Proceedings of the National Conference; INDIACom-2007
Computing for Nation Development, February 23 – 24, 2007
Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

A Flexible Discrete Software Reliability Growth Model With Change-Point

P. K. KAPUR* D. N. GOSWAMI# SUNIL K. KHATRI$ PRASHANT JOHRI+
* Department of Operational Research, University of Delhi, Delhi, pkkapur1@gmail.com
# S.O.S in Computer Science, Jiwaji University, Gwalior-474011
$Mother Teresa Institute of Management, Guru Gobind Singh Indraprastha University, Delhi, sunilkkhatri@gmail.com
+Integrated Academy of Management and Technology, UP Technical University, johri.prashant@gmail.com

ABSTRACT
This paper presents a flexible discrete software reliability growth model (SRGM) and introduces the concept of change-point in fault removal rate(FRR). Most of the discrete SRGMs discussed in the literature seldom consider the change in FRR. In real software development environment, the FRR need not be same throughout the testing process. Due to the complexity of the software system and the incomplete understanding of the software requirements, specifications and structure, the testing team may not be able to detect the failures at same rate. The model adopts the number of test occasions (cases) as a unit of fault detection (removal) period. The model has been validated, evaluated and compared by applying it on actual failure/fault removal data sets cited from real software development projects. The results show that the proposed model provides improved goodness of fit and predictive validity for software failure/fault removal data.

KEY WORDS
Software reliability, Software reliability growth model (SRGM), Non-homogeneous Poisson process (NHPP), Software testing, Test occasions (cases).

Acronyms

SRGM	Software Reliability Growth Model
NHPP	Non-homogeneous Poisson Process
FRR	Fault Removal Rate
MLE	Maximum Likelihood Estimate
PGF	Probability Generating Function.
SSE	Sum of Squared Errors
CP	Change-Point
DS	Data Set
RPE	The Relative Prediction Error

1. Introduction

Computer systems covers every aspect of our daily life. Although this had benefited the society but it has also made our lives more critically dependent on their correct functioning. Software reliability assessment is important to evaluate and predict the reliability and performance of software system. Several SRGMs have been developed in the literature to estimate the fault content and fault removal rate per fault in software[4,6]. Models have been developed under various sets of assumptions representing factors affecting the testing phase[4, 6, 11]. Goel and Okumoto [3] have proposed NHPP based SRGM assuming that the failure intensity is proportional to the number of faults remaining in the software. The model is very simple and can describe exponential failure curves. Ohba [13] refined the Goel-Okumoto model by assuming that the fault detection / removal rate increases with time and that there are two types of faults in the software. SRGM proposed by Bittanti et al. [1] and Kapur and Garg [7] have similar forms as that of Ohba [13] but are developed under different set of assumptions. Bittanti et al. [1] proposed an SRGM exploiting the fault removal (exposure) rate during the initial and final time epochs of testing. Whereas, Kapur and Garg [7] describe a fault removal phenomenon, where they assume that during a removal process of a fault some of the remaining faults may also be removed. These models can describe both exponential and S-shaped growth curves and therefore are termed as flexible models.

NHPP based SRGMs are generally classified into two groups. The first group contains models, which use the execution time (i.e., CPU time) or calendar time. Such models are called continuous time models.

The second group contains models, which use the test cases as a unit of fault removal period. Such models are called discrete time models, since the unit of software fault removal period is countable (Yamada et. al.[18], Inoue and Yamada [5]; Kapur et. al. [7]; Pham [14]; Musa [12]; Yamada et. al. [19]). A test case can be a single computer test run executed in an hour, day, week or even month. Therefore, it includes the computer test run and length of time spent to visually inspect the software source code. A large number of models have been developed in the first group while fewer are there in the second group due to the difficulties in terms of mathematical complexity involved.

Lately attempts have been made to develop flexible discrete SRGMs. In this paper a discrete flexible SRGM is developed using Probability Generating Function (P.G.F) incorporating the concept of change-point. It is further shown, how continuous time SRGM can be derived from the discrete model.

