  Vol. 2(1), Jan 2012, pp. 23-31

On the Development of Software Reliability Growth Model Based on Features Enhancement

Jagvinder Singh[1], Ompal Singh[1], Deepti Aggrawal[1] and Indarpal Singh[2,*]
[1]Department of Operational Research, University of Delhi, India
[2,*]D.N.(P.G.) College, Gulaothi, Bulandshahar, UP, India

The software industry can be considered as the typical high technology industry where rate of innovation and knowledge creation plays a pivotal role for continued firm growth. In the last few decades it has been observed that the world of software development management has evolved rapidly due to the intensified market competition. In particular the use of feature-addition model of software products in the industry is fast becoming the commonplace. The up-gradation model can be characterized by increasing the number of features in the software that will give the firm competitive edge in the market. The up-gradation of the system is done by extending it through add-ons, interfacing with other applications etc. Continuous up-gradation of software's also brings complexity in the systems once it failed to work properly. In recent years, there has been a growing interest to predict the link between the rates of failure and the reliability of software. Many software reliability growth models (SRGMs) have been proposed over past three decades that estimate the reliability of a software system as it undergoes changes through the removal of failure causing faults. But unfortunately most of the models didn't consider anything about the increase in failure rate once an up-gradation is made on the software. The objective of this paper is to propose the software reliability growth model that incorporates the effect of enhancement of features on software during testing and debugging process. Results have been supplemented with numerical examples.

Keywords: Software Reliability Growth Model (SRGM), Fault-Removal, Up-Gradation.

1. INTRODUCTION

The software industry can be considered as the archetypal high technology industry where innovation and knowledge creation form the primary fuel for continued firm growth. Often the rate of innovation achieved by a firm shapes its evolutionary path as well as its future growth. The innovation related factors like development process, software testing and debugging process and team structure have significant impact on a firm's future growth potential. In the last few decades it has been observed that the world of software development management (i.e. new product development, technology alliance etc.) has evolved rapidly due to the intensified market competition. In particular, the use of continuous up-gradation model of software products fast becoming commonplace due to the shrinking budget, expanding system requirements and on the other hand accelerating rate of software enhancement. The up-gradation of the system is done by extending it through add-ons, interfacing with other applications etc. The growing trend towards up-gradation of softwares has taken the original concept of reprocess it into a completely different arena and due to that it has also presented many challenges to software developers attempting to enter this

new arena. Due to the continuous up-gradation of softwares, they are rapidly becoming an integral part of nearly every engineered product, controlling the manufacturing process for products, commercial aircrafts, nuclear power plants, medical devices, weapon systems, aerospace systems, automobiles, public transportation systems, and so on. At the same time it brings complexity in the systems once it fails to work properly. Software failures may occur due to errors, indistinctness, oversights or misspecification that the software is supposed to satisfy, incompetence in writing code, inadequate testing, incorrect or unexpected usage of diction of software failures can contribute substantially to long-term financial success and are an effective strategy to increase the reliability of the software. Software Reliability is an important attribute of software quality. It is hard to achieve as the complexity associated with software tends to be high. Pan[1] argued that due to the rapid growth of software size and ease of doing so by upgrading the software, a high degree of complexity is brought in it and due to that it becomes hard to reach a certain level of reliability. Mathematical models have been proved to be useful tools for understanding the structure and functioning of software, predicting the reliability of software and prescribing the best course of actions under known constraints. In recent years, there has been a growing interest to predict the link between the rates of failure and the reliability of software. Software reliability is the probability that given software will be functioning correctly under a given environment during a specified period of time. Many software reliability growth models (SRGM) have been proposed over the past three decades that estimate the reliability of a software system as it undergoes changes through the removal of failure causing faults. The Goel-Okumoto model[2] is among the most quoted publications in this area. The model is a purely exponential and based on the assumption that the faults are uniformly distributed where each fault has an equal chance of detection. Yamada, Obha and Osaki[3] and Kapur, Younes and Agrawal[4] proposed a software reliability growth model assuming that each fault detection leads to exactly one removal. Kapur and Garg[5], Obha[6] and Bittani *et al.*[7] proposed models where fault detection rate is defined as function of number of faults already removed. Many software reliability growth models were proposed in literature based on non homogenous Poisson process (NHPP) (Obha and Yamada[8]; Yamada, Ohtera and Narihisa[9]; Kapur, Garg and Kumar[10]; Yamada, Tamura and Kimura[11] etc.) on the assumption that the initial fault contents of software remain the same for entire life cycle. But unfortunately, these models do not consider anything about the drastic increase in failure rate, a software will experience each time an upgrade is made. One of the objectives of this paper is to understand the role of enhancement of features during test-phase that aid to increase in failure contents of software. The article is divided into the following sections: contribution of the study, model development, data, parameter estimation and comparison and conclusion.

