Software Reliability Growth modeling with Generalized Exponential testing-effort and optimal SOFTWARE RELEASE policy

By S.M. K. Quadri, N. Ahmad, Sheikh Umar Farooq
Kashmir University, Srinagar, India

Abstract - In this paper, a scheme for constructing software reliability growth model based on Non-Homogeneous Poisson Process is proposed. Although various testing-effort functions for software reliability growth model based on non-homogeneous Poisson process (NHPP) have been proposed. These software reliability growth models are quite helpful for software developers and have been widely accepted and applied by the industry people and by the software developers. We still need to put more testing-effort functions into software reliability growth model for accuracy on estimate of the parameters. In this paper, we will consider the case where the time dependent behaviors of testing-effort expenditures are described by Generalized Exponential Distribution (GED). Software Reliability Growth Models (SRGM) based on the NHPP are developed which incorporates the (GED) testing-effort expenditure during the software-testing phase. It is assumed that the error detection rate to the amount of testing-effort spent during the testing phase is proportional to the current error content. Models parameters are estimated by the Least Square and the Maximum Likelihood estimation methods, and software measures are investigated through numerical experiments on real data from various software projects. The evaluation results are analyzed and compared with other existing models to show that the proposed SRGM with (GED) testing-effort has a fairly and better faults prediction capability and it depicts the real-life situation more fairly. This model can be applied to a wide range of software system. The optimal release policy for this model, based on cost-reliability criterion is also discussed.

Keywords: Software reliability growth model, optimal software release policy, estimation method, testing-effort function, mean value function, non-homogeneous Poisson process.

Classification: GJCST D.2.4
Software Reliability Growth Modeling With Generalized Exponential Testing–Effort And Optimal Software Release Policy

S. M. K. Quadri¹, N. Ahmad², Sheikh Umar Farooq³

Abstract: In this paper, a scheme for constructing software reliability growth model based on Non-Homogeneous Poisson Process is proposed. Although various testing-effort functions for software reliability growth model based on non-homogeneous Poisson process (NHPP) have been proposed. These software reliability growth models are quite helpful for software developers and have been widely accepted and applied by the industry people and by the software developers. We still need to put more testing-effort functions into software reliability growth model for accuracy on estimate of the parameters. In this paper, we will consider the case where the time dependent behaviors of testing-effort expenditures are described by Generalized Exponential Distribution (GED). Software Reliability Growth Models (SRGM) based on the NHPP are developed which incorporates the (GED) testing-effort expenditure during the software-testing phase. It is assumed that the error detection rate to the amount of testing-effort spent during the testing phase is proportional to the current error content. Models parameters are estimated by the Least Square and the Maximum Likelihood estimation methods, and software measures are investigated through numerical experiments on real data from various software projects. The evaluation results are analyzed and compared with other existing models to show that the proposed SRGM with (GED) testing-effort has a fairly and better faults prediction capability and it depicts the real-life situation more fairly. This model can be applied to a wide range of software system. The optimal release policy for this model, based on cost-reliability criterion is also discussed.

Keywords: Software reliability growth model, optimal software release policy, estimation method, testing-effort function, mean value function, non-homogeneous Poisson process.

I. INTRODUCTION

SRGM is mathematical model. It shows how software reliability improves as faults are detected and repaired. SRGM can be used to predict when a particular level of reliability is likely to be attained. Thus, SRGM is used to determine when to stop testing to attain a given reliability level. Software reliability is the probability that the given software functions correctly under a given environment during the specified period of time (Musa et al., 1987; Musa, 1999; Lyu, 1996; Kapur et al., 1999). Therefore, modeling of software reliability accurately and predicting its possible trends are essential for determining overall reliability of the software. Numerous SRGMs been developed during the last three decades (Musa et al., 1987; Xie 1991; Lyu, 1996; Pham, 2000). It is easy to determine some important metrics like time period, number of remaining faults, mean time between failures (MTBF), and mean time to failure (MTTF) through SRGMs. Various SRGMs have been proposed by many authors (Yamada et al., 1986; 1987; 1990; 1991; 1993; Kapur and Garg, 1990; 1996; Kapur and Younes, 1994; Huang et al., 1997, 1999; 2000; Kuo et al., 2001; Huang and Kuo, 2002; Huang, 2005; Bokhari and Ahmad, 2006; Quadri et al., 2006) based on Non-homogeneous Poisson process (NHPP) which incorporates the testing-effort. The testing-effort can be represented as the number of CPU hours; the number of executed test cases; etc. (Yamada and Osaki, 1985b; Yamada et al., 1986; 1993). Most of works on these SRGMs are based on NHPP assuming that the time-dependent behavior of test-effort expenditure is either exponential, Weibull, logistic or generalized logistic curve. However, in many software testing situations, it is sometimes difficult to describe the testing-effort expenditure only by these curves, since actual software data show various expenditure patterns. This paper takes Software reliability growth modeling for the case where the time-dependent behavior of testing-effort expenditures is described by the Generalized Exponential failure model. Its curve is flexible with a wide variety of possible expenditure patterns. Hence, these curves are called Generalized Exponential curve. The proposed framework is a generalization over the previous works on SRGMs with testing-efforts such as Yamada et al. (1986; 1987; 1990; 1991; 1993), Kapur and Garg (1990; 1996), Kapur and Younes (1994), Putnam (1978), Tian et al. (1995), and Quadri et al. (2006). It is assumed that the error detection rate in software testing, is proportional to the current error content and the proportionality is dependent on the current testing-effort expenditures at an arbitrary testing time.
time. SRGMs parameters are estimated by Least Square Estimation (LSE) and Maximum Likelihood Estimation (MLE) methods. Experiments have been carried out based on actual software data from various projects and the results show that the proposed SRGMs with Generalized Exponential Testing-effort function is wider and effective models for software testing phase and is more realistic. Comparative of predictive capabilities between various models are presented and the results reveal that the SRGM with Generalized Exponential testing-effort function can estimate the number of initial faults better that that of other models. In addition, the optimal release policy of this model based on cost-reliability criterion is also discussed.

