Distribution Based Change-Point Problem with Two Types of Imperfect Debugging in Software Reliability

P. K. Kapur[1], Sameer Anand[2] and V. B. Singh[3]

Abstract - Software testing is an important phase of software development life cycle. It controls the quality of software product. Due to the complexity of software system and incomplete understanding of software, the testing team may not be able to remove/correct the fault perfectly on observation/detection of a failure and the original fault may remain resulting in a phenomenon known as imperfect debugging, or get replaced by another fault causing fault generation. In case of imperfect debugging, the fault content of the software remains same while in case of fault generation, the fault content increases as the testing progresses and removal/correction results in introduction of new faults while removing/correcting old ones. During software testing fault detection /correction rate may not be same throughout the whole testing process, but it may change at any time moment. In the literature various software reliability models have been proposed incorporating change-point concept. In this paper we propose a distribution based change-point problem with two types of imperfect debugging in software reliability. The models developed have been validated and verified using real data sets. Estimated Parameters and comparison criteria results have also been presented

Index Terms - Non-homogenous Poisson process, software reliability growth model, hazard rate, imperfect debugging.

NOTATION

$m(t)$	:	the mean value function or the expected number of faults detected or removed by time t.
$a(t)$	:	total fault content of software dependent on time.
p	:	the probability of fault removal on a failure (i.e., the probability of perfect debugging).
α	:	the rate at which the faults/errors may be introduced during the debugging process.
b	:	fault removal/correction rate.
$\lambda(t)$	:	intensity function for NHPP models or fault detection rate per unit time.
$F(t)$	:	distribution functions for fault removal/correction times.
$f(t)$	:	density functions for fault removal/correction times.
$z(t)$	:	hazard rate function.
β	:	learning parameter in logistic function.

[1]*Department of Operational Research, University of Delhi, India*
[2]*S. S. College of Business Studies, University of Delhi, India*
[3]*Delhi College of Arts & Commerce, University of Delhi, India*
E-Mail: [1]pkkapur1@gmail.com, [2]sanand_or@yahoo.com and [3]singh_vb@rediffmail.com

1. INTRODUCTION

Computer software is embedded in systems of all kinds: transportation, medical, telecommunications, military, industrial processes, entertainment, office products….the list is almost endless. Software is virtually inescapable in a modern world. And as we move into the twenty-first century, it will become the driver for new advances in everything from elementary education to genetic engineering. Software development consists of different phases: requirement analysis, design, coding, testing, implementation and maintenance called SDLC. Research has been conducted in software reliability engineering over the past three decades and many software reliability growth models (SRGM) have been proposed. The Software Reliability Growth Model (SRGM) is the tool, which can be used to evaluate the software quantitatively, develop test status, schedule status and monitor the changes in reliability performance.

Research has been conducted in software reliability engineering over the past three decades and many software reliability growth models (SRGM) have been proposed. The pioneering attempt in non-homogenous Poisson process based on SRGM was made by Goel and Okumoto (G-O) [1]. The model describes the failure observation phenomenon by an exponential curve. There are also SRGM that describe either S-shaped curves or a mixture of exponential and S-shaped curves (flexible). Some of the important contributions of these type of models are due to Yamada *et al.* [27], Ohba [18], Bittanti *et al.* [3], Kapur and Garg [16], Kapur *et al.* [17], Pham [24] etc.

In most of the models discussed above it is assumed that whenever an attempt is made to remove a fault, it is removed with certainty i.e. a case of perfect debugging. But the debugging activity is not always perfect because of number of factors like tester's skill/expertise etc. In practical software development scenario, the number of failures observed/detected may not be necessarily same as the number of errors removed/corrected. Kapur and Garg [16] have discussed in their error removal phenomenon model that as testing grows and testing team gains experience, additional numbers of faults are removed without them causing any failure.