2. CONTRIBUTION OF STUDY

Significant improvement in software reliability calls for innovative methods for developing software, determining its readiness for release, and predicting field performance. Software products which are introduced in the market can be one-off type (i.e. no enhancements is made in the software) or can be a product which is periodically upgraded with new features upto a level. To acquire the ability to produce and market new products, companies make

huge investments and the failure of such products can be damaging. Therefore to optimize the companies' goals, they try to reduce the risk by staggering the new ideas to a sequence of product-features introduced over a period of time and satisfying multiple market segments.

Since the pioneering work of Goel-Okumoto model[2], many authors have proposed software reliability growth models to measure the failure rate of software. This high level of interest in measuring the failure rate has resulted in a large body of publications (Musa, Iannino and Okumoto[12], Abdel, Chen and Littlewood[13], Kapur and Garg[14] etc.). All the models are based on the assumption that when the software is first manufactured, the initial number of faults is high but then decreases as the fault components are identified and removed or the components stabilize. The software then enters the useful life phase where more faults are removed and the failure rate levels of gradually. The typical software failure curve experienced by traditional software reliability growth model can be depicted by the Fig. 1. Thus, the traditional software reliability growth model failed to capture the error growth due to the software enhancements in the test phase. In the useful-life phase as the software firm introduces new add-ons or features on the basis of the user needs, software will experience a drastic increase in failure rate each time an upgrade is made. The failure rate levels off 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 the software. Due to the feature upgrades, the complexity of software is likely to be increased as the functionality of software is enhanced. 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[1]. The objective of this paper is to propose a software reliability growth model under the assumption that software will experience a drastic increase in failure content each time an upgrade is made by the software firm.

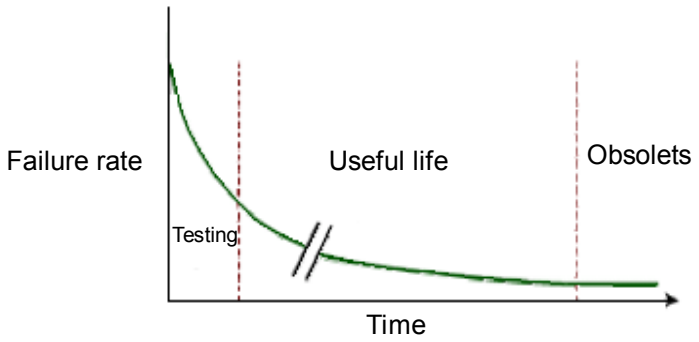

Fig. 1: Traditional Failure Curve

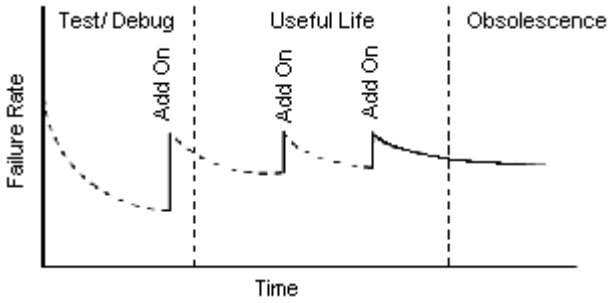

Fig. 2: Failure Curve due to Features Enhacenent

Software Reliability Growth Models (SRGMs) play an important role due to their ability to predict the fault detection / removal phenomenon during testing. Several classes of SRGMs have been proposed and validated on test data in the literature. A proliferation of software reliability models has emerged as people try to understand the characteristics of how and why software fails, and try to quantify software reliability. Over 200 models have been developed since the early 1970s, but how to quantify software reliability still remains largely unsolved. One group of models that has been widely used and researched is the Non-Homogeneous Poisson Process (NHPP) models. The proposed model is also based on NHPP framework. The basic assumption of NHPP process can be given as follows.

