can be written as follow:

$$\begin{aligned} m_2(t) = &\left[a_{21}^* + a_{11}^*\left(1-F(t_1)\right)\right]F_{21}(t-t_1) + \left[a_{22}^* + a_{12}^*\left(1-F(t_1)\right)\right]F_{22}(t-t_1) \\ &+ \left[a_{23}^* + a_{13}^*\left(1-F(t_1)\right)\right]F_{23}(t-t_1) \qquad ; t_1 < t < t_2 \end{aligned} \tag{7}$$

where $a_{21}^* = \frac{\lambda_{21} a_2}{1-\alpha_{21}}$, $a_{22}^* = \frac{\lambda_{22} a_2}{1-\alpha_{22}}$, $a_{23}^* = \frac{(1-\lambda_{21}-\lambda_{22}).a_2}{1-\alpha_{23}}$.

In equation (7), it is a part of our consideration that left over simple, hard and complex faults of previous release will be debugged in correspondingly generated faults of same severity. The same sets of equations can be extended for software with 'n' release, which can be given as follows:

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

$$m_i(t) = \left[a_{i1}^* + a_{(i-1)1}^* . F_{(i-1)1}(t_{i-1})\right] F_{i1}(t - t_{i-1}) + \left[a_{i2}^* + a_{(i-1)2}^* . F_{(i-1)2}(t_{i-1})\right] F_{i2}(t - t_{i-1}) + \left[a_{i3}^* + a_{(i-1)3}^* . F_{(i-1)3}(t_{i-1})\right] F_{i3}(t - t_{i-1}) \quad ; t_{i-1} < t < t_i \tag{8}$$

where $a_{i1}^* = \frac{\lambda_{i1} a_i}{1-\alpha_{i1}}$, $a_{i2}^* = \frac{\lambda_{i2} a_2}{1-\alpha_{i2}}$, $a_{i3}^* = \frac{(1-\lambda_{i1}-\lambda_{i2}).a_i}{1-\alpha_{i3}}$.

4. Model Validation and Data Analysis

The proposed model has been analyzed on real life data set. Two releases of Tandem data set have been used to validate the model (Wood, 1996). We have made use of SAS software package (SAS, 2004). Table 1 & 2 contain the parameter estimation and comparison criteria for two releases. Fig. 1 & 2 shows the graphical representation of proposed methodology. Performance analysis of proposed model is measured by the five common criteria i.e. SSE, MSE, Root MSE, R^2 and $Adj.R^2$ (Kapur et al., 2011a).

Table 1. Parameter estimation for Release I & II

Parameters	Release I	Release II
a_i	141.588	126.994
b_{i1}	0.450	0.207
b_{i2}	0.550	0.206
b_{i3}	0.031	0.0027
λ_{i1}	0.125	0.1784
λ_{i2}	0.157	0.749
α_{i1}	0.022	0.021
α_{i2}	0.775	0.0018
α_{i3}	0.011	0.879

Table 2. Comparison criteria for Release I & II

Parameters	Release I	Release II
SSE	320.9	178.4
MSE	18.87	11.151
Root MSE	4.34	3.339
R^2	0.980	0.992
$Adj.R^2$	0.977	0.992

International Journal of Mathematical, Engineering and Management Sciences
Vol. 3, No. 4, 429–437, 2018
https://dx.doi.org/10.33889/IJMEMS.2018.3.4-030