The utility of discrete reliability growth models cannot be under estimated. As the software failure data sets are discrete, these models many times provide better fit than their continuous time counterparts. Therefore, in spite of difficulties in terms of mathematical complexity involved, discrete models are proposed regularly. Most of discrete models discussed in the literature seldom differentiate between the failure observation and fault removal processes. In real software development scene, the number of failure observed can be less than or more than the number of error removed. Kapur and Garg [7] has discussed the first case in their Error removal phenomenon flexible model which shows as the testing grows and testing team gain experience, additional number of faults are removed without them causing any failure. But if the number of failure observed is more than the number of error removed then we are having the case of imperfect debugging. Flexible SRGM due to Kapur and Garg is able to capture exponential as well as S-shaped failure curve[9, 10].

Due to the complexity of the software system and the incomplete understanding of the software requirements, specifications and structure, the testing team may not be able to detect the faults at same rate. As the testing progresses, the FRR changes. The time at which FRR change is called change-point. There can be multiple change-points in the testing process[15, 16, 20, 21].

In this paper, a flexible discrete SRGM incorporating change-point concept has been proposed. The proposed model has been validated and evaluated on actual software failure/fault removal DS and compared with discrete version of KG model. The importance and utility of discrete time modeling have been highlighted.

2. Software Reliability Modeling

2.1. Model Development

Most of the software reliability growth models assume that the fault removal phenomenon also describes the failure phenomenon. In reality this may not always be true. Fault, which is removed consequent to a failure, is known as a leading fault. While removing the leading faults, some other faults are removed which may have caused failure in future. These faults are known as dependent faults.

Kapur and Garg [7] have described the above phenomenon in their SRGM based on the Non-homogeneous Poisson Process (NHPP). The mean value function of the failure phenomenon describes the removal process.

2.2. Model Assumptions

The model developed below is based upon the following basic assumptions:

1. Failure observation / fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. On a failure, the fault causing that failure is immediately removed and no new faults are introduced i.e. fault removal process is perfect..
4. The expected number of faults removed between n^{th} and $(n+1)^{th}$ case is proportional to the expected number of faults remaining.
5. Faults present in the software are of two types: mutually independent and mutually dependent.
6. The fault detection rate is proportional to the current fault content in the software and the proportionality increases linearly with each additional fault removal.

2.3 Model Notations

a	Initial fault content of the software.
$b(n)$	FRR function dependent on the number test cases
$m_r(n)$	Mean number of faults removed by n number of test cases.
β	A constant parameter in the FRR function.
δ	Constant time difference interval.
c	Fault removal rate of additional removed faults.
b_1	FRR before CP
b_2	FRR after CP

2.4 Model Formulation

Under the above assumptions, the expected number of faults removed between n^{th} and $(n+1)^{th}$ test case is proportional to the number of faults remaining after the execution of n^{th} test run, satisfies the following difference equation:

Under these assumptions the fault removal intensity per unit time can be written as:

$$\frac{m_r(n+1)-m_r(n)}{\delta}=b\left(a-m_r(n)\right)+c\frac{m_r(n)}{a}\left(a-m_r(n)\right) \qquad \ldots(1)$$

The mathematical equation describing KG model can be rewritten as

$$\frac{m_r(n+1)-m_r(n)}{\delta}=\left[b+c\frac{m_r(n)}{a}\right]\left(a-m_r(n)\right) \qquad \ldots(2)$$

or, $$\frac{m_r(n+1)-m_r(n)}{\delta}=b(n)\left(a-m_r(n)\right) \qquad \ldots(3)$$

Flexibility in KG model can be captured by proposing a logistic time dependent form for *b(n),* given by

$$b(n)=\frac{b}{1+\beta(1-b\delta)^n} \qquad \ldots(4)$$

Consequently, the model takes the following form

$$\frac{m_r(n+1)-m_r(n)}{\delta}=\frac{b}{1+\beta(1-b\delta)^n}\left(a-m_r(n)\right) \qquad \ldots(5)$$

Solving equation(5), using PGF under the initial condition m_r (n=0) =0, we get the solution as

$$m_r(n)=a\left[\frac{1-(1-b\delta)^n}{1+\beta(1-b\delta)^n}\right] \qquad \ldots(6)$$

The structure of the model is flexible. The shape of the growth curve is determined by the parameters b and c and can be either exponential or S-shaped.

