SRGM with logistic-exponential Testing-effort function with change-point and Analysis of Optimal release policies based on increasing the test efficiency.

Shaik. Mohammad Rafi[1] Dr.K.Nageswara Rao[2]

1. Assoc. Professor Sri Mittapalli College Institute of Technology for Women, NH-5, Tummalapalem, Guntur, A.P, India.

2. Professor& H.O.D in P.V.P.S.I.T, Vijayawada affiliated to J.N.T.U, Kakinada, India.

Abstract:-Reliability is the one of the important factor of software quality. Past few decades several software reliability growth models are proposed to access the quality of the software. Main challenging task of reliability growth model is predicting the reliability, total cost at optimal time at, software released into the market. It has been observed that most of the reliability growth models predict the failure rate to be constant during the software testing, but in reality software failure rate changes with testing time. In this paper we have investigated software reliability growth model by incorporating the both change point and testing effort. We incorporated logistic-exponential TEF in software reliability growth model with change-point. We also investigated the how testing efficiency can be increased by adopting the new automated testing tools into the software testing and its effect on the total cost of the software. Experiments are done on real datasets. Parameters are estimated. Results show the better fit.

Keywords: Software reliability; Software testing; Non-Homogeneous Poisson Process (NHPP); Change-point; Testing-effort

1. Introduction

Software has been ruling this world from past few decades. Today we require a more sophisticated and complex software in our computer systems. Generally software has been developed and maintained by humans for that, there is chance that errors might be propagated into the software. So we require a high technology for developing reliable software. Software reliability is the probability that software will provide failure-free operation in a fixed environment for a fixed interval of time [15, 18].Future failure conditions of the software can be estimated from the past failure conditions which is available. Several papers are published in this context. Like Musa, Xie, Pham and Singapurvalla and Wilson among many others. The software reliability growth models (SRGM) are designed to make the predictions. These predictions include failure rate and to reach the required reliability target. A very important class of (NHPP) non-homogeneous Poisson process models like Goel and Okumato, Ohba, Yamada, Yamada and Osaki. All these models had appropriate failure intensity function. Once the failure intensity function is defined we can estimated the quantities like number of failures remains in the software , number of initial faults and reliability level in a given time period.

Most SRGMs use calendar time as the unit of fault detection/removal period. Very few SRGMs use the human power, number of test case runs, or CPU time as the unit [8][18][28-34]. Recently, we proposed a new SRGM that incorporates the concept of logistic developer of the software and an independent test group (ITG) (Pressman, 2001). In the vast literature, most researchers assume a constant detection rate per fault in deriving their SRGMs. That is, they assume that all faults have equal probability of being detected during the software testing process, and the rate remains constant over the intervals between fault occurrences. In fact, a successful test is one that uncovers an as-yet-undiscovered error. It is impossible to execute every combination of paths during testing. Moreover, Pareto principle implies that 80% of all errors uncovered during testing will likely be traceable to 20% of all program components (Pressman, 2001). In practice, the fault detection rate strongly depends on the skill of test teams, program size, and software testability. Thus, it may not be smooth and could be changed [16].

On the other hand, if we want to detect more additional faults, it is advisable to purchase new equipments or introduce new tools/techniques, which are use. These external new methods can give a detailed description of the test methodology, a complete test report, or an expert analysis of the findings to the clients. If the software techniques/tools can be considered in software cost model and viewed as the investment required improving the long-term competitiveness. In this paper, we will review a SRGM with logistic-exponential TEF. Furthermore, we propose a methodology to incorporate both logistic-exponential TEF and change-point (CP) into software reliability growth modeling. Change-point problems have been studied by many authors [1, 2, 6, 22, 35, 37].

In the remaining of this paper, there are four more sections. In Section 2, we give a brief review of the

SRGM with a logistic-exponential TEF. Furthermore, we will investigate how to incorporate both logistic-exponential TEF and change-point into software reliability modeling in Section 3. We estimate these parameters of the proposed SRGM based on the actual observed software failure data, plot the mean value functions, and give a detailed comparison with other existing well- known SRGMs in Section 4. Finally, conclusions are presented in Section 5.

2. Software reliability modeling and testing-effort function

In this section, an NHPP model with TEF is present. The following assumptions are made for software reliability modeling [7, 8, 9, 12, 14, 29, 31, 32]

(i) The fault removal process follows the Non-Homogeneous Poisson process (NHPP)

(ii) The software system is subjected to failure at random time caused by faults remaining in the system.

(iii) The mean time number of faults detected in the time interval (t, t+Δt) by the current test effort is proportional for the mean number of remaining faults in the system.

(iv) The proportionality is constant over the time.

(v) Consumption curve of testing effort is modeled by Logistic-exponential TEF.

(vi) Each time a failure occurs, the fault that caused it is immediately removed and no new faults are introduced.

We can describe the mathematical expression of a testing-effort based on following for stochastic modeling of a software error detection phenomenon, we define a counting process [N(t), t>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 assumptions of Goel and Okumoto (1979) as:

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

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 are remaining in the system because software failure and the errors are detected and corrected by test personnel. Based on the assumptions if the numbers of detected errors by the current testing effort expenditure are proportional to the number of remaining errors, and then we obtain the following differential equations.

$$\frac{dm(t)}{dt} \times \frac{1}{w(t)} = r \times (a - m(t)) \quad (2)$$

Where m(t) represents the expected mean number of errors detected in time (0,t), w(t) current testing effort consumption at time t, a is the expected number of initial faults , and r is a fault detection rate per unit testing effort at testing time t. solving above equation under boundary conditions m(0)=0 and W(0)=0 we get the following equation

$$m(t) = a \times \left(1 - e^{-r \times [W(t) - W(0)]}\right) \quad (3)$$

The relation between current and cumulative testing-effort given by

$$W(t) = \int_0^t w(t)dt \quad (4)$$

The Eq.(3) represents the MVF incorporated with testing-effort. Generally testing-effort describes the how effectively the faults are detected and can be modeled by different expenditure curves.

