Reliability Analysis of a Software with Non Homogeneous Poisson Process (NHPP) Failure Intensity

M. Jain[1] and Kriti Priya[2*]

[1]Department of Mathematics, Indian Institute of Technology, Roorkee, India
[2]Institute of Advanced Management and Research, Ghaziabad, India
e-mail: madhujain@sancharnet.in; kriti.priya@trediffmail.com

Abstract:

This paper deals with the reliability prediction of a large and complex software system consisting of a number of modules in which each module has a different failure pattern. A Non-Homogeneous Poisson Process (NHPP) arising from the superposition of power law processes (S-PLPs) is proposed for determining the failure intensity of the software system. A graphical method is illustrated for estimating the model parameters. Finally, the optimal testing time of the software subject to the reliability requirement is computed. Numerical illustrations are also provided to examine the suggested optimal policies.

Key words: Reliability Prediction, Non-Homogeneous Poisson Process, Failure Intensity

1. Introduction

Software reliability is one of the important metrics for software quality assessment. The most commonly used models in the reliability analysis of a software are the Non-Homogeneous Poisson Processes (NHPPs) which are based on the assumption that the errors or bugs which have been incorporated in the software during the development phase, are removed during the testing phase. A great deal of research efforts in the past has been focused on modeling the software reliability growth during the testing phase. These reliability models which are referred to as black box models, treat the software as a whole; considering its interactions only with the external environment, without regarding the internal structure of the software. These models are based on the stochastic modeling of the failure process, assuming a parametric model of cumulative number of failures over a finite time interval. An extensive survey on black box reliability growth models can be found in [23, 5, 16, 26, 4]. Another approach to estimate the software reliability is

* Corresponding author

the white box approach or architecture-based approach. In such an approach, it is assumed that the software comprises of a number of modules or components whose interactions affect the reliability of the software. With the growing emphasis on reuse, software development is moving towards component-based software design. Consequently, architecture-based or component-based modeling approaches are developing which estimate the reliability of the software by considering the interactions between the components, the reliability of the components and their interfaces with other components. A number of analytical models have been proposed for the architecture-based approach. Some important contributions in this direction are due to [1, 14, 13, 7]. All these models are state-based models in which the architecture of the software is combined with the failure behavior into a composite model, which is then solved to predict the reliability of the software. A detailed account of various types of architecture-based approaches is given in [18].

The level of software reliability to be achieved greatly affects the duration of the testing phase. More testing is required if the software has to be made more reliable. But this leads to an increase in the maintenance cost of the software as well as delay in releasing the software. Hence, determining the optimal testing time or software release time is a major issue in software management. Several authors have studied various software release problems in different scenarios. Yamada and Osaki determined an optimal software release policy for a non-homogeneous software error detection rate model [27]. Kimura et al. analyzed the software release problems with warranty cost and reliability requirement [12]. Jain and Priya proposed the optimal policies for software release time by employing a Delayed S-shaped model for software reliability growth [11]. They followed the white-box approach for analyzing the software reliability. Dai et al studied optimal testing resource allocation with generic algorithm for modular software systems [3]. Chin-Yu et al studied optimal allocation of testing resource considering cost, reliability and testing effort [2]. Worwa et al studied a discrete-time software reliability growth model and its applications for predicting the number of errors encounter during program testing [25]. Jain et al discussed optimal release policies for software reliability growth model (SRGM) with maintenance costs [10]. Gokhale et al studied incorporating fault debugging activities into software reliability models [6]. Quadri et al studied software optimal release policy and reliability growth modeling [21]. Prasad et al studied measurement of software reliability using Sequential Bayesian technique [19]. Quadri and Ahmad studied software reliability growth modeling with new modified weibull testing [22].

In this study, an NHPP arising from the superposition of power law processes (PLPs) is proposed to analyze the failure pattern of a complex software system, which shows a bathtub behavior of the intensity function. We follow an architecture-based approach and

assume that the software consists of a number of modules or components. The reliability constraint based optimal release policies of the software are suggested by analyzing the failure pattern of different modules separately.

The rest of the paper is organized as follows. Section 2, describes the formulation of the failure intensity model along with the notations and assumptions. The estimation of the parameters of the failure intensity model is suggested in section 3. In section 4, the software maintenance cost model is developed to obtain the optimal testing time by minimizing the total maintenance cost of the software. The optimal testing time subject to the reliability constraint is determined in section 5. In section 6, the numerical illustrations are provided to examine the suggested optimal policies for the testing time.

2. The failure intensity model

We assume that the software has n number of modules and that each of the modules consists of different number of initial errors. The failure pattern of each module is modeled by a PLP with scale parameter s_i and shape parameter $h_i(i = 1,2, \dots, n)$.

