Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 06, June -2018

SOFTWARE RELIABILITY GROWTH MODEL WITH NEW DYNAMIC LEARNING WITH FLUCTUATING TESTERS EXPERIENCE AND RELEASE TIME DETERMINATION

[1]Shaik.Mohammad Rafi, [2]Dr.B. Srinivasa Rao, [3]Dr Shaheda Akthar

[1]*Research scholar in the department of computer science ,Acharya Nagarjuna University Guntur.*
[2]*Professor in the Department of Computer Science and Engineering.*
[3]*Working as Registrar of Urdu University Kurnool, Andhrapradesh.*

Abstract: *Software Reliability growth models are a mimic models of both statistics and mathematics. which are developed to estimate the errors during the software testing . In the Past several people have developed different models based on certain assumptions. In one recent paper Chiu proposes a new learning based imperfect debugging environment. Learning during software testing is a continuous process. No testers were initially fully aware of software testing and its related internal errors which were hidden inside the software product. The experience of software testers are very useful during testing and debugging process. Software testing it is fluctuating and Complex environment because when a new errors are generated during software testing even experience testers cannot able to recognize those errors. In this paper we proposes a new software reliability growth model based on incorporating the fluctuating experience of testers and fluctuating learning experience in our models.*

Keywords: *Software Reliability, Software Testing, Testing Effort, Non-homogeneous Poisson Process (NHPP), Software Cost.*

I. INTRODUCTION

Past many years software industry is struggling to produce new quality and reliable software. Software development process itself a complex process where each phase has its own constraint related to time and cost. Among different phases testing is concerned to be more important and heart of development process where plenty of resources are consumed. [8]Software reliability is defined as probability of failure free software over period of time before it fails. Many researches were developed different mathematical models to understand the testing environment. These mathematical models developed are mimic of original testing phase in given environmental conditions. Researches struggles to predict reliability of the software product, because software development phases are very complex and variable in nature. To define a software product as quality one through testing is to be done. Errors are common in software product and error are introduced in software by software developers. All the errors which are introduced by software developer are caught during the testing phase. Testing is very costly phenomenon where total cost of testing can affect the total development cost. Aim software testing is to detects and correct the errors during development and maintenance phases. Errors which are unable to detect during testing , can be caught during operation phase where cost of detection and correction becomes three folds. Many authors were developed varieties of software reliability models either be on failure count data or time count data[8]. Some of researches believe that software failure detection and correction is constant and no new errors were introduced during testing those models termed ad perfect debugging environment. Some researches proposes during software testing can produce new errors which makes us to think imperfect debugging environment. some authors incorporated testing efforts into their models to capture the testing effort spend by the software testers[6]. It is observed that testers experience and their capability of learning about testing environment can make drastic effect on software testing. So some authors proposed software reliability models base on testers experience and learning capability. Chui [5] proposed new reliability model incorporating the both experience and learners capacity into software reliability model. Some authors proposed time varying learning effects into software reliability models etal Chiu and Chen(2013). some authors proposes the learners negligence in to their reliability growth models[1]. Javid Iqbal [7] who has integrated learning functions into imperfect debugging software reliability growth models. Now in this paper we proposed testers learning capacity and their experience treated to be dynamic functions which are incorporated into software reliability growth model. The reason for testers learning and experience capacity will be considered as time varying functions, tests are struggling a lots during software testing their testing capacity will vary with time because of complex testing environment Proposed model validations and performance is estimated on real time datasets. Parameters are estimated through least square estimation with numerical estimation is done as the model is complex in nature.

