International Journal of Reliability, Quality and Safety Engineering
Vol. 14, No. 4 (2007) 311–326
© World Scientific Publishing Company

SOFTWARE RELIABILITY GROWTH MODELLING FOR ERRORS OF DIFFERENT SEVERITY USING CHANGE POINT

P. K. KAPUR*,§, ARCHANA KUMAR*, KALPANA YADAV† and SUNIL K. KHATRI‡

*Department of Operational Research
University of Delhi, Delhi-110007
§pkkapur@or.du.ac.in

†Indira Gandhi Institute of Technology
G.G.S.I.P. University, Delhi-110006

‡Mother Teresa Institute of Management
G.G.S.I.P. University, Delhi-110006

Received 15 January 2007
Revised 16 May 2007

During the last two decades many researchers have analyzed the reliability growth of software during the testing and operational phases and proposed the mathematical models to estimate and predict the reliability measures. During the software testing on the detection of a failure the fault that has caused the failure is isolated and removed. Most of the existing research in this area considers that similar testing efforts and strategy are required on each debugging effort. However this may not be true in practice. Different faults may require different amount of testing efforts and testing strategy for their removal. In software reliability modeling in order to incorporate this phenomenon faults are classified into different categories as simple, hard and/or complex faults. This categorization is also extended to n-types of faults. Some of the existing research incorporates this phenomenon considering that the fault removal rate is different for different types of faults and remains constant during the overall period of testing. However this assumption may not apply in general testing environment in practice. It is a common observation that as the testing progresses the fault detection and/or removal rate changes. This change can be due to a number of reasons. The changing testing environment, testing strategy, skill, motivation and constitution of the testing and debugging personnel etc. are some of the major reasons behind this change. In this paper we have formulated the model for the software system developed for safety critical application under a specific testing environment. The model is validated on real life data sets.

Keywords: Software reliability; Software Reliability Growth Model (SRGM); severity, change point; Non Homogeneous Poisson Process (NHPP).

Acronyms

AIC	=	Akaike Information Criterion
DS	=	Data Set
FDR	=	Fault Detection Rate
FRR	=	Fault Removal Rate
LOC	=	Lines of Source Code
MLE	=	Maximum Likelihood Estimate
MSE	=	Mean Square Error
RMSPE	=	Root Mean Square Prediction Error
SRE	=	Software Reliability Engineering

1. Introduction

The development of high quality software satisfying cost, schedule, and resource requirements is an essential prerequisite for improved competitiveness of any organization. One major difficulty to master this challenge is the inevitability of defects in software products during its use in operational phase. It is the testing phase of the software that aims to remove most of the faults lying in software. The testing of software systems is subject to strong conflicting forces. Therefore the software developer attempts to have a tight control over the testing of the software. One of the most effective ways to do this is to apply SRE tools and techniques to testing and development.

SRE delivers the desired functionality for a product much more efficiently by quantitatively characterizing its expected use. The software reliability engineering tools prove to be major assistance in increasing the reliability while decreasing the development time and cost. Thus software reliability engineering balances customer needs for the major quality characteristics of reliability, availability, delivery time, and life cycle cost more effectively.

The Software Reliability Growth Model (SRGM) is a tool of SRE, which can be used to evaluate the software quantitatively, develop test status, schedule status and monitor the changes in reliability performance.[5] In the last two decades several Software Reliability models have been developed in the literature to estimate the fault content, failure rate and fault removal rate per fault in software and to predict the reliability of the software at the release time. Most of these are characterized by the mean value function of a NHPP and utilize historical failure data collected during the testing phase to evaluate the quality of software. Goel and Okumoto[6] have proposed first time dependent 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 describes exponential failure curves. In general most of the existing SRGM describes either an exponential or s-shaped failure curves. Ohba[17] 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.*[2] and Kapur and Garg[13] have similar forms as that of Ohba[17] but are developed under different set of assumptions. Bittanti *et al.*[2] proposed an SRGM exploiting the fault removal (exposure) rate during the initial and final time epochs of testing. Whereas, Kapur and Garg[13] describe a fault removal phenomenon, where they assume that during a removal process of a fault some of the additional faults might be removed without these faults causing any failure. These models can describe both exponential and S-shaped growth curves and therefore are termed as flexible models.[2,13,17]

Most of these models were proposed under the assumption that similar testing efforts and testing strategy is required for removing each of the faults. However this assumption may not be true in practice, different faults may require different amount of testing efforts and testing strategy for their removal from the system. In literature to incorporate this phenomenon faults are categorized as of different types and are analyzed separately. Yamada[21] proposed a modified exponential SRGM assuming that there are two types of faults in the software and exponential failure curve. Pham *et al.*[18] proposed a SRGM with multiple failure types. Later Kapur *et al.*[8] introduced a flexible model called the generalized Erlang SRGM by classifying the faults in the software system as simple, hard and complex faults. It is assumed that the time delay between the failure observation and its subsequent removal represent the severity of faults. The model is extended to n-types of faults. Another model due to Kapur *et al.*[14] describes the implicit categorization of faults based on the time of detection of fault. However an SRGM should explicitly define the errors of different severity as it is expected that any type of fault can be detected at any point of testing time. Therefore it is desired to study the testing and debugging process of each type of faults separately.[9,12] The mean value function of the SRGM is described by the joint effect of the type of faults present in the system. Such an approach can capture the variability in the reliability growth curve due to the errors of different severity depending on the testing environment.