The following notations are used for the mathematical formulation purpose:

n	Number of modules in the software
a_i	Number of initial faults in module i
α_i	Scale parameter for the failure pattern of module i
β_i	Shape parameter for the failure pattern of module i
m(t)	Expected number of failures detected at time t
EC(T)	Total expected software maintenance cost
T	Software release time
T*	Optimum software release time or testing time
T_w	Warranty period
R	Software reliability
c_o	Initial testing cost
c_t	Testing cost per unit time
c_w	Maintenance cost per fault during the warranty period
$C_w(T)$	Maintenance cost during the warranty period
θ	Discount rate of the cost

The failure intensity of the software can be given by

$$\lambda(t) = \sum_{i=1}^{n} a_i \frac{\beta_i}{\alpha_i}\left(\frac{t}{\alpha_i}\right)^{\beta_i - 1} \quad (1)$$

The expected number of failures or errors up to time t is

$$m(t) = \sum_{i=1}^{n} a_i \left(\frac{t}{\alpha_i}\right)^{\beta_i} \quad (2)$$

The reliability of the software can be computed by

$$R_{i\theta}(x|t) = \exp[-\{m(x+t) - m(t)\}] \quad (3)$$

3. Parameter estimation

Parameter estimation is equally important issue to fit real failure data to predict the failure pattern. For this purpose, we present two approaches for estimating the shape and scale parameters of the S-PLP failure intensity function. The two approaches are based on maximum likelihood (ML) estimation and the graphical method as described below:

Maximum likelihood method

Let $t_1 < t_2 < \cdots < t_N$ denote the times of fault occurrence in the software, observed during interval [0, T]. I is assumed that the failure pattern follows S-PLP with intensity function given by eqn (1). Assuming that (j-1)th failure occurred at t = t_{j-1}, the conditional probability density function of the jth failure time is given by

$$f(t_j|t_{j-1}) = \lambda(t_j)\exp\{-[m(t_j) - m(t_{j-1})]\}, t_j \geq t_{j-1}$$

Taking $t_0 = 0$, the likelihood function can be obtained as

$$L = \prod_{j=1}^{N}\left[\sum_{i=1}^{n}\frac{\beta_i}{\alpha_i}\left(\frac{t_j}{\alpha_i}\right)^{\beta_i - 1}\right] \times exp\left[-\sum_{i=1}^{n}\left(\frac{T}{\alpha_i}\right)^{\beta_i}\right] \quad (4)$$

The log likelihood function is given by

$$l = lnL = \sum_{j=1}^{N} ln\left[\sum_{i=1}^{n}\frac{\beta_i}{\alpha_i}\left(\frac{t_j}{\alpha_i}\right)^{\beta_i - 1}\right] - \left[\sum_{i=1}^{n}\left(\frac{T}{\alpha_i}\right)^{\beta_i}\right] \quad (5)$$

The partial derivatives of the log likelihood function with respect to the model parameters are

$$\frac{\partial l}{\partial \alpha_k} = \sum_{j=1}^{N} \frac{-(\beta_k/\alpha_k)^2 (t_j/\alpha_k)^{\beta_k - 1}}{\sum_{i=1}^{n} (\beta_i/\alpha_i)(t_j/\alpha_i)^{\beta_i - 1}} + \frac{\beta_k}{\alpha_k}\left(\frac{T}{\alpha_k}\right)^{\beta_k}, k = 1,2,\dots,n \tag{6}$$

And

$$\frac{\partial l}{\partial \beta_k} = \sum_{j=1}^{N} \frac{[1+\beta_k ln(t_j/\alpha_k)](t_j/\alpha_k)^{\beta_k - 1}/\alpha_k}{\sum_{i=1}^{n} (\beta_i/\alpha_i)(t_j/\alpha_i)^{\beta_i - 1}} - \left(\frac{T}{\alpha_k}\right)^{\beta_k} ln\left(\frac{T}{\alpha_k}\right), k = 1,2,\dots,n \tag{7}$$

Let $\hat{\alpha}_k$ and $\hat{\beta}_k$ (k = 1,2,…,n) be the required ML estimators. From eqn (6), we get

$$\left(\frac{T}{\hat{\alpha}_k}\right)^{\hat{\beta}_k} = \sum_{j=1}^{N} \frac{(\hat{\beta}_k/\hat{\alpha}_k)(t_j/\hat{\alpha}_k)^{\hat{\beta}_k - 1}}{\sum_{i=1}^{n} (\hat{\beta}_i/\hat{\alpha}_i)(t_j/\hat{\alpha}_i)^{\hat{\beta}_i - 1}}, k = 1,2,\dots,n \tag{8}$$

It is noted from eq^{n} (2) that the ML estimate of the expected number of failures up to T is equal to the observed number of failures, i.e.