By substituting $\beta=(c/b)$ and replacing b by $(b+c)$ we observe equation (6) is identical to the form given by Kapur and Garg. The S-shapedness in the cumulative curve is created by S-shaped $b(n)$.

FRR during testing may vary because of changes in testing skill, testing strategy and testing environment. As a consequence, fault removal rate before the change-point is different from the fault removal rate after change-point.:

$$\frac{m_r(n+1)-m_r(n)}{\delta}=b(n)\left(a-m_r(n)\right) \qquad \ldots(7)$$

where $$b(n)=\frac{b_1}{1+\beta(1-b_1\delta)^n};0\le n<\eta_1 \qquad \ldots(8)$$

$$b(n)=\frac{b_2}{1+\beta(1-b_2\delta)^n};n\ge\eta_1 \qquad \ldots(9)$$

where η_1 is the change-point.

Case 1: $(0 \le n < \eta_1)$

Solving the difference equation (7) substituting $b(n)$ from (8), using the probability generating function under the initial condition at $n = 0$, $m(n) = 0$, we get

$$m_r(n)=a\left[\frac{1-(1-b_1\delta)^n}{1+\beta(1-b_1\delta)^n}\right] \qquad \ldots(10)$$

Case 2: $(n \ge \eta_1)$

Solving the difference equation (7) substituting $b(n)$ from (9), using the probability generating function with the initial condition at $n = \eta_1$, $m_r(n) = m_r(\eta_1)$, we get

$$m_r(n)=a\left[\frac{1-(1+\beta)\left(1+\beta(1-\delta b_2)^{n_1}\right)(1-\delta b_2)^{(n-\eta_1)}(1-\delta b_1)^{\eta_1}}{1+\beta(1-\delta b_1)^{n_1}\left(1+\beta(1-\delta b_2)^{n}\right)}\right] \qquad \ldots(11)$$

3. Estimation of Parameters

Parameters estimation is of primary concern in software reliability prediction. For this, the failure data is collected and is recorded in either of the following two formats-the first approach is to record the time between successive failures while second way is to collect the number of failures experienced at regular testing intervals. If failure data is available then the values of the unknown parameters can be estimated by using either maximum Likelihood Method or by using the technique of least square method. The brief description of these two techniques is:

Maximum Likelihood Method: The MLE procedure when the failure data is given in the form (n_i, x_i), $i=1,2,3\ldots k$, where x_i is the cumulative number of faults removed by n_i test cases ($0< n_1 < n_2 <\ldots< n_k$) and n_i is the accumulated test runs spent to remove x_i faults. The Likelihood function L is given as:

$$L\left(Parameters|(n_i,x_i)\right)=\prod_{i=1}^{k}\frac{\left[m(n_i)-m(n_{i-1})\right]^{x_i-x_{i-1}}}{(x_i-x_{i-1})!}e^{-m(n_k)}$$

The MLE of the parameters of SRGMs can be obtained by maximizing L with respect to the model parameters.