The models discussed above are derived under the assumption that the fault detection and/or removal rate remains constant over the entire testing period. But during the period of testing, it is observed that the FDR or FRR may not be constant and can change as the testing progresses. The changes in the FDR or FRR can be accounted due to the changes in the testing environment, testing strategy, complexity and size of the functions under testing, skill, motivation and constitution of the testing and debugging team etc. The change in FDR or FRR can be analyzed using the "change point concept". The idea behind the change point concept is that it divides the testing period into subintervals and assumes that during a particular subinterval the testing strategy and testing environment are more or less similar and are slightly different from the other subintervals. The FDR and/or FRR is either assumed to be constant or a function of testing time during each subinterval but varies (constant but different or a different relation to time) from the other subintervals. The concept of change point was started by Zhao[22] who introduced the change-point analysis in Hardware and Software reliability. Shyur,[20] Kapur *et al.*[7,11] also made their contributions in this area. Shyur[20] has developed an SRGM for multiple types of faults incorporating the concept of change point keeping the FDR constant and different for different types of faults.

In this paper we propose a general software reliability growth model considering three types of faults (simple, hard and complex) in the software system incorporating the effect of changing fault-debugging rate using the change point concept. The general framework of the model can be reformulated for specific applications and testing environment with ease. Further we have formulated the model for the software system developed for critical application under a specific testing environment. The model is validated on real life data sets.

This paper is organized as follows: Section 2 presents the model formulation for the proposed model. Section 3 gives the method used for parameter estimation and criteria used for validation and evaluation of the proposed model. The goodness-of-fit of the proposed model is compared with Yamada's[21] exponential model with two types of faults and Shyur's Model[20] for three types of faults with one change point. For comparison purpose the Yamada's[21] exponential model is modified assuming three types of faults in the system. We conclude this paper in Sec. 4.

2. Software Reliability Growth Modelling

2.1. *Proposed model*

All faults lying in a software are not similar. Some of them may be critical from the users point of view i.e. failure due to these faults results in total system failure while others may be critical from the developers point of view in the sense that large amount of testing resources are required for their removal as they are difficult to remove due to complexity of the underlying code. In this paper we have described the severity of faults from the developers view point. Faults are categorized with respect to time they take for isolation and removal after their observation. Faults are classified as "simple" if the time between their observation and removal is negligible else if more efforts and time is required for the removal the fault is classified as "hard fault" and if on the detection of a fault the amount of effort and time required to remove it is much more as compared to hard faults, the fault is classified as "complex fault."[10] This classification implies simple faults are easy to detect and remove. For hard faults the FRR is slower as compared to simple faults, which may increase as the testing progresses and becomes comparable to simple faults towards the end of testing phase, while for complex faults FRR is even slower than hard faults in the beginning which increases with the time span due to learning and may become stable towards the end of testing phase. Applying this type of model to the initial failure data (or past data) gives an idea about the nature of remaining faults and their respective proportions. If this information is made available to the test manager, testing effort can be streamlined to achieve better fault detection and appropriate control can be initiated for a particular category.[10,16]

Here we model the removal phenomenon of the testing and debugging process assuming that the fault removal rate per remaining fault of the each type of fault is different and the rate changes with the change point. The general assumptions and notations of the model are as follows:

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. Each time a failure is observed, an immediate effort takes place to decide the cause of the failure in order to remove it.
4. The time delay between the failure observation and its subsequent removal is assumed to represent the severity of faults. The more severe the fault, more the time delay.
5. During the fault isolation/removal, no new fault is introduced into the system.
6. The fault removal process is perfect.
7. The fault removal rate per remaining fault of the each type of fault is different and the rate changes with the change point.