II. Generalized Exponential Testing-effort Functions

During software testing phase, much testing-effort is consumed itself. The consumed testing-effort indicates how the errors are detected effectively in the software and can be modified by different distributions (Putnam, 1978; Musa et al., 1987; Musa, 1999; Yamada et al., 1986; 1990; 1993; Kapur et al., 1999). Many author reported that Yamada Weibull-Type testing-effort curves may have an apparent peak phenomenon during the software development process when shape parameter \( m = 3, 4, \) or 5 (Hung et al., 1997; 2000; Hung and Kuo, 2002). Basically, the software reliability is highly related to the amount of testing-effort expenditures spent on detecting and correcting software errors. The cumulative testing-effort expenditure consumed in time \((0, t]\) (Ahmad et al. 2007; Yamada & Osaki, 1985a; Yamada et al., 1986; 1993; Huang & Kuo, 2002; Bokhari & Ahmad, 2005) is

\[
W(t) = \alpha \cdot (1 - e^{-\beta t})^\theta, \alpha > 0, \beta > 0, \theta \geq 0.
\]

And the current testing-effort consumed at testing time \( t \) is

\[
w(t) = W(t)' = \alpha \cdot \beta \cdot \theta \cdot e^{-\beta t} \cdot (1 - e^{-\beta t})^{\theta-1}
\]

Where \( \alpha, \beta, \) and \( \theta \) are constant parameters, \( \alpha \) is the total amount of testing-effort expenditures; \( \beta \) is shape and \( \theta \) is scale parameters.

When \( \theta > 1 \), the Generalized Exponential testing-effort function \( w(t) \) reaches its maximum value at the time

\[
t_{\text{max}} = \left[ \frac{2(\theta - 1)}{\beta (\theta + 1)} \right]
\]

Special case:

Yamada exponential curve (Yamada et al., 1986): For \( \theta = 1 \), there is an exponential testing-effort function, and the cumulative testing-effort consumed in time \((0, t]\) is

\[
W(t) = \alpha \cdot (1 - e^{-\beta t}), \alpha > 0, \beta > 0.
\]

III. Software Reliability Growth Model

1) Model Description

We have the following assumptions for Software reliability growth modeling (Yamada and Osaki, 1985a; Yamada et al., 1986; 1993; Kapur et al., 1999; Kuo et al., 2001; Huang and Kuo, 2002; Huang, 2005): The software system is subject to failures at random times caused by errors remaining in the system. Each time failure occurs, the error that caused it is immediately removed and no new errors are introduced. The mean number of errors detected in the time interval \((t, t+\Delta t)\) to the current testing-effort expenditures is proportional to the mean number of remaining errors in the system.

The error detection phenomenon in software testing is modeled by an NHPP. The proportionality is a constant over time. For stochastic modeling of software error detection phenomenon, we define a counting process \( \{N(t), t \geq 0\} \), where \( N(t) \) represents the cumulative number of software errors detected by testing time \( t \) with mean value function \( m(t) \). We can then formulate a SRGM based on NHPP under the assumption of Goel and Okumoto (1979) as

\[
\Pr\{N(t) = n\} = \frac{[m(t)]^n \cdot e^{-m(t)}}{n!}, n = 0, 1, 2, ...
\]

In general, an implemented software system is tested to detect and correct software error in the software development process. During the testing phase software errors remaining in the system cause software failure and the errors are detected and corrected by test personnel. Based on the assumptions, if the number of detected errors by the current testing-effort expenditures is proportional to the number of remaining errors, then we obtain the following different equation Yamada and Osaki, 1985a; Yamada et al., 1986; 1993; Yamada and Ohtera, 1990; Huang and Kuo, 2002; Huang, 2005; Bokhari and Ahmad, 2006):

\[
\frac{dm(t)}{dt} / w(t) = r \cdot [a - m(t)], a > 0, 0 < r < 1.
\]
is the current testing-effort expenditure at time \( t \), \( a \) is the expected number of initial error in the system, and \( r \) is the error detection rate per unit testing-effort at time \( t \). Solving the above differential equation, we have

\[
m(t) = a \cdot (1 - e^{-rW(t)}) .
\]

Substituting \( W(t) \) from (1), we get

\[
m(t) = a \cdot (1 - e^{-\alpha (1-e^{-\beta t})}) .
\]

This is an NHPP model with mean value function considering the Generalized Exponential testing-effort expenditure.

In addition, the failure intensity at testing time \( t \) of the NHPP is given by

\[
\lambda(t) = \frac{dm(t)}{dt} = a \cdot r \cdot w(t) \cdot e^{-rW(t)}
\]

The expected number of errors to be detected eventually is

\[
m(\infty) = a \cdot (1 - e^{-r\bar{a}}) .
\]

This implies that even if a software system is tested during an infinitely long duration, all errors remaining in the system cannot be detected (Yamada et al., 1986; 1993). Thus, the mean number of undetected errors if a test is applied for an infinite amount of time is

\[
a - m(\infty) = a - a \cdot (1 - e^{-r\bar{a}}) = a \cdot e^{-r\bar{a}} .
\]

That is, not all the original errors in a software system can be fully tested with a finite testing effort since the effort expenditure is limited to \( a \).

2) Software Reliability Measures

Based on the NHPP model with \( m(t) \), we can derive the following quantitative measures for reliability assessments (Goe and Okumoto, 1979; Yamada et al., 1993). If \( \overline{N}(t) \) represent the number of errors remaining in the system at testing time \( t \), then the mean of \( \overline{N}(t) \) and its variance are given by

\[
r(t) = E[\overline{N}(t)] = E[N(\infty) - N(t)]
\]

\[
= m(\infty) - m(t) = a \cdot (e^{-rW(t)} - e^{-rW(\infty)})
\]

\[
= Var[\overline{N}(t)].
\]

The software reliability representing the probability that no failures occur in the time interval \( (t, t + \Delta t) \) given that the last failure occurred at testing time \( \Delta t \), is given by

\[
R = R(\Delta t) = e^{-m(t) - m(t + \Delta t)} = e^{-a \cdot rW(t) - e^{-rW(t + \Delta t)}}
\]

The instantaneous mean time between failures (MTBF) at arbitrary testing can be defined as a reciprocal of error detection rate in (8). Then, the instantaneous MTBF is given by

\[
MTBF(t) = \frac{1}{\lambda(t)} = \frac{e^{(\beta t + r \alpha (1 - e^{-\beta t}) - \beta \bar{a})}}{a \cdot r \cdot \alpha \cdot \beta \cdot \theta (1 - e^{-\beta t}) \cdot \beta} .
\]

IV. ESTIMATION OF PARAMETERS

In order to validate the proposed model and to compare its performances with other existing models, experiments on actual software failure data will be performed. MLE and LSE techniques are used to estimate the model parameters (Musa et al., 1987; Musa, 1999; Lyu, 1996). Sometimes, however, the likelihood equations may be complicated and difficult to solve explicitly. In that case one may have to solve with some numerical methods to obtain the estimates. On the other hand, LSE, like MLE, is fairly general technique which can be applied in most practical situations for small or medium sample sizes and may provide better estimates (Musa et al., 1987; Huang et al., 1997; Huang and Kuo, 2002). It minimizes the sum of squares of the deviations between what we expect and what we actually observe.

