Flexible software reliability growth model with testing effort dependent learning process

P.K. Kapur a, D.N. Goswami b,*, Amit Bardhan c, Ompal Singh a

a Department of Operational Research, University of Delhi, Delhi 7, India
b S.O.S. in Computer Science, Jiwaji University, Gwalior 11, India
c Department of Operational Research, University of Delhi-South Campus, New Delhi 21, India

Received 1 April 2006; received in revised form 1 November 2006; accepted 20 April 2007
Available online 1 May 2007

Abstract

A lot of importance has been attached to the testing phase of the Software Development Life Cycle (SDLC). It is during this phase it is checked whether the software product meets user requirements or not. Any discrepancies that are identified are removed. But testing needs to be monitored to increase its effectiveness. Software Reliability Growth Models (SRGMs) that specify mathematical relationships between the failure phenomenon and time have proved useful. SRGMs that include factors that affect failure process are more realistic and useful. Software fault detection and removal during the testing phase of SDLC depend on how testing resources (test cases, manpower and time) are used and also on previously identified faults. With this motivation a Non-Homogeneous Poisson Process (NHPP) based SRGM is proposed in this paper which is flexible enough to describe various software failure/reliability curves. Both testing efforts and time dependent fault detection rate (FDR) are considered for software reliability modeling. The time lag between fault identification and removal has also been depicted. The applicability of our model is shown by validating it on software failure data sets obtained from different real software development projects. The comparisons with established models in terms of goodness of fit, the Akaike Information Criterion (AIC), Mean of Squared Errors (MSE), etc. have been presented.

© 2007 Published by Elsevier Inc.

Keywords: Software testing; Testing effort; Software reliability; Software Reliability Growth Model (SRGM); Non-Homogeneous Poisson Process (NHPP)

1. Introduction

The subject Software Engineering has evolved to make possible the formidable task of developing quality software economically. Reliability which measures the chance of failure-free operation of software is the most important quality metric. Failures occur due to the dormant faults of the software. Therefore, the emphasis is
on fault avoidance during the software development process. But it is also a fact that software cannot be made bug-free due to the uncertainties involved and the amount of manual effort involved in the process. This underlines the importance of testing phase during the Software Development Life Cycle (SDLC). During this phase attempt is made to identify and remove as many latent faults as may be possible during the limited time. Hence it is very important to monitor the progress of the testing phase and to estimate the reliability. Software Reliability Growth Models (SRGMs) have proved their utility in this regard [1–4].

According to the ANSI definition [3], “Software reliability is the probability of failure-free software operation for a specified period of time in a specified environment”. An SRGM provides a mathematical relationship between time span of testing or using the software and the cumulative number of faults detected. It is used to assess the reliability of the software during testing and operational phases. Software testing involves running the software and checking for unexpected behavior in software output. The successful test can be considered to be one, which reveals the presence of the latent faults. The process of locating the faults and designing the procedures to detect them is called the debugging process. The chronology of failure occurrence and fault detections can be utilized to provide an estimate of the software reliability and the level of fault content. The software reliability model is the tool, which can be used to evaluate the software quantitatively, develop test status, schedule status and monitor the change in reliability performance.

Numerous Software Reliability Growth Models (SRGMs), which relate the number of failures (fault identified) and the Execution time (CPU time/Calendar time) have been discussed in the literature [5–8,1–3,9,10]. These models are used to predict fault content and reliability of the software. At the same time, they are also used to predict release time of the software [8,11]. SRGMs have also been used to control the way testing resources are applied i.e. in controlling the testing effort [12]. Failure curves are either exponential or S-shaped in nature. Many SRGMS exist that describe any one of these curves. There are others known as flexible SRGMS which can depict, depending upon parameter values, both exponential and S-shaped growth curves [5,1,13]. Most SRGMs do not distinguish between software failure and fault isolation/removal processes. But in reality the actual removal of a fault is done after a failure is reported and corresponding fault is isolated. Hence the removal is done in two phases. In the first phase, the failure identification team isolates a failure. In the second phase another team primarily consisting of programmers removes the fault causing that failure. The management allocates resources to both the teams. Yamada et al. [10] developed an S-shaped SRGM taking into consideration the time lag between the two stages, i.e. failure and removal. However, the removal rate is assumed to be constant. Huang and Lin [8] have presented modeling framework where more general models could be formulated. The framework recognizes the dependency between faults and also the tile lag between failure identification and fault removal. They have also given a fairly exhaustive survey of similar models from literature.