Notations

$m(t)$:	Expected number of faults identified in the time interval $(0, t]$ during testing phase
τ_i:	Change Points (time from where a change in FRR is observed, $i = 1, 2, 3$)
a:	Total fault content
a_i:	Initial fault content of type i faults (simple, hard and complex) $i = 1, 2, 3$
$b_{ij}(t)$:	Fault removal rate for a fault type i in jth time interval (each time interval corresponding to each change point), $i = 1, 2, 3$; $j = 1, 2, \ldots, n$
$m_i(t)$:	Mean value function of type i faults (simple, hard and complex) $i = 1, 2, 3$
p_i:	Proportion of type i faults in the software's, $i = 1, 2, 3$

The differential equation describing the model can be given as

$$m_i'(t) = b_{ij}[ap_i - m_i(t)] \quad i = 1, 2, 3; \;\; j = 1, 2, \ldots, n \tag{2.1.1}$$

where

$$bij = \begin{cases} b_{i1}(t) & 0 \le t \le \tau_1 \\ b_{i2}(t) & \tau_1 < t \le \tau_2 \\ \ldots & \\ \ldots & \\ b_{in}(t) & t > \tau_n. \end{cases} \tag{2.1.2}$$

The exact solution of the above model equations (2.1.1) can be obtained on substituting the functional forms of the FRR in (2.1.2) and defining the number of change points based on past data or by the experience. Now the mean value function of the expected total number of faults removed from the system is given as

$$m(t) = \sum_{i=1}^{3} m_i(t). \tag{2.1.3}$$

Various diverse testing environment and testing strategies existing for different type of software can be analyzed from the above model by choosing the appropriate forms of the fault removal rates $b_{ij}(t)$ (based on the past failure data and experience of the developer). One of the most simple and general case would be the one if we consider FRR for each type of fault in each change point interval to be constant but distinct for each i and j if we observe exponential failure curve growth pattern for each type of fault. However in case of a general purpose software we may expect that the FRR for each type of fault may increase with time as the testing team gains experience with the code and learning occurs and reaches a certain constant level towards the end of the testing phase. The fault detection rate of hard and/or complex faults is slightly less than that of simple fault type. We may also observe a decreasing FRR towards the end of testing phase since most of the faults lying in the software are removed and failure intensity has become very less. Increasing and/or decreasing trend in FRR can be depicted with the time dependent forms of $b_{ij}(t)$. From the literature study of various fault detection rates used in reliability growth modelling we summarize in the Table 2.1 an interesting application and develop its model assuming three change points. This type of testing environment and testing strategy is usually seen while testing of safety critical systems.

In the beginning of the testing phase the testing team start with some constant FRR for each type of fault with $b_{11} \geq b_{21} \geq b_{31}$ due to motivation of the testing and debugging personnel to remove the critical faults (from users point of view) from the system. After a certain period of time the fault detection rate may increase or decrease due to the various possible changes in the testing process. Here we consider the case when fault detection rate first decreases due to the reason such as addition of new testing personnel, modifications in testing strategy etc. to further improve the overall efficiency of the testing, then start increasing as the testing progresses and the learning occurs ultimately reaching a certain constant level towards the end of the testing phase. For simple faults it is reasonable to assume constant FRR in each change point interval since not much learning testing strategies are applied

Table 2.1. Severity of faults with three change points.

Time interval / Type of fault	Fault detection rates		
	Simple	Hard	Complex
$0 \leq t \leq \tau_1$	b_{11}	b_{21}	b_{31}
$\tau_1 < t \leq \tau_2$	b_{12}	$(b_{22}^2 t)/(1 + b_{22} t)$	$(b_{32}^3 t^2/2)/(1 + b_{32} t + b_{32}^3 t^2/2)$
$\tau_2 < t \leq \tau_3$	b_{13}	b_{23}	$(b_{33}^2 t)/(b_{33}^2 t)$
$t > \tau_3$	b_{14}	b_{24}	b_{34}

for their removals. Here it may be noted that the above explanation reflects one particular situation, various other possibilities also exist depending upon the testing environment and testing strategies employed.

2.2. *Framework for modelling proposed SRGM*

In this section we derive the closed form solution of the model discussed above in Table 2.1. The example suits well to the real time safety critical systems however the testing environment might not be exactly similar as discussed above but the model can be modified accordingly. Here we observe at time τ_1 fault detection rate shows a decrease due to addition of new testing personnel, modifications in testing strategy, fault density etc. carried to further improve the efficiency of the testing team. Then at time moment τ_2, fault detection rate starts increasing as the testing progresses due to the learning of the testing team and ultimately reaching a stable value after τ_3.

Case 1: Simple Faults

Case 1.1. $0 \le t \le \tau_1$

$$m_1'(t) = b_{11}[a_1 - m_1(t)] \tag{2.2.1}$$

Solving the Eq. (2.2.1) under the boundary conditions at $t = 0$, $m_1(t) = 0$ we get

$$m_1(t) = a_1(1 - e^{-b_{11}t}) \tag{2.2.2}$$

Case 1.2. $\tau_1 < t \le \tau_2$

$$m_1'(t) = b_{12}[a_1 - m_1(t)] \tag{2.2.3}$$

Solving the Eq. (2.2.3) under the boundary conditions at $t = \tau_1$, $m_1(t) = m_1(\tau_1)$ we get

$$m_1(t) = a_1(1 - e^{-b_{11}\tau_1 - b_{12}(t-\tau_1)}) \tag{2.2.4}$$

Case 1.3. $\tau_2 < t \le \tau_3$

$$m_1'(t) = b_{13}[a_1 - m_1(t)] \tag{2.2.5}$$

Solving the Eq. (2.2.5) under the boundary conditions at $t = \tau_2$, $m_1(t) = m_1(\tau_2)$ we get

$$m_1(t) = a_1(1 - e^{-b_{11}\tau_1 - b_{12}(\tau_2-\tau_1) - b_{13}(t-\tau_2)}) \tag{2.2.6}$$

