Fault Removal Phenomenon Using Different Distribution Functions for Each Release

Ompal Singh, Jyotish N. P. Singh, Anshul Tickoo, and P. K. Kapur

***Abstract*—Software is everywhere and has become a major worldwide industry. We find software embedded, for example, in watches, coffee makers, cars, televisions, airplanes, telephones, reservation systems, and medical equipment. Software not only pervades a multitude of products, but also is an important corporate asset, and demand is increasing. With dynamic markets and evolving business models, organizations need to stay agile to maintain and improve their competitive edge. First release of software products includes enough features and functionality to make it useful for the customers. Later, software companies have to come up with up-gradation or add-ons in their software to survive in the market through a series of releases. They plan successive releases by adding new features or new functionalities or try to improve performance of system as compared to previous releases by removing faults from existing software. Removing maximum faults from existing release and delivering reliable software is most important. In this paper we have used different fault removal process based on generalized Erlang model for different releases. The model is validated on real software data set.**

***Index Terms*—Software reliability growth model, distribution function, multi release.**

I. Introduction

The In recent years, with the rapid development of computer technology, more and more software systems are widely used in high reliability field. It is presented great demanding for software reliability. In order to evaluate the reliability of software, a lot of software reliability models are presented [1]-[8]. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. In last three decades several reliability growth (SRGMS)models have been proposed, and some realistic issues such as imperfect debugging, coverage and learning phenomena of software developers have been studied and incorporated in software reliability assessment [1], [6], [9]-[16].

The intense global competition software developing companies like Microsoft, IBM, Adobe and Wipro etc. are trying very hard to provide better value to its customers. They are trying to make their market presence by Up-gradations/add-ons or by adding some new functionality to the existing software system periodically. Technological breakthroughs are happening rapidly and these new innovations often take form of a new product. The concept of performance of an upgraded software system over its life cycle has been explained by using well known sigmoid curve [14]. And it has been seen that in the initial period of the software more efforts are put increasingly so that overall performance of the technology can be improved till attaining its natural performance limit. In general when software reaches a level when it attains its operational reliability level desired by the company, new upgraded software is introduced in the market. Due to demand of upgraded software in competitive environment the software developing department's sharp eye is always on market competition keeping in their mind the quality of software with the user's needs and requirements. Also upgrading a software application is a complex task. The upgraded and existing system may differ in the performance, interface and functionality etc. Although the developers upgrades the software in order to improve the software product, which also includes the possibility that the upgrade version will worsen, That's why there is risk involved into upgrading the software system. While upgrading an existing software system, only selected components of the software system are changed while the other will remain same to function. This process leads to an increase in the fault contents and the testing team is always interested in knowing the bugs present in the software which will decide the utility of up-graded software. Safe up-gradation can improve the behavior of the system and can preserve market for company however risky up-gradation can cause critical error in system. for example in October 2005, a glitch in a software upgrade caused trading on the Tokyo Stock Exchange to shut down for most of the day [1], in 1991 after changing three lines code in a signaling program which contained millions lines of code, the local telephone systems in California and the eastern seaboard came to stop]. Similar gaffes have occurred from important government systems [15], [17] to freeware on the internet. Sometimes Upgrades can worsen a product and user may prefer an older version. The typical software failure curve experienced by traditional software reliability growth model can be depicted by the Fig. 1. The traditional software reliability growth model fails to capture the error growth due to the software enhancements in user-end. In the useful-life phase, software firm introduces new add-ons or features on the basis of the user need. Software will experience an increase in failure rate, each time an upgrade is made. The failure rate decreases gradually, partly because of the defects found and fixed after the upgrades. Fig. 2 depicts the increase in failure rate due to the addition of new features in

Manuscript received August 9, 2013; revised October 13, 2013.

Ompal Singh and Jyotish N. P. Singh are with the Department of Operational Research, University of Delhi, Delhi, India (e-mail: drompalsingh@live.com, jyotishdu@gmail.com).

