Determination of Software Release Instant of Three-Tier Client Server Software System

Yogesh Singh ys66@rediffmail.com
Professor & COE University School of Information Technology
Guru Gobind Singh Indraprastha University, Kashmere Gate,
Delhi - 110403, India

Pradeep Kumar pksharma26@rediffmail.com
Associate Professor, Department of Information Technology
ABES Engineering College affiliated to UPTU Lucknow,
Ghaziabad - 201009, India

Abstract

Quality of any software system mainly depends on how much time testing take place, what kind of testing methodologies are used, the complexity of software and the amount of efforts put by software developers subject to the cost and time constraint. More time developers spend on testing more errors can be removed leading to better reliable software. On the contrary, if testing time is too short, the software cost could be reduced, but in that case the customers may take a higher risk of buying unreliable software. However, this will increase the cost during operational phase since it is more expensive to fix an error during operational phase than during testing phase. Therefore it is essentially important to decide when to stop testing and release the software to customers based on cost and reliability assessment. In this paper we present a mechanism of when to stop testing process and release the software to end-user by developing software cost model with risk factor. Based on the proposed method we specifically address the issues of how to decide that now we should stop testing and release the software that is based on three-tier client server architecture which would facilitates software developers to ensure on-time delivery of a software product matching the criteria of attaining a predefined level of reliability and minimizing the cost. A numerical example has been cited to illustrate the experimental results showing significant improvements over the conventional statistical models based on NHPP.

Keywords: Software Reliability Growth Model (SRGM), Optimal Release Policy, Three-tier Client server System

1. INTRODUCTION

Several software cost models and optimal release policies have been studied for modeling software reliability growth trends with different predictive capabilities at different phases of testing. Software Reliability Growth Models (SRGMs) have been known as most widely used mathematical tools for measuring, assessing, and predicting software reliability quantitatively. The project managers and practitioners of software development have a great challenge of how to develop a reliable software system economically that can be used for reliability assessment in a

realistic environment. As one of the major issues is to decide when to stop testing and release the software to customer timely at low price with high degree of reliability.
SRGMs associated with software reliability measurement structure enhance both developer and customer understanding of software quality and the factors affecting it. The factors include time for how long a program has been executing, software product characteristics, development process characteristics including resources, and operational environment in which the software is used. Since early 1970s, software reliability modeling has been in practice to model past failure data to predict future behavior. This approach employs either the observed number of failures discovered per time period or observed time between failures of software. Software reliability models therefore fall into two basic classes, depending upon types of data the model uses: failures per time period and time between failures. Basically one of the well-known and most important applications of SRGMs is to determine the software release instant [1, 2, 3, 7, 8, 9, 10, 11, 12, 14, 17]. In our study we investigate that how software faults detection process can be employed to develop software reliability models to predict the behavior of failure occurrences and the fault content of a software product that can be used in the determination of software release instant.

Rest of the paper is organized as follows: In section 2, we discuss in detail the motivational work done in the field of software reliability growth modeling and release policy. Section 3 describes the mathematical formulation of software risk cost model and in section 4, numerical example is provided to examine the optimal testing policies for proposed model. The concluding remarks and directions for future work are discussed in section 5.

2. RELATED WORK

Many researchers and practitioners have addressed the problem of software release instant over the years particularly Okumoto and Goel (1980) discussed a cost model addressing linear development cost during testing and operational phase. Yamada (1983) developed S-shaped reliability growth model for software error detection. Yamada and Osaki (1986) presented an optimal software release policy for a non-homogeneous software error detection rate model. Othera and Yamada (1990) discussed optimum software release time problem with fault-detection during operation by introducing two evaluation criteria for the problem, first software reliability and second mean time between failures. Yamada (1991) discussed software reliability measurement and assessment of various software reliability growth models and data analysis.

KK Aggarwal and Y Singh (1993) presented a method for determination of software release instant using a non-homogeneous error detection rate model based on the fact that some faults can be regenerated during the process of correction. Pham (1996) developed a cost model with an imperfect debugging and random life cycle besides a penalty cost to determine optimal release policies for a software system. Kimura et al. (1999) discussed optimal software release policy with consideration of an operational warranty period during which developer has to pay the cost for fixing any detected errors. Pham and Zhang (1998) developed a generalized cost model including fault removal cost, warranty cost and software risk cost due to software failures. They also developed a GUI tool to determine the optimal software release time. Pham and Zhang (1999) reviewed optimum release policy literature and concluded that quality of software system depends on how much time testing takes and what kind of testing methodologies are used.