The testing team, however, may not be able to remove/correct fault perfectly on observation/detection of a failure and the original fault may remain leading to a phenomenon known as imperfect debugging, or replaced by another fault resulting in fault generation. In case of imperfect debugging the fault content of the software is not changed, but because of incomplete understanding of the software, the original detected fault is not removed perfectly. But in case of fault generation, the total fault content increases as the testing progresses

because new faults are introduced in the system while removing the old original faults.

Model due to Obha and Chou [18] is an fault generation model applied on G-O model and has been also named as Imperfect debugging model. Kapur and Garg [22] introduced the imperfect debugging in G-O model. They assumed that the FDR per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed/detected by time infinity is more than the initial fault content. Although these two models describe the imperfect debugging phenomenon yet the software reliability growth curve of these models is always exponential. Moreover, they assume that the probability of imperfect debugging is independent of the testing time. Thus, they ignore the role of the learning process during the testing phase by not accounting for the experience gained with the progress of software testing. Pham [24] developed an SRGM for multiple failure types incorporating fault generation. Zhang et al. [26] proposed a testing efficiency model which includes both imperfect debugging and fault generation, modeling it on the number of failures experienced/observed/detected, however both imperfect debugging and fault generation are actually seen during fault removal/correction. Recently, Kapur et al. [12] proposed a flexible SRGM with imperfect debugging and fault generation using a logistic function for fault detection rate which reflects the efficiency of the testing/removal team.

We execute the program in specific environment and improve its quality by detecting and correcting faults. Many SRGM assume that, during the fault detection process, each failure caused by a fault occurs independently and randomly in time according to the same distribution Musa et al. [21]. But the failure distribution can be affected by many factors such as running environment, testing strategy, defect density and resource allocation. On the other hand, in practice, if we want to detect more faults for a short period of time, we may introduce new techniques or tools that are not yet used, or bring in consultants to make a radical software risk analysis. In addition, there are newly proposed automated testing tools for increasing test coverage and can be used to replace traditional manual software testing regularly. The benefits to software developers/testers include increased software quality, reduced testing costs, improved release time to market, repeatable test steps, and improved testing productivity. These technologies can make software testing and correction easier, detect more bugs, save more time, and reduce much expense. Altogether, we wish that the consultants, new automated test tools or techniques could greatly help us in detecting additional faults that are difficult to find during regular testing and usage, in identifying and correcting faults most cost effectively and in assisting clients to improve their software development process. Thus, the fault detection rate may not be smooth and can be changed at some time moment τ called change-point. Many researchers have incorporated change point in software reliability growth modeling. Firstly Zhao [28] incorporated change-point in software and hardware reliability. Huang et al. [7] used change-point in software reliability growth modeling with testing effort functions. The imperfect debugging with change-point has been introduced in software reliability growth modeling by Shyur [25]. Kapur et al. [9,14] introduced various testing effort functions and testing effort control with change-point in software reliability growth modeling. Goswami et al.[6] and Kapur et al.[15] proposed a software reliability growth model for errors of different severity using change-point. The multiple change-points in software reliability growth modeling for fielded software has been proposed by Kapur et al. [11]. Later on SRGM based on stochastic differential equations incorporating change-point concept has been proposed by Kapur et al. [10].

2. BASIC ASSUMPTION

The NHPP models are based on the assumption that the software system is subject to failures at random times caused by manifestation of remaining faults in the system. Hence NHPP are used to describe the failure phenomenon during the testing phase. The counting process $\{N(t), t \geq 0\}$ of an NHPP process is given as follows.

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

and

$$m(t) = \int_0^t \lambda(x)\, dx \tag{2}$$

The intensity function $\lambda(x)$ (or the mean value function $m(t)$) is the basic building block of all the NHPP models existing in the software reliability engineering literature.