II. SOFTWARE RELIABILITY GROWTH MODELS WITH LEARNING FACTORS

A) Chiu , Huang and Lee learning model (2008)[5]

in this model authors proposed a imperfect debugging environment based software reliability model based on casual loop diagram. they incorporated learning and experience of software testers in their models. they feel that learning and experience of software during software testing can effect on software testing during defect identification in constant environment. they feel that learning and experience factors are constant.

$$f(t) = (\alpha + \eta * F(t)) * (1 - F(t)) \quad (1)$$

above equation solved by assuming F(0)=0 then

$$F(t) = \frac{e^{(\alpha+\eta)*t}-1}{\frac{\eta}{\alpha}+e^{(\alpha+\eta)*t}} \quad (2)$$

B) Kuei-Chen Chiu (2012) and chiu , Kuei Chen 2013[2][5]

In this paper author proposed new model based on time varying learning phenomenon by introducing new learning factors. where they introdued two new time varying learning factors into their model. $\eta(t) = (1 + \xi * t)\ and\ \eta(t) = e^{\xi * t}$. where ξ represents coefficient of accelerating factor.

$$F(t) = \frac{e^{1+\frac{\eta+\xi*t}{\alpha'}}-1}{\frac{\eta}{\alpha}+e^{1+\frac{\eta+\xi*t}{\alpha'}}} \quad (3)$$

$$F(t) = \frac{e^{1+\frac{\eta+e^{\xi*t}}{\alpha'}}-1}{\frac{\eta}{\alpha}+e^{1+\frac{\eta+e^{\xi*t}}{\alpha'}}} \quad (4)$$

here learning and experience factors varies with time.

C) Javid Iqbal , N.Ahmad and S.M.K Quadri 2013[1]

in this paper authors assumes that software testers ate little negligent during testing process where it has adverse effect on software testing . they incorporated an negligent factor into their model.

$$\alpha'(t) = \eta_1 * \alpha - \tau \quad (5)$$

now the

$$F(t) = \frac{e^{1+\frac{\eta_2}{\alpha'(t)}}-1}{\frac{\eta}{\alpha}+e^{1+\frac{\eta_2}{\alpha'(t)}}} \quad (6)$$

D) Proposed Model

in this paper we proposed a new model by integrating both time varying functions $\eta(t)$dynamic learning function and $\alpha(t)$ experience of testers. we assume that software learning phenomenon depending on the environmental conditions of testing phase. testing phase environment is dynamic in nature so we have incorporated both functions into software reliability growth models.

$$\frac{dF(t)}{dt} = ([\alpha(t) + \eta(t) * F(t)]) * ([1 - F(t)]) \quad (7)$$

as we have integrated $\eta(t)$ as dynamic time oriented function which can varies with time. depending on testing environment the learning function also varies with time.

$$z(t) = \int_0^t \frac{dF(t)}{1-F(t)} = \int_0^t (\alpha(t) + \eta(t) * F(t))dt \quad (8)$$

$$\theta * F(t) = m(t) = \theta * (1 - e^{-((\phi_1(t)+\phi_2(t))}) \quad (9)$$

$$\lambda(t) = (\phi_1{}'(t) + \phi_2{}'(t)) * e^{-((\phi_1(t)+\phi_2(t))} \quad (10)$$

$$\phi_2(t) = \int_0^t \eta(t) * F(t)dt \quad (11)$$

$$\phi_1(t) = \int_0^t \alpha(t)\, dt \quad (12)$$

from above equation $z(t)$ represents Hazard Rate function which completely depending on testers learning and experience capability. It is observed that hazard rate function z(t) depends on functions concerned to learning capacity function $\eta(t)$, and $\alpha(t)$ learners experience function to select error from software product without performing the testing , *a adjusting* distribution function *F(t).* By assigning suitable distribution functions into these functions can give dynamic effects into deriving models. for that we assumed three different functions represented in eq.(13)

$$\eta(t) = \eta * (1 + r * t) \;\; and \;\; F(t) = (1 - e^{-\beta * t}) \qquad and \;\; \alpha(t) = \frac{\alpha}{(1+C*exp(-\gamma*t))} \quad (13)$$

In learning function $\eta(t)$ r represents learning accelerating factor, η learning factor ,
F(t) an adjustment distribution function which adjusts the given testing environment, in this paper we assumed adjustment function as exponential in nature. β represents distribution parameter.
$\alpha(t)$ represents experience function where α represents experience factor. In this paper we assumed an S shaped function as experience function represents experience of testers in software testing.