$$\hat{m}(T) = \sum_{k=1}^{n} \left(\frac{T}{\hat{\alpha}_k}\right)^{\hat{\beta}_k} = \sum_{k=1}^{n} \sum_{j=1}^{N} \frac{(\hat{\beta}_k/\hat{\alpha}_k)(t_j/\hat{\alpha}_k)^{\hat{\beta}_k - 1}}{\sum_{i=1}^{n} (\hat{\beta}_i/\hat{\alpha}_i)(t_j/\hat{\alpha}_i)^{\hat{\beta}_i - 1}} = \text{N}$$

In particular, for n = 2, we have

$$\hat{m}(T) = \sum_{k=1}^{2} \left(\frac{T}{\hat{\alpha}_k}\right)^{\hat{\beta}_k} = \sum_{k=1}^{2} \sum_{j=1}^{N} \frac{(\hat{\beta}_k/\hat{\alpha}_k)(t_j/\hat{\alpha}_k)^{\hat{\beta}_k - 1}}{\sum_{i=1}^{n} (\hat{\beta}_i/\hat{\alpha}_i)(t_j/\hat{\alpha}_i)^{\hat{\beta}_i - 1}} = \text{N}$$

From the above result, one ML estimator say α_2, can be expressed as a function of the other three ML estimators, i.e.

$$\hat{\alpha}_2 = \frac{T}{\left[N - \left(T/\hat{\alpha}_1\right)^{\hat{\beta}_1}\right]^{1/\hat{\beta}_2}} \tag{9}$$

More detailed explanation of the above method can be found in [20].

The graphical approach

Graphical methods are extensively used for obtaining easy estimates and model parameters. Several researchers have used the graphical methods for the analysis of repairable equipment failing according to a PLP. The relevant references are [15, 8, 24, 17, 20]. Now, we discuss the graphical method for estimating the model parameters in case of n = 2 as follows:

Consider the transformations

x = ln(t) and y = ln[m(t)]

which gives

$$y = y(x) = ln\left[\left(\frac{e^x}{\alpha_2}\right)^{\beta_1} + \left(\frac{e^x}{\alpha_2}\right)^{\beta_2}\right] \quad (10)$$

Fig. (1) depicts the plot of y versus x, which defines the curve C.

From eqn (10), we get

$$y = y(x) = \beta_1(x - ln\alpha_1) + ln\left[1 + e^{(\beta_2-\beta_1)x}\frac{{\alpha_1}^{\beta_1}}{{\alpha_2}^{\beta_2}}\right] \quad (11)$$

Similarly, we have

$$y = y(x) = \beta_2(x - ln\alpha_2) + ln\left[1 + e^{(\beta_1-\beta_2)x}\frac{{\alpha_2}^{\beta_2}}{{\alpha_1}^{\beta_1}}\right] \quad (12)$$

Assuming that $\beta_2 < \beta_1$, we have

$$\lim_{x\to\infty} ln\left[1 + e^{(\beta_2-\beta_1)x}\frac{{\alpha_1}^{\beta_1}}{{\alpha_2}^{\beta_2}}\right] = 0 \quad (13)$$

And

$$\lim_{x\to-\infty} ln\left[1 + e^{(\beta_1-\beta_2)x}\frac{{\alpha_2}^{\beta_2}}{{\alpha_1}^{\beta_1}}\right] = 0 \quad (14)$$

From eqns (11) to (14), the asymptotes of the curve C are given by,

$L_1: y = y(x) = \beta_1(x - ln\alpha_1)$ as $x \to \infty$

$L_2: y = y(x) = \beta_2(x - ln\alpha_2)$ as $x \to \infty$

Hence the curve C has asymptotic slopes equal to β_1 and β_2 as x tends to ∞ and $-\infty$ respectively.

From eqn (10), we have

$$\acute{y}(x) = \beta_1 F_1(x) + \beta_2 F_2(x) \quad (15)$$

Where

$F_k(x) = \frac{(e^x/\alpha_k)^{\beta_k}}{(e^x/\alpha_1)^{\beta_1} + (e^x/\alpha_2)^{\beta_2}}$, k = 1, 2

For any x, $F_1(x) + F_2(x) = 1$ so that $\beta_2 < \acute{y}(x) < \beta_1$ and hence, the curve C always has a slope between β_2 and β_1. The coordinates x_P and y_P of the intersection point P of the asymptotes L_1 and L_2 are

$$x_p = (\beta_1 ln\alpha_1 - \beta_2 ln\alpha_2)/(\beta_1 - \beta_2) \quad (16(a))$$

$$y_p = \beta_1\beta_2 ln(\alpha_1/\alpha_2)/(\beta_1 - \beta_2) \quad (16(b))$$

The shape parameters β_1 and β_2 can be estimated using above equations 16(a-b).

4. Optimal testing time

In this section, we obtain the optimal testing time by minimizing the total maintenance cost of the software. We assume that after the delivery of the software, there is a warranty period (T_w), in which the maintenance cost has to be paid by the developer. We also incorporate a discount rate in the testing cost and maintenance cost so as to determine the present value of the total cost of the software. A detailed description on cost prediction of warranty reserve by including the effects of present value of money can be found in [9].