Hoang Pham (2003) categorically studied software reliability modeling based on nonhomogeneous Poisson process (NHPP) with environmental factors and cost factors. Chin Huang (2005) reviewed software reliability growth modeling with generalized logistic testing-effort function and concluded that generalized logistic testing-effort function can be used to describe actual consumption of resources during the software development process. Kuei-Chen Chiu et al. (2007) proposed in their study that perspective of learning effects can influence the process of learning effect that comes from inspecting the testing /debugging codes. Chu and Huang (2008) further enhanced the predictive capabilities of testing effort dependent software reliability models by introducing multiple change-points into Weibull-type testing-effort functions.

2.1 Software Reliability Growth Model for Three-Tier Client Server System

In a distributed computing environment to improve the process of reliability estimation and prediction of software products we discuss and describe a three-tier client server architecture based system for error detection process during testing phase. However reliability can be enhanced through various means such as improving the process of designing, effectiveness of testing, manual & automated inspections, familiarization with developers, users & product, and improving the management processes & decisions [1, 2]. The rate of reliability growth depends on the factors related to how rapidly defects are identified, how fast corrective action take place & how soon the impact of the changes is implemented in the operational phase. Nevertheless all preventive measures need to be taken during fault detection in order to correct and freeze them. To formulate our methodology we consider a conventional client server architecture where presentation logic and application logic are split off into separate components resulting into three-tier system shown in figure1.

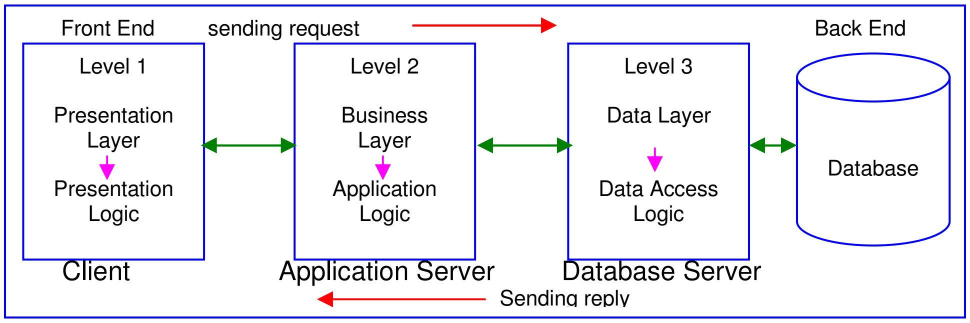

FIGURE 1: A Three-Tier Client Server Architecture View of the proposed model

The presentation layer of proposed model contains forms providing user interface, display data, collect user inputs and sends the requests to next layer. Application layer provides the support services to receive the requests for data from user tier, evaluates against business rules, and pass on them to data tier. Data layer includes data access logic and to store the data at backend. In modern computing system particularly for web based applications where various modules of software are executed on different machine under different network architecture and operating conditions we apply software cost model with risk factor to make a realistic reliability prediction and assessment.

2.2 Terminology

NHPP: nonhomogeneous Poisson process represents the number of failures experienced up to time *t i.e.,* $\{N(t), t \geq 0\}$. The NHPP based model provides an analytical framework for describing the software failure phenomenon during testing phase.
Testing-effort: resource expenditures spent on software testing, e.g., test cases, man-power, CPU time etc.
Fault: an incorrect logic, incorrect instruction, or inadequate instruction that upon execution will cause a failure.
Error: a cause of a failure, which is unacceptable departure from nominal program operation.
Software error: an error made by a programmer or designer, such as a typographical error or an incorrect numerical value or an omission, etc.
Operational profile: the set of operations that the software can execute, given the probabilities of their occurrence.

2.3 Acronyms

MLE	maximum likelihood estimation
MVF	mean value function

SRGM	software reliability growth model
SSE	sum of squared errors

2.4 Notations used

$m(t)$	mean value function in NHPP model
a	Total number of software errors to be detected
b_i	error correction rate during initial testing phase of i^{th} layer of model for i=1,2,3
r_i	error generation factor due to correction of errors in initial testing phase of i^{th} layer of the model
t_i	time spent in initial testing phase at i^{th} layer of model for i=1,2,3
t	total time spent in all three phases of testing
$\lambda(t)$	Fault detection rate per unit time
T	Software release time
C_1	Software test cost per unit time
C_2	Cost of removing each error per unit time during testing
C_3	Cost of risk due to software failure
$E(T)$	Expected total cost of a software system by time T
$N(T)$	Number of errors to be detected by time T
μ_y	Expected time to remove an error during testing phase which is E(Y)
Y	Time to remove an error during testing phase
$R(x\|t)$	conditional software reliability