substituting (13) into (9) and assuming F(0)=0 we derived the following equation

$$m(t) = \theta * \left\{1 - \left(\frac{1+C*e^{\gamma*t}}{(1+C)*e^{\gamma*t}}\right)^{-\frac{\alpha}{\gamma}} * e^{-\left(t+\frac{1}{2}*r*t^2\right)*\eta*\left(1-e^{-\beta*t}\right)+\beta*\left(-\frac{1}{2}*\frac{e^{-\beta*t}[\beta^2*t^2+2*\beta^2*t+2*\beta*t+2*\beta+2]}{\beta^3}\right)} + \frac{(\beta+1)}{\beta^3}\right\} \quad (14)$$

difference between Kuei-Chen Chiu (2012) and chiu , Kuei Chen 2013 and our Proposed model is we are incorporated dynamic functions in to software reliability growth model where as they assumed and substituted learning factors into their respective models.

III PARAMETER ESTIMATION

in this paper we used standard procedure as least squate estimation to validate our proposed. As the equation is little complex in nature we used numerical approximations.

$$SSE = \left\{(m_i - \theta * (1 - e^{-((\phi_1(t)+\phi_2(t))}))\right\}^2 \quad (15)$$

$$\frac{d(SSE)}{d\theta} = -2 * \sum_{i=1}^{n}\left\{(m_i - \theta * (1 - e^{-((\phi_1(t)+\phi_2(t))}) * \left\{(1 - e^{-((\phi_1(t)+\phi_2(t))})\right\}\right\} = 0 \quad (16)$$

$$\frac{d(SSE)}{d\alpha} = -2 * \sum_{i=1}^{n}\left\{(m_i - \theta * (1 - e^{-((\phi_1(t)+\phi_2(t))}) * \left\{\left(\left(\theta * e^{-((\phi_1(t)+\phi_2(t))}\right) * \frac{d(-((((\phi_1(t)+\phi_2(t))))}{d\alpha}\right)\right\}\right\} = 0 \quad (17)$$

$$\frac{d(SSE)}{d\eta} = -2 * \sum_{i=1}^{n}\left\{(m_i - \theta * (1 - e^{-((\phi_1(t)+\phi_2(t))}) * \left\{\theta * \left(e^{-((\phi_1(t)+\phi_2(t))} * \frac{d(-((((\phi_1(t)+\phi_2(t))))}{d\eta}\right)\right\}\right\} = 0 \quad (18)$$

$$\frac{d(SSE)}{dr} = -2 * \sum_{i=1}^{n}\left\{(m_i - \theta * (1 - e^{-((\phi_1(t)+\phi_2(t)))}) * \left\{\theta * \left(e^{-((\phi_1(t)+\phi_2(t))} * \frac{d(-((((\phi_1(t)+\phi_2(t))))}{dr}\right)\right\}\right\} = 0 \quad (19)$$

$$\frac{d(SSE)}{d\beta} = -2 * \sum_{i=1}^{n}\left\{(m_i - \theta * (1 - e^{-((\phi_1(t)+\phi_2(t))}) * \left\{\theta * \left(e^{-((\phi_1(t)+\phi_2(t))} * \frac{d(-((((\phi_1(t)+\phi_2(t))))}{d\beta}\right)\right\}\right\} = 0 \quad (20)$$

IV EVALUATION CRITERIA

A) 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.

$$R^2 = 1 - \frac{(Residual\ Sum\ of\ Squares)}{(Corrected\ Sum\ of\ Squares)} \quad (21)$$

$$R^2 = 1 - \sum_{i=1}^{n} \frac{(m_i - m(t_i))^2}{\sum_{i=1}^{n}(m_i - \sum_{j=1}^{n}\frac{m_j}{n})^2} \quad (22)$$

IV MODEL PERFORMANCE ANALYSIS

A) DATA SETS

In this paper we used standard datasets used by various authors in their research paper. we have taken the reference of datasets 1 and 2 from research paper proposed by Chiu (2008)[5] .