Anshul Tickoo is with the ASE, Amity University, Noida, Uttar Pradesh, India (e-mail: anshultickoo@hotmail.com).

P. K. Kapur is with the Amity International Business School, Amity University, Noida, India (e-mail: pkkapur1@gmail.com).

the software. Even fixing bugs may induce more software failures by fetching other defects into software. But if the goal of the firm is to upgrade the software by enhancing its reliability, then it is possible to incur a drop in software failure rate that can be done by redesigning or re-implementing some modules using better engineering approaches [17].

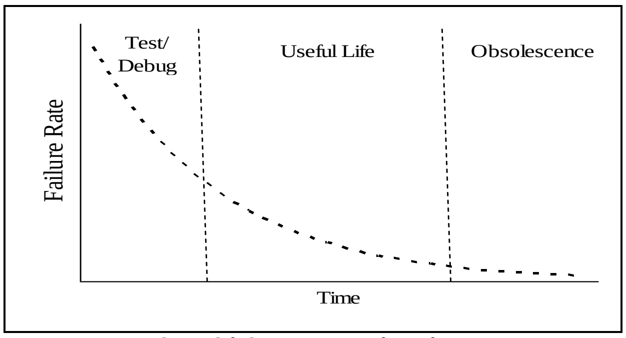

Fig. 1. Traditional failure rate curve for software systems.

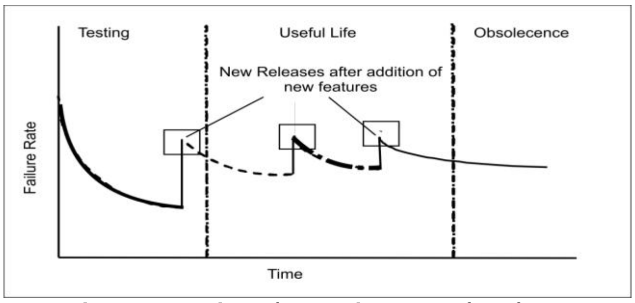

Fig. 2. Failure rate curve due to feature enhancements for software systems.

Recently Kapur *et al.*, [16], [17] developed a multi up-gradation reliability model, considering that cumulative faults in each generation depend on all previous releases and also assumes that fault is removed with certainty. But the proposed model is based on the assumption that the overall fault removal of the new release depends on the reported faults from the just previous release of the software and on the faults generated due to adding some new functionalities (add-ons/up-gradations) to the existing software system. Removal of maximum number of faults from each release is most important part in software development. Up gradation increases the complexity of software. All the developed multi up gradation modeling phenomenon consider one distribution function for fault removal process in every release i.e. one fault detection rate for all the release. As the complexity of the software increases due to enhancement in features fault removal process become harder. We have taken different detection rate for each release. In other words we have considered different distribution function for each release.

In this paper, we propose a general framework for multi up-gradation software reliability growth model incorporating different distribution function for each release. The rest of this paper is organized as follows. Section II describes assumptions and notations. Section II C briefly reviews literature on Software Reliability. In section II D, we propose a general framework for up-gradation problem. Section III discuses about how can derive new model in this environment. Section IV shows the experimental results through real data sets. We also analyze about parameter in each release. Finally, conclusions are given in Section V.

II. Modeling the Software Reliability Multi Up-Gradations

A. Assumptions

The basic assumptions of the model are as follows:

- The fault detection/ correction are modeled by non homogeneous poison process (NHPP).
- The number of faults detected at any time is proportional to the remaining number of faults in the software.
- Failure introduction rate is equally affected by faults remaining in the software.
- The number of faults in the beginning of the testing phase is finite.
- All faults are mutually independent from failure detection point of view.
- Software systems are subject to failure during execution caused by a fault remaining in the system.

B. Notations

$m(t)$	Expected mean number of faults removed by time *t*
$f(t)$	Probability density function.
$F_i(t)$	Probability distribution functions for i^{th} release.
t_{i-1}	Time for i^{th} release (*i*=1 to 4).
a_i	Initial fault content for i^{th} release (*i*=1 to 4).
$b_i(t)$	Time dependent fault detection rate function for i^{th} release.
b_i	Constant parameter for i^{th} release.