The total expected software maintenance cost is given by

$$EC(T) = c_0 + c_t \int_0^T \exp(-\theta t)\, dt + C_W(T) \tag{17}$$

where $C_w(T) = c_W \int_T^{T+T_W} \lambda(T)\exp(-\theta t)dt$ (18)

To minimize the total maintenance cost, we differentiate eqn (17) w.r.t T and equate it to zero which gives

$$c_t + \frac{c_W(1-e^{-\theta T_W})}{\theta}\left[-\theta \sum_{i=1}^{n} a_i \left(\frac{\beta_i}{\alpha_i}\right)\left(\frac{T}{\alpha_i}\right)^{\beta_i - 1} + \frac{\sum_{i=1}^{n} a_i \beta_i (\beta_i - 1) T^{\beta_i - 2}}{{\alpha_i}^{\beta_i}}\right] = 0 \tag{19}$$

Particular cases:

i) For n=1:

In this case, there is only one module in the software. Let T_1 be the value of testing time satisfying eqn (19) so that

$${T_1}^{\beta_1 - 1}\left((\beta_1 - 1)T^{-1} - \theta\right) = \frac{-c_t \theta {\alpha_1}^{\beta_1}}{c_w(1-e^{-\theta T_w}) a_1 \beta_1 (\beta_1 - 1)} \tag{20}$$

It can be easily verified that $\left|\frac{d^2 EC(T)}{dT^2}\right|_{T=T_1} > 0$.

Hence, EC(T) has a minimum value at T_1, so that $T^* = T_1$.

Now, $\lambda(T_1) = a_1 \frac{\beta_1}{\alpha_1}\left(\frac{t}{\alpha_1}\right)^{\beta_1 - 1}$

Thus the optimal release policy 1 in this case is stated as follows:

$P^{(1)}{}_{11}$: $T^* = T_1$ when $\lambda(0) > \lambda(T_1)$

$P^{(1)}{}_{12}$: $T^* = 0$ when $\lambda(0) \leq \lambda(T_1)$

ii) For n=2:

This implies that there are two modules in the software. The optimal release time can be computed by using

$$\frac{a_1 \beta_1}{{\alpha_1}^{\beta_1}} {T_2}^{(\beta_1 - 1)}\left\{(\beta_1 - 1){T_2}^{-1} - \theta\right\} + \frac{a_2 \beta_2}{{\alpha_2}^{\beta_2}} {T_2}^{(\beta_2 - 1)}$$

$$\left\{(\beta_2 - 1){T_2}^{-2} - \theta\right\} = -\frac{c_t \theta}{c_w(1-e^{-\theta T_w})} \tag{21}$$

Again, $\left|\frac{d^2EC(T)}{dT^2}\right|_{T=T_2} > 0$and we obtain the minimum cost at T_2.

Here, $\lambda(T_2) = a_1\frac{\beta_1}{\alpha_1}\left(\frac{t}{\alpha_1}\right)^{\beta_1-1} + a_2\frac{\beta_2}{\alpha_2}\left(\frac{t}{\alpha_2}\right)^{\beta_2-1}$ and the optimal release policy 1 is stated as follows:

$P_{11}{}^{(2)}: T^* = T_2$ when $\lambda(0) > \lambda(T_2)$

$P_{12}{}^{(2)}: T^* = 0$ when $\lambda(0) \leq \lambda(T_2)$

4. Optimal testing time with reliability constraint

Here, we obtain the optimal testing time by imposing the reliability constraint. Let R_0 be the minimum reliability to be achieved by the software developer. Now the constrained optimization problem can be stated as:
Minimize $EC(T)$

Subject to $R(x|T) \geq R_0$ (22)

Substituting the value of R(x|T) from eqn (3) in eqn (22), we get

$$exp[-\{m(x+T) - m(T)\}] \geq R_0 \quad (23)$$

Let T_R denote the testing time satisfying

$$exp[-\{m(x+T) - m(T)\}] = R_0$$

The optimal release policy 2 for the above problem is as follows:

i) For n=1:

$P_{21}{}^{(1)}$: If $\lambda(0) > \lambda(T_1)$ and $R(x|0) < R_0$, then $T^* = \max\{T_1, T_R\}$.

$P_{22}{}^{(1)}$: If $\lambda(0) > \lambda(T_1)$ and $R(x|0) \geq R_0$, then $T^* = T_1$.

$P_{23}{}^{(1)}$: If $\lambda(0) \leq \lambda(T_1)$ and $R(x|0) < R_0$, then $T^* = T_R$.

$P_{24}{}^{(1)}$: If $\lambda(0) \leq \lambda(T_1)$ and $R(x|0) \geq R_0$, then $T^* = 0$.

ii) For n=2:

$P_{21}{}^{(2)}$: If $\lambda(0) > \lambda(T_2)$ and $R(x|0) < R_0$, then $T^* = \max\{T_2, T_R\}$.