Case 1.4. $t > \tau_3$

$$m_1'(t) = b_{14}[a_1 - m_1(t)] \tag{2.2.7}$$

Solving the Eq. (2.2.7) under the boundary conditions at $t = \tau_3$, $m_1(t) = m_1(\tau_3)$ we get

$$m_1(t) = a_1(1 - e^{-b_{11}\tau_1 - b_{12}(\tau_2-\tau_1) - b_{13}(\tau_3-\tau_2) - b_{14}(t-\tau_3)}) \tag{2.2.8}$$

Case 2: Hard Faults

Case 2.1. $0 \leq t \leq \tau_1$

$$m_2'(t) = b_{21}[a_2 - m_2(t)] \tag{2.2.9}$$

Solving the Eq. (2.2.9) under the boundary conditions at $t = 0$, $m_2(t) = 0$ we get

$$m_2(t) = a_2(1 - e^{-b_{21}t}) \tag{2.2.10}$$

Case 2.2. $\tau_1 < t \leq \tau_2$

$$m_2'(t) = \left((b_{22}^2 t)/(1 + b_{22}t)\right)[a_2 - m_2(t)] \tag{2.2.11}$$

Solving the Eq. (2.2.11) under the boundary conditions at $t = \tau_1$, $m_2(t) = m_2(\tau_1)$ we get

$$m_2(t) = a_2[1 - ((1 + b_{22}t)/(1 + b_{22}\tau_1))\ e^{-b_{21}\tau_1 - b_{22}(t-\tau_1)}] \tag{2.2.12}$$

Case 2.3. $\tau_2 < t \leq \tau_3$

$$m_2'(t) = b_{23}[a_2 - m_2(t)] \tag{2.2.13}$$

Solving the Eq. (2.2.13) under the boundary conditions at $t = \tau_2$, $m_2(t) = m_2(\tau_2)$ we get

$$m_2(t) = a_2[1 - ((1 + b_{22}\tau_2)/(1 + b_{22}\tau_1))\, e^{-b_{21}\tau_1 - b_{22}(\tau_2-\tau_1) - b_{23}(t-\tau_2)}] \tag{2.2.14}$$

Case 2.4. $t > \tau_3$

$$m_2'(t) = b_{24}[a_2 - m_2(t)] \tag{2.2.15}$$

Solving the Eq. (2.2.15) under the boundary conditions at $t = \tau_3$, $m_2(t) = m_2(\tau_3)$ we get

$$m_2(t) = a_2[1 - ((1 + b_{22}\tau_2)/(1 + b_{22}\tau_1))e^{-b_{21}\tau_1 - b_{22}(\tau_2-\tau_1) - b_{23}(\tau_3-\tau_2) - b_{24}(t-\tau_3)}] \tag{2.2.16}$$

Case 3: Complex Faults

Case 3.1. $0 \leq t \leq \tau_1$

$$m_3'(t) = b_{31}[a_3 - m_3(t)] \tag{2.2.17}$$

Solving the Eq. (2.2.17) under the boundary conditions at $t = 0$, $m_3(t) = 0$ we get

$$m_3(t) = a_3(1 - e^{-b_{31}t}) \tag{2.2.18}$$

Case 3.2. $\tau_1 < t \leq \tau_2$

$$m_3'(t) = \left(\left(\frac{b_{32}^3 t^2}{2}\right)\Big/\left(1 + b_{32}t + \frac{b_{32}^2 t^2}{2}\right)\right)[a_3 - m_3\,(t)] \tag{2.2.19}$$

Solving the Eq. (2.2.19) under the boundary conditions at $t = \tau_1$, $m_3(t) = m_3(\tau_1)$ we get

$$m_3(t) = a_3\left[1 - \left(\left(1 + b_{32}t + \frac{b_{32}^2 t^2}{2}\right)\Big/\left(1 + b_{32}\tau_1 + \frac{b_{32}^2\tau_1^2}{2}\right)\right) e^{-b_{31}\tau_1 - b_{32}(t-\tau_1)}\right] \tag{2.2.20}$$

