International Journal of Computer Applications (0975 – 8887)
Volume 6– No.12, September 2010

Software Reliability Growth Modeling with New Modified Weibull Testing–effort and Optimal Release Policy

S. M. K. Quadri
P. G. Department of Computer Sciences,
Kashmir University, Srinagar-190006, India

N. Ahmad
Department of Statistics & Computer Applications,
T. M. Bhagalpur University, Bhagalpur, India

ABSTRACT
In software development life cycle, software testing is one of the most important tasks; and in the testing, software reliably is very important aspect for any category of software systems. A number of testing-effort functions for software reliability growth model based on non-homogeneous Poisson process (NHPP) have been proposed in the past. Although these models are quite helpful for software developers and have been widely accepted and applied in the industries and research centers, 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 New Modified Weibull Distribution (NMWD). Software Reliability Growth Models (SRGM) based on the NHPP are developed which incorporates the (NMWD) 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. Model parameters are estimated by Least Square and Maximum Likelihood estimation techniques, 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 (NMWD) testing-effort has a fairly better faults prediction capability and it depicts the real-life situation more faithfully. This model can be applied to a wide range of software system. In addition, the optimal release policy for this model, based on reliability criterion is discussed.

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

1. INTRODUCTION
Software Reliability Growth Model (SRGM) is a mathematical model of how the 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 [19], [18], [17], [14]. Therefore, modeling of software reliability accurately and predicting its possible trends are essential for determining overall reliability of the software. Various SRGM have been developed during the last three decades and they can provide very useful information about how to improve reliability [19], [28], [17], [22]. One can be easily determined some important metrics like time period, number of remaining faults, mean time between failures (MTBF), and mean time to failure (MTTF) through SRGM.

Several SRGM based on Non-homogeneous Poisson process (NHPP) have been proposed by many authors [36], [35], [29], [32], [33], [11], [12], [13], [9], [7], [8], [16], [5], [4], [2], [24] which incorporates the testing-efforts. The testing-effort can be represented as the number of CPU hours; the number of executed test cases; etc. [31], [36], [33]. Most of these works on SRGM 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.

The proposed framework is a generalization over the previous works on SRGM with testing-effort such as [36], [35], [29], [32], [33], [11], [12], [13], [23], [25], [24]. In this paper, we consider software reliability growth modeling for the case where the time-dependent behavior of testing-effort expenditures is described by the New Modified Weibull (NMW) failure model [26]. Its curve is flexible with a wide variety of possible expenditure patterns. Hence, these curves are called NMW testing-effort, which includes the exponential, Rayleigh, Weibull and log-gamma curves.

SRGM 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 SRGM with NMW testing-effort function is wider and effective model for software testing phase and is more realistic. Comparative predictive capabilities between various models are presented. The results reveal that the SRGM with NMW testing-effort function can estimate the number of initial faults better than that of other models. In addition, the optimal release policy of this model based on cost-reliability criterion is also discussed.

2. NMW TESTING EFFORT FUNCTION
Much testing-effort is consumed during software testing phase itself. The consumed testing-effort indicates how the errors are detected effectively in the software and can be modified by different distributions [23], [19], [18], [36], [29], [33], [14]. Many authors 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 [9], [8], [5]. Basically, the software reliability is highly related to the amount of testing-effort expenditures spent on detecting and correcting software errors. Therefore, we propose the NMW curve as a more flexible testing-effort function. The cumulative testing-effort expenditure consumed in time $(0, t]$ [30], [36], [33], [5], [1] is

$$W(t) = \alpha \cdot (1 - e^{-\beta \cdot t^m \cdot e^{\delta \cdot t}}), \alpha > 0, \beta > 0, m \geq 0, \delta > 0. \quad (1)$$

And the current testing-effort consumed at testing time t is

$$w(t) = \alpha \cdot \beta \cdot (m + \delta \cdot t) \cdot t^{m-1} \cdot e^{\delta \cdot t} \cdot e^{-\beta \cdot t^m \cdot e^{\delta \cdot t}} \quad (2)$$

Where α, β, m, δ are constant parameters, α is the total amount of testing-effort expenditures; β and δ are the scale parameters, and m is shape parameter.

3. SOFTWARE RELIABILITY GROWTH MODEL

For Software reliability growth modeling we have the following assumptions [30], [36], [33], [14], [16], [5], [6]:

1. The software system is subject to failures at random times caused by errors remaining in the system.
2. Each time failure occurs, the error that caused it is immediately removed and no new errors are introduced.
3. Testing-effort expenditures are described by the NMW curve.
4. 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.
5. The error detection phenomenon in software testing is modeled by an NHPP.
6. 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 [3] as

$$\Pr\{N(t) = n\} = \frac{[m(t)]^n \cdot e^{-m(t)}}{n!}, n = 0, 1, 2, \ldots \quad (3)$$

In general, an implemented software system is tested to detect and correct software errors 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 above assumptions, we obtain the following different equation [30], [36], [33], [29], [16], [6], [2]:

$$\frac{dm(t)}{dt} / w(t) = r \cdot [a - m(t)], a > 0, \ 0 < r < 1 \quad (4)$$

Where $m(t)$ represent the expected mean number of errors detected in time $(0, t]$ which is assumed to be a bounded non-decreasing function of t with $m(0) = 0, w(t)$ is the current testing-effort expenditure at time t, a is 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 get

$$m(t) = a.(1 - e^{r \cdot W(t)}). \quad (5)$$

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

$$m(t) = a \cdot (1 - e^{r \cdot \alpha \cdot (1 - e^{-\beta \cdot t^m \cdot e^{\delta \cdot t}})}). \quad (6)$$

This is an NHPP model with mean value function incorporating the NMW testing-effort expenditure.