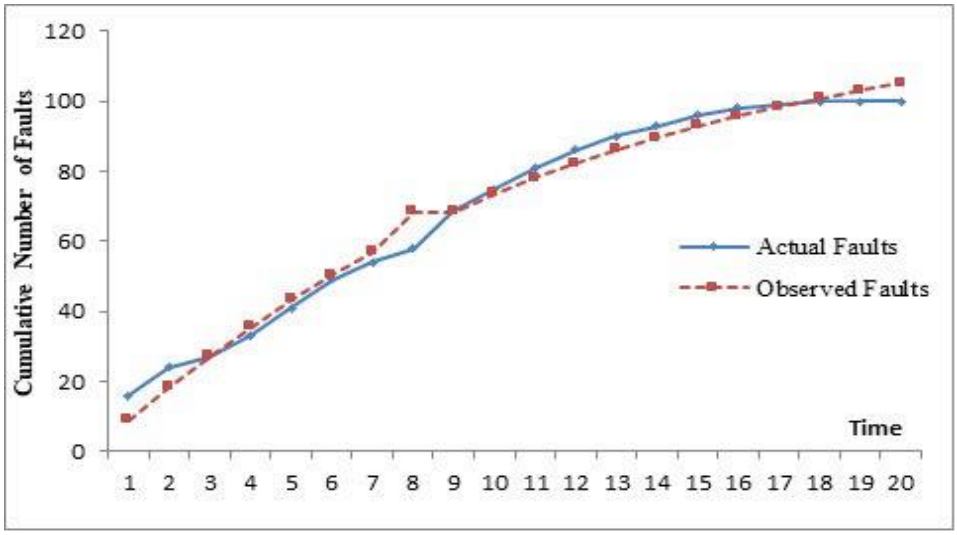

Fig. 1. Goodness of fit curve for Release I

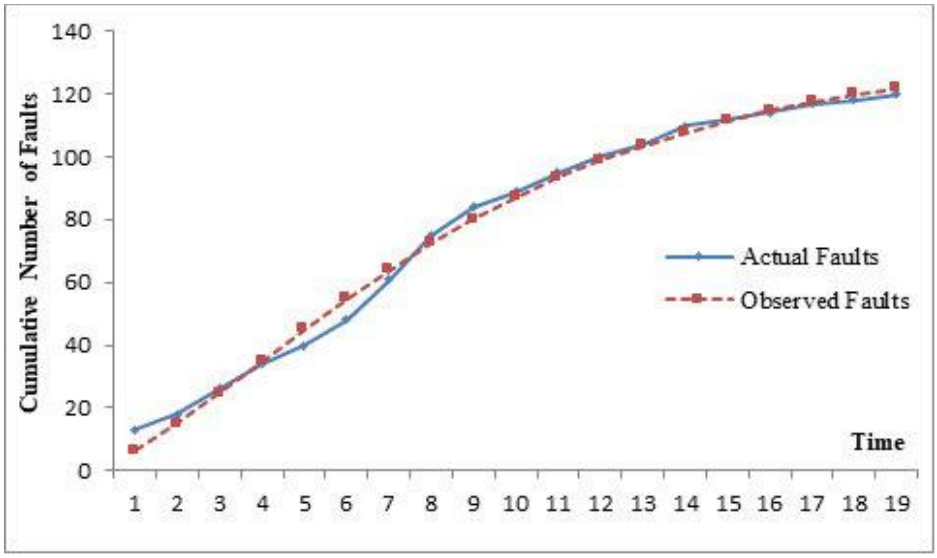

Fig. 2. Goodness of fit curve for Release II

5. Conclusion

The cut-throat competition in the market has given birth to multiple versions of the software. There is relentlessness pressure to beat the competitors in the daily grind resulting in the process of sporadic up gradation of the software with various releases. The need for improvement in the

functions and removal of bugs from the earlier versions is principal reason for these up gradations. The errors in the up-graded version can either generate from the newly added code or might be some previous remaining error. The categorization into three kinds of errors namely simple, hard and complex is associated with the severity of the errors. The one which is convenient to remove is known as simple, the one which is difficult and time-consuming is known as hard and one which is comparatively more time- consuming, effort taking and requires quite an expertise is known as complex. The testing team articulates different strategies for different forms of errors rooted in the software sometimes resulting into addition of more faults. Virtually it is not possible to eliminate all the bugs at a time, as some bugs are passed on from the previous release. In this article; embracing the concept of fault severity and error generation we have proposed a mathematical framework of multi-upgradation wherein all the faults generated would be captured and concurrently removed with the various kinds of remaining faults from the earlier release. Here, we have assumed that remaining simple (hard or complex) faults will be removed with simple (hard or complex) faults of its succeeding release.

References

Aggarwal, A. G., Kapur, P. K., & Garmabaki, A. S. (2011). Imperfect Debugging software reliability growth model for multiple releases. *Proceedings of the 5th National Conference on Computing for Nation Development-INDIACOM, New Delhi*.