Least square method: In this method, the sum of square of the difference between observed value and the value estimated by the model is minimized. If the failure data consists of k pairs of sample values (n_i, x_i), $i=1,2,3\ldots k$, where x_i is the cumulative number of faults removed by n_i test cases ($0< n_1 < n_2 <\ldots< n_k$) and n_i is the accumulated test runs spent to remove x_i faults. Let the estimated value of the number of faults removed by n_i test cases be $\hat{m}(n_i)$. Then parameter estimation by least square method consists of minimizing the sum of squares of the deviation between actual and estimated values i.e.

$$Sum\,of\,Squares=\sum_{i=1}^{k}\left(\hat{m}(n_i)-x_i\right)^2$$

Bayesian Analysis: When no failure data or very small amount of the failure data is available then it is not possible to estimate the values of the parameters by using above two specified techniques. In such case, the parameters are not assumed to be fixed at some unknown value, but they are assumed to follow some probability distribution, known as prior distribution. Given the software reliability model and the assumption about the distribution of the model parameters, it is possible to obtain the distribution of random variable $N(n)$ (known as posterior distribution) and its expected value $m(n)$ i.e. mean value function.

3.1 Parameter Estimation for the proposed model

In this paper, the maximum likelihood method is used to estimate the parameters *(a_0, b_0, p, α, β)* of the proposed model. Since the DS used in this paper are given in the form of pairs (n_i, x_i), $i=1,2,3\ldots k$, where x_i is the cumulative number of faults removed by n_i test cases ($0< n_1 < n_2 <\ldots< n_k$) and n_i is the accumulated test runs spent to remove x_i faults. The Likelihood function L is given as:

$$L\left(Parameters|(n_i, x_i)\right)=\prod_{i=1}^{k}\frac{\left[m(n_i)-m(n_{i-1})\right]^{x_i-x_{i-1}}}{(x_i-x_{i-1})!}\,e^{-m(n_k)} \qquad \ldots (12)$$

The likelihood function or the log Likelihood function of (12) can be maximized with respect to the parameters to find their estimates. Following constraints can also be used: $a_0>0$, $0<b_0<1$, $0<p\leq 1$, $\alpha\geq 0$, $\beta\geq 0$.

Taking natural logarithm of (12) we get:

$$\ln\ L=\sum_{i=1}^{k}(x_i-x_{i-1})\ln\left[m(n_i-n_{i-1})\right]-m(n_k)-\sum_{i=1}^{k}\ln\left[(x_i-x_{i-1})!\right] \qquad \ldots (13)$$

The MLE of the parameters of SRGMs can be obtained by maximizing (12) or (13) with respect to the model parameters.

4. Model Validation

To check the validity of the proposed model to describe the software reliability growth, it has been tested on two DS. The DS-I is cited from (Woods [17]) in which 100 faults were detected after testing for 20 weeks. Change-point is taken at . The DS-II is cited from (Brooks and Motley [2]) in which 1301 faults were detected after testing for 35 months. Change-point is taken at

4.1. Model Evaluation

The performance of SRGM is judged by their ability to fit the past software failure occurrence / fault removal data and to predict satisfactorily the future behavior of the software failure occurrence/fault removal process (Musa et al. [12], Kapur et al. [8]). Therefore, we use two types of comparison criteria:

1. The Goodness of Fit Criteria.
2. The Predictive Validity Criterion.

4.2. The Goodness of Fit Criteria

The Sum of Squared Error (SSE): SSE measures the distance of a model estimate value from the actual data, as follows:

$$SSE=\sum_{i=1}^{k}\left(\hat{m}(n_i)-x_i\right)^2 \qquad \ldots (14)$$

Where k is the number of observations, $\hat{m}(n_i)$ is the estimated cumulative number of failures by n_i test case obtained from the fitted mean value function (i.e., SRGM), and x_i is the total number of failures observed by n_i test cases. Lower value of SSE indicates less fitting error, thus better goodness of fit.

The smaller the metric value the better the model fits relative to other models run on the same DS.

R Squared (R^2): Goodness-of-fit measure of a linear model, sometimes called the coefficient of determination. It is the proportion of variation in the dependent variable explained by the regression model. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well.