Recently we proposed a SRGM with logistic-exponential testing-effort function [22]. The cumulative testing effort consumption is [38]

$$W(t) = \alpha \times \frac{(e^{\lambda t} - 1)^k}{(1 + (e^{\lambda t} - 1)^k)} \quad (5)$$

Current testing effort is

$$w(t) = \frac{\alpha \times k \times \lambda \times e^{\lambda \times t} \times \left(e^{\lambda \times t} - 1\right)^{k-1}}{\left(1 + \left(e^{\lambda \times t} - 1\right)^k\right)^2} \quad t>0 \quad (6)$$

α is the total expenditure , λ is the effort consumption rate and k is the structuring index. The intensity function at a time t is

$$\lambda(t) = \frac{dm(t)}{dt} \quad (7)$$

3.1 SRGM with Logistic-exponential TEF and change-point

During a software testing process, the nature of the failure data can be affected by many factors such as the testing environment, test strategy, resource allocation and so on. The factors are unlikely to all be kept stable during the whole process of software testing. The detection rate may not be smooth and can be changing when the testing environment and resource allocation is changed. The testing effort can be described by amount CPU hours, man power and number test cases. During the software development process the fault detection rate may not be constant; it may change after some time moment called change point [6, 22]. Here we will incorporate the logistic-exponential testing effort and change point into the SRGM. SRGM based on testing-effort and change point is given by

$$\frac{dm_1(t)}{dt} \times \frac{1}{w(t)} = r_1 \times (a - m_1(t)) \qquad (8)$$

$$r(t) = \begin{cases} r_1 & 0 \leq t \leq \tau \\ r_2 & t > \tau \end{cases} \qquad (9)$$

$$m_1(t) = a \times \left(1 - e^{-r1 \times [W(\tau) - W(0)]}\right) \quad 0 < t < \tau \qquad (10)$$

$$m_2(t) = a \times (1 - e^{(-[r_1 \times (W(\tau) - W(0)) + r_2 \times (W(t) - W(\tau))])}) \text{ , } t > \tau \qquad (11)$$

Complete solutions for Eq.(9) are present in Appendix A

3.2 New technique for increasing the software testing efficiency

Any testing strategy must incorporate test planning, test cases design, test execution, and resultant data collection and evaluation. The increasing visibility of software as a system element and the attendant costs associated with a software failure are motivating forces for well planed through testing. It is not usual for software development organization to expend between 30 to 40 percent of total project effort on testing [Pressman 2001]. Once the all faults are removed the software is deploy to the customer. A software engineer needs more rigorous criteria for determining when sufficient testing has been conducted. Testing has been conducted in properly, by adopting the new testing tools we can speed up the testing, although it increases the extra cost. Here we study the change point problem in a different angle. When a change point occurs there the developer will adopt a new automated testing tool which speeds up the process even though it effects the cost. A gain parameter is proposed by Huang 2006[6, 9] which is defined fraction of additional faults found by using the automated testing tool. In this assumption he proposed that fraction of fault detection is constant. But it is observed that no testing tool is efficient all time, so fraction of faults detected might not be constant.

The modified mean value is depicted as [6, 9]

$$m_e(t) = a \times (1 - e^{-\sigma \times r \times W^*(t)}) \qquad (12)$$

Where t > τ and σ(t) is the gain parameter(GP). Therefore from Eq(11) and Eq (12), we have

$$\sigma = \frac{r1\ [W(\tau) - W(0)] + r2\ [W(t) - W(\tau)]}{r1\ [W(t) - W(0)]} \qquad (13)$$

Also from Eq (3) , Eq (11) , and (12) we can also re-define the gain effect of employing new automated techniques /tools and depicted it as follows.

$$a \times \frac{\left(1 - e^{-(r1 \times [W(\tau) - W(0)] + r2 \times [W(t) - W(\tau)])}\right)}{a \times \left(1 - e^{-r1 \times [W(t) - W(0)]}\right)} = 1 + P \qquad (14)$$

Hence we can conclude that

$$m_1(t) = a \times \left(1 - e^{-(r1 \times [W(\tau) - W(0)] + r2 \times [W(t) - W(\tau)])}\right) \qquad (15)$$

$$= a \times (1 - e^{-\sigma \times r \times W^*(t)}) \qquad (16)$$

$$= (1 + P) \times a \times (1 - e^{-\sigma(t) \times r \times W^*(t)}) \qquad (17)$$

$$= (1 + P) \times m(t) \qquad (18)$$

Where P is the additional of faults detected by using new automated tools or techniques during the testing [9, 10]. Depending on the characteristic of tool the value of P varies. The nature of the testing tool will characterize the value of P at that time.

4) EVALUATION CRITERIA

4.1) a) The goodness of fit technique

Here we used MSE [21,23]which gives real measure of the difference between actual and predicted values. The MSE defined as

$$MSE = \sum_{i=1}^{k} \frac{\left[m(t_i) - m_i\right]^2}{k} \qquad (19)$$

A smaller MSE indicate a smaller fitting error and better performance.

b) Coefficient of multiple determinations (R^2) which measures the percentage of total variation about mean accounted for the fitted model and tells us how well a curve fits the data. It is frequently employed to compare model and access which model provies the best fit to the data. The best model is that which proves higher R^2. that is closer to 1.

c) The predictive Validity Criterion

The capability of the model to predict failure behavior from present & past failure behavior is called predictive validity. This approach, which was proposed by [7], can be represented by computing RE for a data set

$$RE = \frac{(m(t_q) - q)}{q} \qquad (20)$$

In order to check the performance of the logistic-exponential software reliability growth model and make a comparison criteria for our evaluations.

SSE criteria: SSE can be calculated as: [21]

$$SSE = \sum_{i=1}^{n} \left[y_i - m(t_i) \right]^2 \tag{21}$$

Where y_i is total number of failures observed at a time t_i according to the actual data and $m(t_i)$ is the estimated cumulative number of failures at a time t_i for i=1,2,…..,n.

4.2) Model comparisons with real applications

DS1: the first set of actual data is from the study by Ohba 1984 [19].the system is PL/1 data base application software , consisting of approximately 1,317,000lines of code

.During nineteen weeks of experiments, 47.65 CPU hours were consumed and about 328 software errors are removed. Fitting the model to the actual data means by estimating the model parameter from actual failure data.