Anand, A., Singh, O., & Das, S. (2015). Fault severity based multi up-gradation modeling considering testing and operational profile. *International Journal of Computer Applications*, *124*(4), 9-15.

Garmabaki, A. H. S., Aggarwal, A. G., Kapur, P. K., & Yadavali, V. S. S. (2014). The impact of bugs reported from operational phase on successive software releases. *International Journal of Productivity and Quality Management*, *14*(4), 423-440.

Goel, A. L., & Okumoto, K. (1979). Time dependent error detection rate model for software reliability and other performance measures. *IEEE Transaction Reliability*, *R-28*(3), 206-211.

Kapur, P. K., Anand, A., & Singh, O. (2011b). Modeling successive software up-gradations with faults of different severity. *Proceedings of the 5th National Conference on Computing for Nation Development*, *INDIACom* (pp. 351-356).

Kapur, P. K., Pham, H., Gupta, A., & Jha, P. C. (2011a). *Software reliability assessment with OR application*. Springer, Berlin.

Kapur, P. K., Shrivastava, A. K., Kumar, A., & Shivhare, B. D. (2014, October). Multi up gradation model under distributed environment. *In Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014 3rd International Conference on* (pp. 1-6). IEEE.

Kapur, P. K., Singh, O., Garmabaki, A., & Singh, J. (2010b). Multi up-gradation software reliability growth model with imperfect debugging. *International Journal of Systems Assurance Engineering and Management*, *1*(4), 299-306.

Kapur, P. K., Tandon, A., & Kaur, G. (2010a, December). Multi up-gradation software reliability model. In *Reliability, Safety and Hazard (ICRESH), 2010 2nd International Conference on* (pp. 468-474). IEEE.

Pham, H. (1993). Software reliability assessment: imperfect debugging and multiple failure types in software development. *EGandG-RAAM-10737, Idaho National Engineering Laboratory*.

SAS Institute Inc. (2004). *SAS/ETS user's guide version 9.*1, Cary, NC: SAS Institute Inc.

Singh, J., Singh, O., & Kapur, P. K. (2015b). Multi up-gradation software reliability growth model with learning effect and severity of faults using SDE. *International Journal of System Assurance Engineering and Management*, *6*(1), 18-25.

Singh, O., Aggrawal, D., Anand, A., & Kapur, P. K. (2015a). Fault severity based multi-release SRGM with testing resources. International Journal of System Assurance Engineering and Management, *6*(1), 36-43.

Singh, O., Anand, A., Aggrawal, D., & Singh, J. (2014c). Modeling multi up-gradations of software with fault severity and measuring reliability for each release. *International Journal of System Assurance Engineering and Management*, *5*(2), 195-203.

Singh, O., Kapur, P. K., & Anand, A. (2011). A stochastic formulation of successive software releases with faults severity, *2011 IEEE International Conference on Industrial Engineering and Engineering Management*, Singapore, 2011, pp. 136-140

Singh, O., Kapur, P. K., & Singh, J. N. P. (2012b). Testing-Effort based multi up-gradation software reliability growth model. *Communication in Dependability and Quality Management-An International Journal*, *15*(1), 88-100.

Singh, O., Kapur, P. K., Khatri, S. K., & Singh, J. N. P. (2012a). Software reliability growth modeling for successive releases. *Proceeding of 4th International Conference on Quality, Reliability and Infocom Technology (ICQRIT)*, 77-87.

Singh, O., Kapur, P. K., Shrivastava, A. K., & Das, L. (2014b, October). A unified approach for successive release of a software under two types of imperfect debugging. *In Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014 3rd International Conference on* (pp. 1-6). IEEE.

Singh, O., Singh, J. N., Tickoo, A., & Kapur, P. K. (2014a). Fault removal phenomenon using different distribution functions for each release. *International Journal of Modeling and Optimization, 4*(1), 5-9.

Wood, A. (1996). Predicting software reliability. *IEEE Computer*, *29*(11) 69-77.

Yamada, S., Ohba, M., & Osaki, S. (1983). S-shaped software reliability growth modelling for software error detection. *IEEE Transaction Reliability*, *R-32*(5), 475-484.