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

$$\lambda(t) = \frac{dm(t)}{dt} = a \cdot r \cdot w(t) \cdot e^{-r \cdot W(t)} \quad (7)$$

The expected number of errors to be detected eventually is

$$m(\infty) = a \cdot (1 - e^{-r \cdot a}) \quad (8)$$

This implies that even if a software system is tested during an infinitely long duration, all errors remaining in the system cannot be detected [36], [33] Thus, the mean number of undetected errors if a test is applied for an infinite amount of time is

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

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 α.

4. SOFTWARE RELIABILITY MEASURES

Based on the NHPP model with $m(t)$, we can derive the following quantitative measures for reliability assessments [3], [33]. If $\bar{N}(t)$ represent the number of errors remaining in the system at testing time t, then the mean of $\bar{N}(t)$ and its variance are given by

$$r(t) = E[\bar{N}(t)] = E[N(\infty)] - N(t)] = m(\infty) - m(t) = a \cdot (e^{-r \cdot W(t)} - e^{-r \cdot W(\infty)}) = Var[\bar{N}(t)] \quad (9)$$

The software reliability represents the probability that no failure occurs in the time interval $(t, t + \Delta t)$ given that the last failure occurred at time Δt is given by

$$R = R(\Delta t \mid t) = e^{[m(t+\Delta t) - m(t)]} = e^{-a[e^{-r \cdot W(t)} - e^{-r \cdot W(t+\Delta t)}]} \quad (10)$$

The instantaneous mean time between failures (MTBF) at arbitrary testing can be defined as a reciprocal of error detection rate in (7). Then, the instantaneous MTBF is given by

$$MTBF(t) = \frac{e^{r \cdot \alpha \cdot (1 - e^{\beta \cdot t^m \cdot e^{\delta \cdot t}})}}{a \cdot r \cdot \alpha \cdot \beta \cdot (m + \delta \cdot t) \cdot t^{m-1} e^{\delta \cdot t} e^{-\beta \cdot t^m \cdot e^{\delta \cdot t}}} \quad (11)$$

5. ESTIMATION OF PARAMETERS

MLE and LSE techniques are used to estimate the model parameters [19], [18], [17]. 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 [19], [9], [5].

5.1 Least Square Method

The parameters $\alpha, \beta, m,$ and δ in the NMW current testing-effort function (2) can be estimated by the method of LSE. These parameters are determined for n observed data pairs in the form $(t_k, W_k)(k = 1, 2, \ldots, n; \quad 0 < t_1 < t_2 < \ldots < t_n)$, where $\hat{\alpha}, \hat{\beta}, \hat{m},$ and $\hat{\delta}$ can be obtained by minimizing:

$$S(\alpha,\beta,m,\delta)=\sum_{k=1}^{n}[\ln w_k-\ln\alpha-\ln\beta-\ln(m+\delta\cdot t_k)-(m-1)\ln t_k)-\delta\cdot t_k+\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}]^2 \quad (12)$$

Differentiating S partially with respect to α, β, m and δ respectively. The least square estimators $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$, setting the partial derivatives to zero, we obtain the set of nonlinear equations, respectively.

$$\frac{\partial S}{\partial\alpha}=\sum[\ln w_k-\ln\alpha-\ln\beta-\ln(m+\delta\cdot t_k)-(m-1)\cdot\ln t_k-\delta\cdot t_k+\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}]\cdot\frac{-1}{\alpha}=0 \quad (13)$$

Solving the above equation, we can get

$$\hat{\alpha}=\exp[\sum\{[\ln w_k-\ln\beta-\ln(m+\delta\cdot t_k)-(m-1)\cdot\ln t_k-\delta\cdot t_k+\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}\}/n]$$

When we differentiate S with respect to β, we can get

$$\frac{\partial S}{\partial\beta}=\sum[\ln w_k-\ln\alpha-\ln\beta-\ln(m+\delta\cdot t_k)-(m-1)\cdot\ln t_k-\delta\cdot t_k+\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}]\ X\ [\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}-1]=0 \quad (14)$$

When we differentiate S with respect to δ, we can get

$$\frac{\partial S}{\partial\delta}=\sum[\ln w_k-\ln\alpha-\ln\beta-\ln(m+\delta\cdot t_k)-(m-1)\cdot\ln t_k-\delta\cdot t_k+\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}]\cdot[\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}\cdot t_k-t_k-\frac{t_k}{m+\delta\cdot t_k}]=0 \quad (15)$$

and finally, after differentiating S with respect to m we can get

$$\frac{\partial S}{\partial m}=\sum[\ln w_k-\ln\alpha-\ln\beta-\ln(m+\delta\cdot t_k)-(m-1)\cdot\ln t_k-\delta\cdot t_k+\beta\cdot t_k^m\cdot e^{\delta\cdot t_k}]\cdot[\beta\cdot m\cdot t_k^{m-1}\cdot e^{\delta\cdot t_k}\cdot(-\ln t_k)-\frac{1}{m+\delta\cdot t_k}]=0 \quad (16)$$

One can get the estimates of β, δ and m after solving equations (14), (15) and (16) respectively using any suitable technique of numerical method.