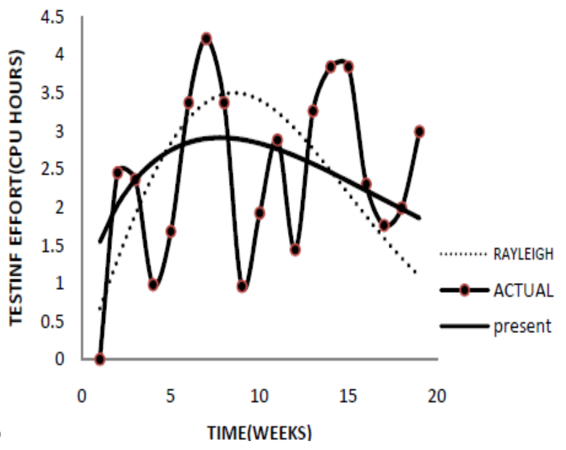
Here we used the LSE (non-linear least square estimation) and MLE to estimate the parameters. The unknown parameters of Logistic-exponential TEF are α=72(CPU hours), λ=0.04847, and k=1.387.Calculations are given in appendix A. from the table 2 it is observed that proposed model fits better than other models(Goel and Okumoto , Yamda Delayed S shaped model). In this we will take change-point is occurred at τ=6 and estimated values are given in the table. DS 2: the dataset used here presented by wood [25] from a subset of products for four separate software releases at Tandem Computer Company. Wood Reported that the specific products & releases are not identified and the test data has been suitably transformed in order to avoid Confidentiality issue. Here we use release 1 for illustrations. Over the course of 20 weeks, 10000 CPU hours are consumed and 100 software faults are removed. Similarly the least square estimates of the parameters for logistic-exponential TEF in the case of DS2 are α=12600(CPU hours), λ=0.06352, and k=1.391.

Table 1
Parameters of logistic-exponential
TEF for the dataset-1

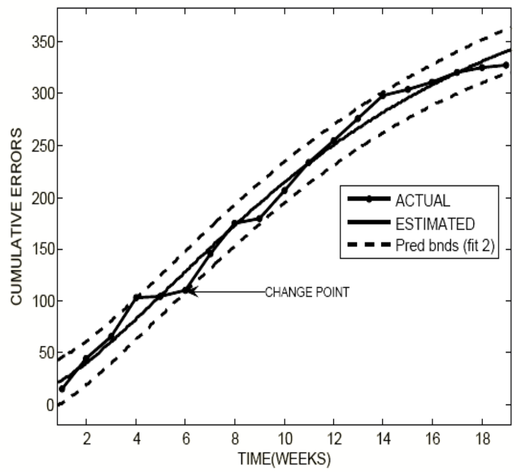
Model	α	λ	k
E.q (5)	72	0.04847	1.387

Table 3
Parameters of logistic-exponential
TEF for the dataset-2

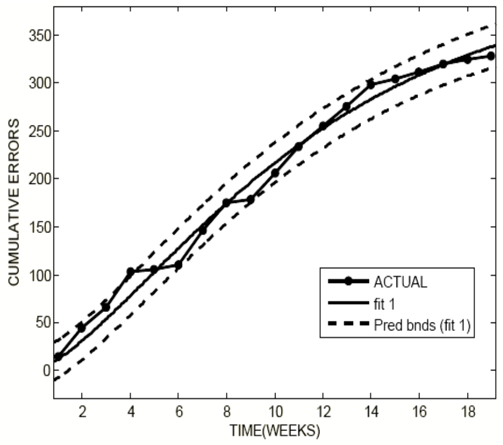
Model	α	λ	k
E.q (5)	12600	0.06352	1.391



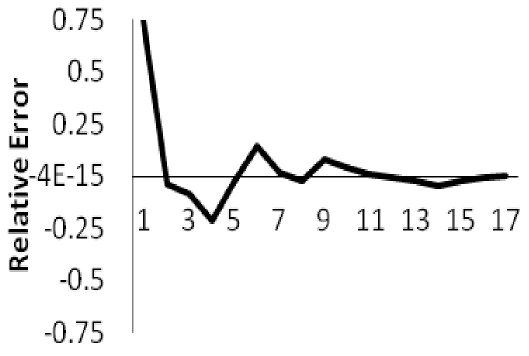
(a)



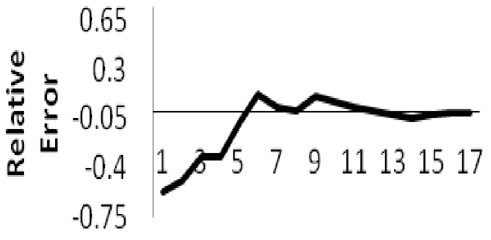
(c)



(b)



(d)



(e)

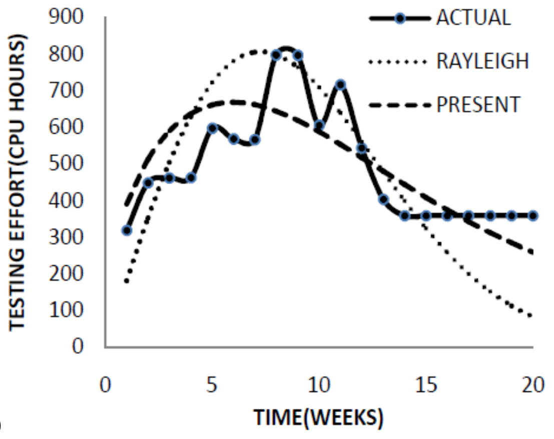
Fig 1.(a) Observed/estimated TE vs time for dataset 1 (b) mean value function for Eq 3 (c) Mean value function for Eq 10 and Eq 11 (d) RE curves for proposed model at τ=6 (e) RE curve for Yamada Delayed S shaped model

Table 2
Comparison results of different SRGMs for the first dataset

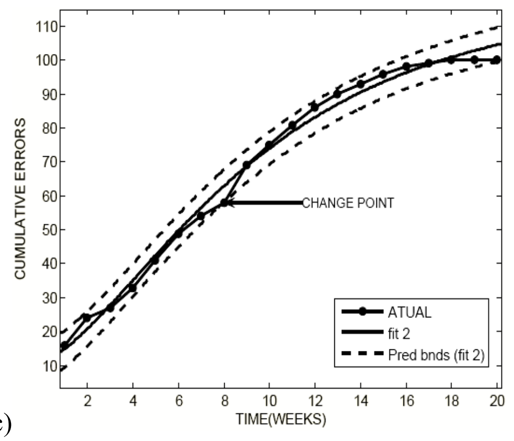
Model	a	r	τ	MSE	R^2	SSE
Proposed model Eq. (10)	578.8	0.01903	-------	128.36	0.9889	2183
Proposed model Eq (11)	703.9	$r_1 = 0.01642$	4	114.70	0.9906	1835
		$r_2 = 0.01397$				
Proposed model Eq (11)	703.9	$r_1 = 0.01578$	5	114.70	0.9906	1835
		$r_2 = 0.01397$				
Proposed model Eq (11)	703.6	$r_1 = 0.01539$	6	113.96	0.992	1833
		$r_2 = 0.01397$				
Proposed model Eq (11)	703.6	$r_1 = 0.01513$	7	113.96	0.992	1833
		$r_2 = 0.01397$				
Proposed model Eq (11)	703.6	$r_1 - 0.01495$	8	113.96	0.992	1833
		$r_2 - 0.01397$				
Yamada Delayed S shaped	374.1	0.1977	------	188.51	0.9837	3205