The proposed models are based upon the following basic assumptions:

1. Failure fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. Failure rate is equally affected by all the faults remaining in the software.
4. When a software failure occurs, an instantaneous repair effort starts and the following may occur:
   (a) Fault content is reduced by one with probability p
   (b) Fault content remains unchanged with probability $1-p$.
5. During the fault removal process, whether the fault is removed successfully or not, new faults are generated with a constant probability α.
6. Fault detection / removal rate may change at any time moment.

Assumption 4 and 5 captures the effect of imperfect debugging and fault generation respectively.

3. MODEL DEVELOPMENT

In this section, we formulate distribution based software reliability growth models incorporating change-point and two types of imperfect debugging. Since the faults in the software

systems are detected and eliminated during the testing phase, the number of faults remaining in the software system gradually decreases as the testing procedure goes on. Thus under the common assumptions for software reliability growth modeling, we consider the following linear differential equation.

$$\frac{dm(t)}{dt} = b(t)\big(a - m(t)\big) \qquad (3)$$

Where b(t) is a fault detection rate per remaining faults at testing time t. Here we consider the fault detection rate as hazard rate z(t), initial fault is not the constant but the function of time and incorporating the imperfect debugging. So the above equation can be written as

$$\frac{dm(t)}{dt} = z(t)p\big(a(t) - m(t)\big)$$

We assume that faults can be introduced during the debugging phase with a constant fault introduction rate α. Therefore, the fault content rate function, *a(t)*, is a linear function of the expected number of faults detected by time *t.* That is $a(t) = a + \alpha m(t)$, above equation becomes

$$\frac{dm(t)}{dt} = z(t)p\big(a + \alpha m(t) - m(t)\big) \qquad (4)$$

In the proposed model we assume that the hazard rate $z(t) = \dfrac{f(t)}{1 - F(t)}$, the fault introduction rate α and probability of perfect debugging p, may be changed at some time moment τ called change point.

After incorporating change-point, we get the following form of fault detection /removal, probability of perfect debugging and fault generation rate.

$$z(t) = \begin{cases} \dfrac{f_1(t)}{1 - F_1(t)} & for\ t \le \tau \\ \dfrac{f_2(t)}{1 - F_2(t)} & for\ t > \tau \end{cases} \qquad (5)$$

Where F_1, f_1 and F_2, f_2 are the distributions, density functions before and after change point respectively.

The equation (5) can be rewritten as

Probability of perfect debugging rate will be

$$p = \begin{cases} p_1 & for\ t \le \tau \\ p_2 & for\ t > \tau \end{cases} \qquad (6)$$

and fault content rate

$$a(t) = \begin{cases} a + \alpha_1 m(t) & for\ t \le \tau \\ a + \alpha_1 m(\tau) + \alpha_2\big(m(t) - m(\tau)\big) & for\ t > \tau \end{cases} \qquad (7)$$

The Equation (5) can be rewritten as

$$z(t) = \frac{f_1(t)}{1 - F_1(t)}U(\tau - t) + \frac{f_2(t)}{1 - F_2(t)}U(t - \tau)$$

Using unit step function give by

$$U(x) = \begin{cases} 0 & if\ \ x < 0 \\ 1 & if\ \ x > 1 \end{cases}$$

Similarly equation (6) & (7) can also be rewritten as

$$p = p_1 U(\tau - t) + p_2 U(t - \tau)$$

$$\& a(t) = a + \alpha_1 m(t)U(\tau - t) + \big(a + \alpha_1 m(\tau) + \alpha_2\big(m(t) - m(\tau)\big)\big)U(t - \tau)$$

Now using equation (5), (6) and (7), the equation (4) can be rewritten as,

$$\frac{dm(t)}{dt} = \begin{cases} \dfrac{f_1(t)}{1 - F_1(t)} p_1\big(a + \alpha_1 m(t) - m(t)\big) & for\ t \le \tau \\ \dfrac{f_2(t)}{1 - F_2(t)} p_2\big(a + \alpha_1 m(\tau) + \alpha_2\big(m(t) - m(\tau)\big) - m(t)\big) & for\ t > \tau \end{cases} \qquad (8)$$