1) Least Square Estimation Method

The parameters \( \alpha, \beta, \text{ and } \theta \) in the Generalized Exponential test-effort function (1) can be estimated by the method of LSE. These parameters are determined for \( n \) observed data pairs in the form \((t_k, W_k)(k = 1, 2, \ldots, n; 0 < t_1 < t_2 < \ldots < t_n)\), where \( W_k \) is the cumulative testing-effort consumed in time \((0, t_k]\). The least square estimators \( \alpha, \beta, \text{ and } \theta \) can be obtained by minimizing:

\[
S(\alpha, \beta, \theta) = \sum_{k=1}^{n} [\ln W_k - \ln \alpha - \theta \cdot \ln(1 - e^{-\beta t_k})]^2
\]
Differentiating $S$ partially with respect to $\alpha$, $\beta$ and $\theta$, setting the partial derivatives to zero we obtain the set of nonlinear equations, respectively. After some algebraic simplification, we get the estimates of $\alpha$ and $\theta$:

$$\hat{\alpha} = e^{\left\{ \sum W_k - \theta \cdot \ln(1 - e^{-\beta \cdot t_k}) \right\} / n}$$

The estimate of other parameter $\beta$ can be obtained by substituting values of $\hat{\alpha}$ and $\hat{\theta}$ into the following equation:

$$\sum_{k=1}^{n} \left[ \frac{\ln W_k - \ln(1 - \alpha - \theta \cdot \ln(1 - e^{-\beta \cdot t_k}))}{1 - e^{-\beta \cdot t_k}} \right] = 0$$

2) Maximum Likelihood Estimation Method

Suppose that the estimated testing-effort parameters $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\theta}$ in the Generalized Exponential testing-effort function have been obtained by the method of least squares discussed earlier. The estimators for $a$ and $r$ are determined for $n$ observed data pairs in the form $(t_k, y_k) (k = 1, 2, ..., n; 0 < t_1 < t_2 < ... < t_n)$, where $y_k$ is the cumulative number of software errors detected up to time $t_k$ or $(0, t_k)$. Then the likelihood function for the unknown parameters $a$ and $r$ in the NHPP model with $m(t)$ in (6), is given (Kapur et al., 1999; Musa et al., 1987) by

$$L'(a, r) = P\{N(t_k) = y_i, i = 1, 2, ..., n\}$$

$$= \prod_{k=1}^{n} \frac{[m(t_k) - m(t_{k-1})]^{(y_k - y_{k-1})}}{(y_k - y_{k-1})!} \cdot e^{-[m(t_k) - m(t_{k-1})]}$$

$$(y_0 = 0, y_n = n)$$

Thus, $L = \ln L' = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln m(t_k) - m(t_{k-1}) - \sum_{k=1}^{n} [m(t_k) - m(t_{k-1})] - \sum \ln(y_i - y_{i-1})$!

From (5) we know that $m(t_k) - m(t_{k-1}) = a \cdot [e^{-r \cdot W(t_k)} - e^{-r \cdot W(t_{k-1})}]$ and then we have

$$\sum_{k=1}^{n} [m(t_k) - m(t_{k-1})] = m(t_n) = a[1 - e^{-r \cdot W(t_n)}]$$

Thus, $L = \sum_{k=1}^{n} (y_k - y_{k-1}) \cdot \ln a + \sum_{k=1}^{n} (y_k - y_{k-1}) \cdot \ln \left[ e^{-r \cdot W(t_k)} - e^{-r \cdot W(t_{k-1})} \right] - a[1 - e^{-r \cdot W(t_n)}] \cdot \sum_{k=1}^{n} \ln(y_k - y_{k-1})$

$$(13)$$
The maximum-likelihood estimates (MLE) of reliability growth model’s parameters \(a\) and \(r\) can be obtained by solving the following equations, that is
\[
\frac{\partial L}{\partial a} = \sum_{k=1}^{n} \frac{(y_k - y_{k-1})}{a} - 1 + e^{-rW(t_k)} = 0,
\]
\[
\frac{\partial L}{\partial r} = \sum_{k=1}^{n} \left( \frac{-e^{-rW(t_k)}}{e^{-rW(t_k)} - e^{-rW(t_{k-1})}} \right) \left( -e^{-rW(t_k)} W(t_k) + e^{-rW(t_{k-1})} W(t_{k-1}) - a W(t_k) e^{-rW(t_k)} \right) = 0.
\]
After some algebraic simplification, we get
\[
\hat{a} = \frac{y_n}{1 - e^{-rW(t_n)}} = \frac{y_n}{1 - \phi_n}, \quad \text{(15)}
\]
\[
a \cdot W(t_n) \cdot \phi_n = \sum_{k=1}^{n} \frac{(y_k - y_{k-1}) [W(t_k) \phi_n - W(t_{n-1}) \phi_{k-1}]}{\phi_{k-1} - \phi_k} = \text{ (16)}
\]
Where \(\phi_k = e^{-rW(t_k)}, k = 1, 2, \ldots, n\).

By solving (15) and (16) using appropriate technique of numerical method, one can get the \(a\) and \(r\). If the sample size \(n\) of \((t_k, y_k)\) is sufficiently large, then the maximum-likelihood estimates \(\hat{a}\) and \(\hat{r}\) asymptotically follow bivariate s-normal (BVN) distribution (Nelson, 1982; Okumoto and Goel, 1980).

The variance-covariance matrix \(\Sigma\) in the asymptotic properties of (17) is useful in qualifying the variability of the estimated parameters \(\hat{a}\) and \(\hat{r}\), and is the inverse of the Fisher information matrix \(F\), i.e., \(\Sigma = F^{-1}\), given by the expectation of the negative of second partial derivative of \(L\) as
\[
F = \begin{bmatrix}
E\left(\frac{\partial^2 L}{\partial a^2}\right) & E\left(\frac{\partial^2 L}{\partial a \partial r}\right) \\
E\left(\frac{\partial^2 L}{\partial r \partial a}\right) & E\left(\frac{\partial^2 L}{\partial r^2}\right)
\end{bmatrix} = \begin{bmatrix}
f_a & g_s \\
g_s & a \cdot \sum_{k=1}^{n} \frac{(g_k - g_{k-1})^2}{(f_k - f_{k-1})}
\end{bmatrix} \quad \text{(18)}
\]
Where \(g_k = W(t_k) \cdot e^{-rW(t_k)}\) and \(f_k = 1 - e^{-rW(t_k)}\) for \(k = 1, 2, \ldots, n\).