C. Review of Software Reliability Models

Several SRGMs have been proposed in software reliability literature under different set of assumption and testing environment to capture the cumulative number of faults removed in the software[1], [3], [5]-[7], [9]-[14], [16]. Further With the help of the hazard rate we can derive the mean value function of cumulative number of faults removed.

Let $N(t)$ $t \geq 0$ be a counting process representing the cumulative number of software failures by time *t*. The counting process *N* (*t*) is shown to be a NHPP with a mean value function $m(t)$ which represents the number of faults removed by time *t*.

Based on the NHPP assumption, it can be shown that *N* (*t*) has Poisson distribution with mean $m(t)$, *i.e.*,

$$Pr\{N(t)=n\}=\frac{m(t)^n \cdot \exp(-m(t))}{n!}, \quad n=0,1,2,...$$

By definition, the mean value function of cumulative number of failures, *m* (*t*), can be expressed in terms of the failure intensity function of the software, i.e.,

$$m(t)=\int_0^t \lambda(s)ds$$

Based on previous assumptions the differential equation describing the removal phenomenon can be given by:

$$\frac{dm(t)}{dt} = \frac{F'(t)}{1-F(t)}.\rho.\left[a-m(t)\right] = s(t).\left[a-m(t)\right] \quad (1)$$

Solving the above differential equation (1), under initial condition *m* (0) =0, we get mean value function as:

$$m(t) = a.F(t) \quad (2)$$

D. General Framework for Multi Up-Gradations Model

Our modeling framework with different type of distribution function is developed based on a unified framework proposed by literatures [6] and Mathematical model related to each release are given separately.

1) Modeling for first release

The First Release of software is released at $t = t_0$. Note that there is not any effect of another release on the in this release of the software and we consider tasting phase as classical SRGM. The mathematical equation of these finite numbers of faults removed is given as:

$$m_1(t) = a_1.F_1(t) \quad (3)$$

2) Modeling for second release

After first release, the company has information about the reported bugs from the users who used release 1, hence we must consider the effect of release 1 on this release. In this model note that testing phase for second release and operational phase of release1 happened in same interval time as $t_0 \le t < t_1$.Also in order to attract more customers, a company adds some new functionality to the existing software system. Adding some new functionality to the software leads to change in the code. These new specifications in the code lead to increase in the fault content. Now the testing team starts testing the upgraded system. In this period when there are two versions of the software, $a_1.(1-F_1(t))$ is the leftover fault content of the first version which interacts with new portion of detected faults i.e. $F_2(t-t_1)$. In addition a fraction of faults generated due to enhancement of the features are removed with new rate. The mathematical equation of these finite numbers of faults removed can be given by:

$$m_2(t) = (a_2 + a_1(1-F_1\ (t_1)).F_2\ (t-t_1)\ \ ,t_1 \le t \le t_2 \quad (4)$$

3) Modeling for third release

Similarly for release 3, we consider faults generated in third release and remaining number of faults from the second release and the corresponding mathematical equation can be represented as follows:

$$m_3(t) = (a_3 + a_2(1-F_2\ (t_2-t_1)).F_3\ (t-t_2)\ \ ,t_2 \le t \le t_3 \quad (5)$$

4) Modeling for fourth release

And similarly for release 4, the corresponding mathematical expression can be given by:

$$m_4(t) = (a_4 + a_3(1-F_3(t_3-t_2)).F_4(t-t_3)\ \ ,t_3 \le t \le t_4 \quad (6)$$

5) Modeling for i^{th} release

In general for release *i*, the corresponding mathematical expression can be given as:

$$m_i(t) = (a_i + a_{i-1}(1-F_{i-1}(t_{i-1}-t_{i-2})).F_i(t-t_{i-1})\ \ ,t_{i-1} \le t \le t_i \quad (7)$$

III. Derivation of New Model

In this section based on structure which we make in release 1 to 4, we derive multi up-gradation growth models with different detection rate in each release. The detection rate for first release is constant. The distribution function for the first release follows exponential distribution. The detection rate for the second release of the software is time dependent and the distribution function for second release is 2-stage Erlang distribution function. Similarly the mean value functions for the third and fourth release follow 3-stage Erlang growth curve and 4-stage Erlang growth curve respectively. Table I show the complete list of detection rate, distribution function and mean value function for each release.