After solving the above equations, we get the following solutions

$$m(t) = \begin{cases} \dfrac{a}{(1 - \alpha_1)}\Big[1 - \big(1 - F_1(t)\big)^{p_1(1-\alpha_1)}\Big] & for\ t \le \tau \\ \dfrac{a}{(1 - \alpha_2)}\left[1 - \big(1 - F_1(\tau)\big)^{p_1(1-\alpha_1)}\left(\dfrac{\big(1 - F_2(t)\big)}{1 - F_2(\tau)}\right)^{p_2(1-\alpha_2)}\right] + \left(\dfrac{(\alpha_1 - \alpha_2)}{(1 - \alpha_2)}\right)m(\tau) & for\ t > \tau \end{cases} \qquad (9)$$

SRGM-1

The following exponential distribution function is used to model SRGM-1:

Let T~ exp(b_1) $\quad for\ t \le \tau$

$$i.e\quad F_1(t) = 1 - e\exp\big(-b_1 t\big) \qquad for \le \tau \qquad (10)$$

and

Let T~ exp(b_2) $\quad for\ t > \tau$

$$F_2(t) = 1 - \exp\big(-b_2 t\big)\ for\ t > \tau \qquad (11)$$

Substituting the value of $F_1(t)$ and $F_2(t)$ from Equation (10) and (11) into Equation (9), we get:

$$m(t) = \begin{cases} \dfrac{a}{(1 - \alpha_1)}\Big[1 - \exp\big(-b_1 p_1 (1 - \alpha_1)t\big)\Big] & for\ t \le \tau \\ \dfrac{a}{(1 - \alpha_2)}\Big[1 - \exp\big(-b_1 p_1 (1 - \alpha_1)\tau - b_2 p_2 (1 - \alpha_2)(t - \tau)\big)\Big] & \\ \quad + \dfrac{(\alpha_1 - \alpha_2)}{(1 - \alpha_2)}m(\tau) & for\ t > \tau \end{cases} \qquad (12)$$

The above model can be reduced to the model given by Shyur [25] if we consider the perfect debugging and no fault generation.

SRGM-2

Let *F(t)* be a two-stage Erlangian distribution function i.e. ,

T~ Erlang-2(b_1) $for\ t \le \tau$

$$i.e.\ F_1(t) = 1 - (1 + b_1 t)\exp(-b_1 t)\ for\ t \le \tau \quad (13)$$

And

T~ Erlang-2(b_2) $for\ t > \tau$

$$i.e.\ F_2(t) = 1 - (1 + b_2 t)\exp(-b_2 t)\ for\ t > \tau \quad (14)$$

Substituting the value of $F_1(t)$ and $F_2(t)$ from Equation (13) and (14) into Equation (9), we get:

$$m(t) = \begin{cases} \dfrac{a}{(1-\alpha_1)}\left[1 - \left((1+b_1 t)\exp(-b_1 t)\right)^{p_1(1-\alpha_1)}\right] & for\ t \le \tau \\ \dfrac{a}{(1-\alpha_2)}\left[1 - (1+b_1\tau)^{p_1(1-\alpha_1)}\left(\dfrac{(1+b_2 t)}{(1+b_2\tau)}\right)^{p_2(1-\alpha_2)}\exp\left(-b_1 p_1(1-\alpha_1)\tau - b_2 p_2(1-\alpha_2)(t-\tau)\right)\right] & \\ + \dfrac{(\alpha_1-\alpha_2)}{(1-\alpha_2)} m(\tau) & for\ t > \tau \end{cases} \quad (15)$$

The above model can be reduced to the model given by Archana [2] if we consider the perfect debugging and no fault generation.