After substituting the values of the estimates of \(a\) and \(r\) in (18) one can estimate \(F^{-1}\). The estimated asymptotic variance-covariance matrix is
\[
\Sigma = F^{-1} = \begin{bmatrix}
\text{Var}(\hat{a}) & \text{Cov}(\hat{a}, \hat{r}) \\
\text{Cov}(\hat{r}, \hat{a}) & \text{Var}(\hat{r})
\end{bmatrix} \quad \text{(19)}
\]
V. Performance Analysis

1) Comparison Criteria

To evaluate the performance of our software reliability growth model and to make a fair comparison with the other existing SRGM, we describe the following comparison criteria.

1. The Accuracy of Estimate (AE) is defined (Musa et al., 1987; Yamada and Osaki, 1985a; Huang and Kuo, 2002; Kuo et al., 2001) as

\[ AE = \frac{M_a - a}{M_a} \]

Where \( M_a \) is the actual cumulative number of detected errors after the test, and \( a \) is the estimated number of initial errors. For practical purposes, \( M_a \) is obtained from software error tracking after software testing.

2. The mean of Squared Errors (Long-term predictions) is defined (Lyu, 1996; Huang and Kuo, 2002; Kuo et al., 2001) as

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

Where \( m(t_i) \) is the expected number of errors at time \( t_i \) estimated by a model, and \( m_i \) is the observed number of errors at time \( t_i \). MSE gives the qualitative comparison for long-term predictions. A smaller MSE indicates a minimum fitting error and better performance (Huang et al., 1997; Kapur and Garg, 1996; Kapur et al., 1999).

3. The Coefficient of Multiple Determination is defined (Musa et al., 1987; Musa, 1999) as

\[ R^2 = \frac{S(\hat{\alpha},0,1) - S(\hat{\alpha},\hat{\beta},\hat{\theta})}{S(\hat{\alpha},0,1)} \]

Where \( \hat{\alpha} \) is the LSE of \( \alpha \) for the model with only a constant term, that is, \( \beta = 0 \) and \( \theta = 1 \) in (12). It is given by \( \hat{\alpha} = \frac{1}{n} \sum_{i=1}^{n} \ln W_k \). Therefore, \( R^2 \) measures the percentage of total variation about the mean accounted for by the fitted model and tells us well a curve fits the data. It is frequently employed to compare models and assess which model provides the best fit to the data. The best model is the one which provides the higher \( R^2 \), that is, closer to 1 (Kumar et al., 2005). To investigate whether a significant trend exists in the estimated testing-effort, one could test the hypotheses \( H_0: \beta = 0 \) and \( \theta = 1 \) against \( H_1: \beta \neq 0 \) or \( \theta 
eq 1 \) using F-test by merely forming the ratio

\[ F = \frac{S(\hat{\alpha},0,1) - S(\hat{\alpha},\hat{\beta},\hat{\theta})}{S(\hat{\alpha},0,1)/(n-3)} \]

If the value of \( F \) is greater that \( F_{a}(2, n-3) \), which is the \( \alpha \) percentile of the \( F \) distribution with degrees of freedom 2 and \( n-3 \), we can be \((1-\alpha)100\) percent confident that \( H_0 \) should be rejected, that is, there is a significant trend in the testing-effort curve.

4. The Predictive Validity is defined (Musa et al., 1987; Musa, 1999) as the capability of the model to predict future behavior from present and past failure behavior. Assume that we have observed \( q \) failures by the end of test time \( t_q \). We use the failure data up to time \( t_0 (\leq t_q) \) to determine the parameter of \( m(t) \). Substituting the estimates of these parameters in the mean value function yields the estimate of the number of failures \( \hat{m}(t_q) \) by \( t_q \). The estimate is compared with the actually observed number \( q \). This procedure is represented for various values of \( t_q \). The ratio \( \frac{\hat{m}(t_q) - q}{q} \) is called the relative error. Values close to zero for relative error indicate more accurate prediction and hence a better model. We can virtually check the predictive validity by plotting the relative error for normalized test time \( t_q/t_0 \).

2) Numerical Examples

**DS1:** The first set of actual data is from the study by Ohba (1984). The system is PL/1 data base application software, consisting of approximately 1,317,000 lines of code. During the nineteen weeks experiments, 47.65 Cpu hours were consumed and about 328 software errors were removed. The study reports that the total cumulative number of detected faults after a long period of testing is 358. In order to estimate the parameters \( \alpha, \beta, \) and \( \theta \), of the Generalized Exponential testing-effort function; we fit the actual testing effort data into (1) and solve it by using the method of least squares. That is, we minimize the sum of squares given in (12) and the estimated parameters are obtained as:

\[ \hat{\alpha} = 1077.44223, \hat{\beta} = 0.0033447 \text{ and } \hat{\theta} = 0.115837241 \] (21)

Figures 1-2 graphically illustrate the comparisons between the observed failure data and the estimated Generalized Exponential testing-effort data. Here, the fitted curves are shown as a dotted line and the solid line represents actual software data.
Using the estimated parameters $\alpha$, $\beta$, and $\theta$ the other parameters $a, r$ in (6) can be solved numerically by the MLE method. These estimated parameters are $\hat{a} = 565.777, \hat{r} = 0.019634$

Table 1 summarizes the estimated values of parameters with their standard errors and 95% confidence limits for the proposed model.

A fitted curve of the estimated mean value function with the actual software data is plotted in Figure 3. The $R^2$ value for proposed Generalized Exponential testing-effort is 0.989. It can therefore be observed that the Generalized Exponential testing-effort function is suitable for modeling the software reliability of this data set. We also observed that the fitted testing-effort curve is significant since the calculated value $F(=7.90936)$ is greater than $F_{0.05}(2,16)$ and $F_{0.01}(2,16)$. Secondly, the selected models are compared with each other based on objective criteria. Table 2 lists the performance of various SRGM investigated. Kolmogorov Smirnov goodness-of-fit test shows that our proposed SRGM fits pretty well at the 5% level of significance. Figure 4 depicts the estimated intensity function $\hat{\lambda}(t)$ from (7). Following the work in Musa et al. (1987), we compute the relative error in prediction for this data set and the results are plotted in Figure 5. We observed that relative error approaches zero as $t_e$ approaches $t_q$ and the error curve is usually within $\pm 5$ percent. Altogether, from Figures 1-5 and Tables 1-2, we can see that the proposed model has better performance and predicts the future behavior well.