TABLE I: List of Detection Rates

Release	Detection Rate $b(t)$	Distribution $F(t)$ Function	Mean Value Function $m(t)$
First	b_1	$(1-e^{-b_1 t})$	$a_1(1-e^{-b_1 t})$
Second	$\frac{b_2{}^2 t}{1+b_2 t}$	$(1-(1+b_2 t)e^{-b_2 t})$	$a_2(1-(1+b_2 t)e^{-b_2 t})$
Third	$\frac{b_3{}^3 t^2}{2(1+b_3 t+\frac{b_3{}^2 t^2}{2})}$	$(1-(1+b_3 t+\frac{b_3{}^2 t^2}{2})e^{-b_3 t})$	$a_3(1-(1+b_3 t+\frac{b_3{}^2 t^2}{2})e^{-b_3 t})$
Fourth	$\frac{b_4{}^4 t^3}{2(1+b_4 t+\frac{b_4{}^2 t^2}{2}+\frac{b_4{}^3 t^3}{2})}$	$(1-(1+b_4 t+\frac{b_4{}^2 t^2}{2}+\frac{b_4{}^3 t^3}{3})e^{-b_4 t})$	$a_4(1-(1+b_4 t+\frac{b_4{}^2 t^2}{2}+\frac{b_4{}^3 t^3}{3})e^{-b_4 t})$

A. Multi Up-Gradations Based on Generalized Erlang Distribution (i=1 to 4)

1) First release

$$m_1(t) = a_1(1-e^{-b_1 t})$$
$$= a_1 . F_1(t)\,, \qquad 0 \le t \le t_1$$

2) Second release

$$m_2(t) = (a_2 + a_1(1-F_1(t_1))).\left[1-(1+b_2 t)e^{-b_2 t}\right]$$
$$= (a_2 + a_1(1-F_1(t_1))F_2(t-t_1)\,, \qquad t_1 \le t \le t_2$$

3) Third release

$$m_3(t) = (a_3 + a_2(1-F_2(t_2-t_1))).\left[1-(1+b_3 t+\frac{b_3{}^2 t^2}{2})e^{-b_3 t}\right]$$
$$= (a_3 + a_2(1-F_2(t_2-t_1)).F_3(t-t_2), \quad t_2 \le t \le t_3$$

4) Fourth release

$$m_4(t) = (a_4 + a_3(1-F_3(t_3-t_2))).\left[1-(1+b_4 t+\frac{b_4{}^2 t^2}{2}+\frac{b_4{}^3 t^3}{3})e^{-b_4 t}\right]$$
$$= (a_4 + a_3(1-F_3(t_3-t_2))F_4(t-t_3)\,, \qquad t_3 \le t \le t_4$$

IV. Parameter Analysis

To check the validity of the proposed model and to describe the software reliability growth, it has been tested on tandem computer [1] four release data set. Also we have used non linear least square technique in SPSS software for estimation of parameters. Estimated value of parameters of each releases are given in Table II. Table III shows the comparison criterion of the four software releases. Fig. 3-Fig. 6 shows the estimated and the actual values of the number of faults removed for four releases. Based on data available given in Table II, the performance analysis of proposed model is measured by the four common criteria MSE, Bias, RMSPE, R2, and Variation.