5.2 Maximum Likelihood Method

Suppose that the estimated testing-effort parameters $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$ in the NMW current 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 [14], [19] by

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

Where $t_0=0$ and $y_0=0$. Taking logarithm both the sides, we get

$$L=\sum_{k=1}^{n}(y_k-y_{k-1})\cdot\ln[m(t)-m(t_{k-1})]-\sum_{k=1}^{n}[m(t_k)-m(t_{k-1})]-\sum_{k=1}^{n}\ln[(y_k-y_{k-1})!] \quad (17)$$

From (5) we know that

$m(t_k)-m(t_{k-1})=a\cdot[e^{-rW(t_{k-1})}-e^{-rW(t_k)}]$ and then we have

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

Thus,

$$L=\sum_{k=1}^{n}(y_k-y_{k-1})\cdot\ln a+\sum_{k=1}^{n}(y_k-y_{k-1})\ln[e^{-rW(t_{k-1})}-e^{-rW(t_k)}]\quad -a[1-e^{-rW(t_n)}]-\sum_{k=1}^{n}\ln(y_k-y_{k-1}) \quad (18)$$

The maximum-likelihood estimates (MLE) of reliability growth model's parameters a and r can be obtained by solving the following equations, that is

$$\frac{\partial L}{\partial a}=\frac{\sum_{k=1}^{n}(y_k-y_{k-1})}{a}-1+e^{-rW(t_n)}=0,$$

$$\frac{\partial L}{\partial r}=\sum_{k=1}^{n}\frac{(y_k-y_{k-1})}{e^{-rW(t_{k-1})}-e^{-rW(t_k)}}(-e^{-rW(t_{k-1})}.W(t_{k-1})+e^{-rW(t_k)}.W(t_k))-aW(t_n)\cdot e^{-rW(t_n)}=0$$

After some algebraic simplification, we get

$$\hat{a}=\frac{y_n}{1-e^{-rW(t_n)}}=\frac{y_n}{1-\phi_n}, \quad (19)$$

$$a\cdot W(t_n)\cdot\phi_n=\sum_{k=1}^{n}\frac{(y_k-y_{k-1})[W(t_k)\phi_n-W(t_{n-1})\phi_{k-1}}{\phi_{k-1}-\phi_k} \quad (20)$$

Where $\phi_k=e^{-rW(t_k)}, k=1,2,...n$

By solving (19) and (20) using appropriate technique of numerical method, one can get the $\hat{a}$ and $\hat{r}$. 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 [83], [90].

$$\begin{pmatrix}\hat{a}\\ \hat{r}\end{pmatrix}\square\ BVN\left(\begin{pmatrix}\hat{a}\\ \hat{r}\end{pmatrix}\Sigma\right),\ \text{as } n\to\infty. \quad (21)$$

The variance-covariance matrix $\sum$ in the asymptotic properties of (21) 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., $\sum=F^{-1}$, given by the expectation of the negative of second partial derivative of L as

$$F = \begin{bmatrix} E(-\frac{\delta^2 L}{\delta a^2}) & E(-\frac{\delta^2 L}{\delta a \delta r}) \\ E(-\frac{\delta^2 L}{\delta r \delta a}) & E(-\frac{\delta^2 L}{\delta r^2}) \end{bmatrix} = \begin{bmatrix} \frac{f_n}{a} & g_n \\ g_n & \frac{a \cdot \sum_{k=1}^{n} (g_k - g_{k-1})^2}{(f_k - f_{k-1})} \end{bmatrix} \quad (22)$$

Where $g_k = W(t) \cdot e^{-rW(t_k)}$ and

$f_k = 1 - e^{-rW(t_k)}$ for $K = 1, 2, ..., n$.

After substituting the values of the estimates of a and r in (22) one can estimate F^{-1}. The estimated asymptotic variance-covariance matrix is

$$\hat{\Sigma} = F^{-1} = \begin{bmatrix} Var(\hat{a}) & Cov(\hat{a}, \hat{r}) \\ Cov(\hat{r}, \hat{a}) & Var(\hat{r}) \end{bmatrix} \quad (23)$$

6. PERFORMANCE ANALYSIS

In order to validate the proposed model and to compare its performances with other existing models, experiments on actual software failure data have been performed.

6.1 Criteria for Model Comparison

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 [19], [30], [5], [16] as AE $= \left| \frac{M_a - a}{M_a} \right|$, 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 [17], [5], [16] as

MSE = $\frac{1}{k}\sum_{i=1}^{k}[m(t_i) - m_1]^2$, where $m(t_i)$ is the expected is the expected number of errors at time t_i estimated by a model, and $m(t_i)$ 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 [9], [12], [14].

3. The Coefficient of Multiple Determination is defined [19], [18] as

$$R^2 = \frac{S\ \hat{\hat{\alpha}}, 0, 1, 1\ - S\ \hat{\alpha}, \hat{\beta}, \hat{m}, \hat{\delta}}{S\ \hat{\hat{\alpha}}, 0, 1, 1},$$

Where $\hat{\hat{\alpha}}$ is the LSE of α for the model with only a constant term, that is, $\beta = 0$, $m = 1$ and $\delta = 1$ in (12). It is given by ln $\hat{\hat{\alpha}} = \frac{1}{n}\sum_{k=1}^{n} \ln W_k$. 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 [15]. To investigate whether a significant trend exists in the estimated testing-effort, one could test the hypothesizes H_0: $\beta = 0, m = 1$ and $\delta = 1$, against $H_1 : \beta \neq 0$ or at least m or $\delta \neq 0$ using F-test by merely forming the ratio

$$F = \frac{\left[S\ \hat{\hat{\alpha}}, 0, 1, 1\ - S\ \hat{\alpha}, \hat{\beta}, \hat{m}, \hat{\delta} \right] / 3}{S\ \hat{\hat{\alpha}}, 0, 1, 1\ /(n-4)}$$

If the value of F is greater that $F_\alpha(3, n-4)$, which is the α percentile of the F distribution with degrees of freedom 3 and n-4, 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 [19], [18] 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_e. 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_e / t_q.

6.2 Application Examples

DS 1: The first set of actual data is from the study by Ohba [20]. 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 $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$ of the NMW current testing-effort function; we fit the actual testing effort data into (2) 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:

$\hat{\alpha}$ =64.4667, $\hat{\beta}$ = 0.0341, $\hat{m}$ = 0.8851, $\hat{\delta}$ = 0.0561 (24)

Figure 1 graphically illustrates the comparisons between the observed failure data and the estimated NMW testing-effort data. Here, the fitted curve is shown as a doted line and the solid line represents actual software data.