SRGM-3

Let *F(t)* be a logistic distribution function i.e. ,

T~ logistic distribution (b_1, β_1) $for\ t \le \tau$

$$i.e.\ F_1(t) = \frac{\left(1 - \exp(-b_1 t)\right)}{\left(1 + \beta_1 \exp(-b_1 t)\right)}\ for\ t \le \tau \quad (16)$$

And

T~ logistic distribution (b_2, β_2) $for\ t \le \tau$

$$i.e.\ F_2(t) = \frac{\left(1 - \exp(-b_2 t)\right)}{\left(1 + \beta_2 \exp(-b_2 t)\right)}\ for\ t > \tau \quad (17)$$

Substituting the value of $F_1(t)$ and $F_2(t)$ from Equation (16) and (17) into Equation (9), we get:

$$m(t) = \begin{cases} \dfrac{a}{(1-\alpha_1)}\left[1 - \left(\dfrac{(1+\beta_1)}{(1+\beta_1\exp(-b_1 t))}\right)^{p_1(1-\alpha_1)}\exp\left(-b_1 p_1(1-\alpha_1)t\right)\right] & for\ \ 0 \le t \le \tau \\ \dfrac{a}{(1-\alpha_2)}\left[1 - \left(\dfrac{(1+\beta_1)}{(1+\beta_1\exp(-b_1\tau))}\right)^{p_1(1-\alpha_1)}\left(\dfrac{(1+\beta_2\exp(-b_2\tau))}{(1+\beta_2\exp(-b_2 t))}\right)^{p_2(1-\alpha_2)}\right. & \\ \left. * e\exp\left(-b_1 p_1(1-\alpha_1)\tau - b_2 p_2(1-\alpha_2)(t-\tau)\right)\right] & \\ + \dfrac{(\alpha_1-\alpha_2)}{(1-\alpha_2)} m(\tau) & for\ \ t > \tau \end{cases} \quad (18)$$

For further simplifying the estimation procedure we may assume $\alpha_1 = \alpha_2 = \alpha$ *and* $p_1 = p_2 = p$.

4. MODEL VALIDATION, COMPARISON CRITERIA AND DATA ANALYSES

Model Validation

To illustrate the estimation procedure and application of the SRGM (existing as well as proposed) we have carried out the data analysis of real software data set. The parameters of the models have been estimated using statistical package SPSS and the change-point of the data sets have been judged by using change-point analyzer.

Data set 1(DS-1)

The first data set (DS-1) had been collected during 35 months of testing a radar system of size 124 KLOC and 1301 faults were detected during testing. This data is cited from Brooks and Motley [4]. The change-point for this data set is 17th month.

Data set 2(DS-2)

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

5. COMPARISON CRITERIA FOR SRGM

The performance of SRGM are judged by their ability to fit the past software fault data (goodness of fit) and predicting the future behavior of the fault.

Goodness of Fit criteria

The term goodness of fit is used in two different contexts. In one context, it denotes the question if a sample of data came from a population with a specific distribution. In another context, it denotes the question of "How good does a mathematical model (for example a linear regression model) fit to the data"?

The Mean Square -Error (MSE):

The model under comparison is used to simulate the fault data, the difference between the expected values, $\hat{m}(t_i)$ and the observed data y_i is measured by MSE as follows.

$$MSE = \sum_{i=1}^{k} \frac{(\hat{m}(t_i) - y_i)^2}{k} \quad (19)$$

where k is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [17].

Coefficient of Multiple Determination (R2):

We define this coefficient as the ratio of the sum of squares resulting from the trend model to that from constant model subtracted from 1.

$$\text{i.e. } R^2 = 1 - \frac{\text{residual SS}}{\text{corrected SS}}.$$

R^2 measures the percentage of the total variation about the mean accounted for the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well. The larger R^2, the better the model explains the variation in the data [17].

Bias

The difference between the observation and prediction of number of failures at any instant of time i is known as PE_i(prediction error). The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit [8].