Table 1

S.No	Reference	Datasets
1	Zhang and Pham (1998)	Failure data of misra system
2	Pham (2003)	Failure data real time control system

model comparisons are done through MSE and R^2 .

B) RESULTS

Following *Table 2* indicates parameters of our proposed models.. Model parameters are estimated through least square estimation with numerical approximations. *Table 3* indicates all fitted results of comparisons of different models based on R^2 values. table 4 shows the results of various models fitted on Zhang and Pham 1998 model data set. as from the given table 4 it seems proposed models better predicts the software failures. hence a good fit model.

Table 2

Datasets	Proposed model
Zhang and Pham (1998)	$\alpha = 2.555, \beta = 0.04973, \eta = 0.269, r = 0.38, \theta = 138.9, \gamma = 0.2406, C = 13.47$
Pham (2003)	$\alpha = 0.6316, \beta = 0.007514, \eta = 0.3136, r = 0.8653, \theta = 131.7, \gamma = 0.99, C = 10.59$

Table 3

Models	Sources of datasets	
	Zhang and Pham (1998)	**Pham(2003)**
Pham and Zhang(2003)	0.966	0.975
Huang (2005)	0.973	0.982
Chiu (2008)	0.966	0.975
Chiu and Kuei -Chen linear model(2013)	0.975	0.987
Chiu and Kuei -Chen exponential model(2013)	0.986	0.989
Javaid Iqbal, N. Ahmad and S.M.K Quadri (2013)	0.966	0.978
Proposed Model $m(t)$	0.997	0.995

Table 4

Total defects predicted by the following models based on Zhang and Pham (1998)

Testing time (per hour)	Defects found	Pham and Zhang(2003)	Huang (2005)	Chiu (2008)	Chiu and Huang and Lee 2013	Chiu and Huang and Lee2013	Proposed model m(t)
1	27	17.515178	18.753639	17.527226	17.527305	17.527226	23.02711
2	43	32.789511	34.611824	32.795171	32.795691	32.795171	41.96998
3	54	46.105543	48.073478	46.095057	46.096522	46.095057	56.49929
4	64	57.711199	59.544218	57.680570	57.683470	57.680571	67.18791
5	75	67.823776	69.354962	67.772696	67.777430	67.772700	74.87942
6	82	76.633546	77.776637	76.563933	76.570776	76.563948	80.41629
7	84	84.306973	85.031825	84.221968	84.231067	84.222016	84.55608
8	89	90.989586	91.304011	90.892873	90.904255	90.893015	87.95523
9	92	96.808530	96.744969	96.703889	96.717483	96.704300	91.15600
10	93	101.874820	101.480660	101.765860	101.781512	101.767016	94.55560
11	97	106.285370	105.615980	106.175330	106.192839	106.178534	98.36696
12	104	110.124690	109.238530	110.016420	110.035532	110.025130	102.60376
13	106	113.466510	112.421750	113.362390	113.382835	113.385768	107.11081

14	111	116.375100	115.227370	116.277050	116.298563	116.339097	111.63887
15	116	118.906460	117.707460	118.816010	118.838314	118.978846	115.93349
16	122	121.109420	119.906050	121.027700	121.050527	121.449567	119.80249
17	122	123.026490	121.860500	122.954300	122.977408	124.023841	123.14401
18	127	124.694700	123.602620	124.632560	124.655724	127.209285	125.93970
19	128	126.146320	125.159520	126.094480	126.117512	131.484050	128.22979
20	129	127.409420	126.554440	127.367970	127.390682	135.255654	130.08523
21	131	128.508460	127.807290	128.477290	128.499548	135.971130	131.58673
22	132	129.464730	128.935240	129.443630	129.465296	135.974000	132.81070
23	134	130.296760	129.953060	130.285400	130.306378	135.974000	133.82324
24	135	131.020680	130.873580	131.018670	131.038874	135.974000	134.67738
25	136	131.650520	131.707870	131.657420	131.676788	135.974000	135.41299