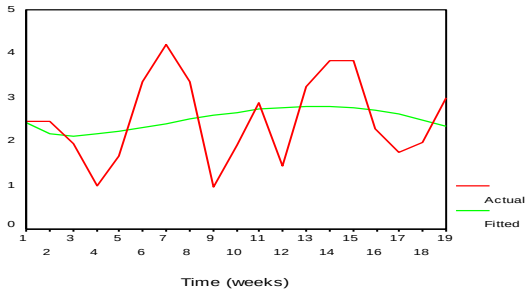

Figure 1: Observed/estimated current test-effort vs time

Using the estimated parameters $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$ the other parameters a, r in (6) can be solved numerically by the MLE method. These estimated parameters are $\hat{a}$ = 566.6613, $\hat{r}$ = 0.0195961

Table 1: Summary of estimates of NHPP model parameters

Parameter	Estimate	Standard Error	95% Confidence Interval	
			Lower	Upper
a	566.6614	61.75	433.29	700.02
r	0.019596	0.0030	0.0130	0.026

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

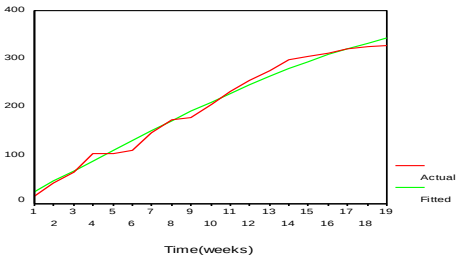

Figure 2: Observed/estimated cumulative number of failures

Table 2: Comparative results of different SRGM

Model	a	r	AE (%)	MSE
Proposed Model (Eqn. (6))	566.66	0.0196	58.28	103.1
Yamada exponential model (Eqn. (5) with exponential curve)	828.25	0.0118	131.4	140.7
Yamada Rayleigh model (Eqn. (5) with Weibull curve)	565.35	0.0197	57.91	122.1
Huang Logistic model	394.08	0.0427	10.06	118.6
Ohba exponential mode	455.37	0.0267	27.09	206.9
Inflection S-shaped model	389.1	0.0935	8.69	133.3
Delayed S-shaped model	374.05	0.1977	4.48	168.7
G-O model	760.0	0.0323	112.3	139.8
Dlayed S-shaped model with Rayleigh	333.14	0.1004	6.93	798.5

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 NMW testing-effort is 0.9901. It can therefore be observed that the NMW 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 (=7.92) is greater than $F_{0.05}(3,15)$ and $F_{0.01}(3,15)$. 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. Following the work in [19], we compute the relative error in prediction for this data set and the results are plotted in Figure 3. We observed that relative error approaches zero as t_e approaches t_q and the error curve is usually within ± 5 percent. Altogether, from Figures 1-3 and Tables 1-2, we can see that the proposed model has better performance and predicts the future behavior well.

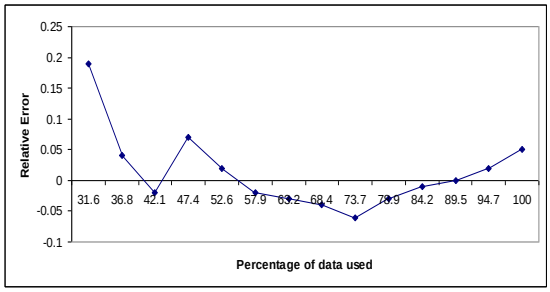

Figure 3: Relative error curve.

DS 2: The second set of actual data is the pattern of discovery of errors [27]. 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 $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$ of the NMW current testing-effort function described in (2) by using the method of least squares. The estimated values of parameters are

$\hat{\alpha}$ =97.1342, $\hat{\beta}$ = 0.0128, $\hat{m}$ = 1.1297, $\hat{\delta}$ = 0.8769 (25)

Figure 4 show the fitting of the estimated testing-effort by using above estimates. The fitted curve and the actual software data are shown by dotted and solid lines, respectively.

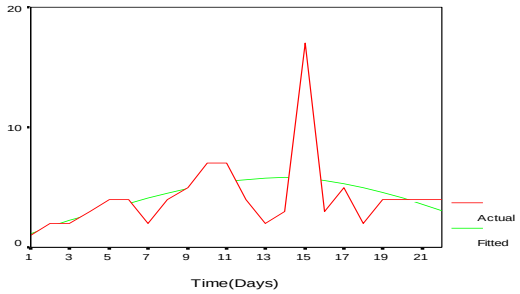

Figure 4: Observed/estimated current test-effort vs time

The other parameters a, r in (6) can be solved numerically using MLE method for these failure data. The estimators are

$\hat{a}$ =94.88667, $\hat{r}$ =0.02524413.

Table 3: Summary of estimates of NHPP model parameters

Parameter	Estimate	Standard Error	95% Confidence Interval	
			Lower	Upper
a	94.8867	2.561	69.45	100.314
r	0.02524	0.0015	0.0219	0.0285

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 5.

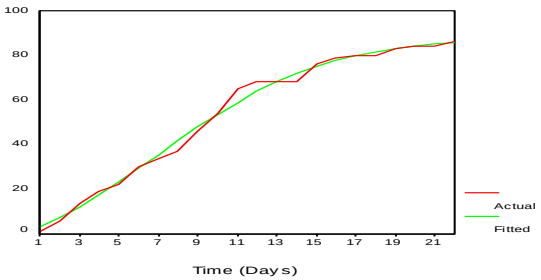

Figure 5: Observed/estimated cumulative number of failures Vs time

The R^2 Value for proposed NMW testing-effort is 0.9931. Therefore, we can say that the proposed curve is suitable for modeling the software reliability. Also, the calculated value F (=8.94) is greater than $F_{0.05}$ (3, 18) and $F_{0.01}$ (3, 18), which concludes that the fitted testing-effort curve is highly significant for this data set.