**DS 2:** The second set of actual data is the pattern of discovery of errors by Tohma et al. (1989). The debugging time and the number of detected faults per day are reported. The cumulative number of discovered faults up to twenty two days is 86 and the total consumed debugging times is 93 CPU hours. All debugging data are used in this experiment. The testing effort data are applied to estimate the parameters $a, \beta$, and $\theta$, of the Generalized Exponential testing-effort function described in (1) by using the method of least squares. The estimated values of parameters are $\hat{a} = 161.5248248, \hat{\beta} = 0.070971106$, and $\hat{\theta} = 2.287629712$.

Figures 6-7 show the fitting of the estimated testing-effort by using above estimates. The fitted curves and the actual software data are shown by dotted and solid lines, respectively. The other parameters $a, r$ in (6) can be solved numerically using MLE method for these failure data. The estimators are $\hat{a} = 94.880344797, \hat{r} = 0.025206813$

Table 3 shows the estimated values of parameters with their standard errors and 95% confidence limits for the proposed model.

The fitted curve of the estimated mean value function with the actual software data has been plotted in Figure 8. The $R^2$ value for proposed Generalized Exponential testing-effort is 0.99025. Therefore, we can say that the proposed curve is suitable for modeling the software reliability. Also, the calculated value $F (=9.41)$ is greater than $F_{0.05}(2,19)$ and $F_{0.01}(2,19)$, which concludes that the fitted testing-effort curve is significant for this data set. Table 4 lists the comparisons of proposed model with different SRGMs which reveal that the proposed model has better performance. Kolmogorov Smirnov goodness-of-fit test shows that the proposed SRGM fits pretty well at the 5% level of significance. Finally, we compute the relative error in prediction of proposed model for this data set. Figure 9 shows the estimated intensity function $\hat{\lambda}(t)$ from (7). Figure 10 shows the relative error plotted against the percentage of data used (that is $t_e / t_q$). We observed that relative error approaches zero as $t_e$ approaches $t_q$ and the error curve is usually within $\pm 5$ percent. Therefore, from the Figures 6-10 and Tables 3-4 discussed, it can be concluded that the proposed model gets reasonable prediction in estimating the number of software errors and fits the observed data better than the others.

**DS 3:** The third set of actual data in this paper is the System T1 data of the Rome Air Development Center (RADC) projects and cited from Musa et al. (1987), Musa (1999). The number of object instructions for the system T1 which is used for a real-time command and control application. The size of the software is approximately 21,700 object instructions and developed by Bell Laboratories. The software was tested for twenty one weeks with 9 programmers. During the testing phase, about 25.3 CPU hours were consumed and 136 software errors were removed. The number of errors removed after 3.5 years of test was reported to be 188 (Huang, 2005). Similarly, parameters $a, \beta$, and $\theta$, of Generalized Exponential testing-effort function for this data set can be obtained by using the method of LSE. The estimated values are $\hat{a} = 40.046306265, \hat{\beta} = 0.2153323906$ and $\hat{\theta} = 39.845501631$.

Figures 11-12 show the fitting of the estimated testing-effort by using these estimates. The fitted
curves are shown as a dotted line and the solid line is for actual software data in the graphs. Using the estimated parameters \( \alpha, \beta, \) and \( \theta \), the other parameters \( a, r \) in (6) can be solved numerically by MLE method. The estimates are \( \hat{a} = 134.56655874, \hat{r} = 0.149400537 \).

Table 5 summarizes the experimental results of estimated parameters with their standard errors and 95% confidence limits of parameters for the proposed model.

A fitted curve of the estimated mean value function with the actual software data is plotted in figure 13. The R2 value for proposed Generalized Exponential testing-effort curve is 0.94331 and calculated F value is 8.49, which is greater than \( F_{0.05}(2,18) and F_{0.01}(2,18) \). It can therefore be observed that the proposed model is suitable for modeling the software reliability and the fitted testing-effort curve is highly significant for this data set. Also, Table 6 compares the performance of various SRGM for this data set. The Kolmogorov Smirnov goodness-of-fit test shows that the proposed SRGM fits pretty well at the 5% level of significance. Similarly, we compute the relative error in prediction for proposed model at the 5% level of significance. Similarly, we compute the relative error in prediction for proposed model for this data set. Figure 14 depicts the estimated intensity function \( \hat{\lambda}(t) \) from (7). Figure 15 depicts the relative error plotted against the percentage of data used (that is \( t_e / t_q \)). It is noted that the relative error of the proposed model approaches zero as \( t_e \) approaches \( t_q \).

Finally, Figures 11-15 and Tables 5-6 reveal that the proposed model has better performance than the other models. The proposed model fits the observed data better, and predicts the future behavior well.

VI. OPTIMAL SOFTWARE RELEASE POLICIES

Recently, it is becoming increasingly difficult for the developer to produce highly reliable software systems efficiently. If the length of software testing is long, it can remove many software errors in the system and its reliability increases. However, it leads to increase the testing cost and to delay software delivery. In contrast, if the length of software testing is short, a software system with low reliability is delivered and it includes many software errors which have not been removed in the testing phase. So, it is important that we have to find the solution for the optimal length of the software testing that is called optimal release time and the decision process is called an optimal software release problem (Xie, 1991; Yamada and Osaki, 1985b; Okumoto and Goel, 1980; Kapur & Garg, 1990; Kapur et al., 1999).

1) Software release time based on reliability criteria

Generally, the software-release time problem is associated with the reliability of a software system. First, we discuss the release policy based on the reliability criterion. If we know that the software reliability of this computer system has reached an acceptable reliability level, then we can determine the right time to release the software (Pham, 2000). The conditional reliability function is given in (10).

Differentiate (10) with respect to \( t \), we observe that \( \frac{\partial R}{\partial t} \geq 0 \). Hence \( R(\Delta t | t) \) is a monotonic increasing function of \( t \). Taking the logarithm on both sides of (10), we obtain

\[
\ln R = -[m(t + \Delta t) - m(t)].
\]

(24)

We can easily determine the testing time needed to reach a desired R by solving (24) and (6). It is noted that \( R(t) \) is increasing in \( t \).