Case 3.3. $\tau_2 < t \le \tau_3$

$$m_3'(t) = \left((b_{33}^2 t)/(1 + b_{33}t)\right)[a_3 - m_3(t)] \tag{2.2.21}$$

Solving the Eq. (2.2.21) under the boundary conditions at $t = \tau_2$, $m_3(t) = m_3(\tau_2)$ we get

$$m_3(t) = a_3\left[1 - \left(\frac{1 + b_{32}\tau_2 + \frac{b_{32}^2\tau_2^2}{2}}{1 + b_{32}\tau_1 + \frac{b_{32}^2\tau_1^2}{2}}\right)\left(\frac{1 + b_{33}t}{1 + b_{33}\tau_2}\right) e^{-b_{31}\tau_1 - b_{32}(\tau_2-\tau_1) - b_{33}(t-\tau_2)}\right] \tag{2.2.22}$$

Case 3.4. $t > \tau_3$

$$m_2'(t) = b_{34}[a_2 - m_2(t)] \tag{2.2.23}$$

Solving the Eq. (2.2.23) under the boundary conditions at $t = \tau_3$, $m_3(t) = m_3(\tau_3)$ we get

$$\begin{aligned} m_3(t) = a_3\Bigg[1 - &\left(\frac{1 + b_{32}\tau_2 + \frac{b_{32}^2\tau_2^2}{2}}{1 + b_{32}\tau_1 + \frac{b_{32}^2\tau_1^2}{2}}\right)\left(\frac{1 + b_{33}\tau_3}{1 + b_{33}\tau_2}\right) \\ &\times e^{-b_{31}\tau_1 - b_{32}(\tau_2-\tau_1) - b_{33}(\tau_3-\tau_2) - b_{34}(t-\tau_3)}\Bigg] \end{aligned} \tag{2.2.24}$$

Modelling Total Fault Removal Phenomenon

The total fault removal phenomenon of the proposed SRGM is given by the sum of the mean value function of the simple, hard and complex faults.

Thus, the mean value function of superimposed NHPP is

$$m(t) = m_1(t) + m_2(t) + m_3(t) \tag{2.2.25}$$

which implies

$$\begin{aligned} m(t) = {}& a_1(1 - e^{-b_{11}\tau_1 - b_{12}(\tau_2-\tau_1) - b_{13}(\tau_3-\tau_2) - b_{14}(t-\tau_3)}) \\ &+ a_2(1 - ((1 + b_{22}\tau_2)/(1 + b_{22}\tau_1))e^{-b_{21}\tau_1 - b_{22}(\tau_2-\tau_1) - b_{23}(\tau_3-\tau_2) - b_{24}(t-\tau_3)}) \\ &+ a_3\left(\begin{aligned} &1 - \left(\left(\left(1 + b_{32}\tau_2 + \tfrac{b_{32}^2\tau_2^2}{2}\right)\Big/\left(1 + b_{32}\tau_1 + \tfrac{b_{32}^2\tau_1^2}{2}\right)\right)\right. \\ &\quad \left.\times (1 + b_{33}\tau_3)/(1 + b_{33}\tau_2)\right) e^{-b_{31}\tau_1 - b_{32}(\tau_2-\tau_1) - b_{33}(\tau_3-\tau_2) - b_{34}(t-\tau_3)} \end{aligned}\right) \\ &\text{where } a_1 = ap_1,\ a_2 = ap_2,\ a_3 = a(1 - p_1 - p_2). \end{aligned} \tag{2.2.26}$$

The model represents a particular testing environment and can be modified according to the testing environment under consideration. Thus the model provides flexibility in capturing the varying testing environments in practice.

3. Parameter Estimation

Parameter estimation and model validation is an important aspect of modelling. The mathematical equations of the proposed SRGM are non-linear. Technically, it is more difficult to find the solution for non-linear models using Least Square method and requires numerical algorithms to solve it. Statistical software packages such as SPSS helps to overcome this problem. SPSS is a Statistical Package for Social Sciences. For the estimation of the parameters of the proposed model, Method of Least Square (Non Linear Regression method) has been used. Non Linear Regression is a method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike traditional linear regression, which is restricted to estimating linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables.

3.1. *Comparison criteria for SRGMs*

The performance of SRGMs are judged by their ability to fit the past software fault data (goodness of fit).

3.1.1. *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"?

a. The Mean Square Fitting 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[10] as follows. $MSE = \sum_{i=1}^{k} \frac{(\hat{m}(t_i)-y_i)^2}{k}$ where k is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit.

b. The Akaike Information Criterion (AIC)
The criteria is defined as $AIC = -2$ (the value of the maximum log likelihood function) $+2$ (the number of the parameters used in the model). This index[1,10] takes into account both the statistical goodness of fit and the number of parameters that are estimated in competing models. Lower values of AIC indicate the preferred model.

c. Coefficient of Multiple Determination ($\mathbf{R}^2$)
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.[10] I.e. $\mathrm{R}^2 = 1-\frac{\text{residual SS}}{\text{corrected SS}}$. 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.

d. Prediction Error (PE)