$P_{22}{}^{(2)}$: If $\lambda(0) > \lambda(T_2)$ and $R(x|0) \geq R_0$, then $T^* = T_2$.

$P_{23}{}^{(2)}$: If $\lambda(0) \leq \lambda(T_2)$ and R(x|0) < R_0, then T*=T_R.

$P_{24}{}^{(4)}$: If $\lambda(0) \leq \lambda(T_2)$ and R(x|0) ≥ R_0, then T*=0.

5. Numerical Illustrations

We provide numerical illustration to examine the optimal release policies suggested in previous section. Fig. 2 depicts the variation of the total maintenance cost EC(T) of the software with the testing time T for n=1 and n=2. It can be noticed that EC(T) first decreases and then increases with T which implies that increase in the duration of the testing phase causes an increase in the total maintenance cost of the software. Also, software having more number of modules requires more maintenance cost as compared to that having lesser number of modules. The effect of the c_w on EC(T) can be seen in Fig. 3. EC(T) increases with c_w which is obvious. Fig. 4 illustrates that increase in the discount rate λ decreases the maintenance cost of the software. Figures 5–6 depict the effect of various parameters on the software reliability (R). The variation of R for n=1 and n=2, with the testing time is shown in Fig. 5. The reliability of the software increases with the testing time but it becomes constant after a certain level. Fig. 6 demonstrates the effect of initial number of errors in module 2 i.e. a_2 on R. Evidently, R decreases with a which implies that the software is less reliable if there are more number of errors present initially in it.

Now, we examine the suggested optimal policies for the testing time (T). Table 1 demonstrates the optimal release policy 1 for n=1 and n=2. The optimal testing time, T* decreases with the testing cost per unit time (c_t). This means that the testing of the software can be completed in a shorter duration if more money is invested on testing phase, which is obvious. Also, T* shows an increasing trend with the warranty period, T_w. Fig. 7 illustrates the optimal testing time for n=2 with reliability objective R_0. In the case of c_t=50, θ =. 001 and T_w=1000, we derive T_2= 308.517 from Table 1(Optimal release policy 2). When the reliability objective R_0=. 8 and operational time x=1, we obtain T_R=1306.522. Hence, from optimal release policy 2, the optimal testing time is T*= max {T_2, T_R}= max {308.517,1306.522}=1306.522.

It is noted from the numerical results that a desired level of software reliability can be achieved by increasing the testing time thereby minimizing the maintenance cost of the software.

N	T_w	c_t				
		10	**20**	**30**	**40**	**50**
1	500	266.741	150.609	110.116	88.766	75.327
	600	299.930	167.899	122.118	98.286	83.238
	700	330.986	183.483	132.923	106.767	90.292
	800	359.750	197.049	142.707	114.248	96.528
	900	385.471	210.230	151.287	120.913	102.050
	1000	408.629	221.385	158.881	127.019	107.061
2	500	793.901	446.185	317.781	253.132	213.495
	600	988.914	503.385	4584.088	281.731	236.872
	700	1113.203	555.846	389.458	307.425	257.768
	800	1229.665	604.592	420.253	330.505	276.457
	900	1338.229	648.528	448.489	351.475	293.337
	1000	1442.126	688.788	474.064	370.267	308.517

Table 1: Optimal testing time for parameters
a_1=20, a_2=50,α_1=3,α_2=2,β_1=. 5,β_2=. 3,θ=. 001,c_0=200,c_w=50