Table 4: Comparative results of different SRGM

Model	a	r	MSE
Proposed Model (Eqn. (6))	94.887	0.0252	5.55
Generalized exponential model (Eqn. (5) with Generalized exponential curve)	94.88	0.025	7.557
Yamada Rayleigh model (Eqn. (5) with Rayleigh curve)	87.032	0.0345	7.772
Delayed S-shaped model	88.653	0.2282	6.313
Huang Logistic model	88.893	0.0391	25.23
G-O model	137.07	0.0515	25.33
HGDM	88.30	--	33.68

Table 4 lists the comparisons of proposed model with different SRGM 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 6 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 ± 5 percent. Therefore, from the Figures4-6 and Tables 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.

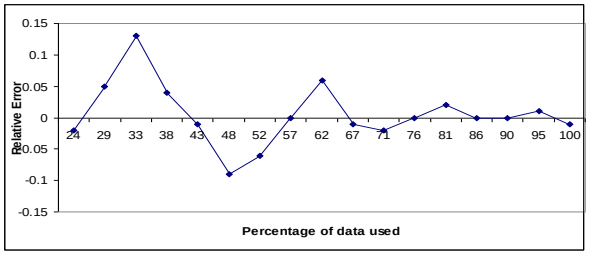

Figure 6: Relative error curve

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 [19], [18]. 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 [6]. Similarly, parameters $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$ of the NMW current testing-effort function for this data set can be obtained by using the method of LSE. The estimated values are

$\hat{\alpha}$ =25.5879, $\hat{\beta}$ =0.000098, $\hat{m}$ =1.1401, $\hat{\delta}$ = 0.3389 (26)

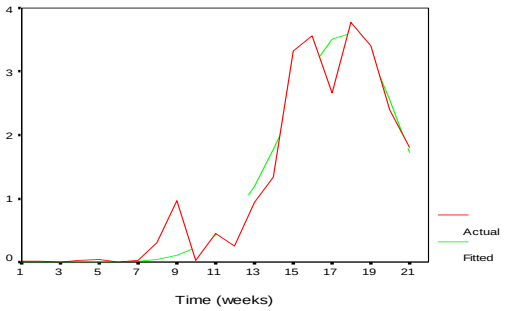

Figure 7: Observed/estimated current test-effort Vs time

Figure 7 shows the fitting of the estimated testing-effort by using these estimates. The fitted curve is shown as a dotted line and the solid line is for actual software data in the graphs. Using the estimated parameters $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$, the other parameters a, r in (6) can be solved numerically by MLE method. The estimates are $\hat{a}$ = 134.61, $\hat{r}$ =0.151572.

Table 5: Summary of estimate of NHPP model parameters

Parameter	Estimate	Standard Error	95% Confidence Interval	
			Lower	Upper
a	134.609	5.121	69	145.5
r	0.15157	0.01685	0.456	0.187

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

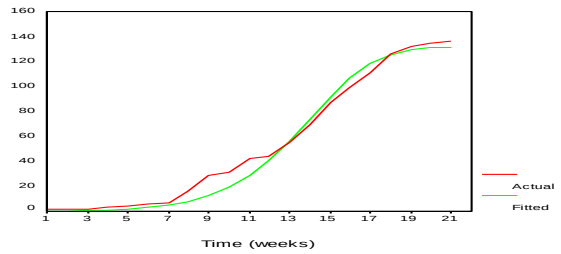

Figure 8: Observed/estimated cumulative number of failures Vs time

A fitted curve of the estimated mean value function with the actual software data is plotted in figure 8.

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 for this data set. Figure 9 depicts the relative error plotted against the percentage of data used (that is, t_e / t_q).

Finally, Figures 1-9 and Tables 1-6 reveal that the proposed model has better performance than the other models. This model fits the observed data better, and predicts the future behavior well.

Table 6: Comparative results of different SRGM

Model	a	r	AE (%)	MSE
Proposed Model (Eqn. (6))	134.61	0.1516	28.4	27.15
Yamada Rayleigh model (Eqn. (5) with Rayleigh curve)	866.94	0.0096	25.1	89.24
Delayed S-shaped model	237.20	0.0963	26.16	245.25
Huang Logistic model	138.03	0.1451	26.58	62.41
G-O model	142.32	0.1246	24.29	2438.3
Inflection S-shaped model	159.11	0.0765	15.36	118.3
Ohba exponential model	137.20	0.156	27.12	3019.6

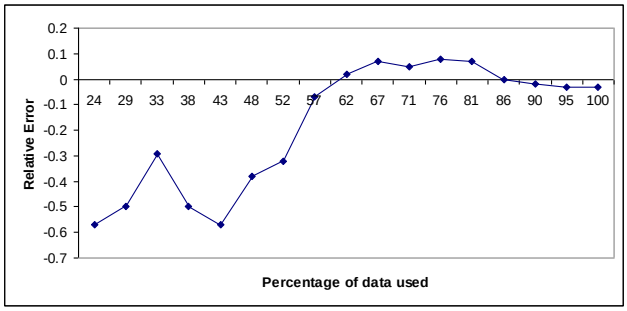

Figure 9: Relative error curve

The R^2 value for proposed NMW testing-effort curve is 0.9823 and calculated F value is 8.84, which is greater than $F_{0.05}$ (3,117) and $F_{0.01}$ (3, 17). 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 for this data set. Figure 9 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, Figure7-9 and Tables 5-6 reveal that the proposed model has better performance than the other models. This model fits the observed data better, and predicts the future behavior well.