The difference between the observation and prediction of number of failures at any instant of time i is known as PE_i. Lower the value of Prediction Error better is the goodness of fit.[19]

e. Bias

The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit.[19]

f. Variation

The standard deviation of PE is known as variation.
$Variation = \sqrt{(1/N-1)\sum(PE_i - Bias)^2}$ Lower the value of Variation better is the goodness of fit.[19]

g. Root Mean Square Prediction Error

It is a measure of closeness with which a model predicts the observation.
$RMSPE = \sqrt{(Bias^2 + Variation^2)}$ Lower the value of Root Mean Square Prediction Error better is the goodness of fit.[19]

h. The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test (K-S test)[4] is a non-parametric test. It tries to determine if two datasets differ significantly. It is based on the empirical distribution function (ECDF). Since it is non-parametric, it treats individual observations directly and is applicable even in the case of very small sample size, which is usually the case with SRGM validation. Lower the value of Kolmogorov-Smirnov test better is the goodness of fit.

3.2. *Model validation and data description*

To check the validity of the proposed SRGMs and to find out their software reliability growth, it has been tested on two data sets. The Proposed SRGM has been compared with Yamada's modified exponential model and Shyur's model.[20] In the Yamada's two types of faults are assumed in the system, therefore for comparison purpose we have recomputed the mean value function of the removal phenomenon of the model assuming three types of faults in the system. The recomputed mean value function for the model is given by

$$m(t) = \sum_{i=1}^{3} m_i(t) = \sum_{i=1}^{3} ap_i(1 - \exp^{-b_i t}) \tag{3.1}$$

Where b_i; $i = 1, 2, 3$ are the removal rates per remaining fault for the simple, hard and complex faults. The Shyur's model[20] is based on Yamada's model with one change point. Here b_i; $i = 1, 3, 5$ are the FRR before the change point and

Table 3.1. Severity of faults with three change points.

Time interval / Type of fault	Fault detection rates		
	Simple	Hard	Complex
$0 \le t \le \tau_1$	b_1	b_2	b_3
$\tau_1 < t \le \tau_2$	b_1	$(b_2^2 t)/(1 + b_2 t)$	$(b_3^3 t^2/2)/(1 + b_3 t + (b_3^2 t^2/2))$
$\tau_2 < t \le \tau_3$	b_1	b_2	$(b_3^2 t)/(1 + b_3 t)$
$t > \tau_3$	b_1	b_2	b_3

$b_i;\ i = 2, 4, 6$ after the change point. Since only a few data points are available and the number of unknown parameters is sixteen therefore to yield better estimates we assume $b_{11} = b_{12} = b_{13} = b_{14} = b_1$ (say), $b_{21} = b_{22} = b_{23} = b_{24} = b_2$ (say), $b_{31} = b_{32} = b_{33} = b_{34} = b_3$ (say) for the proposed model. The fault removal rates for each type of faults in each time interval are now as given in Table 3.1.

The above assumption implies that the fault removal rate of simple faults doesn't change over time whereas it changes for the hard and complex faults. For simple faults we may observe that not much learning is required for their removals. However it is difficult to remove hard and complex faults and require more resources for their removal therefore the developer puts more efforts to increase the learning of the testing and debugging team to remove these faults. In Shyur's model, the fault removal rates before and after the change point are all taken to be constant for each type of faults whereas in the proposed SRGM the fault removal rate may change with respect to time according to the dynamic testing environment due to the time dependent forms of FRR for each type of fault in each change point interval. The locations of change points are identified by plotting the cumulative number of faults versus time. The Proposed SRGM provides better goodness of fit for both the datasets due to its applicability and flexibility. However, the increased accuracy achieved shows the capability of the model to capture different types of failure datasets. The real time data sets used for estimation are:

DS-1

This data is cited from Ref. 15. The software was tested for 38 weeks during which 2456.4 computer hours were used and 231 faults were removed. The Parameter Estimation result and the goodness of fit results for the proposed SRGM are given in Table 3.2. The goodness of fit curve is given in Fig. 3.1. In this dataset we have taken $\tau_1 = 13$, $\tau_2 = 21$, and $\tau_3 = 29$ for the Proposed SRGM and values of p_1, p_2 and p_3 are computed from the actual data set since data was available separately for each type of faults.

DS-2

This data is cited from Ref. 3. The software was tested for 12 months during which 2657 faults were removed. The Parameter Estimation result and the goodness of fit results for the proposed SRGM are given in Table 3.3. The goodness of fit curve is

Table 3.2. Parameter estimates for DS-1.