TABLE 4
Comparison results of different SRGMs for the second dataset

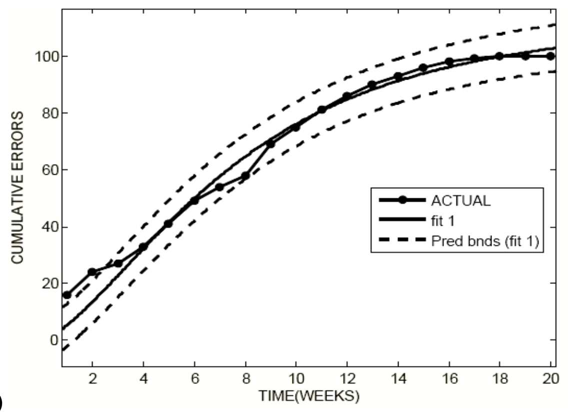
Model	a	r	τ	MSE	R^2	SSE
Proposed model Eq. (10)	135.6	0.0001423	-------	18.413	0.9796	331.4
Proposed model Eq (11)	183.8	$r_1 = 0.000111$	4	6.80	0.9929	115.7
		$r_2 = 0.000078$				
Proposed model Eq (11)	183.8	$r_1 = 0.0001028$	5	6.80	0.9929	115.7
		$r_2 = 0.000078$				
Proposed model Eq (11)	183.8	$r_1 = 0.0000977$	6	6.80	0.9929	115.7
		$r_2 = 0.000078$				
Proposed model Eq (11)	183.8	$r_1 - 0.0000944$	7	6.77	0.9929	114.7
		$r_2 = 0.000078$				
Proposed model Eq (11)	183.8	$r_1 = 0.0000921$	8	6.77	0.9929	114.7
		$r_2 = 0.01397$				
Yamada Delayed S shaped	99.4	0.0005434	------	107.12	0.9857	232.3



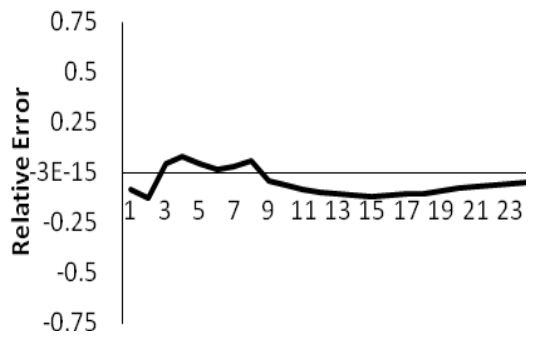
(a)



(c)



(b)



(d)

Fig 2.(a) Observed/estimated TE vs time for dataset 2 (b) mean value function for Eq 3 (c) Mean value function for Eq 10 and Eq 11 (d) RE curves for proposed model at τ=7

6. Optimal release policy

One of the major problems in software industry is at what time and when the software is released in to market. This problem is defined in many papers and a solution is defined. If more testing is conducted on the software ultimately it increases the cost related to it, at the same if testing period is less than the software cannot be reliable enough. For that we have to know at what optimal time the software has to be released into market. Many people like [Musa, Yamada] had studied the above problem and they had given their respective solutions. In certain context people explicitly stated the scheduled delivery of the software analyzed its penalty cost [31]. Recently some authors had proposed warranty cost, cost based model. The software testing process is either can be done manually are automated testing tools are used between the process. It is observed that the software development company has to keep track of the scheduled time of the software, if the testing takes more time, the current time increases greater than scheduled time. In order to cop-up with scheduled time testers will adopt the new automated testing tools which are more efficient than manual testing. By incorporating the new testing

tools into testing can increase the test efficiency. Use of testing tools speed up the process but increases the software cost.

6.1 Optimal release policy based on cost

This section deals with the release policy based on the cost-reliability criterion. Using the total software cost evaluated by cost criterion, the cost of testing-effort expenditures during software testing/development phase and the cost of fixing errors before and after release are: [17, 18, 19, 20, 21].

Where C_1 the cost of correcting an error during testing, C_2 is the cost of correcting an error during the operation, $C_2 > C_1$, C_3 is the cost of testing per unit testing effort expenditure and T_{LC} is the software life-cycle length.

$$C1(T) = C_1 \times m(T) + C_2 \times [m(T_{LC}) - m(T)] + C_3 \times \int_0^T w(t)dt$$

(22)

Now testing time reaches the time τ they adopt the new automated tool. New cost of adopting the new automated testing tool is added along with the previous costs [9].

Now new cost equation

$$C2(T) = C_0(T) + C_1 \times (1+P) \times m(T) +$$
$$C_2 \times [m(T_{LC}) - (1+P) \times m(T)] + C_3 \times \int_0^T w(T)dT$$

(23)

Now above equation C2(T) is total cost of the software by incorporating the new automated testing tools. Now subtract Eq.(22) and E.q(23) (C1(T)-C2(T)) ≥ 0 then

$$P \times m(T) \times [C_2 - C_1] \geq C_0(T) \quad (24)$$

From the equation (24) we can decide that adopting new automated tools can be beneficiary or not. Now differentiate Eq (23) with respect to T then

$$\frac{d}{dT} C2(T) = \frac{d}{dT} C_0(T) + C_1 (1+P) \left(\frac{d}{dT} m(T) \right) - C_2 (1+P) \left(\frac{d}{dT} m(T) \right)$$
$$+ C_3 w(T)$$

(25)

By making the above equation to zero now we will get a fine unique solution T_0. From the mean value function E.q (3)

$$\frac{d}{dT} C2(T) = \frac{d}{dT} C_0(T) + C_1 \times (1+P) \times a \times r \times w(T) \times e^{-r \times W\circledast(T)}$$
$$- C_2 \times (1+P) \times a \times r \times w(T) \times e^{-r \times W\circledast(T)} + C_3 \times w(T)$$

(26)