2.5 Assumptions

The proposed software cost model is developed based on the following assumptions under different circumstances as follows:

1. Initially there is a set-up cost of the software development process.
2. Cost to perform testing is proportional to testing time.
3. Cost to remove errors during testing phase is proportional to total time of removing all errors detected by the end of testing phase.
4. Time to remove each error during testing follows a truncated exponential distribution.
5. There is a risk cost related to the reliability at each release time point.

2.6 A nonhomogeneous Poisson process model

The counting process $\{N(t), t \geq 0\}$ is known as NHPP with an intensity function $\lambda(t)$, $t \geq 0$ and N(t) has a Poisson distribution with a mean value function m(t) given by:

$$\Pr\{N(t) = k\} = \left\{[m(t)]^k \exp\{-m(t)\} / k!\right\} \quad \text{, where } k = 0,1,2,\ldots n. \text{ and} \tag{1}$$

$m(t) = E[N(t)]$ is the mean value function.

The Pr {N(t)} denotes the probability of event N(t), the mean value function m(t) represents expected cumulative number of faults detected during testing time interval (0,t] and intensity function $\lambda(t)$ representing fault detection rate per fault. Using Goel-Okumoto NHPP reliability model the mean value function m(T) can be written as follows:

$$m(T) = \left\{ a\,(1 - \exp\{-bT\}) \right\} \quad \text{, where } a>0 \text{ and } b>0 \tag{2}$$

3. SOFTWARE COST MODEL WITH RISK FACTOR

Here we describe mathematically a software cost model with risk factor for three-tier client server system consisting of three type of faults where some faults are easier to detect then others based on the amount of efforts required to detect causes of failure in order to fix and remove it. These faults are associated with presentation layer, business layer and database layer during testing phase addressing risk level and time to remove errors. The optimal release policy that minimizes the expected total software cost is obtained without loss of generality, by using mean value function m (T) given as follows:

$$m(T) = a\left[\sum_{i=1}^{3}\left\{(1 - \exp\{-b_i T_i\}) *(1- r_i)\right\}\right] \qquad (3)$$

Where $t = t_1 + t_2 + t_3$, $a > 0$, and $0 < b_3 < b_2 < b_1 < 1$, $0 < r_i < 1$

For three types of fault at each layer the error detection rate function dm(T) / dT can be written as:

$$\lambda(T) = a\left[\sum_{i=1}^{3} b_i \exp\{-b_iT_i\} *(1- r_i)\right] \qquad (4)$$

The probability of a software failure which does not occur in (T, T+x], given that last failure occurred in T >=0 (x>=0) is defined as:

$$R(x \mid T) = \exp[-\{m(T+x) - m(T)\}] \qquad (5)$$

By substituting the values from eqn. (3) we get

$$R(x \mid T) = \left(\exp(-)\left\{a\sum_{i=1}^{3} (1 - \exp\{-b_i (T_i +x)\})*(1- r_i) - a\sum_{i=1}^{3} \{1 - \exp(-b_iT_i)\}*(1- r_i)\right\}\right) \qquad (6)$$

Also it is observed that R(x | T) and λ(T) are strictly decreasing function of T, i.e.,

$$R(x \mid 0) = \left(\exp(-)\left\{a\sum_{i=1}^{3} (1 - \exp\{-b_ix\})*(1- r_i)\right\}\right) \qquad (7)$$

$$\lambda(0) = a\sum_{i=1}^{3} b_i(1- r_i) \quad \text{and } \lambda(\infty) =0 \qquad (8)$$

Therefore the total expected software system cost, E(T) can be defined as: (i) cost to perform testing; (ii) cost incurred in removing errors during the testing phase; and (iii) a risk cost due to software failure.