Variation

The standard deviation of prediction error is known as variation.

$$Variation = \sqrt{\left(1/N-1\right)\sum\left(PE_i - Bias\right)^2}$$

Lower the value of Variation better is the goodness of fit [8].

Root Mean Square Prediction Error

It is a measure of closeness with which a model predicts the observation.

$$RMSPE = \sqrt{\left(Bias^2 + Variation^2\right)}$$

Lower the value of Root Mean Square Prediction Error better is the goodness of fit [8].

Data Analyses

For DS-1

The parameter estimation and comparison criteria results for DS-1 of all the models under consideration can be viewed through Table I(a) and I(b). It is clear from the table that the value of R^2 for SRGM-3 is higher and value of MSE is lower in comparison with other models and provides better goodness of fit for DS-1.

For DS-2

The parameter estimation and comparison criteria results for DS-2 of all the models under consideration can be viewed through Table II(a) and II(b) . It is clear from the table that the value of R^2 for SRGM-3 is higher and value of MSE is lower in comparison with other models and provides better goodness of fit for DS-2.

Models	a	b_1	b_2	β_1	β_2	p	α
SRGM-1	1686	.060	.072	-	-	.377	.515
SRGM-2	1650	.098	.101	-	-	.910	.001
SRGM-3	1335	.123	.243	2.03	26.9	.724	.001

Table I(a): Model Parameter Estimation Results (DS-1)

Models	R^2	MSE
SRGM-1	.974	5608.09
SRGM-2	.988	2544.43
SRGM-3	.999	207.4532

Table I(b): Model Comparison Results (DS-1)

Models	a	b_1	b_2	β_1	β_2	p	α
SRGM-1	388	.202	.222	-	-	319	.368
SRGM-2	467	.602	.562	-	-	.156	.001
SRGM-3	362	.352	.281	5	3	.567	.063

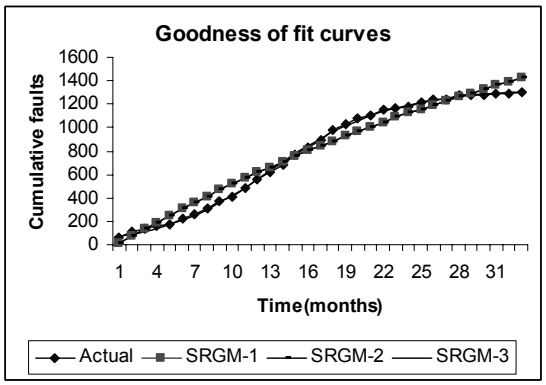
Table II(a): Model Parameter Estimation Results (DS-2)

Models	R^2	MSE
SRGM-1	.988	122.666
SRGM-2	.990	104.906
SRGM-3	.992	83.111

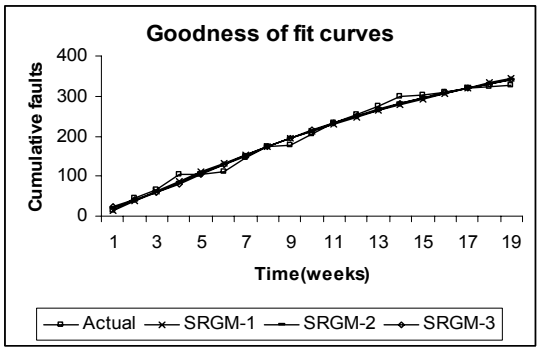
Table II(b): Model Comparison Results (DS-2)

6. GOODNESS OF FIT CURVES

For DS-1



For DS-2



7. CONCLUSION

In this paper we have developed a distribution based change-point problem with two types of imperfect debugging in software reliability. With this approach, we can derive existing models and propose new model. All these models have been validated and verified using real data sets. Parameter estimates, comparison results and goodness of fit curves have also been presented.