Figure 1

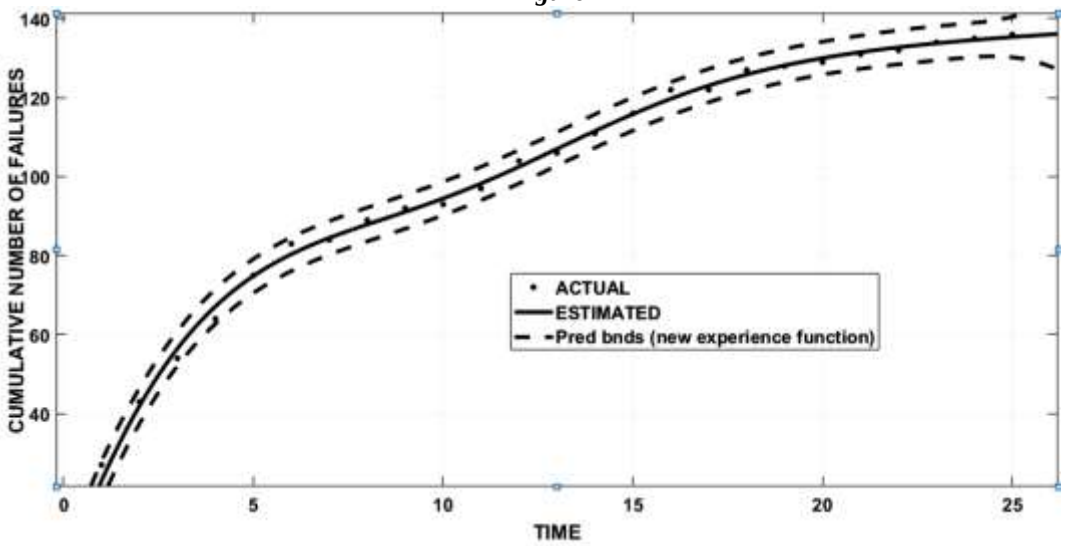
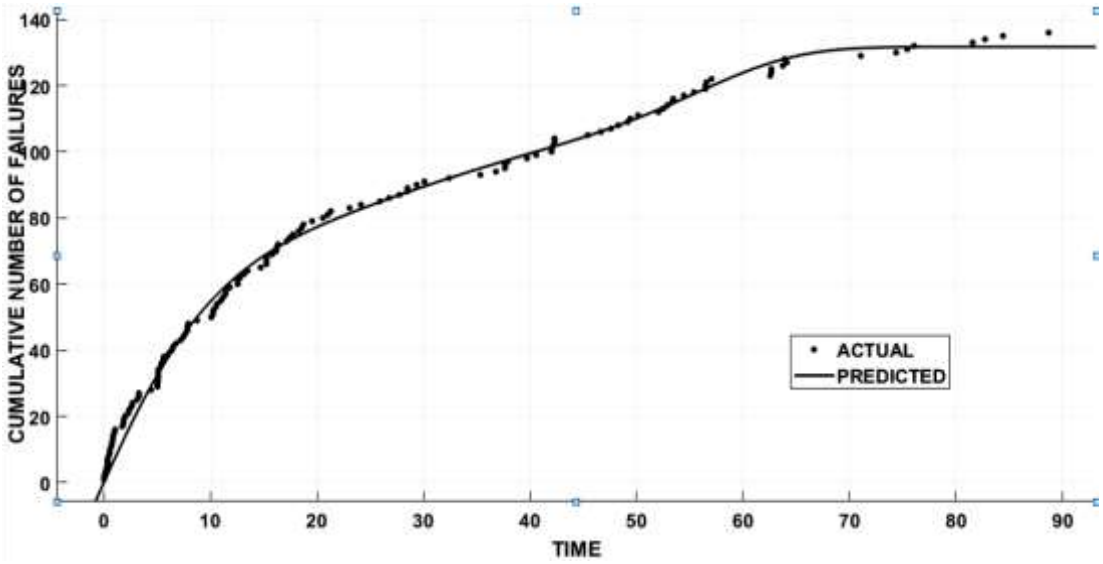