The cost to perform testing can be defined as

$$E_1(T) = C_1T \qquad (9)$$

The expected total time to remove all N(T) can be expressed using Zhang [8] as:

$$E_2(T) = E\sum_{i=1}^{N(T)} Y_i = E[N(T)]* E[Yi] = m(T)\mu_y \qquad (10)$$

where $\mu_y = \left\{[1 - (\lambda T_0 +1)*\exp\{-\lambda T_0\}]\right\} / \left\{[\lambda(1 - \exp\{-\lambda T_0\}]\right\}$

Also the expected cost to remove all errors detected by time T can be written as:

$$E_2(T) = C_2\, E[\sum_{i=1}^{N(T)} Y_i] = C_2m(T)\mu_y \qquad (11)$$

The risk cost due to software failure after releasing software is $E_3 (T) = C_3 [1- R(x |T)]$, where C_3 is cost due to software failure. Assuming T is to be release time of the software, total cost incurred during SDLC, the expected total software cost can be expressed using Zhang 1998 [8] as follows:

$$E(T) = C_1(T) + C_2\, m(T)\, \mu_y + C_3[1 - R(x|T)] \qquad (12)$$

By substituting the values from eqn. (6), (7) and (8) we get

$$E(T) = C_1(T) + C_2\left(a\sum_{i=1}^{3} [\{1 - \exp(-b_iT_i)\} *(1- r_i)] * [1 - (\lambda T_0 +1)*\exp\{-\lambda T_0\}] / [\lambda(1 - \exp\{-\lambda T_0\}]\right)$$

$$+ C_3\left(1 - \left\{\exp(-)\left\{a\sum_{i=1}^{3} (1 - \exp\{-b_i (T_i +x)\})*(1- r_i) - a\sum_{i=1}^{3} \{1 - \exp(-b_iT_i)\}*(1- r_i)\right\}\right\}\right)$$

(13)

3.2 Optimal Release Policy

Here we discuss the behavior of the software cost model given in eq. (12) and determine optimal release time T* that minimizes the expected software cost of the system subject to attaining a desired reliability level, R_0 the optimization problem can be characterized as follows:

Minimizing E(T) given as in eq.(12) subject to $\{R(x \mid T) \geq R_0\}$

Differentiating eq.(12) with respect to T and equating them to zero we get the optimal testing time T* as follows:

$$\frac{d}{dT}E(T) = C_1 + C_2\left(a\sum_{i=1}^{3}[b_i\exp\{-b_iT_i\}]^*[(1-r_i)]*[1-(\lambda T_0+1)*\exp\{-\lambda T_0\}]/[\lambda(1-\exp\{-\lambda T_0\}]\right)$$
$$+ C_3\left(\left\{\exp(-)\left\{a\sum_{i=1}^{3}(1-\exp\{-b_i(T_i+x)\})*(1-r_i) - a\sum_{i=1}^{3}(1-\exp\{-b_iT_i\})*(1-r_i)\right\}\right\}\right.$$
$$\left.*\left(a\sum_{i=1}^{3}b_i\exp\{-b_iT_i\}*(1-r_i)\left\{\exp\{-b_ix\}-1\right\}\right)\right) = 0 \qquad (14)$$

T* = T_1 can be represented as:

$$C_1 = -\{C_2A + C_3B\} \qquad (15)$$

Where $A = a\sum_{i=1}^{3}[b_i\exp\{-b_iT_i\}]^*[(1-r_i)]*\left\{[1-(\lambda T_0+1)*\exp\{-\lambda T_0\}]/[\lambda(1-\exp\{-\lambda T_0\}]\right\}$

$$B = \left(\left\{\exp(-)\left\{a\sum_{i=1}^{3}(1-\exp\{-b_i(T_i+x)\})*(1-r_i) - a\sum_{i=1}^{3}(1-\exp\{-b_iT_i\})*(1-r_i)\right\}\right\}\right.$$
$$\left.*\left(a\sum_{i=1}^{3}b_i\exp\{-b_iT_i\}*(1-r_i)\left\{\exp\{-b_ix\}-1\right\}\right)\right)$$

The second derivative with respect to T of equation (12) yields:

$$\frac{d^2}{dT^2}\left\{E(T)\right\} = C_2\left(a\sum_{i=1}^{3}[(-)b_i^2\exp\{-b_iT_i\}]^*[(1-r_i)]*[1-(\lambda T_0+1)*\exp\{-\lambda T_0\}]/[\lambda(1-\exp\{-\lambda T_0\}]\right)$$
$$+ C_3\left(\left(\exp(-)\left\{a\sum_{i=1}^{3}(1-\exp\{-b_i(T_i+x)\})*(1-r_i) - a\sum_{i=1}^{3}(1-\exp\{-b_iT_i\})*(1-r_i)\right\}\right)\right.$$
$$\left.*\left(a\sum_{i=1}^{3}b_i\exp\{-b_iT_i\}*(1-r_i)\left\{\exp\{-b_ix\}-1\right\}\right)\right)\left\{a\sum_{i=1}^{3}\left(b_i\exp\{-b_iT_i\}*(1-r_i)\exp\{-b_ix\}-1-b_i\right)\right\} \qquad (16)$$