Models	Parameter estimation									
	a	b_1	b_2	b_3	b_4	b_5	b_6	p_1	p_2	p_3
Yamada SRGM (3.1)	263	0.038	0.031	0.030	—	—	—	0.640	0.342	0.017
Shyur SRGM[20]	264	0.050	0.047	0.020	0.031	0.006	0.046	0.640	0.342	0.017
Proposed SRGM	260	0.049	0.030	0.010	—	—	—	0.640	0.342	0.017

Goodness of Fit Metrics

Models	Comparison criteria						
	R^2	MSE	AIC	Bias	Variation	RMSPE	K-S Test
Yamada SRGM (3.1)	0.96935	113.79	228.72	−2.41	10.53	10.80	0.198
Shyur SRGM[20]	0.96937	113.70	223.32	−2.80	10.62	10.80	0.234
Proposed SRGM	0.98051	72.37	193.57	−1.61	8.46	8.61	0.096

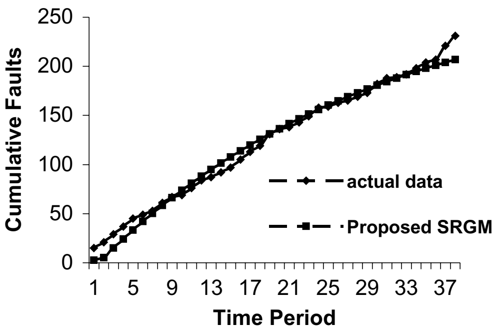

Fig. 3.1. Goodness of fit curve for DS-1 (231 faults).

Table 3.3. Parameter estimates for DS-2.

Models	Parameter estimation									
	a	b_1	b_2	b_3	b_4	b_5	b_6	p_1	p_2	p_3
Yamada SRGM (3.1)	2899	0.190	0.185	0.103	—	—	—	0.445	0.364	0.190
Shyur SRGM[20]	3386	0.143	0.127	0.133	0.122	0.128	0.125	0.442	0.433	0.125
Proposed SRGM	3289	0.176	0.089	0.048	—	—	—	0.779	0.167	0.058

Goodness of Fit Metrics

Models	Comparison criteria						
	R^2	MSE	AIC	Bias	Variation	RMSPE	K-S Test
Yamada SRG (3.1)	0.98987	4971.95	202.17	7.42	73.23	73.61	0.510
Shyur SRGM[20]	0.99097	4445.64	238.21	−6.92	55.84	68.59	0.686
Proposed SRGM	0.99525	2329.77	184.91	−3.96	43.76	49.89	0.467

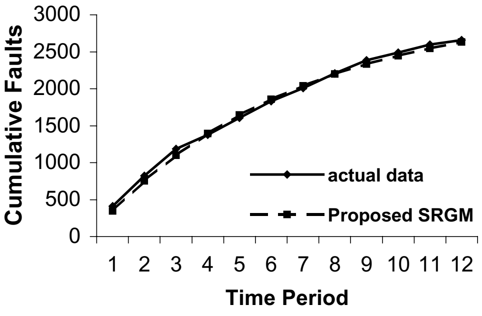

Fig. 3.2. Goodness of fit curve for DS-2 (2657 faults).

given in Fig. 3.2. In this dataset we have taken $\tau_1 = 3$, $\tau_2 = 5$, and $\tau_3 = 11$ for the Proposed SRGM.

It is evidently seen from the tables that the proposed SRGM fits better than both Yamada SRGM (Eq. (3.1)) and Shyur's Model[20] in terms of MSE, AIC, Bias, Variation, RMSPE, R^2, K-S Test. Here it can also be seen that $b_1 > b_2 > b_3$ as the testing teams have to spend more time to analyze and remove the cause of failure of hard and complex faults and therefore require greater efforts to remove them as the faults in the components comprising a complete software can be of different severity. For the first data set the results of Yamada model and Shyur model are very close. For the proposed SRGM the results are better for both the data sets. The values of initial fault contents a_1, a_2, a_3 can be calculated from the Tables 3.2–3.3 for both the datasets i.e DS-1 and DS-2 using $a_i = ap_i$; $i = 1, 2, 3$.

4. Conclusion

In this paper, we have proposed SRGM defining errors of different severity using Change Point concept. The goodness-of-fit of the proposed SRGM has been compared with Yamada SRGM. The proposed SRGM can depict different types of failure growth curves depending on the estimated values of its parameters. The results obtained show better fit and wider applicability of the model to different types of failure datasets. The higher level of accuracy and better predictability for the reliability of the software being tested makes the model quite valuable for the situations where safe and risk free operation of the system are the high priority. From the numerical illustrations, we see that the models provide with considerably improved results with better predictability because of lower MSE, AIC, Variation, RMSPE, Bias and higher R^2. Application of the model is shown for a particular testing environment. The model can be modified according to the underlying testing environment. Changing the functional forms of the various fault removal rates for each type of fault in time interval captures the variability of testing environment. Proposed SRGM with suitable changes can be applied to distributed environment and will be brought out in future.