Figure 1and Figure 2 indicates the estimated model based on original dataset1 and dataset 2

Figure 2



V. OPTIMAL SOFTWARE RELEASE POLICY

software release time determination is an important concern to many software development process. Software release time determination is concerned with time at which software has to be delivered to the customer such that released software product should have quality and error free. in order to determine the exact release time we must know its reliability and concerned cost of testing of the product. once they have determined reliability and cost we can predict the release time based on cost and reliability which are predicted.

A) Software Release-Time Based on Reliability Criteria[9]

software reliability can be estimated based on the change in a mean value function over a period of time. for that following equations represents the concerned reliability expression

$$R(t) = e^{[m(t+\Delta t)-m(t)]} \quad (23)$$

Lets consider the required R_0 reliability to release the software product. the expression 18 changed as

$$R_0 = e^{[m(t+\Delta t)-m(t)]} \; and \; [m(t+\Delta t) - m(t)] = \ln(R_0) \quad (24)$$

$$\theta * [\left(1 - e^{-(\alpha*(t+\Delta t)+\phi(t+\Delta t))}\right) - (1 - e^{-(\alpha*(t)+\phi(t))})] = ln(R_0) \quad (25)$$

$$[e^{-(\alpha*(t)+\phi(t))} - e^{-(\alpha*(t+\Delta t)+\phi(t+\Delta t))}] = \frac{ln(R_0)}{a} \quad (26)$$

solving the above equation we will optimal time T_{R_0} at which the reliability could reach R_0. *Figureb3* indicates the reliability of dataset 1 through proposed model with mean value function $m(t)$. from the table4 the concern product can reach $R_0 = 0.95$ at $T_{R_0} = 28$

Figure 3

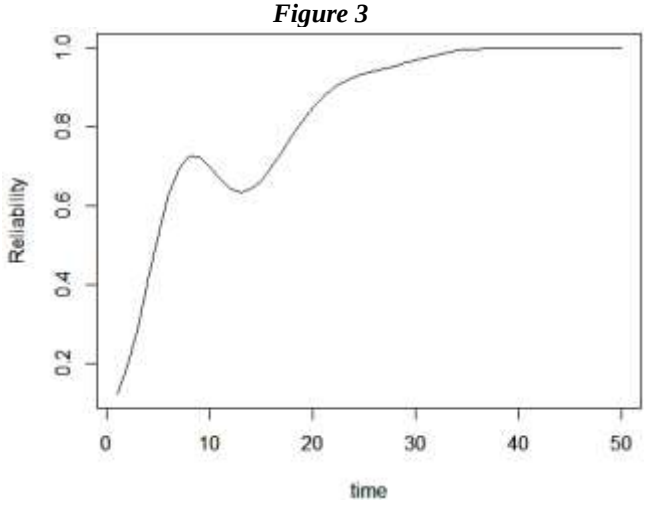


Table 5

Time	Reliability	Time	Cost
22	**0.8960**	**23**	**567.23**
23	**0.9123**	**24**	**566.25**
24	**0.9246**	**25**	**566.10**
25	**0.9340**	**26**	**566.59**
26	**0.9415**	**27**	**567.60**
27	**0.9480**	**28**	**569.07**
28	**0.9542**	**29**	**571.00**