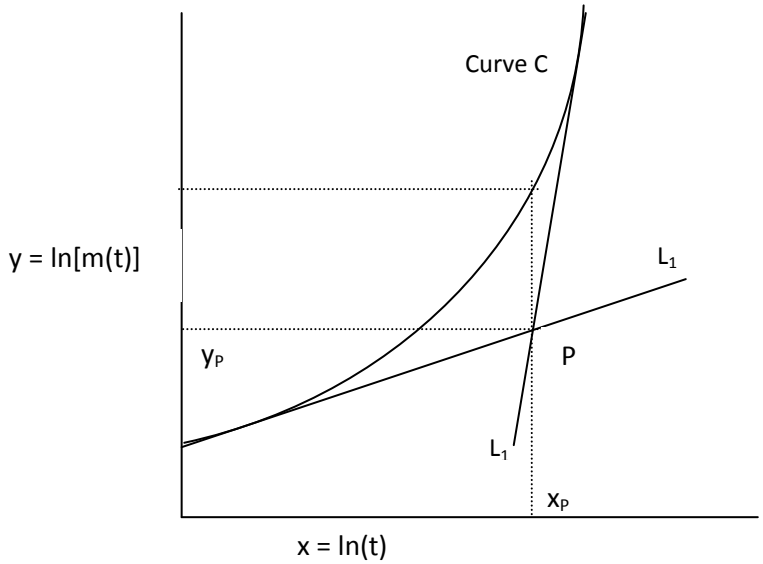

Fig. 1: Curve C and its asymptotes L_1 and L_2

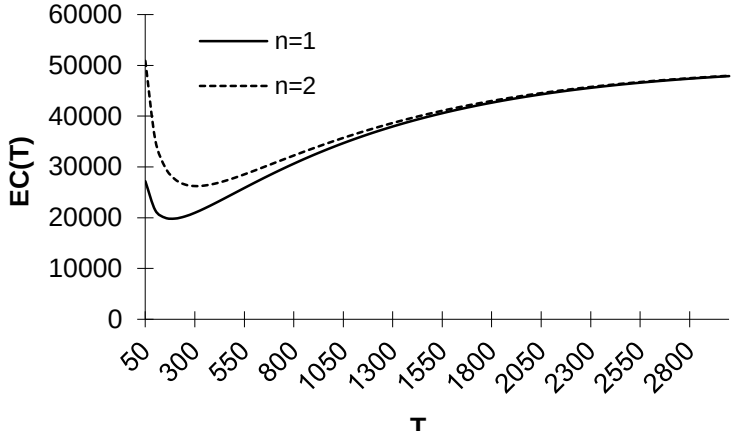

Fig.2: Total maintenance cost Vs. Testing time for parameter $a_1=20, a_2=50, \alpha_1=3, \alpha_2=2, \beta_1= .5, \beta_2= .3, \theta=.001, c_w=50, c_0=200, T_w=1000, c_t=50$

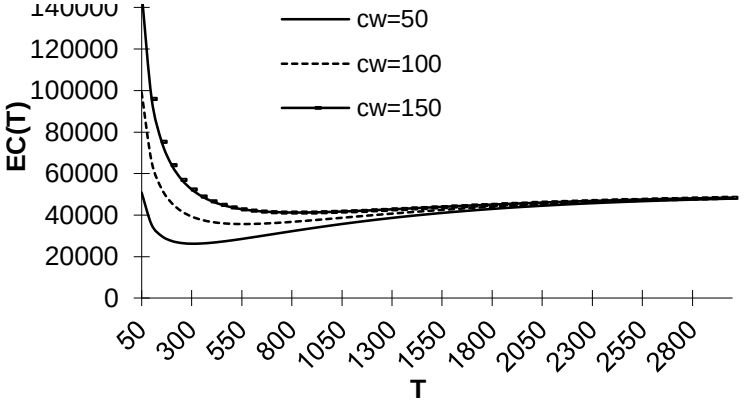

Fig.3: Total maintenance cost Vs. Testing time for parameters $a_1=20, a_2=50, \alpha_1=3, \alpha_2=2, \beta_1=.5, \beta_2=.3, \theta=.001, c_0=200, T_w=1000, c_t=50$

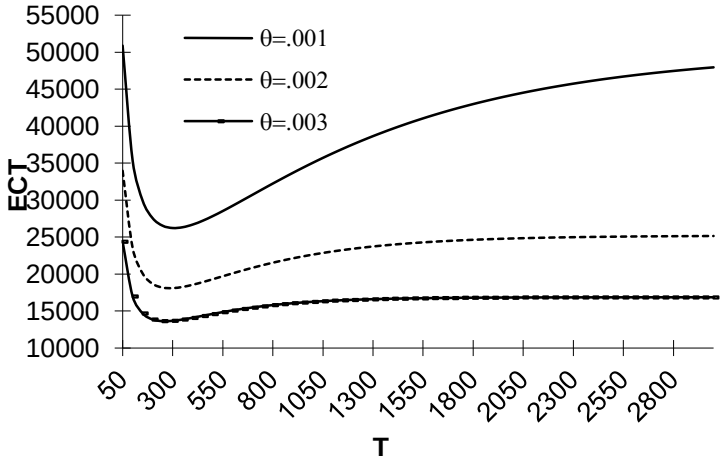

Fig.4: Total maintenance cost Vs. Testing time for parameters $a_1=20, a_2=50, \alpha_1=3, \alpha_2=2, \beta_1=.5, \beta_2=.3, c_w=50, c_0=200, T_w=1000, c_t=50$