We can consider several possibilities of $C_0(T)$

1) If $C_0(T)$ is constant : in this case $C_0(T)$ $= C_0$, T ≥ τ; $C_0(T) = 0$ for T<τ where τ is the starting time of adopting new technique and method and $C_1 > 0, C_2 > 0, C_3 > 0$ *and* $C_2 > C_1$ then we have the following cases
   a) From Eq.(26) by assigning $\frac{dC2(T)}{dt} = 0$ then
   $(C_2 - C_1) \times (1+P) ar\, e^{-rW*(T_\tau)} > C_3$ And
   $(C_2 - C_1) \times (1+P) ar\, e^{-rW*(T_\tau)} < C_3$ (27)
   And there exist a unique solution

$$T_0 = \frac{\ln\left(\left(- \frac{\ln\left(- \frac{C_3}{a\ r\ (-C_2 + C_1)} \right)}{r\ \alpha + \ln\left(- \frac{C_3}{a\ r\ (-C_2 + C_1)} \right)} \right)^{\frac{1}{k}} + 1 \right)}{\lambda}$$

Optimal release time $T^* = T_0$

   (b)

$if\,(C_2 - C_1) \times (1+P) ar\, e^{-rW*(T_\tau)} \leq C_3$ There exist $(C_2 - C_1) \times (1+P) ar\, e^{-rW*(T_\tau)} < C_3$ for τ<T<T_{LC} Therefore optimal release time $T^* = \tau$ since $\frac{dC2(T)}{dT} > 0$ for τ < T < T_{LC}.

2) In this the cost of adopting new testing tool is linearly related to the effort

$C_0(T) = C_{01} + C_0\int_{\tau}^{T} w(t)dt$, T $\geq$ τ ; $C_0(T) = 0$, T < τ , where C_{01} is the cost of adopting new technique.

$$\frac{C2(T)}{dT} = C_0\, w(t) + C_1\,(1+P)\,a\,r\,w(T)\,e^{-r\,W\,T} - C_2\,(1 + P)\,a\,r\,w(T)\,e^{-r\,W\,T} + C_3\,w(T)$$

(28)

Since w (t)>0 for 0≤ T ≤ ∞ , $\frac{dC2(T)}{dT} = 0$ if

$$a \times r \times (1+P) \times e^{-r\times[W(T)-W(0)]} = C_3 + C_0$$

(29)

Because the left side is monotonically decreasing function of T if

$$a \times r \times (1+P) \times e^{-r\times[W(T)-W(0)]} > C_3 + C_0$$

and

$$a \times r \times (1+P) \times e^{-r\times[W(T)-W(0)]} < C_3 + C_0$$,

there exist unique solution for Eq.(29)

$$T_0 = \frac{\ln\left(\left(-\frac{\ln\left(-\frac{C3 + C0}{a\,r\,(1+P)\,(-C2+C1)}\right)}{r\,\alpha + \ln\left(-\frac{C3 + C0}{a\,r\,(1+P)\,(-C2+C1)}\right)}\right)^{\frac{1}{k}} + 1\right)}{\lambda}$$

(30)

3) In this the cost of incorporating new testing tool is exponentially related to the effort

$$C_0(T) = C_{01} + C_0\left(\left[\int_{\tau}^{T} w(t)dt\right]^m\right)$$, T $\geq$ τ;

$C_0(T) = 0$, T < τ, because w(t)>0 for 0$\leq$ T <∞ , $\frac{dC2(T)}{dT} = 0$, if

$$a \times r \times (1+P) \times (c_2 - c_1) = c_3 + c_0 \times m \times \left(\left[\int_{\tau}^{T} w(t)dt\right]^{m-1}\right)$$

(31)

6.2 software release time based on reliability Criterion

Generally software is release into the market when its reliability reaches its acceptance level. So we need to determine the time at which our software has to be released to market. Reliability of the software is given by [6, 9, 20]

$$R(T) = \begin{cases} \frac{m(T)}{a} & 0 \leq t \leq \tau \\ \frac{m_1(T)}{a} = \frac{(1+P)m(T)}{a} & , t > \tau \end{cases}$$

(32)

Solve the above equation and obtain the unique solution T_1 satisfying R(T_1)=R_0 .

6.3 Software release time based on cost-reliability criterion with efficiency

This section deals with minimizing the total software cost with respect to the desired reliability, then calculate the software release time. In this minimize the C(T) with respect to the R(T) where 0 < R_0 < 1.

T^* =Optimal release time $= \max(T_0, T_1)$ where T_0 is satisfying the equations (27) ,(30) ,(31) and T_1 satisfying Eq.(32).

Theorem 1:

Assume $C_0(T)=C_0$ (constant) , $C_0 > 0$, $C_1 > 0$,$C_2 > 0$, $C_3>0$, and $C_2 > C_1$ we have

i) $(C_2 - C_1)(1+P)are^{-rW(\tau)} < C_3$, And $(C_2 - C_1)(1+P)are^{-rW(\tau)} > C_3$, T^* =max(T_0 ,T_1) for R(τ)<R_0 <1 or T^* =T_0 for 0<R_0≤ R(τ).

ii) If $(C_2 \quad C_1)(1+P)are^{-rW(\tau)} < C_3$, , T^* =T_1 for R (τ)<R_0 <1 or T^* =τ for 0<R_0 ≤R(τ).

iii) If $(C_2 - C_1)(1+P)are^{-rW(\tau)} > C_3$, , T^* ≥T_1 for R(τ)<R_0 <1 or T^* ≥τ for 0<R_0 ≤R(τ).

Theorem 2:

Assume C_0 (T)=C_{01} +$C_0\int_{\tau}^{T} w(t)dt$, C_{01} , C_0 >0, C_1 >0, C_2 > 0, C_3>0, and C_2 >C_1 we have

i) If $(C_2 - C_1)(1+P)are^{-rW(\tau)} > C_3 + C_0$ and $(C_2 - C_1)(1 \mid P)are^{-rW(T_{LC})} < C_3 \mid C_0$ and T^* =max(T_0 ,T_1) for R(τ)<R_0 <1 or T^* =T_0 for 0<R_0≤R(τ).

ii) If $(C_2 - C_1)(1+P)are^{-rW(\tau)} < C_3 + C_0$ then T^* =T_1 for R(τ)<R_0 <1 or T^* =τ for 0<R_0≤R(τ).

iii) If $(C_2 - C_1)(1+P)are^{-rW(T_{LC})} > C_3 + C_0$, T^* ≥T_1 for R(τ)<R_0 <1 or T^* ≥τ for 0<R_0≤R(τ).