In this paper, we develop an SRGM with the additional assumptions that:

1. Software faults are of two types, namely, Type-I and Type-II which are of different severity. Testing effort required depends on the severity of faults.
2. The fault removal rate is time dependent with respect to testing effort. This can account for learning which increases with testing time.

During the testing phase fault exposure rate strongly depends on the skill of test teams, program size and software testability. Thus in this paper we treat the fault removal rate as a function of time and testing effort. This assumption makes the model more flexible and it can capture variety of software reliability growth curves. As discussed, the proposed SRGM incorporates both time lag and the nature of testing effort.

During testing, resources such as manpower and time (computer time) are consumed. The failure (fault identification) and removal are dependent upon the nature and amount of resources spent. The time dependent behavior of the testing effort has been studied earlier [7,1,3,14]. The exponential and Rayleigh curves are used to describe the relationship between the testing effort consumption and testing time (the calendar time). Exponential curve is used if the testing resources are uniformly consumed with respect to the testing time and Rayleigh curve otherwise. Logistic and Weibull type functions have also been used to describe testing effort. The SRGM with testing effort proposed in this paper is based on Non-Homogeneous Poisson Process
(NHPP) assumption [6,1,3,4]. The model has been validated on three software fault identification data sets.

**Notations used**

- \( X(t) \): cumulative testing effort in the time interval \((0,t]\)
- \( c(t) \): time dependent testing effort rate
- \( \alpha, \gamma \): constants
- \( a \): initial fault content
- \( i (i = 1,2) \): type of fault (Type-I and Type-II)
  - \( a_i \): initial fault content of Type \(i\) faults \(\sum_{i=1}^{2} a_i = a\)
- \( b_i \): proportionality constant failure rate/fault isolation rate per fault for Type \(i\) faults
- \( b_2(t) \): logistic learning function, i.e. fault removal rate per fault for Type-II faults
- \( m_d(t) \): mean number of failures caused by Type \(i\) faults by time \(t\)
- \( m_r(t) \): mean number of faults removed of Type \(i\) faults by time \(t\)
- \( \eta \): a constant parameter in the logistic learning function

**2. Modelling testing effort**

A Software Reliability Growth Model (SRGM) explains the time dependent behavior of fault removal. Several SRGMs have been proposed in software reliability literature under different sets of assumptions and testing environment, yet more are being proposed. The proposed SRGM in this paper takes into account the time dependent variation in testing effort. The testing effort (resources) that govern the pace of testing for almost all the software projects are [4]:

(a) Manpower which includes
   - (i) Failure identification personnel.
   - (ii) Failure correction personnel.
(b) Computer time.

The key function of manpower engaged in software testing is to run test cases and compare the test results with desired specifications. Any departure from the specifications is termed as a failure. On a failure the fault causing it is identified and then removed by failure correction personnel. The computer facilities represent the computer time, which is necessary for failure identification and correction.

To describe the behavior of testing effort, either Exponential, Rayleigh, Logistic or Weibull function has been used. Exponential and Rayleigh models can be derived from the following differential equation:

\[
\frac{dX(t)}{dt} = c(t)[\alpha - X(t)], \tag{1}
\]

where \(c(t)\) is the time dependent rate at which testing resources are consumed, with respect to the remaining available resources. Solving Eq. (1) under the initial condition \(X(0) = 0\) we get

\[
X(t) = \alpha \left[1 - \exp \int_0^t c(k)dk\right]. \tag{2}
\]

When \(c(t) = \beta\) a constant

\[
X(t) = \alpha \left(1 - e^{-\beta t}\right). \tag{3}
\]

If \(c(t) = \beta \cdot t\) (1) gives a Rayleigh type curve

\[
X(t) = \alpha \left(1 - e^{-\beta^2 t^2}\right). \tag{4}
\]
Huang et al. [7] developed an SRGM, based upon NHPP with logistic testing effort function. SRGM with logistic testing effort function provides better result on some failure data sets. The cumulative testing effort consumed in the interval \((0, t]\) has the following form:

\[
X(t) = \frac{a}{1 + \beta e^{-\alpha t}},
\]

where \(a\), \(\beta\) and \(c\) are constants.