7. 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 e 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 [28], [30], [21], [14], [11].

7.1 S/w Release- Time Based on Reliability Criterion

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 [93]. 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 $R(\Delta t \mid t)$ is a monotonic increasing function of t. Taking the logarithm on both sides of (10), we obtain

$$\ln R = -[m(t+\Delta t) - m(t)] \qquad (27)$$

We can easily determine the testing time needed to reach a desired R by solving (27) and (6). It is noted that $R(t)$ is increasing in t.

7.2 Reliability Analysis For Real Data Set

DS 1: In first data set, it is known that $\hat{\alpha}$ =64.466695, $\hat{\beta}$ = 0.034101264, $\hat{m}$ = 0.885114496, $\hat{\delta}$ = 0.0561145, $\hat{a}$ = 566.6613, and $\hat{r}$ = 0.0195961. Suppose the software system desires that the testing would be continued till the operational reliability is equal to 0.80 (at Δt = 0.1), from (27) and (6), we get t = 28.57 weeks. If the desired reliability is 0.85, then t = 29.90 weeks. If the desired reliability is 0.95, then t = 33.74 weeks and if the desired reliability is 0.99, then t = 37.65 weeks.

DS 2: In second data set, from (24) and (6), for $\hat{\alpha}$ =97.13424, $\hat{\beta}$ = 0.0128194, $\hat{m}$ =1.1296671, $\hat{\delta}$ =0.087694, $\hat{a}$ = 94.88667, and $\hat{r}$ = 0.02524413, we get testing time t = 20.66 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 Δt = 0.1). If the desired reliability is 0.99, then t = 21.80 days.

DS 3: From the previous estimated parameters: $\hat{\alpha}$ =25.5878, $\hat{\beta}$ = 0.000099, $\hat{m}$ = 1.14012, $\hat{\delta}$ =0.3389, $\hat{a}$ = 134.61, and $\hat{r}$ = 0.1516, suppose the software system desires that the testing would be continued till the operational reliability is equal to 0.8 (at Δt = 0.1), from (27) and (6), we get testing time = 19.10 weeks. Similarly, the desired reliability is 0.99, then $_t$ = 21.80 weeks.

7.3 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 [34], [33], [31], [21], [14], [10], [11], [5], [9], [7].

$$C(T) = C_1 m(T) + C_2[m(T_{lc}) - m(T)] + C_3 \int_0^T w(x)dx. \quad (28)$$

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 life-cycle 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:

$$\begin{cases} Minimize\ C(T) \\ Subject\ to\ R(t+\Delta t \mid t) \geq R_0 \\ for\ C_2 > C_1 > 0, C_3 > 0 \\ \Delta t > 0, 0 < R_0 < 1. \end{cases} \quad (29)$$

The procedures to derive the optimal release policy for this problem are evolved step by step and are shown hereafter. By differentiating (28) with respect to T and equating to zero, yields

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

$$\frac{C_3}{C_2 - C_1} = \frac{\lambda(T)}{w(T)} = a \cdot r \cdot e^{-r \cdot W(T)} = r \cdot (a - m(T)) \quad (30)$$

When $T = 0$ then $m(0) = 0$ and $\frac{\lambda(T)}{w(T)} = ar$. When $T \to \infty$, then $m(\infty) = a(1 - e^{-r\alpha})$ and $\frac{\lambda(T)}{w(T)} = a.r.e^{-r\alpha}$. Therefore, $\frac{\lambda(T)}{w(T)}$ is monotonically decreasing in T. To analyze for the minimum value of $C(T)$, (30) is used to explore two cases of $\frac{\lambda(T)}{w(T)}$ at $T = 0$.

Case 1: If $\frac{\lambda(0)}{w(0)} = a \cdot r \leq \frac{C_3}{C_2 - C_1}$, then.

$\frac{\lambda(\lambda)}{w(T)} \leq \frac{C_3}{C_2 - C_1}$ for $0 < T < T_{lc}$

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

Case 2: If, there can be found a finite T such that

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

$$= a \cdot r \cdot e^{-r \cdot W(T)} = a \cdot r \cdot e^{r \cdot a \cdot (1 - e^{-\beta \cdot T} \cdot e^{\lambda \cdot T})}$$

Solving this, we can get

$$T^m \cdot e^{\lambda \cdot t} = \frac{-1}{\beta} \cdot \ln[1 - \frac{1}{\lambda \cdot \alpha} \cdot \ln\{\frac{a \cdot r \cdot (c_2 - c_1)}{c_3}\}] \quad (31)$$

The equation (31) is solved using numerical technique and get the value of T_0 satisfying (33), $\frac{dC(T)}{dT} < 0$ 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 $C(T)$ c 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 $R(t + \Delta t \mid t) = R_0$ 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 $C_1 > 0, C_2 > 0, C_3 > 0, C_2 > C_1, x > 0, 0 < R_0 < 1$, then

1. *If* $\frac{\lambda(0)}{w(0)} \leq \frac{C_3}{C_2 - C_1}$ *and*

$\frac{\lambda(T)}{w(T)} = \alpha \cdot r \cdot e^{-r \cdot \alpha} < \frac{C_3}{C_2 - C_1}$ *for* $0 < T < T_{lc}$

then $T^* = \max[T_0, T_1]$ for $R(x \mid 0) < R_0 < 1$ or

$T^* = T_0$ *for* $0 < R_0 < R(x \mid t = 0)$.