References

1. H. Akaike, A new look at statistical model identification, *IEEE Transactions on Automatic Control* **19** (1974) 716–723.
2. Bittanti *et al.*, A flexible modelling approach for software reliability growth, in *Software Reliability Modelling and Identification*, (eds.) G. Goos and J. Harmanis, Springer Verlag, Berlin (1988) 101–140.
3. W. D. Brooks and R. W. Motley, Analysis of discrete software reliability models, Technical Report (RADC-TR-80-84), New York, Rome Air Development Center (1980).
4. Chakravart, Laha and Roy, *Handbook of Methods of Applied Statistics,* Vol. I, John Wiley and Sons (1967) 392–394.
5. T. Downs and A. Scott, Evaluating the performance of software reliability models, *IEEE Transactions on Reliability* **41**(4) (1992) 532–538.
6. A. L. Goel and K. Okumoto Time dependent error detection rate model for software reliability and other performance measure, *IEEE Transactions on Reliability* **R-28** (3) (1979) 206–211.
7. P. K. Kapur, A. Gupta, O. Shatnawi and V. S. S. Yadavalli, 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* **2**(3) (2006).
8. P. K. Kapur, S. Younes and S. Agarwala, Generalised Erlang Model with n-types of faults, ASOR Bulletin **14** (1995) 5–11.
9. P. K. Kapur, A. K. Bardhan and O. Shatnawi, Why software reliability growth modelling should define errors of different severity, *Journal of the Indian Statistical Association* **40**(2) (2002) 119–142.
10. P. K. Kapur, R. B. Garg and S. Kumar, *Contributions to Hardware and Software Reliability*, World Scientific, Singapore (1999).
11. P. K. Kapur, V. B. Singh, S. Anand and V. S. S. Yadavalli Software reliability growth model with change point and effort control using a power function of the testing time, *International Journal of Production Research*, preview article (2006) 1–7.
12. P. K. Kapur, S. Younes and P. S. Grover, Software reliability growth model with errors of different severity, *Computer Science and Informatics* (India) **25**(3) (1995) 51–65.
13. P. K. Kapur and R. B. Garg, A software reliability growth model for an error removal phenomenon, *Software Engineering Journal* **7** (1992) 291–294.
14. P. K. Kapur, A. K. Bardhan and S. Kumar, On categorization of errors in a software, *Int. Journal of Management and System* **16**(1) (2000) 37–48.
15. P. N. Misra Software reliability analysis, *IBM System Journal* **22**(3) (1983) 262–270.
16. J. D. Musa, A. Iannini and K. Okumoto, *Software Reliability: Measurement, Prediction, Applications*, Professional Edition: Software Engineering Series, McGraw–Hill, New York (1990).
17. M. Ohba, Software reliability analysis models, *IBM Journal of Research and Development* **28** (1984) 428–443.
18. H. Pham, *Software Reliability Assessment: Imperfect Debugging and Multiple Fauilure Types in Software Development*, EG & G-RAMM-10737, Idaho National Engineering Laboratory (1993).
19. K. Pillai and V. S. S. Nair, A model for software development effort and cost estimation, *IEEE Transactions on Software Engineering* **23**(8) (1997) 485–497.
20. H. J. Shyur, A stochastic software reliability model with imperfect-debugging and change-point, *Journal of Systems and Software Society Press* **66** (2003) 135–141.

21. S. Yamada, S. Osaki and H. Narithisa, A software reliability growth model with two types of errors, *RAIRO* **19** (1985) 87–104.
22. M. Zhao, Change-point problems in software and hardware reliability, *Communications in Statistics-Theory and Methods* **22**(3) (1993) 757–768.

About the Main Author

P. K. Kapur is a Professor and former Head, Department of Operational Research, University of Delhi. He obtained his Ph.D. degree from University of Delhi in 1977. He has published extensively in Indian Journals and abroad in the areas of Hardware reliability, Optimization, Queuing theory, Maintenance and Software Reliability. He has also contributed chapters/papers in edited volumes. He has edited a volume on Operations Research (1996) and two volumes on Quality, Reliability and IT (Trends and Future Directions) in the years 2004 and 2005 and is currently editing another volume on the same theme to be published by Macmillan India, 2007. He has also co-authored a book "Contributions to Hardware and Software Reliability" published from World Scientific, Singapore. He has edited special issues of International Journal of Quality Reliability and Safety Engineering, OPSEARCH and International Journal of Performability Engineering. He is also invited to edit a special issue of International Journal of Communications in Dependability and Quality Management. He has guided M.Tech/Ph.D. theses in Computer Science as well as in Operations Research. He is former President of Operational Research Society of India. He is on the Editorial Board of the International Journal of Performability Engineering and SA Journal of Industrial Engineering and Advisory Editor of Journal of Information and Optimization Sciences. He has been Patron, General Chair/Chair of several National and International Conference.

Copyright of International Journal of Reliability, Quality & Safety Engineering is the property of World Scientific Publishing Company and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.