4.3. The Predictive Validity Criterion

Predictive validity is defined as the ability of the model to determine the future failure behavior from present and past failure behavior. This criterion was proposed by Musa et al. [12]. Suppose n_k be the last test case, x_k is number of faults detected during the interval $(0, n_k]$, and $\hat{m}(n_k)$ is the estimated value of the mean value function $m_r(n)$ at n_k, which is determined using the actually observed data up to an arbitrary test case $n_e(0<n_e\leq n_k)$, in which (n_e/n_k) denotes the testing progress ratio. In other words, the number of failures by n_k can be predicted by the SRGM and then compared with the actually observed number x_k. The difference between the predicted value $\hat{m}(n_k)$ and the reported value x_k measures the prediction fault. The ratio $\{(\hat{m}(n_k)-x_k)/x_k\}$ is called Relative Prediction Error (RPE). If the RPE value is negative / positive the SRGM is said to underestimate / overestimate the future failure phenomenon. A value close to zero for RPE indicates more accurate prediction, thus more confidence in the model and better predictive validity. The value of RPE is said to be acceptable if it is within $\pm(10\%)$ (Kapur et al. [8]).

5. Data Analysis And Model Comparison

5.1 Goodness of Fit Analysis

Using MLE method, the estimation values of the model parameters for both DS are given in table 1. The fitting of the model to both DS is graphically illustrated in figures 1-4. It is clearly seen from the figures that the model fits both DS excellently. Comparison of the proposed model and other well-documented discrete SRGM due to Kapur et al. [8] based on NHPP in terms of goodness of fit for both DS has been worked out. The results are presented in table 2. It is clearly seen from the tables that the proposed model is better under comparison in terms of MSE and R^2.

5.2 Predictive Validity Analysis

All the DS are truncated into different proportions and used to estimate the parameters of the proposed model. For each truncation, one value of RPE is obtained and given in tables 3 & 4. The tables give the results of the predictive validity. It is observed that the predictive validity of the model varies from one truncation to another. It is clearly seen from table 4 that 60% of the total test runs is sufficient to predict the future reasonably and from the table 3 that 70% of the total test runs is sufficient to predict the future reasonably.

5.3 Estimation Results

Table 1:

Models under Comparisons		Parameter Estimation				
		a	β	b	b_1	b_2
DS-I (Pham-Tandem) 100 Faults	Discrete K-G Model	109.73	1.4107	.1667	-----	-----
	Proposed Model with Change-Point	104.49	5.0583	-----	.2467	.2341
DS – II (Brooks-DS2) 1301 Faults	Discrete K-G Model	1331.05	20.1629	.1817	-----	-----
	Proposed Model with Change-Point	1321.89	25.3379	-----	.1946	.1909

Table 2:

Models under Comparisons		Comparison Criteria				
		R^2	MSE	Bias	Variation	RMSPE
DS-I (Pham - Tandem) 100 Faults	Discrete K-G Model	.9923	6.505	0.546	9.212	9.2285
	Proposed Model with Change-Point	.9964	3.053	0.250	2.403	2.4165
DS – II (Brooks-DS2) 1301 Faults	Discrete K-G Model	.9990	203.7	2.146	14.319	14.479
	Proposed Model with Change-Point	.9993	149.51	5.9E-7	12.4063	12.406

Table 3: (Predictive Validity on DS-I)

Model	(n_e/n_k)	$m(n_k)$	RPE
Proposed Model	100%	101.7977	1.7977
	95%	101.9674	1.9674
	90%	102.6314	2.6314
	80%	104.1919	4.1919
	70%	106.4687	6.4687
	60%	117.7655	17.7655

Table 4: (Predictive Validity on DS-II)

Model	(n_e/n_k)	$m(n_k)$	RPE
Proposed Model	100%	1301.6070	0.0466
	95%	1303.5930	0.1933
	90%	1305.5010	0.3459
	80%	1312.7177	0.9006
	70%	1350.3480	3.7930
	60%	1430.7331	9.9717

6.4. Goodness of Fit Curves for Datasets

The goodness of fit for the proposed model corresponding to datasets DS-I and DS-II is graphically presented in Figures 1-4 . The graphs have been plotted between actual and estimated values for the cumulative number of faults for the two data sets under consideration. The curves show excellent fit for the proposed model with the estimated values very near to observed failure data.