More flexible and general testing effort function can be obtained using Weibull function and the cumulative testing effort consumed in the interval \((0, t]\) has the following form:

\[
X(t) = a(1 - e^{-\beta c ^{b}t}),
\]

where \(a\), \(\beta\) and \(c\) are constants.

In the next section, we discuss the Software Reliability Growth Modeling using one of these testing effort functions.

### 3. Software Reliability Growth Modelling

The influence of testing effort has been included in some SRGMs [7,8,3,14]. The SRGM with testing effort developed in this paper takes into account the time lag between the failure and fault isolation/removal processes. The model has the following explicit 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. The faults existing in the software are of two types: Type-I and Type-II. They are distinguished by the amount of testing effort needed to remove them and modeled by one and two stage removal processes, respectively.
4. Each time a failure is observed, an immediate (delayed) effort takes place to decide the cause of the failure and to remove it. 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. Fault removal rate is a logistic learning function as it is expected the learning process will grow with time.
7. Failure rate of the software is equally affected by faults remaining in the software.
8. The fault isolation/removal rate with respect to testing effort intensity is proportional to the number of observed failures whose cause are yet to be identified.

#### 3.1. Modelling the Type-I and Type-II faults

Assuming \(a_1\) and \(a_2\) to be Type-I and Type-II faults initially present in the software, respectively.

##### 3.1.1. Modeling Type-I faults

It is assumed that Type-I faults are simple faults which can be removed instantly as soon as they are observed. Hence Type-I faults are modeled as a one-stage process:

\[
\frac{d}{dt}m_{1r}(t) = \frac{b_1}{x(t)}(a_1 - m_{1r}(t)).
\]

The one stage process as modeled in Eq. (7) describes the failure observation, fault isolation and fault removal processes.

Solving the differential equation (7) under the boundary condition \(m_{1r}(t = 0) = 0\). We get

\[
m_{1r}(t) = a_1(1 - e^{-b_1 x(t)}).
\]
3.1.2. Modeling Type-II faults

It is assumed that the Type-II faults consume more testing effort when compared with Type-I faults. This means that the testing team will have to spend more time to analyze the cause of the failure and therefore requires greater efforts to remove them. Hence the removal process for such faults is modeled as a two-stage process:

\[
\frac{d}{dt} m_2(t) \frac{x(t)}{x(t)} = b_2(a_2 - m_2(t)),
\]

\[
\frac{d}{dt} m_2(t) \frac{x(t)}{x(t)} = b_2(t)(m_2(t) - m_2(t)),
\]

where

\[
b_2(t) = \frac{b_2}{1 + \eta e^{-b_2X(t)}}.
\]

The first stage of the two-stage process is given by Eq. (9). This stage describes the failure observation process. The second stage of the two-stage process given by Eq. (10) describes the delayed fault removal process. During this stage the fault removal rate is assumed to be time dependent. The reason for this assumption is to incorporate the effect of learning on the removal personnel. With each fault removal insight is gained into the nature of faults present and function described in Eq. (11) called logistic function can account for that.