2. *If* $\frac{\lambda(0)}{w(0)} \leq \frac{C_3}{C_2 - C_1}$ then,

$T^* = T_1$ *for* $R(x \mid 0) < R_0 < 1$ or

$T^* = 0$ *for* $0 < R_0 < R(x \mid 0)$

3. *If* $\frac{\lambda(0)}{w(0)} \geq \frac{C_3}{C_2 - C_1}$ *then* $T^* \geq T_1$

for $R(x \mid 0) < R_0 < 1$ or $T^* \geq 0$ *for* $0 < R_0 \leq R(x \mid 0))$.

7.4 Application Examples

DS 1: In fist data set, it is known that $\hat{\alpha}$ =64.466695, $\hat{\beta}$ = 0.034101264, $\hat{m}$ = 0.885114496, $\hat{\delta}$ = 0.0561145, $\hat{a}$ = 566.6613, and $\hat{r}$ = 0.0195961. To determine the optimal

software release time, we assume the values of $C_1 = 1, C_2 = 50$, $C_3 = 100,$ $T_{LC} = 100,$ $R_0 = 0.90$ and $\Delta t = 0.1$ for the analysis. Then we get the optimal release time T_0 estimated as 2.96 based on minimizing $C(T)$ of (31), and T_1 is estimated as 29.90 based on satisfying the reliability criterion of $R(t + \Delta t \mid t) = R_0$. These values sustain the relationships of $\frac{\lambda(0)}{w(0)} > \frac{C_3}{C_2 - C_1}$ and $\frac{\lambda(T)}{w(T)} = \alpha.r..e^{-r\alpha} < \frac{C_3}{C_2 - C_1}$ and $R(\Delta t \mid 0) < R_0$ with which one could imply 1 in Theorem 1 to obtain the optimal software release time T^* as max (29.90, 2.96) = 29.90 weeks and the corresponding software cost $C(T^*)$ is 6852.14

DS 2: In second data set, it is known that $\hat{\alpha}$ =97.13424, $\hat{\beta}$ = 0.0128194, $\hat{m}$ =1.1296671, $\hat{\delta}$ =0.087694, $\hat{a}$ = 94.88667, and $\hat{r}$ = 0.02524413. To determine the optimal software release time, we assume the values of $C_1 = 1, C_2 = 50$ $C_3 = 100, T_{LC} = 100, R_0 = 0.90$ and $\Delta t = 0.1$ for the analysis. Then we get the optimal release time T_0 estimated as 2.56 based on minimizing $C(T)$ of (31), and T_1 is estimated as 17.78 based on satisfying the reliability criterion of $R(t + \Delta t \mid t) = R_0$. These values sustain the relationships of $\frac{\lambda(0)}{w(0)} > \frac{C_3}{C_2 - C_1}$ and $\frac{\lambda(T)}{w(T)} = \alpha.r..e^{-r\alpha} < \frac{C_3}{C_2 - C_1}$ and $R(\Delta t \mid 0) < R_0$ with which one could imply 1 in Theorem 1 to obtain the optimal software release time T^* as max (17.78, 2.56) = 17.78 days and the corresponding software cost $C(T^*)$ is 7792.85

DS 3: In third data set, it is known that $\hat{\alpha}$ =25.5879, $\hat{\beta}$ = 0.0000984, $\hat{m}$ =1.140123, $\hat{\delta}$ = 0.3389, $\hat{a}$ = 134.61, and $\hat{r}$ = 0.15157. To determine the optimal software release time, we assume the values of $C_1 = 1, C_2 = 50$ $C_3 = 100, T_{LC} = 100, R_0 = 0.90$ and $\Delta t = 0.1$ for the analysis. Then we get the optimal release time T_0 estimated as 13.73 based on minimizing $C(T)$ of (31), and T_1 is estimated as 19.49 based on satisfying the reliability criterion of $R(t + \Delta t \mid t) = R_0$. These values sustain the relationships of $\frac{\lambda(0)}{w(0)} > \frac{C_3}{C_2 - C_1}$ and $\frac{\lambda(T)}{w(T)} = \alpha.r..e^{-r\alpha} < \frac{C_3}{C_2 - C_1}$ and $R(\Delta t \mid 0) < R_0$ with which one could imply 1 in Theorem 1 to obtain the optimal software release time T^* as max (19.49, 13.73) = 19.49 weeks and the corresponding software cost $C(T^*)$ is 2393.34

8. REFERENCES

[1] Ahmad, N., Khan, M. G. M., Quadri, S. M. K. and Kumar, M., "Modeling and Analysis of Software Reliability with Burr Type X Testing-Effort and Release-Time Determination", Journal of Modeling in Management, Vol. 4 (1), 28 – 54, 2009.

[2] Ahmad, N., Bokhari, M. U., Quadri, S. M. K. and Khan, M. G. M. (2008), "The Exponetiated Weibull Software Reliability Growth Model with various testing-efforts and optimal release policy: a performance analysis", International Journal of Quality and Reliability Management, Vol. 25 (2), pp. 211-235.

[3] Goel, A.L. and Okumoto, K., "Time dependent error-detection rate model for software reliability and other performance measures", IEEE Transactions on Reliability, Vol. R- 28, No. 3, pp. 206-211, 1979.

[4] Huang, C. Y. "Cost-reliability-optimal-release policy for software reliability models incorporating improvements in testing efficiency", Journal of Systems and Software 77(2), pp. 139-155, 2005b.

[5] Huang, C.Y. and Kuo, S. Y. ", Analysis of incorporating logistic testing-effort function into software reliability modeling", IEEE Transactions on Reliability, Vol. 51, no. 3, pp. 261-270, 2002.

[6] Huang, C. Y. "Performance analysis of software reliability growth models with testing-effort and change-point", Journal of Systems and Software, Vol. 76, pp. 181-194, 2005.

[7] Huang, C. Y., Kuo, S.Y. and Lyu, M. R. , "Optimal software release policy based on cost, reliability and testing efficiency", Proceedings of the 23rd IEEE Annual International Computer Software and Applications Conference (COMPSAC'99), Phoenix, Arizona, pp. 468-473, 1999.