Let:

$$h(T) = a\sum_{i=1}^{3}\left\{b_i*(1-r_i)*(\exp\{-b_iT_i\})\right\} \quad \text{and } h(T) \geq 0 \ \forall\, T \qquad (17)$$

$$g(T) = C_3\left(\left\{\exp(-)\left\{a\sum_{i=1}^{3}(1-\exp\{-b_i(T_i+x)\})*(1-r_i) - a\sum_{i=1}^{3}(1-\exp\{-b_iT_i\})*(1-r_i)\right\}\right\}\right)$$
$$*\left\{\left(\sum_{i=1}^{3}\left\{\exp\{-b_ix\}-1\right\}^2\right) - \left(\sum_{i=1}^{3}b_i\left\{\exp\{-b_ix\}-1\right\}\right)\right\} \qquad (18)$$

$$v(T) = -C_2\left(\sum_{i=1}^{3}b_i[1-(\lambda T_0+1)*\exp\{-\lambda T_0\}]/[\lambda(1-\exp\{-\lambda T_0\}\right) \qquad (19)$$

We can rewrite eq. (14) by using eq.(15) to eq. (17) as below:

$$\frac{d^2}{dT^2}E(T) = h(T)\left\{v(T) + g(T)\right\} \qquad (20)$$

Using eq. (17) we can see that $\frac{d^2}{}E(T) \geq 0 \mid T = T_1$

dT^2

Where h(T), g(T), v(T), b_i, T_i , x, r_i all are positive values defined in equations (12) to equations (20) and the objective function E(T) can be strictly decreasing, increasing or both in T depending upon the solutions obtained from these equations respectively. Therefore E(T) yields the minimum value at $T^* = T_1$ for the following policies:

Optimum Release Policy 1:

$$T^* = T_1 \text{ when } \lambda(0) \geq \lambda(T_1)$$

Optimum Release Policy 2:

$$T^* = 0 \text{ when } \lambda(0) < \lambda(T_1)$$

Now let T_R denote the optimal testing time satisfying the condition $\{R(x|T) \geq R_0\}$ we can minimize E(T) as follows:

Optimum Release Policy 3:

(a) If $\lambda(0) \geq \lambda(T_1)$ and $R(x|0) < R_0$ then $T^* = \max(T_1, T_R)$
(b) If $\lambda(0) > \lambda(T_1)$ and $R(x|0) \geq R_0$ then $T^* = T_1$
(c) If $\lambda(0) \leq \lambda(T_1)$ and $R(x|0) < R_0$ then $T^* = T_R$
(d) If $\lambda(0) \leq \lambda(T_1)$ and $R(x|0) \geq R_0$ then $T^* = 0$

4. NUMERICAL EXAMPLE

In this section we present a numerical example to illustrate the determination of optimal release policies of proposed model. Testing data has been collected from Misra [5] summarizing the number of failures per one-hour interval of execution time. We have applied this data to proposed model for fitting the data using MATLAB version 7.0.1 under Windows XP environment, assuming that testing staff are working for 10 hrs per day and five days a week.

Model Name	**Mean Value Function m (T)**	**SSE**
Goel-Okumoto [14]	$m(T) = a\,(1 - \exp\{-bT\})$	766.1
Yamada-Ohba [15]	$m(T) = a\left\{1 - (1 + bT)\exp\{-bT\}\right\}$	592.1
Proposed Model	$m(T) = a\sum_{i=1}^{3}\left\{(1 - \exp\{-b_i T_i\}) * (1 - r_i)\right\}$	241.7

TABLE 1: Comparison of the models

The criteria used for determining the goodness of fit is the Sum of Square Error (SSE). This statistic measures the deviation of the responses from the fitted values of the responses. A value closer to 0 indicates that the model has a smaller random error component and the fit will be more useful for prediction. The model that produces the smallest SSE has the better performance and can be expressed as follows:

$$SSE = \left\{\text{Sum}_{\{i=1 \text{ to } 25\}} [y_i - f_i]^2\right\} \quad (21)$$

where y_i is the observed value and f_i is the predicted value from the fit. From Table 1 we observe that the proposed methodology fit the data to a greater extent than the other two models. Therefore we apply the proposed model to fit the data and in the determination of software release instant.