Solving, the above differential equations under the boundary condition, \( t = 0, X(t) = 0, m_2(t = 0) = 0 \) and \( m_2(t = 0) = 0 \), we get

\[
m_2(t) = a_2 \left[ 1 - \left\{ 1 + b_2 \cdot X(t) \right\} e^{-b_2X(t)} \right] \left[ 1 + \eta e^{-b_2X(t)} \right]^{-1}.
\]

3.2. Modelling total fault removal phenomenon

The proposed model is the superposition of the two NHPP with mean value functions given in Eqs. (8) and (12). Thus, the mean value function of superposed NHPP is

\[
m(t) = m_1(t) + m_2(t).
\]

From Eq. (12), the fault removal rate per fault for Type-II faults can be derived as follows:

\[
\hat{b}_2(t) = \frac{\frac{d}{dt} m_2(t)}{a_2 - m_2(t)} = \frac{b_2(1 + \eta + b_2X(t)) - b_2(1 + \eta e^{-b_2X(t)})}{(1 + \eta e^{-b_2X(t)})(1 + \eta + b_2X(t))}.
\]

Note that \( \hat{b}_2(t) \) increase monotonically with time \( t \) and tend to constant \( b_2 \) as \( t \to \infty \). Thus, in the steady state, Type-II faults growth curve behaves similar to the Type-I faults growth curve and hence there is no loss of generality in assuming the steady state rate \( b_2 \) to be equal to \( b_1 \).

Assuming \( a_1 = ap, a_2 = a(1 - p) \) and \( b_2 = b_1 = b \), Eq. (13) can be written as

\[
m(t) = ap(1 - e^{-bX(t)}) + a(1 - p) \left[ \frac{1 - (1 + bX(t)) e^{-bX(t)}}{1 + \eta e^{-bX(t)}} \right].
\]

4. Estimation of parameters

Method of least square and maximum likelihood method can both be used for estimation of parameters. The parameters of testing effort function against failure time can be estimated first, using the effort data. The function giving the best fit needs to be used for estimating parameters of SRGM (15) [1,14]. The testing effort data are given in the form of testing effort \( x_j(x_1 < x_2 < \cdots < x_n) \) consumed in time \( (0, t_j); j = 1, 2, \ldots, n \). The testing effort model parameters \( \alpha \) and \( \beta \) can be estimated by the method of least squares as follows:
Minimize \[ \sum_{j=1}^{n} [X_j - \bar{X}] \]

subject to \( \bar{X}_n = X_n \) (i.e. the estimated value of testing effort is equal to the actual value).

Once the estimates of \( \alpha \) and \( \beta \) are known, the parameters of the SRGM (15) for the modules can be estimated through maximum likelihood estimation method using the underlying Stochastic Process, which is described by a Non-Homogeneous Poisson Process. During estimation, estimated values of \( \alpha \) and \( \beta \) are kept fixed. If the fault removal data for a module is given in the form of cumulative number of faults removed \( y_j \) in time \((0, t_j]\). The likelihood function for that module is given as

\[
L(a, b, \eta | y, W) = \prod_{j=1}^{n} \left[ \frac{m_r(\hat{X}_j) - m_r(\hat{X}_{j-1})}{(y_j - y_{j-1})!} e^{-[m_r(\hat{X}_j) - m_r(\hat{X}_{j-1})]} \right].
\]  

(16)

The estimated values of \( a, b \) and \( \beta \) can now be obtained by maximizing the above expression.

5. Model validation

To check the validity of the proposed model to describe the software reliability growth, it has been tested on three data sets.

5.1. Data set-I (DS-I)

This failure data set is for a command, control and communication system cited in Brooks and Motley [15]. The software was tested for 12 months and 2657 faults were identified during this period. During each month the testing effort was also quantified as a linear combination of man-hour and computer time. Exponential function (3) was chosen to represent the testing resource consumption during the period. The estimated values of the proposed SRGM (15) are presented in Table 1. Values in the parenthesis are the Asymptotic Standard Error of estimates.

5.2. Data set-II (DS-II)

The data are again cited from Brooks and Motley [15]. The fault data set is for a radar system of size 124 KLOC (Kilo Lines Of Code) tested for 35 months in which 1301 faults were removed. Rayleigh effort function (4) was chosen to represent the testing effort as it provided the best fit on the testing effort data. Based upon these estimated parameters, parameters of SRGM (15) were estimated. The results are given in Table 2.

The failure data set is S-shaped. It is observed that the proposed model fits the data well as apparent from very high \( R^2 \) value.