6.4 Numerical examples.

DS1:For estimated parameters for proposed model for dataset 1we have α=72, λ=0.04847, k=1.387, a=578.8, r=0.01903, $C_0 = 10$, $C_1 = 2$, $C_2 = 10$, $C_3 = 200$, $C_{01} = 1000$, τ=19 and $T_{LC} = 100$ weeks. Table 5 shows the relationship between the cost , Optimal release time and P. as the value of P is increasing the Optimal release time and decreases the total cost of the software.

Table 5 : Relation between T_0^*, $C2(T^*)$ and $R(T^*)$ which is based on P.

$$C_0(T) = 1000 + 10 \times \int_{10}^{400} w(t)dt$$

P	Optimal Time T^*	Optimal Cost C2(T^*)	R(T^*)	P	Optimal Time T^*	Optimal Cost C2(T^*)	R(T^*)
0.01	20.7	14358	0.612	0.2	27.56	10891	0.802
0.02	21.02	14184	0.622	0.21	27.99	10700	0.812
0.03	21.34	14008	0.632	0.22	28.44	10509	0.822
0.04	21.66	13832	0.642	0.23	28.89	10317	0.832
0.05	21.99	13655	0.652	0.24	29.36	10124	0.842
0.06	22.32	13477	0.662	0.25	29.84	9931	0.852
0.07	22.65	13297	0.672	0.26	30.33	9737	0.862
0.08	22.99	13117	0.682	0.27	30.84	9542	0.872
0.09	23.34	12936	0.692	0.28	31.36	9346	0.882
0.10	23.69	12755	0.70	0.29	31.9	9150	0.900
0.11	24.05	12572	0.712	0.3	32.46	8954	0.912
0.12	24.41	12388	0.722	0.31	33.04	8756	0.922
0.13	24.78	12204	0.732	0.32	33.64	8559	0.932
0.14	25.15	12019	0.742	0.33	34.26	8161	0.942
0.15	25.53	11833	0.752	0.34	34.91	7961	0.952
0.16	25.92	11646	0.762	0.35	35.59	7761	0.962
0.17	26.32	11458	0.772	0.36	36.29	7560	0.972
0.18	26.72	11270	0.782	0.37	37.04	7359	0.990
0.19	27.14	11081	0.792				

Table 6 : Relation between T_0^*, $C2(T^*)$ and $R(T^*)$ which is based on P.

$$C_0(T) = 1000 + 10 \times \left(\int_{10}^{400} w(t)\,dt\right)^{1.5}$$

P	Optimal Time T^*	Optimal Cost C2(T^*)	R(T^*)	P	Optimal Time T^*	Optimal Cost C2(T^*)	R(T^*)
0.01	19.17	14399	0.5919	0.2	25.16	10550	0.7819
0.02	19.45	14225	0.6019	0.21	25.52	10358	0.7919
0.03	19.74	14050	0.6119	0.22	25.89	10166	0.8019
0.04	20.03	13873	0.6219	0.23	26.27	9972	0.8119
0.05	20.32	13518	0.6319	0.24	26.65	9778	0.8219
0.06	20.62	13339	0.6419	0.25	27.05	9583	0.8319
0.07	20.92	13159	0.6519	0.26	27.45	9388	0.8419
0.08	21.22	12978	0.6619	0.27	27.85	9192	0.8519
0.09	21.52	12796	0.6719	0.28	28.27	8995	0.8619
0.10	21.83	12613	0.6819	0.29	28.70	8798	0.8719
0.11	22.14	12430	0.6919	0.3	29.14	8600	0.8819
0.12	22.46	12245	0.7019	0.31	29.59	8402	0.8919
0.13	22.78	12060	0.7119	0.32	30.05	8202	0.9019
0.14	23.11	11874	0.7219	0.33	30.52	8003	0.9119
0.15	23.44	11500	0.7319	0.34	31.01	7803	0.9219
0.16	23.77	11311	0.7419	0.35	31.51	7602	0.9319
0.17	24.11	11122	0.7519	0.36	32.03	7400	0.9419
0.18	24.45	10932	0.7619	0.37	32.57	7198	0.9519
0.19	24.80	10742	0.7719				

From above table 5 we observed that optimal cost $C2(T^*)$ =7961 at T^*=34.91 at the same time C1(T^*)=8414 and reliability has been improved from 0.81 to 0.95 at P=0.37. Now the condition of C1(T)-C2(T)>0 is satisfied. From the table 6 in which cost of using the automated tools are exponentially related to the effort involved.

DS2: From the dataset two estimated values of SRGM with Logistic-exponential TEF α=12600(CPU hours), λ=0.06352 /week, k=1.391, a=135.6 and r=0.0001432 C_1=1, C_2 =300, C_3 =2 and T_{LC} =100, $C_{01} = 1000$, τ=9 and $T_{LC} = 100$ weeks. Table 7 shows the relationship between the cost, Optimal release time and P. as the value of P is increasing the Optimal release time and decreases the total cost of the software.

Table 7: Relation between at T_0^*, $C2(T^*)$ and $R(T^*)$ which is based on P.