2) Numerical Examples

In order to calculate the testing time that needed to reach a desired reliability, we consider the three actual data sets (DS1-DS2) described in the previous section in the following numerical examples.

**DS 1:** In first data set, it is known that \( \hat{\alpha} = 1077.44223, \hat{\beta} = 0.0033447 \) and \( \hat{\theta} = 0.115837241, \hat{\alpha} = 565.777, \hat{\beta} = 0.019634 \). Suppose the software system desires that the testing would be continued till the operational reliability is equal to 0.80 (at \( \Delta t = 0.1 \)), from (24) and (6), we get \( t = 48.37 \) weeks. If the desired reliability is 0.85, then \( t = 53.89 \) weeks. If the desired reliability is 0.95, then \( t = 74.25 \) weeks and if the desired reliability is 0.99, then \( t = 104.31 \) weeks.

**DS 2:** In second data set, from (24) and (6), for \( \hat{\alpha} = 161.5248248, \hat{\beta} = 0.070971106, \) and \( \hat{\theta} = 2.287629712, \hat{\alpha} = 94.880344797, \hat{\beta} = 0.025206813, \) we get testing time \( t = 25.89 \) days, if we assume that the testing of the software system is desired to be continued till the operational reliability is equal to 0.95 (at \( \Delta t = 0.1 \)). If the desired reliability is 0.99, then \( t = 39.68 \) days.

**DS 3:** From the previous estimated parameters: \( \hat{\alpha} = 40.046306265, \hat{\beta} = 0.2153323906 \) and \( \hat{\theta} = 39.845501631, \hat{\alpha} = 134.56655874, \hat{\beta} = 0.149400537 \). Suppose the software system desires that the testing would be continued till the operational reliability is equal
to 0.8 (at $\Delta t = 0.1$), from (24) and (6), we get testing time = 19.41 week. Similarly, the desired reliability is 0.99, then $t = 26.64$ weeks.

3) Software release time based on cost-reliability criteria

In this section, we discuss the cost model and release policy based on the cost-reliability criteria. Using the total software cost evaluated by cost criterion, the cost of testing-effort expenditures during software testing and development phase, and the cost of correcting errors before and after release are given by (Yamada et al., 1984; 1993; Yamada and Osaki, 1985b; Okumoto and Goel, 1980; Kapur et al., 1999; Kapur and Garg, 1980; Kapur et al., 1993; Yamada and Osaki, 1985b; Okumoto and Goel, 1980; Kapur et al., 1999; Kapur and Garg, 1984; 1993; Yamada et al., 1995; Huang and Kuo, 1997; 1999; Huang and Kuo, 2002)

$$C(T) = C_1m(T) + C_2[m(T_k) - m(T)] + C_3\int_0^T w(x)dx, \hspace{1cm} \text{………….. (25)}$$

where $C_1$ is the cost of correcting an error during testing, $C_2$ is the cost of correction an error during operation, $C_2 > C_1$, $C_3$ is the cost of testing per unit testing-effort expenditures and $T_{lc}$ is the software lifecycle length.

From reliability criteria, we can obtain the required testing time needed to reach the reliability objective $R_0$. Our aim is to determine the optimal software release time that minimizes the total software cost to achieve the desired software reliability. Therefore, the optimal software release policy for the proposed software reliability can be formulated as follows:

Minimize $C(T)$
Subject to $R(t + \Delta t | t) \geq R_0$

for $C_1 > C_2 > 0, C_3 > 0$

$\Delta t > 0, 0 < R_0 < 1$.

The procedures to derive the optimal release policy for this problem are evolved step by step and are shown hereafter.

By differentiating (25) with respect to $T$ and equating to zero, yields

$$\frac{dC(T)}{dT} = C_1 \frac{dm(T)}{dT} - C_2 \frac{dm(T)}{dT} + C_3 w(T) = 0.$$  

$$C_3 \frac{\lambda(T)}{C_2 - C_1} = \left(\frac{\lambda(T)}{w(T)}\right) = a \cdot r \cdot e^{-\eta(T)} = r \cdot (a - m(T)).$$  

$$\text{………….. (26)}$$

When $T=0$, then $m(0) = 0$ and $\frac{\lambda(T)}{w(T)} = ar$. When $T \to \infty$, then $m(\infty) = a(1 - e^{-\eta})$ and $\frac{\lambda(T)}{w(T)} = a \cdot r \cdot e^{-\eta}$. Therefore, $\frac{\lambda(T)}{w(T)}$ is monotonically decreasing in $T$. To analyze for the minimum value of $C(T)$, (27) is used to explore two cases of $\frac{\lambda(T)}{w(T)}$ at $T=0$.

Case 1: $\frac{\lambda(0)}{w(0)} = a \cdot r \leq \frac{C_1}{C_2 - C_1}$, then.

$$\frac{\lambda(T)}{w(T)} \leq \frac{C_1}{C_2 - C_1} \hspace{0.5cm} \text{for } 0 < T < T_c$$

It can be obtained that $\frac{dC(T)}{dT} > 0$ for $0 < T < T_c$ and the minimum of $C(T)$ can be found at $T=0$.

Case 2: $\frac{\lambda(0)}{w(0)} = a \cdot r > \frac{C_1}{C_2 - C_1}$, there can be found a finite $T$ such that

$$\frac{\lambda(T)}{w(T)} = \frac{C_3}{C_2 - C_1} = r \cdot (a - m(T)).$$

Taking in both the sides, we get

$$r \cdot \alpha(1 - e^{-\beta T})^\theta = \ln \left(\frac{a \cdot r \cdot (C_2 - C_1)}{C_3}\right)$$

$$\left(1 - e^{-\beta T}\right)^\theta = \frac{1}{r \cdot \alpha} \ln \left(\frac{a \cdot r \cdot (C_2 - C_1)}{C_3}\right)$$

$$T = \frac{1}{\beta} \ln \left(1 - \left(\frac{1}{r \cdot \alpha} \ln \left(\frac{a \cdot r \cdot (C_2 - C_1)}{C_3}\right)\right)^{\theta}\right).$$

Satisfying (27), $\frac{dC(T)}{dT} < 0$ for $0 < T < T_0$ and $\frac{dC(T)}{dT} > 0$ for $T_0 < T < T_{lc}$. It also can be shown that $\frac{d^2C(T)}{dT^2} > 0$ and hence $C(T)$ is a convex function. Thus, minimum of $C(T)$ is at $T=T_0$.