5.3. Data set-III (DS-III)

The data set pertains to Release 1 of Tandem Computer Project cited in Wood [16]. The software test data is available for 20 weeks during which 100 faults were identified. During each week the testing efforts in terms of CPU hours were recorded. Weibull testing effort function (6) was chosen and the SRGM estimation results are presented in Table 3.

<table>
<thead>
<tr>
<th>Model</th>
<th>Testing effort model</th>
<th>( a ) (( 0.003 (0.004) ))</th>
<th>( b ) (( 0.03 (0.003) ))</th>
<th>( \eta ) (( 1303 (260) ))</th>
<th>( p ) (( 8.633 (4.05) ))</th>
<th>( R^2 ) (( 0.349 (0.503) ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed model</td>
<td>35236.79 (4814)</td>
<td>0.03 (0.004)</td>
<td>1303 (260)</td>
<td>0.003 (0.0003)</td>
<td>8.633 (4.05)</td>
<td>0.349 (0.503)</td>
</tr>
</tbody>
</table>
The performance of an SRGM is judged by its ability to fit the past software failure/fault removal data (goodness of fit) and to predict satisfactorily the future behavior of the software failure occurrence/fault detection process from present and past data (predictive validity) [8,4].

In DS-I the failure curve is exponential and in other two failure curves are S-shaped, the proposed SRGM gives very good fit proving its flexibility. The plot of actual values and the estimated values for the three data sets are depicted in Figs. 1–3.

5.4. Comparison criteria

5.4.1. The goodness of fit criterion

1. The mean square fitting error (MSE): The model under comparison is used to simulate failure data, the difference between the expected values, \( \hat{m}(t_i) \) and the observed data \( y_i \), is measured by MSE as follows:

\[
MSE = \frac{\sum_{i=1}^{k} (\hat{m}(t_i) - y_i)^2}{k},
\]

where \( k \) is the number of observations. Lower value of MSE indicates less fitting error, thus better goodness of fit.

2. The Akaike Information Criterion (AIC): It is one of the criteria through which the worth of increase in number of parameters can be evaluated [8,3]:

\[
AIC = -2 \times \log (\text{max of likelihood function}) + 2 \times N,
\]
where $N$ is the number of the parameters used in the model. Lower value of AIC indicates more confidence in the model thus a better fit and predictive validity.

5.5. Relative prediction error [4]

The number of faults removed by time $t_k$ can be predicted by the SRGM and compared to the reported fault removal, i.e. $y_k$. The difference between the predicted value $\hat{m}(t_k)$ and the reported value measures the fault in prediction. The ratio $[(\hat{m}(t_k) - y_k)/y_k]$ is called the Relative Prediction Error (RPE). If the RPE is negative (positive) the SRGM is said to underestimate (overestimate) the fault removal process. Portions of the failure data are sequentially chosen to calculate the RPE. Values close to zero for RPE indicate more accurate prediction, thus more confidence in the model.

In other words, we evaluate the performance of the models using MSE, AIC and RPE metrics. The smaller the value of the metric the better the model fits relative to other models run on the same data set.

The proposed model is compared with two established models due to Yamada et al. [10,14]. The first reference introduced the concept of lag between the failure and fault removal phenomenon, while the second one for the first time showed how testing effort can be incorporated into SRGMs. While estimating the parameters of the second model, we have followed the suggested procedure and assumed Exponential, Rayleigh and Weibull functions for the three data sets, respectively. The results are produced in Table 4. The results on
RPE are depicted in Fig. 4 and are within reasonable limits (10%). For DS-I due to smaller sample size only two values could be calculated.

6. Conclusion

In this paper a new SRGM based on NHPP has been proposed. The model takes into account different types of errors, the time lag between the failure and the fault removal processes and also incorporates the effect of testing effort. The new SRGM is flexible in nature as it can describe both exponential and S-shaped curves as demonstrated in this paper. This model is being further modified to include the effects of imperfect debugging/error generation. It is also being applied to important management decision making situations like release time problem.

References