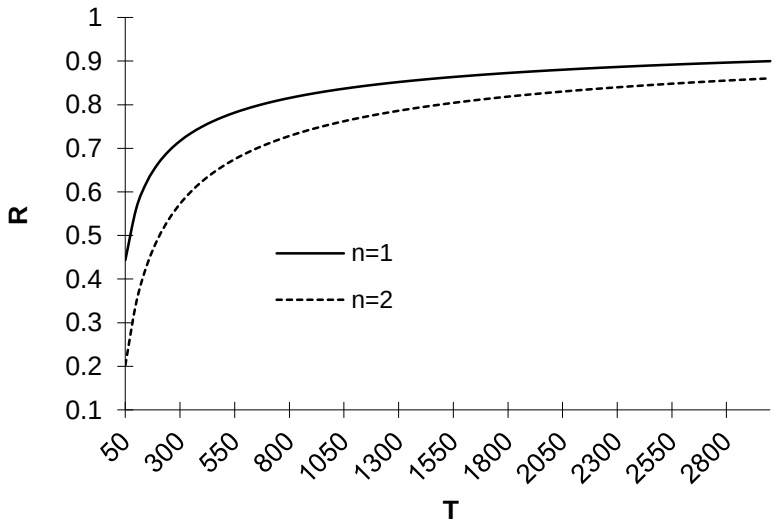

Fig.5: Software Reliability by varying Testing time for parameters $a_1=20, a_2=50, \alpha_1=3, \alpha_2=2, \beta_1=.\ 5, \beta_2=.\ 3, x=1$

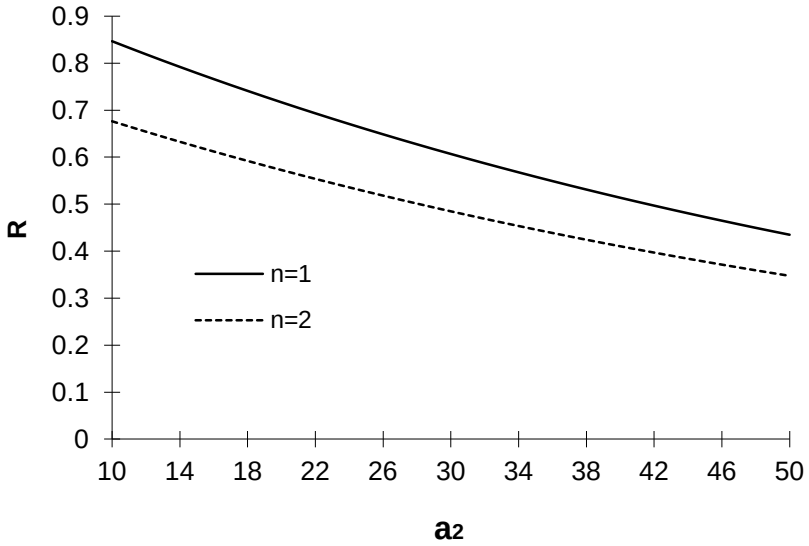

Fig.6: Software Reliability by varying a_2 for parameters $T=300, a_1=20, \alpha_1=3, \alpha_2=2, \beta_1=.5, \beta_2=.3, x=1$

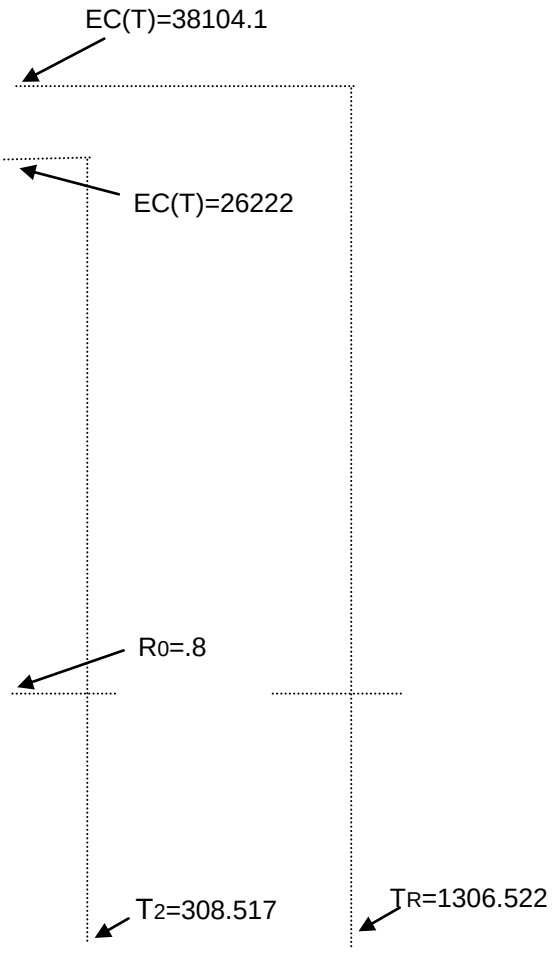

Fig. 7: Optimal release policy 2 by taking parameters
a1=20,a2=50,a1=3,a2=2,b1=.5,b2=.3,q=.001,
cw=50,c0=200,Tw=1000,ct=50, x=1, R0=.8

6. Conclusion

In this paper, we have considered the architecture based software systems. The failure pattern of all the modules has been described by power law processes (PLP) with different shape and scale parameters. We have employed an NHPP arising from the superposition of PLPs to compute the optimal testing time for a modular software system, subject to the reliability constraint. The proposed procedures to estimate the shape and scale parameters of failure intensity function may be easily implemented to fit a S-PLP for real time failure date. The numerical results show that the modular software systems require higher maintenance cost as compared to the non-modular software systems. Also, more testing time is needed to attain a pre-assigned level of reliability in case of modular software systems.