3. BASIC ASSUMPTIONS

Let $\{N(t);\ t = 0\}$ be a counting process representing the cumulative number of software failures by time '*t*'. The $N(t)$ process is shown to be a NHPP with a mean value function $m(t)$. Mean value function represents the expected number of faults removed by time '*t*'.

$$\Pr\{N(t) = n\} = \frac{(m(t))^n}{n!} e^{-m(t)}, \qquad n = 0,1,2... \qquad \text{-(1)}$$

and $$m(t) = \int_0^t \lambda(x)dx \qquad \text{-(2)}$$

The proposed models are based upon the following basic assumptions:

(i) Failure /fault removal phenomenon is modeled by NHPP.
(ii) Software is subject to failures during execution caused by faults remaining in the software.
(ii) Failure rate is equally affected by all the faults remaining in the software.
(iv) Fault detection/removal rate may change at any time moment.
(v) Up-gradation is done continuously after a fix time *'s'.*

4. NOTATIONS

a : Expected number of faults in the software.
b_1 : Detection rate before the time *'s'*.
b_2 : Detection rate after the time *'s'*.
$m(t)$: Expected Number of faults removed.
s : Time after which we start adding new features.
α : Rate of fault addition due to adding a new feature in the software.

5. MODEL DEVELOPMENT

In this section, we formulate probability distribution based software reliability growth models incorporating the affect of adding new features in the software system. As discussed, integrating new features increases complexity in the software and can be the cause of more faults in the system. Even fixing bugs may induce more software faults by fetching other defects into software. In general, before adding any additional features in the system (i.e. before time '*s*') the initial fault contents in the software system remain constant and are detected and eliminated during the testing phase, and the number of faults remaining in the software system gradually decrease as the testing process goes on. On the other hand, adding new features in the software introduces more faults in the system. Assuming the software firm starts incorporating additional features after time '*s*' and each additional feature puts in faults at the rate '*α*', the proposed model shows continuous flow dynamic character. The model for different situations can be described as follows:

$$\frac{dm(t)}{dt} = \begin{cases} b_1(a - m(t)) & 0 \le t < s \\ b_2(a(1+\alpha t) - m(t)) & t > s \end{cases} \quad \text{-(3)}$$

Under the initial condition at:

$t = 0$, $m(t) = 0$ and

$t = s$, $m(t) = m(s)$

The following solution of equation (3) is obtained

$$m(t) = \begin{cases} a(1 - e^{-b_1 .t}) & 0 \le t < s \\ a\left[1 - e^{-b_1 .s - b_2 (t-s)} - \dfrac{\alpha}{b_2}\left\{(1 - b_2 .t) - (1 - b_2 .s)\right\} e^{-b_2 (t-s)}\right] & t > s \end{cases} \quad \text{-(4)}$$

From equation (4) it can be observed that the parameter *α* (i.e. rate of fault addition due to

additional feature added) plays a significant role during estimation of the expected number of faults removal.

6. DATA

To illustrate the estimation procedure and application of the SRGM (existing as well as proposed) we have validated the proposed model on two different data-sets, which are as follows:

6.1. Data Set 1(DS-1)

The first data set (DS-1) was collected during 35 months of testing a radar system of size 124 KLOC where 1301 faults were detected during the testing. This data is cited from Brooks and Motley[15]. The change-point for this data set is 17th month.

6.2. Data Set 2(DS-2)

The second data set (DS-2) was collected during 19 weeks of testing a real time command and control system, where 328 faults were detected during the testing. This data is cited from Ohba[6]. The change-point for this data set is 6th week.

7. PARAMETER ESTIMATION AND COMPARISON

7.1. Model Validation

The parameters of the models have been estimated using the statistical package SPSS. The statistical estimates of coefficients of the proposed model for non-cumulative data are given in Table 1.

7.2. Model Comparison

The proposed model has been compared with the G-O model on the basis of different comparison criteria. These are: Mean Square Fitting Error (MSE), Coefficient of Multiple Determination (R^2), Bias, Variation and Root Mean Square Prediction Error. Table 2 summarizes the value of all the comparison criteria for each of the models for two different datasets. From Table 2, it is clear that SSE, Bias, Variation and RMSPE of the proposed model are the least in comparison to those of the G-O model. Also, the adjusted R^2 value of the proposed model is relatively higher in comparison to that of the G-O model for the two different datasets. From Table 2 and Fig. 3 and 4, it can be concluded that the proposed model works better for the two different datasets. Apart from goodness of fit criteria, the parameters of the proposed model provide important information. From Table 1, it can be seen that the estimates of rate of error generation due to enhancement of features in the software α are relatively high, which suggests that up-gradation of the software plays a vital role in increasing the fault contents.