4.1 The impact of cost coefficients on the expected total cost

The impact of cost coefficients C_1, C_2, and C_3 on expected total cost has been evaluated under different conditions. We increase the values of C_1, C_2, C_3 and keep the values of other parameters unchanged without lack of generality. The parameters of present model are estimated by using maximum likelihood estimation (MLE) method and other related parameters are as follows: Expected total potential error $\hat{a}$ = 143.21, b1=0.8736, b2=0.6094, b3=0.1942, r1=0.7536, r2=0.5104, r3=0.0272 and mean value function m (T) = 0.4248.

Case I:

C_1=\$50/day, C_2=\$100/day, C_3=\$150/day, μ_y =0.1 and x=0.05

Case II:

C_1=\$150/day, C_2=\$100/day, C_3=\$50/day, μ_y =0.1 and x=0.05

Case III:

C_1=\$50/day, C_2=\$100/day, C_3=\$200/day, μ_y =0.2 and x=0.05

4.2 Observations

- From the results for three different cases mentioned above we observed that increasing the cost factor C_1 and C_3 results in increasing total expected cost initially very high but then decreases gradually, which has a significant impact on optimal release policies of software product. In other words if developers don't spend sufficient amount of time for testing before release then it is going to be more risky and unreliable for the customer for obvious reason because to remove an error after delivery require more effort and involve more risk which results in a longer testing time.
- We also observe that even if we consider cost factor C_2 as constant in all three cases, increase in cost factor C_1 still increases cost factor C_3 but then expected time to remove error μ_y becomes double, which is quite encouraging reasonably. We summarize the result of total expected cost of software, expected number of errors to be detected by time T with reliability objective keeping more than 90%.
- Based on the calculations for case I, we find the total expected cost E_1 (T*)=\$382.70 and reliability of the software application at the end of testing on 4th day is 0.9127 that is more than 90%. After changing the cost parameters, we get total expected cost E_2 (T*)=\$661.39 for the reliability assessment at 0.9018 (> 90%) marginally low as in case II.
- Finally in case III, we achieve the reliability level of 0.9239 (> 92%) at the cost of E_3(T*)=\$509.28 after improving the software continuously in operational phase and which is very satisfactory. The summary of results shown in Table 2 and figure 2 to figure 5 show the variation of the total expected testing cost, the reliability achieved at the end of testing phase and the expected number of errors detected at release instant.
- Furthermore the validity of proposed assessment method heavily depends upon the representation of software reliability failure data available through various sources is highly fluctuating and not being updated frequently by the community of researchers which is out of the scope of this paper and need to be addressed separately in near future.

Release time T	Expected total cost E_1 (T) Case I	Expected total cost E_2 (T) Case II	Expected total cost E_3 (T) Case III	Expected no. of errors to be detected m(T)	Conditional Reliability R (x|T)
1	587.24	718.26	1155.50	58.38	0.9002
2	457.21	661.39*	818.59	36.35	0.9018
3	399.48	694.61	644.09	24.22	0.9112
4	382.70*	774.97	557.67	17.11	0.9127
5	388.88	880.73	519.61	12.67	0.9185
6	408.44	1000.84	509.28*	9.70	0.9239
7	436.25	1129.52	515.77	7.62	0.9327
8	469.44	1263.65	533.09	6.08	0.9421
9	506.35	1401.46	557.81	4.90	0.9510
10	545.98	1541.87	587.84	3.98	0.9589
11	587.63	1684.21	621.84	3.25	0.9657
12	630.87	1828.03	658.90	2.66	0.9715
13	675.36	1973.00	698.37	2.18	0.9764
14	720.85	2118.90	739.76	1.79	0.9805
15	767.16	2265.54	782.70	1.47	0.9839
16	814.12	2412.79	826.91	1.21	0.9867
17	861.62	2560.53	872.15	1.00	0.9890
18	909.57	2708.67	918.23	0.82	0.9910
19	957.88	2857.13	965.01	0.68	0.9925
20	1006.49	3005.87	1012.36	0.56	0.9939

TABLE 2: Summary of total expected cost E(T), R(x|T) and m(T)