$$C_0(T) = 1000 + 10 \times \int_0^{100} w(t)dt$$

P	*Optimal Time* T^*	*Optimal Cost* *C2*(T^*)	*R*(T^*)	*P*	*Optimal Time* T^*	*Optimal Cost* *C2*(T^*)	*R*(T^*)
0.01	*5.13*	*26354*	*0.3166*	*0.31*	*7.89*	*17846*	*0.6166*
0.02	*5.233*	*26087*	*0.3266*	*0.32*	*7.97*	*17547*	*0.6266*
0.03	*5.34*	*25818*	*0.3366*	*0.33*	*8.05*	*17248*	*0.6366*
0.04	*5.44*	*25549*	*0.3466*	*0.35*	*8.13*	*16948*	*0.6466*
0.05	*5.54*	*25278*	*0.3566*	*0.35*	*8.21*	*16647*	*0.6566*
0.06	*5.64*	*25006*	*0.3666*	*0.36*	*8.30*	*16345*	*0.6666*
0.07	*5.74*	*24732*	*0.3766*	*0.37*	*8.38*	*16042*	*0.6766*
0.08	*5.83*	*24457*	*0.3866*	*0.38*	*8.46*	*15739*	*0.6866*
0.09	*5.93*	*24182*	*0.3966*	*0.39*	*8.54*	*15435*	*0.6966*
0.10	*6.03*	*23904*	*0.4066*	*0.40*	*8.62*	*15131*	*0.7066*
0.11	*6.12*	*23626*	*0.4166*	*0.41*	*8.70*	*14825*	*0.7166*
0.12	*6.22*	*23347*	*0.4266*	*0.42*	*8.78*	*14519*	*0.7266*
0.13	*6.31*	*23066*	*0.4366*	*0.43*	*8.86*	*14212*	*0.7366*
0.14	*6.40*	*22785*	*0.4466*	*0.44*	*8.93*	*13905*	*0.7466*
0.15	*6.50*	*22502*	*0.4566*	*0.45*	*9.01*	*13597*	*0.7566*
0.16	*6.59*	*22218*	*0.4666*	*0.46*	*9.09*	*13288*	*0.7666*
0.17	*6.68*	*21933*	*0.4766*	*0.47*	*9.17*	*12978*	*0.7766*
0.18	*6.77*	*21648*	*0.4866*	*0.48*	*9.25*	*12668*	*0.7866*
0.19	*6.86*	*21361*	*0.4966*	*0.49*	*9.32*	*12357*	*0.7966*
0.20	*6.95*	*21073*	*0.5066*	*0.50*	*9.40*	*12046*	*0.8066*
0.21	*7.03*	*20784*	*0.5166*	*0.51*	*9.48*	*11734*	*0.8166*
0.22	*7.12*	*20494*	*0.5266*	*0.52*	*9.56*	*11421*	*0.8266*
0.23	*7.21*	*20204*	*0.5366*	*0.53*	*9.63*	*11108*	*0.8366*
0.24	*7.29*	*19912*	*0.5466*	*0.54*	*9.71*	*10794*	*0.8466*
0.25	*7.38*	*19619*	*0.5566*	*0.55*	*9.79*	*10479*	*0.8566*
0.26	*7.47*	*19326*	*0.5666*	*0.56*	*9.86*	*10164*	*0.8666*
0.27	*7.55*	*19302*	*0.5766*	*0.57*	*9.94*	*9849*	*0.8766*
0.28	*7.64*	*18736*	*0.5866*	*0.58*	*10.01*	*9533*	*0.8866*
0.29	*7.72*	*18440*	*0.5966*	*0.59*	*10.093*	*9216*	*0.8966*
0.30	*7.80*	*18143*	*0.6066*	*0.60*	*10.169*	*8898*	*0.9066*

7. Conclusions

In this paper, we proposed a SRGM incorporating the Logistic-exponential testing effort function with change-point. We observed that most of software failure is time dependent. By incorporating testing-effort and change-point into SRGM we can make realistic assumptions about the software failure. In order to speed up the testing process we used the automated testing tools. The experimental results indicate that our proposed model fits fairly well.

Appendix –A

$$S = \sum_{kl=1}^{n} \left(\ln(W_{kl}) - \ln(\alpha) - k \times \ln \frac{\left(e^{\lambda \times t_{kl}} - 1\right)^k}{\left(1 + \left(e^{\lambda \times t_{kl}} - 1\right)^k\right)} \right)^2$$

(33)

$$\frac{\partial S}{\partial \alpha} = 0$$

$$0 = \frac{2\,n\,\ln(\alpha)}{\alpha} + \sum_{kl=1}^{n} \left(-\frac{2\ln(W_{kl})}{\alpha} + \frac{2\,k\,\ln\left(e^{\lambda\,t_{kl}} - 1\right)^k}{\alpha\left(1 + \left(e^{\lambda\,t_{kl}} - 1\right)^k\right)} \right)$$

(34)

$$\alpha = e^{\frac{\sum_{kl=1}^{n} \ln(W_{kl}) - k\ln\left(\sum_{kl=1}^{n} \frac{\left(e^{\lambda\,t_{kl}} - 1\right)^k}{1 + \left(e^{\lambda\,t_{kl}} - 1\right)^k}\right)}{n}} \qquad (35)$$

$$\frac{\partial S}{\partial \lambda} = $$

$$0 = 2\,k^2 \ln\left(\sum_{kl=1}^{n} \frac{1}{\left(1 + \left(e^{\lambda\,t_{kl}} - 1\right)^k\right)^3} \left(t_{kl}\, e^{\lambda\,t_{kl}} \left(-\ln(W_{kl}) \left(e^{\lambda\,t_{kl}} - 1\right)^{k-1} - \ln(W_{kl}) \left(e^{\lambda\,t_{kl}} - 1\right)^{2k-1} + \left(e^{\lambda\,t_{kl}} - 1\right)^{k-1} \ln(\alpha) + \left(e^{\lambda\,t_{kl}} - 1\right)^{2k-1} \ln(\alpha) + k\left(e^{\lambda\,t_{kl}} - 1\right)^{2k-1} \ln \right)\right)\right)$$

$$\frac{dm_1(t)}{dt} + m_1(t) \times r_1 \times w(t) = r_1 \times a \times w(t)$$

(36)

If I.F= $e^{\int_0^t w(t) \times r_1 \, dt}$

(37)

$$m_1(t) \times e^{r_1 \times W(t)} = \int_0^t e^{r_1 \times W(t)} \times a \times r_1 \times w(t)\,dt$$

(38)

Solving the above equation

$$m_1(t) = a \times \left(1 - e^{-r1 \times [W(t) - W(0)]}\right), \quad 0 < t < \tau$$

(39)

$$(dm_2(t))/dt \times 1/(w(t)) = r_2 \times (a - (m_1(\tau) + m_2(t)))$$

(40)

$$\frac{dm2(t)}{dt} = r_2 \times w(t) \times \left(a \times e^{-r_1 \times [W(\tau) - W(0)]} - m_2(t)\right)$$

(41)

$$m_2(t) \times e^{r_2 \times [W(t) - W(\tau)]} = \int_\tau^t e^{r_2 \times [W(t) - W(\tau)]} \times a \times r_2 \times e^{r_1 \times [W(\tau) - W(0)]}$$

(42)

Integrating right side of the equation

$$m_2(t) = a - m_1(t) - a \times e^{-r_1 \times W(\tau) - r_2 \times [W(t) - W(\tau)]}$$

(43)

$$m_1(t) + m_2(t) = a \times (1 - e^{(-[r_1 \times (W(\tau) - W(0)) + r_2 \times (W(t) - W(\tau))])})$$

(44)

$$m(t) = a \times (1 - e^{(-[r_1 \times (w(\tau) - w(0)) + r_2 \times (w(t) - w(\tau))])}) \qquad > \tau$$

(45)