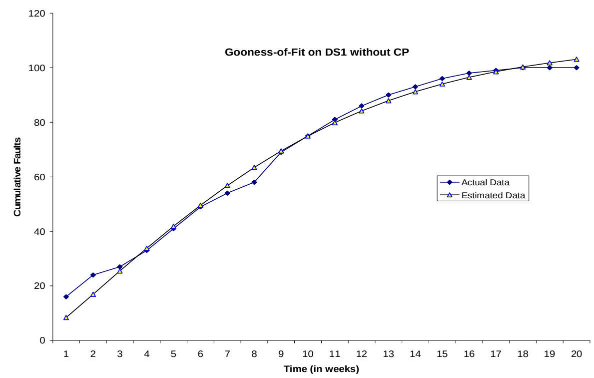

Fig. 1 Goodness-of-Fit on DS-I without CP

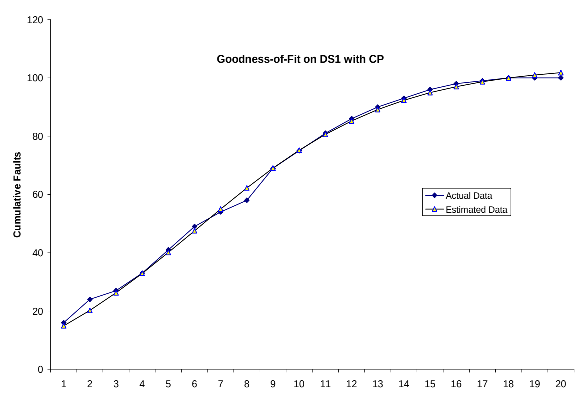

Fig. 2 Goodness-of-Fit on DS-I with CP

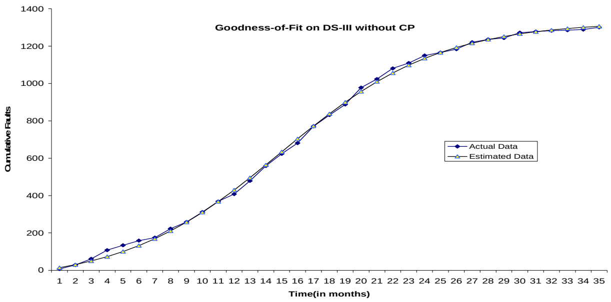

Fig. 3 Goodness-of-Fit on DS-II without CP

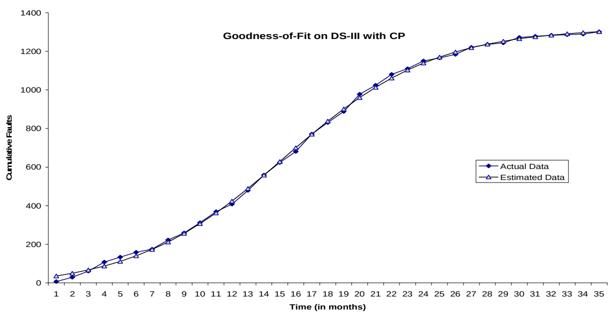

Fig. 4 Goodness-of-Fit on DS-II with CP

7. Conclusion

While testing the software under consideration, fault detection rate is normally assumed to be constant. Whereas, in practice, detection rate varies because of change in testing skill, system environment and testing strategy. Several questions arise - Had a change occurred? Had more than one change occurred? When did the change occur? These questions can be answered by performing a change-point analysis. A change-point analysis is capable of detecting changes. Change-point characterizes the changes and controls the overall error rate.

In this paper, a flexible discrete SRGM incorporating change-point concept has been presented. The proposed model considers that during software testing, FRR does not remain constant. The introduction of change factor in FRR helps in better predictability and more accuracy. The model has been validated, evaluated, and compared with discrete version of Kapur et al. model by applying it on two DS. The results show that the proposed model provides improved goodness of fit and predictive validity for software failure occurrence / fault removal data due to its applicability and flexibility.