Furthermore, to commit the provisions of the optimal software release policy for the proposed software reliability as depicted above, a finite and unique real number $T_1$ is determined such that $R(t + \Delta t | t) = R_0$, where $0 < R_0 < 1$.

Therefore, summarizing the above analysis and combining cost and reliability requirements, we have the following theorem.

Theorem 1: We assume that:

$C_1 > 0, C_2 > 0, C_3 > 0, C_2 > C_1, \alpha > 0, 0 < R_0 < 1$,

then...
1. If \( \frac{\lambda(0)}{w(0)} > \frac{C_i}{C_j - C_i} \) and \( \frac{\lambda(T)}{w(T)} = \alpha \cdot r \cdot e^{-\gamma \cdot T} < \frac{C_i}{C_j - C_i} \), then \( T^* = \max \{T_0, T_1\} \) for \( R(x|0) < R_0 < 1 \) or \( T^* = T_0 \) for \( 0 < R_0 < R(x|0) \).

2. If \( \frac{\lambda(0)}{w(0)} \leq \frac{C_i}{C_j - C_i} \), then \( R(x|0) < R_0 < 1 \) or \( T^* = T_1 \) for \( 0 < R_0 < R(x|0) \).

3. If \( \frac{\lambda(0)}{w(0)} \geq \frac{C_i}{C_j - C_i} \), then \( T^* \geq T_i \) for \( R(x|0) < R_0 < 1 \) or \( T^* \geq 0 \) for \( 0 < R_0 \leq R(x|0) \).

4) \textbf{Numerical Example}

**DS 1:** In the first set of data, it is known that \( \hat{\alpha} = 1077.44223 \), \( \hat{\beta} = 0.0033447 \), and \( \hat{\phi} = 0.115837241 \), \( \hat{\alpha} = 565.777 \), \( \hat{\beta} = 0.019634 \). In order to determine the optimal software release time, we assume the values of \( C_1 = 1, C_2 = 50, C_3 = 100, T_{IC} = 100, R_0 = 0.90 \), and \( \Delta = 0.1 \) for the analysis. Then we get the optimal release time \( T_0 \) estimated as 2.92 based on minimizing \( C(T) \) of (25), and \( T_i \) is estimated as 61.48 based on satisfying the reliability criterion of \( R(t + \Delta t | t) = R_0 \). These values sustain the relationships of \( \frac{\lambda(0)}{w(0)} > \frac{C_i}{C_j - C_i} \) and \( \frac{\lambda(T)}{w(T)} = \alpha \cdot r \cdot e^{-\gamma \cdot T} < \frac{C_i}{C_j - C_i} \) and \( R(\Delta t | 0) < R_0 \), with which one could imply 1 in Theorem 1 to obtain the optimal software release time \( T^* \) as max \( 20.89, 13.63 \) = 20.89 weeks and the corresponding software cost \( C(T^*) \) is 9072.33.

**DS 2:** In the second set of data, it is known that \( \hat{\alpha} = 161.5248248 \), \( \hat{\beta} = 0.070971106 \), and \( \hat{\phi} = 2.287629712 \), \( \hat{\alpha} = 94.880344797 \), \( \hat{\beta} = 0.025206813 \). To determine the optimal software release time, we assume the values of \( C_1 = 1, C_2 = 50, C_3 = 100, T_{IC} = 100, R_0 = 0.90 \), and \( \Delta = 0.1 \) for the analysis. Then we get the optimal release time \( T_0 \) estimated as 3.18 based on minimizing \( C(T) \) of (25), and \( T_i \) is estimated as 20.93 based on satisfying the reliability criterion of \( R(t + \Delta t | t) = R_0 \). These values sustain the relationships of \( \frac{\lambda(0)}{w(0)} > \frac{C_i}{C_j - C_i} \) and \( \frac{\lambda(T)}{w(T)} = \alpha \cdot r \cdot e^{-\gamma \cdot T} < \frac{C_i}{C_j - C_i} \) and \( R(\Delta t | 0) < R_0 \), with which one could imply 1 in Theorem 1 to obtain the optimal software release time \( T^* \) as max \( 20.93, 3.18 \) = 20.93 days and the corresponding software cost \( C(T^*) \) is 2698.06.

**DS 3:** In the third set of data, it is known that \( \hat{\alpha} = 40.046306265 \), \( \hat{\beta} = 0.215323906 \), and \( \hat{\phi} = 39.845501631 \), \( \hat{\alpha} = 134.56655874 \), \( \hat{\beta} = 0.149400537 \). To determine the optimal software release time, we assume the values of \( C_1 = 1, C_2 = 50, C_3 = 100, T_{IC} = 100, R_0 = 0.90 \), and \( \Delta = 0.1 \) for the analysis. Then we get the optimal release time \( T_0 \) estimated as 13.63 based on minimizing \( C(T) \) of (25), and \( T_i \) is estimated as 20.89 based on satisfying the reliability criterion of \( R(t + \Delta t | t) = R_0 \). These values sustain the relationships of \( \frac{\lambda(0)}{w(0)} > \frac{C_i}{C_j - C_i} \) and \( \frac{\lambda(T)}{w(T)} = \alpha \cdot r \cdot e^{-\gamma \cdot T} < \frac{C_i}{C_j - C_i} \) and \( R(\Delta t | 0) < R_0 \), with which one could imply 1 in Theorem 1 to obtain the optimal software release time \( T^* \) as max \( 20.89, 13.63 \) = 20.89 weeks and the corresponding software cost \( C(T^*) \) is 9072.33.

**VII. Conclusion**

Reliability of software system is one of the most important aspects of the software testing phase of the software development life cycle. In this paper, we have discussed a software reliability growth model based on NHPP, which incorporates Generalized Exponential testing-effort function. It is a much more realistic model and more suitable for modeling the software reliability. We conclude that the incorporated testing-effort function is a flexible and can be used to describe the actual expenditure patterns more faithfully during software development. We also conclude that the proposed SRGM has better performance as compared to the other SRGMs and gives a reasonable predictive capability for the actual software failure data. Therefore, this model can be applied to a wide range of software system. In addition, we have also discussed the optimal release policy based on cost and reliability which is demonstrated through examples.