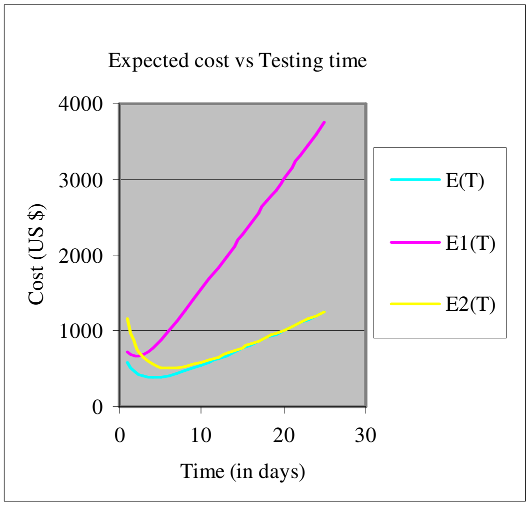

FIGURE 2: Release Instant and Total Expected Cost for Different Cost Factors

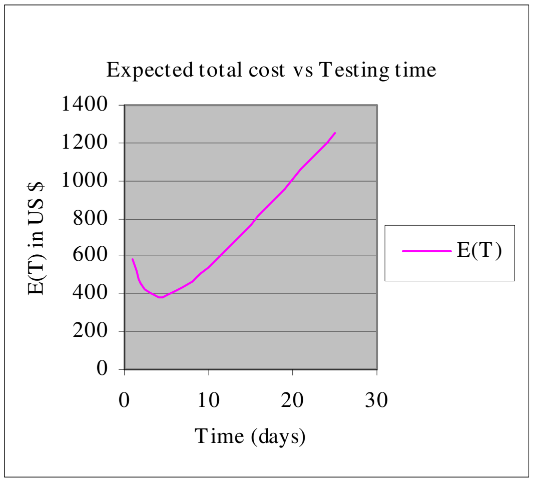

FIGURE 3: Expected Cost During Testing Phase

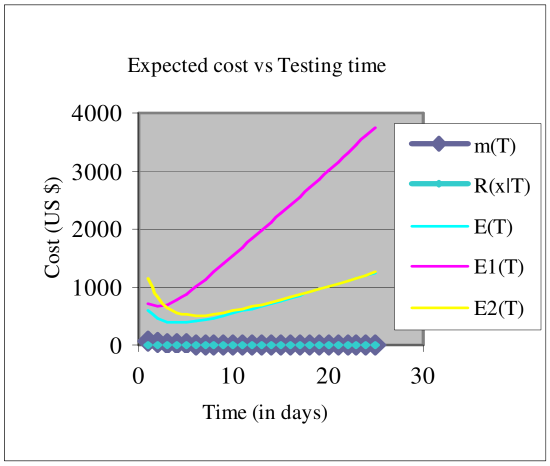

FIGURE 4: Expected Cost with Reliability Achieved During Testing Phase

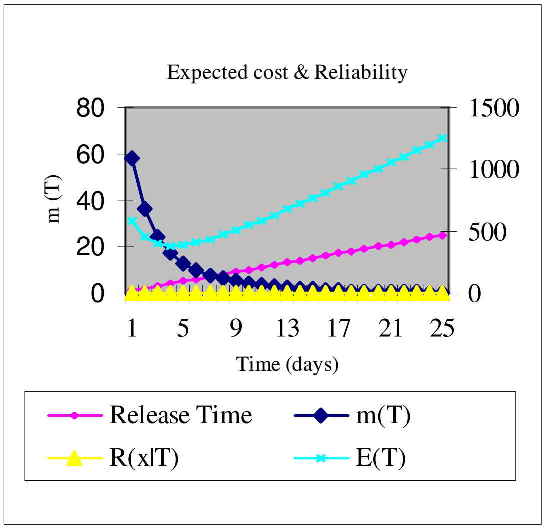

FIGURE 5: Cost and Reliability with Mean Value Function m (T) and Release Instant

5. CONCLUSION & FUTURE WORK

Practically project managers need to know when testing the software can be stopped so that they can deliver the product to customers attaining the requirement of software quality and minimize the related testing costs. In this paper we have formulated a release policy for software reliability growth model under three-tier client server architecture reflecting the cost of postponing software release based on testing efforts. With the help of proposed cost model and designed release policies it can be determined that whether more testing is required or the software has been tested sufficiently to allow its release to the customer for operational use. The results revealed that proposed model not only has a goodness-of-fit but also offers a good explanation of the process of software reliability growth. However a study of comparative analysis to evaluate the effectiveness of the proposed model and other existing software failure models would supplement the present technique by applying more failure data sets of various standard real life projects.