Table 1: Parameter Estimates of the GO Model and the Proposed Model

DS-I		DS-II		
Parameters	GO Model	Proposed Model	GO Model	Proposed Model
a	2986	2297	707.53	500
b_1	0.0177	0.024	0.0322	0.052
b_2	-	0.028	-	0.056
α	-	0.002	-	0.007

Table 2: Model Comparison

DS-I		DS-II		
Parameters	GO Model	Proposed Model	GO Model	Proposed Model
R^2	0.944	0.973	0.986	0.9880
bias	-3.071	1.833	-1.166	0.0149
variation	619.341	76.494	51.290	11.5010
RMSPE	620.07	76.516	51.300	11.5000
MSE	41523	19906	2656.48	2380.74

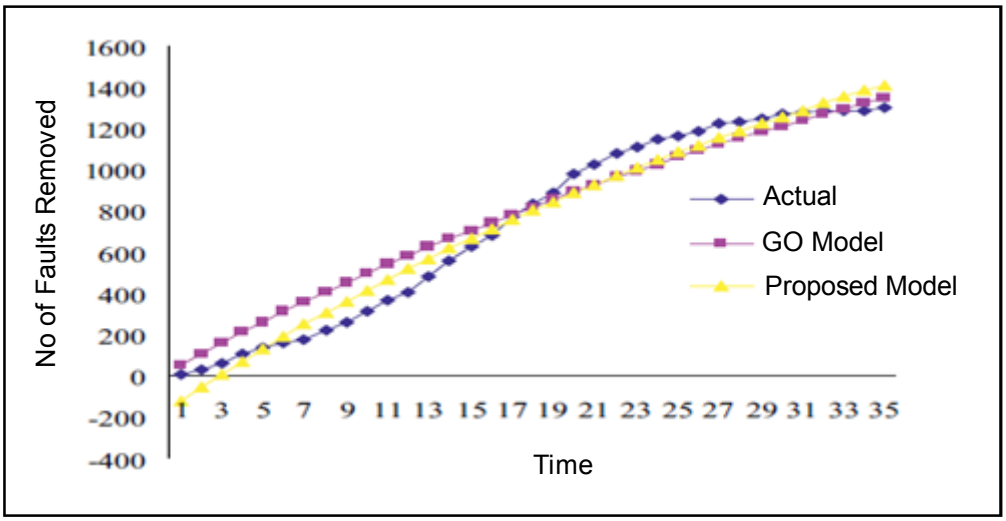

Fig. 3: Actual Faults vs Estimated Faults for DS-I

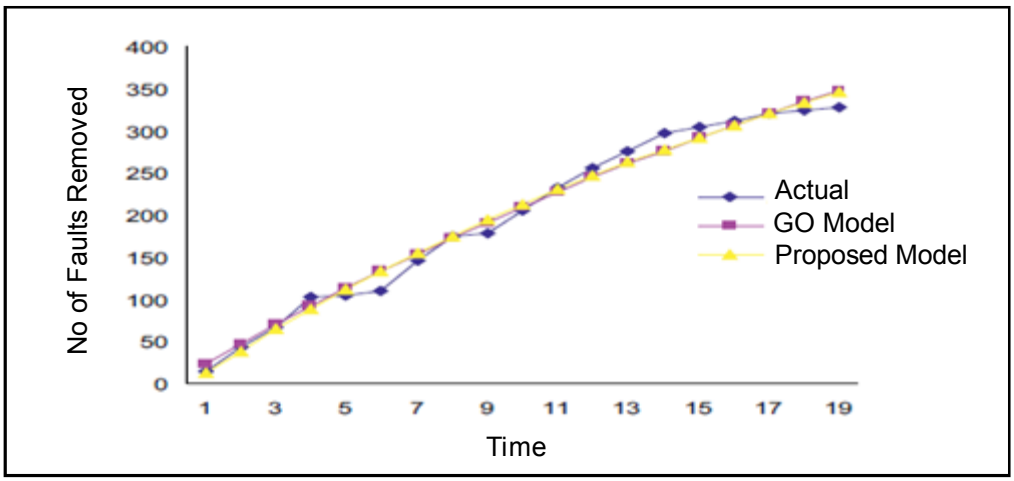

Fig. 4: Actual Faults vs Estimated Faults for DS-II

8. CONCLUSION

In this paper a more general SRGM has been developed based on simple assumptions, consistent with the basic software reliability growth modelling literature. The proposed model embeds a broader theoretical framework which accounts for the interactions between enhancement of features in the software and resultant error growth. In the last few decades it has been observed that the world of software development management has evolved rapidly due to the intensified market competition. In particular, the use of continuous up-gradation model of products in software industry is becoming common-place. The continuous up-gradation model can characterize the introduction of new features of the software in the market place to get the competitive edge. As the software firm introduces new add-ons or features, software will experience a drastic increase in failure rate each time an upgrade is made. Due to the feature upgrades, the complexity of software is likely to be increased as the functionality of software is enhanced. Even fixing bugs may induce more software failures by fetching other defects into software. In this paper, we have proposed a software reliability growth model under the assumption that software will experience a drastic increase in failure content each time an upgrade is made by the software firm.

REFERENCES

[1] Jiantao, Pan; "*Software Reliability*", Carnegie Mellon University (working paper) available at: http://www.ece.cmu.edu/~koopman/des_s99/sw_reliability, 1999.