**References Références Referencias**

Software Reliability Growth modeling with Generalized Exponential testing–effort and optimal SOFTWARE RELEASE policy


Table 1: Summary of estimate of NHPP model parameters for DS1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>565.777</td>
<td>63.195</td>
<td>430.2364, 701.3179</td>
</tr>
<tr>
<td>r</td>
<td>0.019634</td>
<td>0.00311</td>
<td>0.012947, 0.026326</td>
</tr>
</tbody>
</table>

Table 2: Comparative results of different SRGMs for DS1

<table>
<thead>
<tr>
<th>Model</th>
<th>a</th>
<th>r</th>
<th>AE(%)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model (Eqn. (6) )</td>
<td>565.777</td>
<td>0.019634</td>
<td>58.04</td>
<td>111.56</td>
</tr>
<tr>
<td>New Modified Weibull Model (Eqn. (5) with New Modified Weibull curve)</td>
<td>566.66124</td>
<td>0.019596</td>
<td>58.28</td>
<td>103.09</td>
</tr>
<tr>
<td>Yamada exponential model (Eqn. (5) with exponential curve)</td>
<td>828.252</td>
<td>0.0117830</td>
<td>131.35</td>
<td>140.66</td>
</tr>
<tr>
<td>Yamada Rayleigh model (Eqn. (5) with Weibull curve)</td>
<td>565.35</td>
<td>0.0196597</td>
<td>57.91</td>
<td>122.09</td>
</tr>
<tr>
<td>Huang Logistic model</td>
<td>394.08</td>
<td>0.04272</td>
<td>10.06</td>
<td>118.59</td>
</tr>
<tr>
<td>Ohba exponential model</td>
<td>455.37</td>
<td>0.0267368</td>
<td>27.09</td>
<td>206.93</td>
</tr>
<tr>
<td>Inflection S-shaped model</td>
<td>389.1</td>
<td>0.0935493</td>
<td>8.69</td>
<td>133.33</td>
</tr>
<tr>
<td>Delayed S-shaped model</td>
<td>374.05</td>
<td>0.197651</td>
<td>4.48</td>
<td>168.67</td>
</tr>
<tr>
<td>G-O model</td>
<td>760.0</td>
<td>0.0322688</td>
<td>112.29</td>
<td>138.815</td>
</tr>
<tr>
<td>Delayed S-shaped model with Rayleigh</td>
<td>333.14</td>
<td>0.1004</td>
<td>6.93</td>
<td>798.49</td>
</tr>
</tbody>
</table>
Table 3: Summary of estimate of NHPP model parameters for DS2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval</th>
<th>Lower</th>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>94.8803457</td>
<td>3.007957</td>
<td>88.5341</td>
<td>101.22658</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>0.0252068</td>
<td>0.001833</td>
<td>0.02134</td>
<td>0.0290709</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Comparative results of different SRGMs for DS2

<table>
<thead>
<tr>
<th>Model</th>
<th>a</th>
<th>r</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model (Eqn. (6) )</td>
<td>94.8803457</td>
<td>0.0252068</td>
<td>7.557</td>
</tr>
<tr>
<td>New Modified Weibull Model (Eqn. (5) with New Modified Weibull curve)</td>
<td>94.886671</td>
<td>0.001556</td>
<td>5.55</td>
</tr>
<tr>
<td>Yamada Rayleigh model (Eqn. (5) with Rayleigh curve)</td>
<td>87.0318</td>
<td>0.0345413</td>
<td>7.772</td>
</tr>
<tr>
<td>Delayed S-shaped model</td>
<td>88.6533</td>
<td>0.228148</td>
<td>6.31268</td>
</tr>
<tr>
<td>Huang Logistic model</td>
<td>88.8931</td>
<td>0.0390591</td>
<td>25.2279</td>
</tr>
<tr>
<td>G-O model</td>
<td>137.072</td>
<td>0.0515445</td>
<td>25.33</td>
</tr>
<tr>
<td>HGDM</td>
<td>88.30</td>
<td>*</td>
<td>33.6812</td>
</tr>
</tbody>
</table>

Table 5: Summary of estimate of NHPP model parameters for DS3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval</th>
<th>Lower</th>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>134.566 780044</td>
<td>9.3409932</td>
<td>114.76478521</td>
<td>154.36880566</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>0.14940 0537</td>
<td>0.031863061</td>
<td>0.081853865</td>
<td>0.216947210</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Comparative results of different SRGMs for DS3

<table>
<thead>
<tr>
<th>Model</th>
<th>a</th>
<th>r</th>
<th>AE(%)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Model (Eqn. (6) )</td>
<td>134.566780044</td>
<td>0.149400537</td>
<td>28.42</td>
<td>139.60</td>
</tr>
<tr>
<td>Yamada Rayleigh model (Eqn. (5) with Rayleigh curve)</td>
<td>866.94</td>
<td>0.00962</td>
<td>25.11</td>
<td>89.2409</td>
</tr>
<tr>
<td>Delayed S-shaped model</td>
<td>237.196</td>
<td>0.096345</td>
<td>26.16</td>
<td>245.246</td>
</tr>
<tr>
<td>Huang Logistic model</td>
<td>138.026</td>
<td>0.145098</td>
<td>26.58</td>
<td>62.41</td>
</tr>
<tr>
<td>G-O model</td>
<td>142.32</td>
<td>0.1246</td>
<td>24.29</td>
<td>2438.3</td>
</tr>
<tr>
<td>Inflection S-shaped model</td>
<td>159.11</td>
<td>0.0765</td>
<td>15.36</td>
<td>118.3</td>
</tr>
<tr>
<td>Ohba exponential model</td>
<td>137.2</td>
<td>0.156</td>
<td>27.12</td>
<td>3019.66</td>
</tr>
</tbody>
</table>
Figure 1: Observed/Estimated Current Test-effort Vs Time (DS1)

Figure 2: Observed/Estimated Cumulative Test-effort Vs Time (DS1)

Figure 3: Observed/Estimated Cumulative Number of failures Vs Time (DS1)

Figure 4: Estimated Intensity function for DS1

Figure 5: Relative Error Curve for DS1

Figure 6: Observed/Estimated Current Test-effort Vs Time (DS2)
Figure 7: Observed/Estimated Cumulative Test-effort Vs Time (DS2)

Figure 8: Observed/Estimated Cumulative Number of failures Vs Time (DS2)

Figure 9: Estimated Intensity function for DS2

Figure 10: Relative Error Curve for DS2

Figure 11: Observed/Estimated Current Test-effort Vs Time (DS3)

Figure 12: Observed/Estimated Cumulative Test-effort Vs Time (DS3)
Figure 13: Observed/Estimated Cumulative Number of failures Vs Time (DS3)

Figure 14: Estimated Intensity function for DS3

Figure 15: Relative Error Curve for DS3