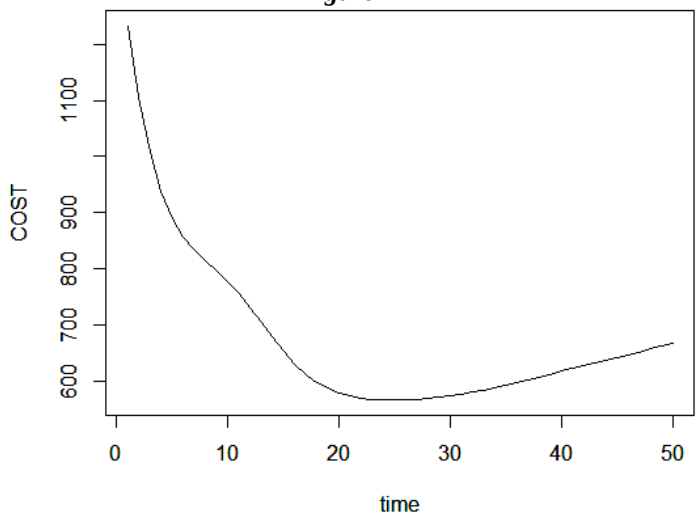
B) Optimal release time based on cost criterion[9]

Software development cost can be estimated from following expression where C1 and C2 and C3 are cost associated with correcting the errors during testing , error correction during operational use of software and miscellaneous cost during entire software development process

$$COST(t) = C1 * m(t) + C2 * [m(t_{LC}) - m(t)] + C3 * t \quad (27)$$

now from $\frac{dCOST(t)}{dt} = 0$ then $\lambda(t) = \frac{C3}{C2-C1}$ find the T_C at which cost the software to be minimized. let us consider the various cost related with C1=3 , C2=10, and C3=5 applied through second model $m_2(t)$ on dataset 1. *Figure 4* show the relation between cost and time . from the table 5 we can determine time at which cost of the software is optimal $T_C = 25$

Figure 4



C) Optimal release time based on Cost and Reliability Criterion[9]

Based on above equations (26) and (27) software release time can be determined based on $max\{T_{R_0}, T_C\} = \{28,25\}$. So the product can be released at 28 where it has optimal quality in it.

VI Conclusions

Software reliability growth models helping the software industries by estimating the remaining number of faults and quality of the software product. Software reliability growth models a mimic models of software testing phase where each reliability growth models capture the actual real time environmental testing in the industries. In this paper we are trying to use testers experience and their capability to identify new errors during testing and learning capacity of testers are integrated in software reliability growth model. Proposed model best fit for software real time failure data. By integrating many functions we want to captures the real time actual environment during testing phase. proposed model can mould it self according to the change in actual environment during testing. In future we integrate some rigorous functions into proposed models .

REFERENCES:

[1] Javaid Iqbal and N. Ahmad and S.M.K. Quadri " A Software Reliability Growth Model with Two Types of Learning" Proceedings of the 2012 IEEE IEEM

[2] Kuei-Chen Chiu " A Study of Software Reliability Growth Model for Time-dependent Learning Effects" Proceedings of the 2012 IEEE IEEM

[3] Javid Iqbal "Analysis of Some Software Reliability Growth Models with Learning Effects" I.J.Mathematical Sciences and Computing 2016.

[4] Javid Iqbal " Software reliability growth models: A comparison of linear and exponential fault content functions for study of imperfect debugging situations" Cogent Engineering (2017).

[5] Kuei-Chen Chiu Yeu-Shiang Huang and Tzai-ZangLee"A study of software reliability growth from the perspective of learning effects" 2008

[6] Huang, C.Y., Kuo, S.Y. and Lyu, M.R. (2000), "Effort-index based software reliability growth models and performance assessment", in Proceedings of the 24th IEEE Annual International Computer Software and Applications Conference (COMPSAC'2000), pp. 454-9.

[7] Chiu ,Kuei Chen " A discussion of software reliability growth models with time varying learning effects" American Journal of Software engineering and Applications 2013

[8] Pham, H. Software Reliability, Springer-Verlag, New York, NY.2000

[9] Yamada, S. and Osaki, S. (1985b), "Cost-reliability optimal release policies for software systems", IEEE Transactions on Reliability, Vol. R-34 No. 5, pp. 422-4.