References

[1] Bittanti et al., *A Flexible Modeling Approach in Software Reliability Growth,* In: Goos G & Hartmanis (Editors), *Software Reliability Modeling & identification*, Springer-Verlag: Berlin; 1988, pp. 101-140.

[2] Brooks WD and Motley RW, *Analysis of discrete software reliability models - Technical Report (RADC-TR-80-84),* 1980, New York: Rome Air Development Center.

[3] Goel AL and Okumoto K., *Time dependent error detection rate model for software reliability and other performance measures, 1979, IEEE Transactions on Reliability*; R-28(3), 1979, pp. 206-211.

[4] Goel AL., *Software reliability models: assumptions, limitations and applicability,* 1985, *IEEE Transactions on Software Engineering*; SE-11, pp. 1411-1423.

[5] Inoue S. and Yamada S. , *Discrete software reliability assessment with discretized NHPP models;* Computer & Mathematics with Applications, 2006, Vol. 51, No. 2, pp. 161-170.

[6] Jelinski Z and Moranda PB., *Software reliability research* In: Freiberger W, *(Ed.) Statistical Computer Performance Evaluation*; 1972, New York: Academic Press: pp. 465-497.

[7] Kapur PK and Garg RB, *A software reliability growth model for an error removal phenomenon, Software Engineering Journal* ; 1992,7: pp. 291-294.

[8] Kapur PK, Garg RB and Kumar S, *Contributions to hardware and software reliability;* Singapore: 1999, World Scientific Publishing Co. Ltd.

[9] Kapur PK and Garg RB, *Optimal software release policies for software reliability growth model under imperfect debugging*, RAIRO; 1990,24: pp. 295-305.

[10] Kapur PK, Shatnawi O and Singh O, *Discrete imperfect software reliability growth models under imperfect debugging environment.* In: Rajaram NJ and Verma AK *(Editors) Proceedings of the International Conference on Multimedia and design*; Arena Multimedia & IIT: Mumbai, 2002, Vol (II): pp. 114-129.

[11] Khoshogoftaar TM and Woodcock TG, S*oftware reliability model selection: A case study, Proceedings of the International Symposium on Software Reliability Engineering*, 1991, pp.183-191.

[12] Musa JD, Iannino A and Okumoto K, *Software reliability: Measurement, Prediction, Applications,* 1987, New York: McGraw-Hill.

[13] Ohba M., *Software reliability analysis models,* IBM Journal of Research and Development; 1984, 28: pp. 428-43.

[14] Pham H, *Software Reliability,* 2000, Springer-Verlag Singapore Pte. Ltd.

[15] Shyur, HJ (2003), A Stochastic Software Reliability Model with Imperfect-Debugging and Change-Point, Journal of Systems and Software; 66, 135-141.

[16] Wang Z, Wang J (2005**)** "Parameter Estimation of some NHPP Software Reliability Models with

Change-Point" Communications in Statistics-Simulation and Computation, 34, pp.121-134.

[17] Wood A (1996), Predicting Software Reliability, IEEE Computers; Vol. 11, pp. 69-77

[18] Yamada S and Osaki S, *Discrete Software reliability growth models, Applied Stochastic models and data analysis*; 1985, 1: pp. 65-77.

[19] Yamada S, Tokuno K and Osaki S., *Imperfect Debugging Models with Fault Introduction Rate for Software Reliability Assessment, Int'l J. System Science*, 1992, Vol. 23, num 12, pp. 2241-2252.

[20] Zhao M (1993), Change-point problems in software and hardware reliability, Communications in Statistics-Theory and Methods, 1993; 22(3), 757-768.

[21] Zou FZ (2003), A Change-Point Perspective on the Software failure process. Software Testing, Verification and Reliability 13, pp 85-93.