The proposed model can be extended by considering the interactions between various modules of the software. The transition probabilities from one module to another can be taken into consideration for this purpose to analyze the reliability of the software.

References

[1] Cheung, R.C., 1980, A user–oriented software reliability model, *IEEE Trans. Software Engg*. **6, 2**, 118-125.

[2] Chin-Yu Huang, Sy-yen Kuo, Lyu, M.R., 2004, Optimal allocation of testing resource considering cost, reliability and testing effort. 10th Pacific Rim International Symposium on Dependable Computing, 2004, 103-112.

[3] Dai Y., Xie M., Poh K. and Yang B., 2003, Optimal testing resource allocation with generic algorithm for modular software systems, *Journal of Systems and software*, **66, 1**, 47-55.

[4] Fan, W., 1996, Software reliability modeling survey, *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 71-117.

[5] Goel, A.L., 1985, Software reliability models: assumptions, limitations and applicability, *IEEE Trans. Software Engg*. **11, 12**, 1411-1423.

[6] Gokhale S.S., Lyu N. and Trivedi K., 2006, Incorporating fault debugging activities into software reliability models: A simulation approach, *IEEE transactions on reliability*, **55, 2**, 281-292.

[7] Gokhale, S., Wong, W.E., Trivedi, K. and Horgan, J.R., 1998, An analytical approach to architecture based software reliability prediction, *Proceedings Third International Computer Performance and Dependability Symposium*, 1998, 13-22.

[8] Hartler, G., 1985, Graphical weibull analysis of repairable systems, *Quality and Reliability Engineering International*, **1, 1**, 23-26.

[9] Jain M and Handa B.R., 2001, Cost analysis for repairable units under hybrid warranty. Recent Developments in Operational Research, (ed. Manju L:ata Agarwal & Kanwar Sen), Narosa Publishing House, New Delhi, 149-165.

[10] Jain M. Maheshwari S. and Priya K., 2005, Optimal release policies for software reliability growth model (SRGM) with maintenance costs, *Journal of ICT*, **l4**, 99-115.

[11] Jain, M. and Priya, K., 2002, Optimal policies for software testing time, *Computer Society of India*, **32,3**, 25-30.

[12] Kimura, M., Toyota, T. and Yamada, S.,1999, Economic analysis of software release problems with warranty cost and reliability requirement, *Reliability Engineering and System Safety*, **66**, 49-55.

[13] Kubat, P., 1989, Assessing reliability of modular software, *Operations Research Letters*, **8**, 35-41.

[14] Laprie, J.C., 1984, Dependability evaluation of software systems in operation, *IEEE Trans. Software Engg.*, **10, 6**, 701-714.

[15] Lilius, W.A., 1979, Graphical analysis of repairable systems, *Proc. Annual Reliability and Maintainability Symposium*, Washington, DC, 1979, 403-406.

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

[17] Nelson, W., 1998, Graphical analysis of system repair data, *Journal of Quality Technology*, **20, 1**, 24-35.

[18] Popstojanova, K.G, and Trivedi, K.S., 2001, Architecture-based approach to reliability assessment of software systems, *Performance Evaluation*, **45**, 179-204.

[19] Prasad L., Gupta A. and Badoria S., 2009, Measurement of Software reliability using Sequential Bayesian Technique, Proceedings of the world congress on Engineering and Computer Science, **11**, 242-246.

[20] Pulcini, G., 2001, Modeling the failure data of a repairable equipment with bathtub type failure intensity, *Reliability Engineering and System Safety*, **71**, 209-218.

[21] Quadri S., Ahmad N. and Peer M., 2008, Software optimal policy and reliability growth modeling, *Computation for national development*, 2008, 247-256.

[22] Quadri S.M.K. and Ahmad N., 2010, Software Reliability Growth Modeling with new modified Weibull testing effort and optimal release policy, *International Journal of Computer Applications*, **6 ,12**, ……..

[23] Ramamoorthy, C.V. and Bastani, F.B., 1982, Software reliability-status and perspectives, *IEEE Trans. Software Eng*., **8, 4**, 354-371.

[24] Wang, C.G., 1991, Graphical analysis of ill-collected interval data for a repairable system in vehicles, *Proc. Annual Reliability and Maintainability Symposium*, Orlando, FL, 93-97.

[25] Worwa K., 2005, A discrete-time software reliability growth model and its applications for predicting the number of errors encountered during program testing, *Control and Cybernetics*, **34, 2**, 589-606.

[26] Xie, M., 1991, Software Reliability Modelling, World Scientific, Singapore.

[27] Yamada, S. and Osaki, S., 1986, Optimal software release policy for a nonhomogeneous software error detection rate model, *Microelectronics and Reliability*, **26**, 691-702.