TABLE II: Parameter Values

Parameter	Release 1	Release 2	Release 3	Release 4
a_i	113.12	67.51	52.607	44.046
b_i	0.1969	0.3948	0.604	0.431

TABLE III: Comparison Criterias

	Release 1	Release 2	Release 3	Release 4
R^2	.982	.995	.996	.995
Bias	.07041	.03703	.000541	0.01289
MSE	17.230	9.6021	3.4182	4.305
Variation	3.5645	2.8690	1.7207	1.9649

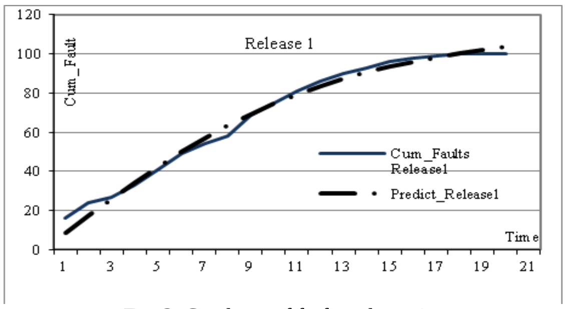

Fig. 3. Goodness of fit for release 1.

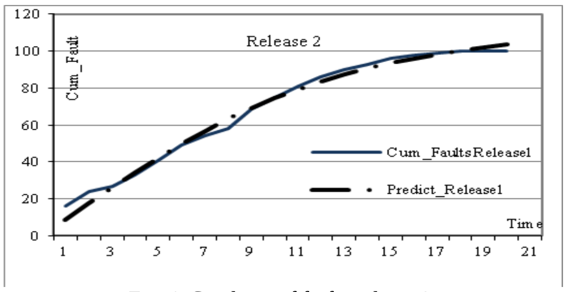

Fig. 4. Goodness of fit for release 2.

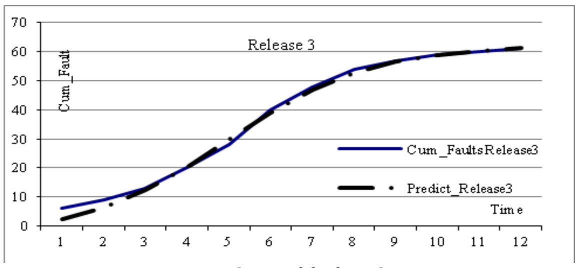

Fig. 5. Goodness of fit for release 3.

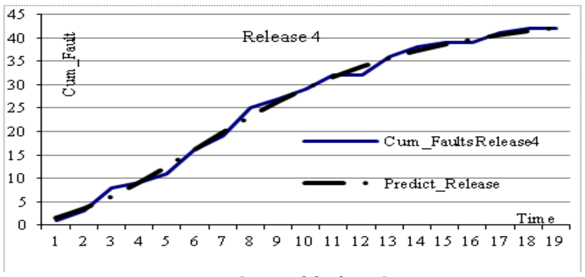

Fig. 6. Goodness of fit for release 4.

V. Conclusion

The modelling frameworks presented in this paper aim at extension of multi up-gradation modelling framework under the different distribution function for each release. The software reliability multi up-gradation model in this paper is based on the assumption that the overall fault removal of the new release depends on the faults generated in that release and on the leftover faults of just previous release (for each release). Experimental results show that the proposed gives a better fit to the observed data.

Acknowledgement

The research work presented in this paper is supported by grants to the forth author from Department of Science and Technology (DST) Grant No SR/S4/MS: 600/09, India.

References

[1] H. Pham, *System Software Reliability*, Springer-Verlag, 2006.

[2] H. Pham and X. Zhang, "NHPP software reliability and cost models with testing coverage," *European Journal of Operational Research*, vol. 145, pp. 443-454, 2003

[3] M. Ohba, "Software reliability analysis models," *IBM Journal of Research and Development*, vol. 28, no. 4, pp. 428–443,1984.

[4] L. V. Bart, Zimmermann and E. R. Marina, "Development of a methodological framework for examining science and technology in flanders," Katholieke Universiteit, Leuven, 2000.

[5] P. K. Kapur. R. B. Garg, and S. Kumar, *Contributions to Hardware and Software Reliability*, World Scientific, Singapore, 1999.

[6] P. K. Kapur, H. Pham, S. Anand, and K. Yadav, "A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation," *Communicated in IEEE Transactions on Reliability*, vol. 2, pp. 1036-1040, 2008.