8. FUTURE SCOPE

In future, we will try to develop more models in the same line by using Erlang normal, weibull and gamma distribution functions. Models can be extended for multiple change-points problem..

REFERENCES

[1] A.L. Goel, Software Reliability Models: Assumptions, Limitations and Applicability, IEEE Transactions on Software Engineering; SE-11, pp. 1411-1423, 1985.

[2] Archana Kumar, Ph.D. Thesis "A Study in Software Reliability Growth Modelling Under Distributed Development Environment", Department of Operational Research, University of Delhi, Delhi, 2007.

[3] Bittanti S., Bolzern, P., Pedrotti, E., Pozzi, N. and Scattolini R., "A flexible modeling approach for software reliability growth", Goos, G., and Harmanis, J., (Ed.), Software Reliability Modelling and Identification, Springer Verlag, Berlin, pp. 101-140, 1988.

[4] Brooks W. D. and Motley R.W., "Analysis of Discrete Software Reliability Models-Technical Report (RADC-TR-80-84)", Rome Air Development Center, New York, 1980.

[5] Goel, A.L. and Okumoto, K, "Time dependent error detection rate model for software reliability and other performance measure", IEEE Transactions on Reliability, 1979.

[6] Goswami D.N., Khatri Sunil K, and Kapur Reecha "Discrete Software Reliability Growth Modeling for Errors of Different Severity incorporating Change-Point Concept" International Journal of Automation and Computing Vol. 4(4) pp.396-405,2007.

[7] Huang Yu Chin "Performance Analysis of Software Reliability Growth Models with Testing Effort and Change-Point ", Journal of Systems and Software Vol. 76, Issue 2, pp 181-194, 2005.

[8] K. Pillai and V.S.S. Nair, "A Model for Software Development effort and Cost Estimation", IEEE Transactions on Software Engineering; vol. 23(8), pp. 485-497, 1997.

[9] Kapur P. K, Singh V. B, Anand Sameer and Yadavalli V.S.S.,"Software Reliability Growth Model with Change-Point and Effort Control Using a Power Function of Testing Time" International Journal of Production Research, Vol. 46 Issue no. 3 pp.771-787,2008.

[10] Kapur P. K, Singh V. B, Anand Sameer "Effect of Change-Point on Software Reliability Growth Models Using Stochastic Differential Equations" published in the proceedings of 3rd International Conference on Reliability and Safety Engineering, (Eds. R.B.Misra, V.N.A. Naikan, S.K.Chaturvedi, and N.K.Goyal), (INCRESE-2007), Udaipur, pp. 320-333, 2007.

[11] Kapur P. K, Singh V. B, Anand Sameer, "Software Reliability Growth Model of Fielded Software Based on Multiple Change-Point Concept Using a Power Function of Testing Time", Quality Reliability and Infocom Technology, (Eds. P.K.Kapur and A.K.Verma), MacMillan India Ltd., pp.171-178, 2007.

[12] Kapur P. K., Kumar D., Gupta A. and Jha P. C., "On How To Model software Reliability Growth in the Presence Of Imperfect Debugging and error Generation", Proceedings of 2nd International Conference on Reliability and safety Engineering, pp. 515-523, 2006.

[13] Kapur P. K., Pham H., Anand S. and Yadav K. "A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation", communicated in IEEE Transactions on Reliability, 2008.

[14] Kapur P.K, Gupta A., Shatnawi O, and Yadavalli V.S.S "Testing Effort Control Using Flexible Software Reliability Growth Model with Change Point" International Journal of Performability Engineering- Special issue on Dependability of Software/ Computing Systems, Vol. 2, No. 3, pp 245-263, 2006

[15] Kapur P.K. Kumar Archana ,Yadav Kalpana and Khatri Sunil "Software Reliability Growth Modelling for Errors of Different Severity using Change Point" International Journal of Quality ,Reliability and Safety Engineering ,Vol.14,No.4, pp 311-326,2007

Continued on page no. 124