References

[1] Chang, Y.P., 2001. Estimation of parameters for non-homogeneous Poisson process software reliability with Chang-point model. Communications in Statistics: Simulation and Computation 30, 623–635.
[2] Chatman, V.V., 1995. Change-points: a proposal for software productivity measurement. Journal of Systems and Software 31, 71–91.
[3] Grottke, M., 2001. Software reliability model study. Research Report, A.2, Project PETS, University of Erlangen-Nuremberg, Germany.
[4] Hou, R.H., Kuo, S.Y., Chang, Y.P., 1994. Applying various learning curves to the hyper-geometric distribution software reliability growth model. In: Proceedings of the 5th International Symposium on Software Reliability Engineering, Monterey, CA, pp. 196– 205.
[5] Hou, R.H., Kuo, S.Y., Chang, Y.P., 1996. Optimal release policy for hyper-geometric distribution software reliability growth model.IEEE Transactions on Reliability 45 (4), 646–651.
[6] Huang, C.Y., in press, Performance analysis of software reliability growth models with testing-effort and change-point. Journal of Systems and Software.
[7] Huang, C.Y., Kuo, S.Y., 2002. Analysis and assessment of incorporating logistic testing effort function into software reliability modeling. IEEE Transactions on Reliability 51 (3), 261–270.
[8] Huang, C.Y., Lo, J.H., Kuo, S.Y., 1998. A pragmatic study of parametric decomposition models for estimating software reliability growth. In: Proceedings of the 9th IEEE International Symposium on Software Reliability Engineering, Paderborn, Germany, pp. 111–123.
[9] Huang, C.Y., Lo, J.H., Kuo, S.Y., Lyu, M.R., 1999a. Software reliability modeling and cost estimation incorporating testing-effort and efficiency. In: Proceedings of the 10th IEEE International Symposium on Software Reliability Engineering, Boca Raton, FL, pp. 62–72.
[10] Huang, C.Y., Kuo, S.Y., Lyu, M.R., 1999b. Optimal software release policy based on cost, reliability and testing efficiency. In: Proceedings of the 23rd IEEE Annual International Computer Software and Applications Conference, Phoenix, AZ, pp. 468–473.

[11] Kapur, P.K., Bardhan, A.K., 2002. Testing effort control through software reliability growth modeling. International Journal of Modeling and Simulation 22 (2), 90–96.

[12] Kapur, P.K., Younes, S., 1995. Software reliability growth model with error dependency. Microelectronics and Reliability 35 (2), 273–278.

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

[14] Framework for modeling software reliability, using various testing-effort and fault-detection rates. IEEE Transactions on Reliability 50 (3), 310–320.

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

[16] Malaiya, Y.K., Mayrhauser, A., Srimani, P., 1993. An examination of fault exposure ratio. IEEE Transactions on Software Engineering
19 (11), 1087–1094.

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

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

[19] Ohba, M., 1984. Software reliability analysis models. IBM Journal of Research and Development 28 (4), 428–443.

[20] Okumoto, K., Goel, A.L., 1980. Optimum release time for software systems based on reliability and cost criteria. Journal of Systems and Software 1, 315–318.

[21] Pham, H., 2000. Software Reliability. Springer-Verlag.

[22] Rafi M.D**,** Dr.K.Nageswara Rao, Shaheda Akthar “Software Reliability Growth Model with Logistic-Exponential Test-Effort Function and Analysis of Software Release Policy.” (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 02, 2010, 368-380.

[23] Shyur, H.J., 2003. A stochastic software reliability model with imperfect-debugging and change-point. Journal of Systems and Software 66, 135–141.

[24] Singpurwalla, N.D., Wilson, S.P., 1999. Software Engineering: Reliability and Risk. Springer-Verlag,

[25] Wood, Predicting software reliability, IEEE computers 11 (1996) 69–77.

[26] Xie, M., 1991. Software Reliability Modeling. World Scientific Publishing Company.

[27] Xie, M., 2000. Software reliability models—past, present and future.

[28] Yamada, S., 2000. Software reliability models and their applications: a survey. In: Proceedings of the International Seminar on SoftwareReliability of Man-Machine Systems, Kyoto University, Kyoto, Japan, pp. 56–80.

[29] Yamada, S., Ohtera, H., 1990. Software reliability growth models for testing effort control. European Journal of Operational Research 46 (3), 343–349.

[30] Yamada, S., Osaki, S., 1985. Software reliability growth modeling: models and applications. IEEE Transactions on Software Engineering 11 (12), 1431–1437.

[31] Yamada, S., Narihisa, H., Osaki, S., 1984. Optimum release policies for a software system with a scheduled software delivery time. International Journal of Systems Science 15 (8), 905–914.

[32] Yamada, S., Ohtera, H., Narihisa, H., 1986. Software reliability growth models with testing effort. IEEE Transactions on Reliability 35 (1), 19–23.

[33] Yamada, S., Hishitani, J., Osaki, S., 1991. Test-effort dependent software reliability measurement. International Journal of Systems Science 22 (1), 73–83.

[34] Yamada, S., Hishitani, J., Osaki, S., 1993. Software reliability growth model with Weibull testing effort: a model and application. IEEE Transactions on Reliability 42, 100–105.

[35] Zhao, M., 1993. Change-point problems in software and hardware reliability. Communications in Statistics—Theory and Methods 22
(3), 757–768.

[36] Zheng, S., 2002. Dynamic release policies for software systems with a reliability constraint. IIE Transactions 34 (3), 253–262.

[37] Zou, F.Z., 2003. A change-point perspective on the software failure process. Software Testing, Verification and Reliability 13, 85–93.

[38] Y. Lan, and L. Leemis, (Aug. 2007) “The Logistic-Exponential Survival Distribution,” Naval Research Logistics (NRL) volume 55, number 3, pp. 252-264.

Sk.MD.Rafi received B.Tech (comp) from Jawaharlal Nehru Technological University, M.Tech (comp) from Acharya Nagarjuna University. Pursuing PhD from Jawaharlal Nehru Technological University. Presently working as Associate. Professor in Sri Mittapalli Institute of Technology for women, affiliated to J.N.T.U, Kakinada. My area of interest is Software Reliability, Software Architecture Recovery, Network Security, and Software Engineering.

Dr.K.Nageswara Rao received B.Tech (electronics) from Karnataka University, M.Tech (comp) from Andhra University and PhD from Andhra University. Presently working as Professor& H.O.D in P.V.P.S.I.T, Vijayawada affiliated to J.N.T.U, Kakinada. My area of interest is Robotics, Software Engineering, Algorithms, and Software Reliability.