[7] S. Bittanti, P. Bolzern, E. Pedrotti, and R. Scattolini, "A flexible modeling approach for software reliability growth," in *Software Reliability Modelling and Identification*, G. Goos and J. Harmanis, Eds., Springer, Berlin, Germany, 1998, pp. 101–140.
[8] S. Yamada, S. Osaki, and Y. Tanio, "Software reliability measurement and assessment methods during operation phase and their comparisons," *System and Computers in Japan*, vol. 23, no. 7, 1992.
[9] C. T. Lin and C. Y. Huang, "Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models," *The Journal of Systems and Software*, vol. 81, pp. 1025–1038, 2008.
[10] L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Trans. on Reliab*., vol. 28, no. 3, pp. 206–211, 1979.
[11] A. L. Goel, "Software Reliability Models: Assumptions, Limitations and Applicability," *IEEE Transactions on Software Engineering*, SE-11, pp. 1411-1423, 1985.
[12] K. Kanoun, B. M. Martini, and J. M. D. Souza, "A method for software reliability analysis and prediction application to the TROPICO-R switching system," *IEEE Trans. Software*, no. 17, no. 4, pp. 334–344, 1991.
[13] P. K. Kapur and R. B. Garg, "Software reliability growth model for an error-removal phenomenon," *Software Engineering Journal*, vol. 7, no. 4, pp. 291–294, 1992.
[14] P. K. Kapur, S. Younes, and S. Agarwala, "Generalized Erlang model with n types of faults," *ASOR Bulletin*, vol. 14, no. 1, pp. 5–11, 1995.
[15] S. Yamada, M. Ohba, and S. Osaki, "S-shaped software reliability growth models and their applications," *IEEE Trans. on Reliability*, vol. 33, no. 4, pp. 289–292, 1984.
[16] P. K. Kapur, H. Pham, A. Gupta, and P. C. Jha, *Software Reliability Assessment with or Applications*, UK: Springer, 2011.
[17] P. K. Kapur, A. Tandon, and G. Kaur, "Multi Up- gradation Software reliability Model," in *Proc. 2nd International Conference on Reliability, Safety and Hazard*, 2010, pp. 468-474.

Ompal Singh is currently a reader in the Department of Operational Research, University of Delhi, India. He has been an active member of the Society for Reliability Engineering, Quality and Operations Management (SREQOM) since 2000. He obtained his Ph.D. degree in software reliability (operational research) from University of Delhi (INDIA) in 2004. He has published extensively in Indian journals and abroad in the areas of Marketing, Software Reliability and Optimization.

Jyotish N. P. Singh is a research scholar in the Department of Operational Research University of Delhi. Presently he is working as a guest lecturer in Ramjas College, University of Delhi. He obtained his master's degree in operational research from University of Delhi, Delhi. His Research interest includes, mathematical modelling in software reliability, software up-gradation modelling and software release. He has published research papers in international and national journal of repute.

Anshul Tickoo is currently an assistant professor in Amity School of Engineering, Amity University, Noida. She is a research scholar in Amity School of Engineering and Technology, Amity University. She obtained his M. tech (CSE) degree from Amity University and has done B. Tech (instrumentation technology) from M. S Ramaiah institute of technology (Bangalore). Her Research interest includes, software up-gradation modelling and Software release.

P. K. Kapur is a professor at Amity International Business School, Amity University, Noida and former head of the Department of Operational Research and former dean of the Faculty of Mathematical Sciences, University of Delhi. He has been the president of Society for Reliability Engineering, Quality and Operations Management (Regd.) since 2000 and former president of Operational Research Society of India. He obtained his Ph.D. degree in reliability theory (operational research) from University of Delhi in 1977. He has published extensively in Indian journals and abroad in the areas of Marketing, Hardware Reliability, Optimization, Queuing Theory and Maintenance and Software Reliability (more than 200 papers). He has recently published a book on "Software Reliability Assessment with OR Applications" (Springer, UK, 2011).