In near future the proposed model can be extended by considering the change-point problem and by introducing extended warranty period. The change-point problem results when some factors of testing process are changed which subsequently can cause the software failure intensity function to decrease or increase. Whereas by extending warranty period the penalty cost may be reduced up to a certain level provided the maintenance cost during operational phase is paid by the customer partially.

Acknowledgement

Authors would like to thank the editor and referees for their useful suggestions and valuable comments.

REFERENCES

[1] K. K. Aggarwal and Yogesh Singh, “Determination of software release instant using a nonhomogeneous error detection rate model”. Microelectron Reliability, Vol. 33. No. 6. pp. 803-807, 1993.

[2] K. K. Aggarwal and Yogesh Singh, “Software Engineering: Programs, Documentation & Operating Procedures”, New Age International Publishers, third edition, pp. 191-324 (2008).

[3] Yogesh Singh and Pradeep Kumar, “A software reliability growth model for three-tier client-server system”. IJCA, Vol. 1. No. 13. doi. 10.5120/289-451,2010.

[4] Hoang Pham, “System Software Reliability”, Springer Series in Reliability Engineering, pp. 315-344 (2006).

[5] Misra, P.N. “Software reliability analysis models”. IBM Systems Journal (1983), 22,262-70.

[6] www.dacs.org “Software Life Cycle Empirical/Experience Database (SLED) published by Data & Analysis Center for Software (DACS)”.

[7] Kuei-Chen Chiu, Yeu-Shiang Huang, Tzai-Zang Lee, “A study of software reliability growth from the perspective of learning effects”. Reliability Engineering and System Safety 93 (2008) 1410-1421.

[8] Xuemei Zhang and Hoang Pham, “A software cost model with warranty cost, error removal times and risk costs”. IIE Transactions (1998) 30, 1135-1142.

[9] Hoang Pham, “Software reliability and cost models: perspectives, comparison, and practice”. European Journal of Operational Research 149 (2003) 475-489.

[10] Chin-Yu Huang, “Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency”. The Journal of Systems and Software 77 (2005) 139-155.

[11] Chu-Ti Lin, Chin-Yu Huang, “Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models”. The Journal of Systems and Software 81 (2008) 1025-1038.

[12] Chin-Yu Huang and Sy-Yen Kuo, “Analysis of incorporating logistic testing-effort function into software reliability modeling”. IEEE Transactions on Reliability, Vol.51, No.3, September 2002.

[13] Yamada S., Othera S and Narihisa H. “Software reliability growth models with testing effort”. IEEE Transactions on Reliability 1986; 35,pp. 19-23.

[14] Goel AL, Okumoto K. “Time-dependent fault detection rate model for software and other performance measures”. IEEE Transactions on Reliability 1979; 28:206-11.

[15] Yamada S. Ohba M. “S-shaped software reliability modeling for software error detection”. IEEE Transactions on Reliability 1983; 32:475-84.

[16] Yamada S., Narihisa H. and Osaki S. “Optimum release policies for a software system with a scheduled software delivery time”. Int. J. System Science 1984, 15, pp. 905-914.

[17] Yamada S., Narihisa H. and Osaki S. “Optimum software release policies with simultaneous cost and reliability requirements”. European Journal of Operation Research 1987, 31, pp. 46-51.

[18] Chin-Yu Huang, Sy-Yen Kuo, Michel R. Lyu, “An assessment of testing-effort dependent software reliability growth model”. IEEE Transactions on Reliability, Vol. 56,No.2, June 2007.

[19] P K Kapur, R B Garg, S K Kumar, “Contributions to Hardware & Software Reliability” World Scientific, pp. 89-147 (1999).

[20] Kapur PK, Bhalla VK. “Optimal release policies for a flexible software reliability growth model”. Reliability Engineering and System Safety 1992; 35:49-54.

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

[22] Pham H. Zhang X. “A software cost model with warranty and risk costs’. IEEE Transaction on Computers 1999; 48:71-75.

[23] Chin-Yu-Huang, Sy-Yen-Kuo and Michael R. Lyu, ‘Optimum software release policy based on cost and reliability with testing efficiency”. IEEE 1999.

[24] Pham, H. and Zhang, X. “A software cost model with error removal times and risk costs”. International Journal of Systems Science (1998), 29, 435-442.

[25] Shinji Inoue and Shigeru Yamada, “Optimal software release policy with change point”. IEEE 978-1-4244-2630-0/08, 2008.