[2] Goel, A.L. and Okumoto, K.; "*Time dependent error detection rate model for software reliability and other performance measures*", IEEE Transactions on Reliability-R, Vol. 28(3), pp. 206–

211, 1979.

[3] Yamada, S., Ohba, M. and Osaki, S.; "*S-shaped software reliability growth modelling for software error detection*", IEEE Trans on Reliability-R, Vol. 32(5), pp. 475–484, 1983.

[4] Kapur, P.K., Younes S. and Agrawala S.; "*Generalized Erlang Software Reliability Growth Model*", ASOR Bulletin, Vol. 14(1), pp. 5–11, 1995.

[5] Kapur, P.K. and Garg, R.B.; "*A software reliability growth model for an error removal phenomenon*", Software Engineering Journal, Vol. 7, pp. 291–294, 1992.

[6] Ohba, M.; "*Software reliability analysis models*", IBM Journal of research and Development, Vol. 28, pp. 428–443, 1984.

[7] Bittanti, S., Bolzern, P., Pedrotti, E. and Scattolini, R.; "A *flexible modelling approach for software reliability growth*", Software Reliability Modelling and Identification, Berlin: Springer Verlag, Vol. 341, pp. 101–140, 1988.

[8] Obha, M. and Yamada, S.; "*S-shaped software Reliability Growth Model*", Proceedings of the 4th National Conference on Reliability and Maintainability, pp. 430–436, 1984.

[9] Yamada, S., Osaki, S. and Narihisa, H.; "*Discrete models for software reliability evaluation",* u: Basy A.P. (ur.) Reliability and quality control, London: Elsevier, str., pp. 401–412, 1986.

[10] Kapur, PK., Garg, R.B. and Kumar, S.; "*Contributions to hardware and software reliability*", Singapore: World Scientific, 1999.

[11] Yamada, S., Tamura, Y. and Kimura, M.; "*A Software Reliability Growth Model for A Distributed Development Environment*", Electronics & Communication in Japan, Part 3, 2003.

[12] Musa, J.D., Iannino, A. and Okumoto, K.; "*Software Reliability: Measurement*", Prediction, Application, McGraw- Hill, New York, 1987.

[13] Abdel, A.A., Chan, P.Y. and Littlewood, B.; "*Evaluation of Competing Software Reliability Predictions*", IEEE Trans. on Software Engineering, Vol. 12(9), pp. 950–967, 1986.

[14] Kapur, P.K. and Garg, R.B.; "*A Software reliability growth model under imperfect debugging*", R.A.I.R.O, Vol. 24, pp. 295-305, 1990.

[15] Brooks, W.D. and Motley, R.W.; "*Analysis of discrete software reliability models–Technical report RADC-TR-80- 84*", New York: Rome Air development center, 1980.