[8] Huang, C. Y., Kuo, S.Y. and Lyu, M. R. , "Effort-index based software reliability growth models and performance assessment", Proceedings of the 24th IEEE Annual International Computer Software and Applications Conference (COMPSAC'2000), pp. 454-459, 2000.

[9] Huang, C.Y., Kuo, S.Y. and Chen, I.Y., "Analysis of software reliability growth model with logistic testing-effort function", Proceeding of 8th International Symposium on Software Reliability Engineering (ISSRE'1997), Albuquerque, New Maxico, pp. 378-388, 1997.

[10] Kapur, P. K. and Bhalla, V. K., "Optimal Release Policy for Flexible Software Reliability Growth Model", Engineering and System Safety, Vol. 35, pp. 49-54, 1992.

[11] Kapur, P.K. and Garg, R.B., "Cost reliability optimum release policies for a software system with testing effort", Operations Research, Vol. 27, no. 2, pp. 109-116, 1990.

[12] Kapur, P.K. and Garg, R.B. "Modeling an imperfect debugging phenomenon in software reliability", Microelectronics and Reliability, Vol. 36, pp. 645-650, 1996.

[13] Kapur, P.K. and Younes, S., "Modeling an imperfect debugging phenomenon with testing effort", Proceedings of 5th International Symposium on Software Reliability Engineering (ISSRE'1994), pp. 178-183, 1994.

[14] Kapur, P.K., Garg, R.B. and Kumar, S., Contributions to Hardware and Software Reliability, World Scientific, Singapore, 1999.

[15] Kumar, M., Ahmad, N. and Quadri, S.M.K., "Software reliability growth models and data analysis with a Pareto test-effort", RAU Journal of Research, Vol., 15 (1-2), pp. 124-128, 2005.

[16] Kuo, S.Y., Hung, C.Y. and Lyu, M.R., "Framework for modeling software reliability, using various testing-efforts

and fault detection rates", IEEE Transactions on Reliability, Vol. 50, no.3, pp 310-320, 2001.

[17] Lyu, M.R., Handbook of Software Reliability Engineering, McGraw- Hill, 1996.

[18] Musa J. D, Software Reliability Engineering: More Reliable Software, Faster Development and Testing, McGraw-Hill, 1999.

[19] Musa, J.D., Iannino, A. and Okumoto, K., Software Reliability: Measurement, Prediction and Application, McGraw-Hill, 1987.

[20] Ohba, M., "Software reliability analysis model" IBM Journal. Research Development, Vol. 28, no. 4, pp. 428-443, 1984.

[21] Okumoto, K. and Goel, A. L., "Optimum release time for software system based on reliability and cost criteria", Journal of Systems and Software, Vol.1, pp. 315-318, 1980.

[22] Pham, H. (2000), Software Reliability, Springer-Verlag, New York, 2000.

[23] Putnam, L., "A general empirical solution to the macro software sizing and estimating problem", IEEE Transactions on Software Engineering, Vol. Se-4, pp. 343-361, 1978.

[24] Quadri, S. M. K., Ahmad, N., Peer, M.A. and Kumar, M., "Non homogeneous Poisson process Software Reliability Growth Model with generalized exponential testing effort function, "RAU Journal of Research, Vol., 16 (1-2), pp. 159-163, 2006.

[25] Tang, Y. et. al. "Statistical Analysis of a Weibull Extension Model", Communications in Statistics, Theory and Analysis, pp. 911-916, 2003.

[26] Tian, J., Lu, P., and Palma, J., "Test-Execution-Based Reliability Measurement and Modeling for Large Commercial Software", IEEE Transaction Software Engineering, Vol. 21, No. 5, pp. 405-414, 1995.

[27] Tohma, Y., Jacoby, R., Murata, Y. and Yamamoto, M., "Hyper-geometric distribution model to estimate the number of residual software fault", Proceeding of COMPSAC-89, Orlando, pp. 610-617, 1989.

[28] Xie, M., "On the determination of optimum software release time". In Proceeding 2nd International Symposium on software reliability engineering, pp. 218-224, 1991a.

[29] Yamada, S. and Ohtera, H., "Software reliability growth models for testing effort control", European Journal of Operational Research, Vol. 46, no. 3, pp. 343-349. 1990.

[30] Yamada, S. and Osaki, S., "Cost-reliability optimal release policies for software systems", IEEE Transaction on Reliability, Vol. R-34, no. 5, pp. 422-424, 1985b.

[31] Yamada, S., and Osaki, S., "Software reliability growth modeling: models and applications", IEEE Transaction on Software Engineering, Vol. SE-11, no. 12, pp. 1431-1437, 1985a.

[32] Yamada, S., Hishitani J. and Osaki, S., "Test-effort dependent software reliability measurement, " International Journal of Systems Science, Vol. 22, no. 1, pp. 73-83, 1991.

[33] Yamada, S., Hishitani J. and Osaki, S., "Software reliability growth model with Weibull testing-effort: a model and application", IEEE Transactions on Reliability, Vol. R-42, pp. 100-105, 1993.

[34] Yamada, S., Narihisa, H. and Osaki, S., "Optimum release policies for a software system with a scheduled software delivery time", International Journal of System Science, Vol.15, pp. 905-914, 1984.

[35] Yamada, S., Ohtera, H. and Narihisa, H., "A testing-effort dependent software reliability model and its application", Microelectronics and Reliability, Vol. 27, no. 3, pp. 507-522, 1987.

[36] Yamada, S., Ohtera, H. and Norihisa, H. , "Software reliability growth model with testing-effort", IEEE Transactions on Reliability, Vol. R-35, no. 1, pp.19-23, 1986.