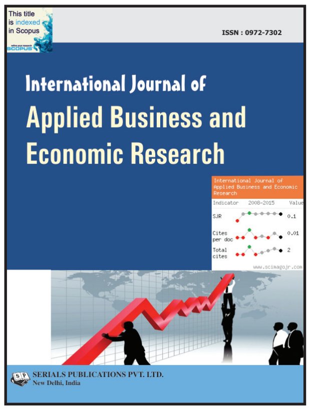

International Journal of Applied Business and Economic Research

ISSN : 0972-7302

available at http: www.serialsjournal.com

© **Serials Publications Pvt. Ltd.**

Volume 15 • Number 14 • 2017

A Comparative Study on the Cost of Software Development Model Based on Inverse Exponential Distribution

Hee-Cheul Kim[1]

[1]*Corresponding author, Associate Professor, Department of Industrial and Management Engineering, Namseoul University, Korea*

ABSTRACT

In finite failure NHPP for failure analysis of the software, the fault occurrence rates may have constant or monotonically increasing or monotonically decreasing pattern. In this study, in the process of testing software products, inverse exponential distribution as a failure distribution was studied in the cost of software reliability model. Software failure model was used NHPP and the parameter estimation was applied maximum likelihood estimation. In this research, the software charge model considering inverse exponential distribution, by applying a software failure time data, was analyzed. Through this study, software operators are considered to be helpful to identify the software development costs.

Keywords: Software Cost Model, NHPP, Inverse Exponential Distribution, Finite Failure Model, Laplace Trend Test.

1. INTRODUCTION

Software reliability is a likelihood that can be operated lacking a malfunction born over a sector of time at conservation settings. Therefore, software reliability is an important factor affecting system reliability and in terms of hardware reliability, design attributes have different properties. Therefore, it may cause enormous losses outstanding to a failure of the computer organization, software breakdown in our society. Therefore, software reliability in the software growth progression is a major underlying problem. This environment needs to analyze the operator software requirements. So as to efficiently achieve the costs in terms of software difficult, a cost and reliability of the testing of the software is the efficient development process if the trend was estimated in advance. Therefore, reliability, cost, and software considering the estimated time for the release time, the development process are indispensable process.

Therefore, in terms of the error search process, the software reliability model (Gokhale & Trivedi, 1999; Goel & Okumoto, 1979) using the Non-Homogeneous Poisson Process (NHPP) was regard as excellent model and if a new fault occurs, it is immediately was removed and have the presumption that new defects does not occurred from the debugging process. Huang, (2005) was presented the effective integration of software reliability prediction technology.

In another aspect, the S-shape model can be analyzed from the knowledge process in the software manager may be utilized in software failure detection tool (Kuei-Chen, Yeu-Shiang & Tzai-Zang, 2008).

In this working out, software failure time data was applied to NHPP life distribution in the reliability model analysis to compare the cost of a software reliability model considering the inverse exponential distribution.

2. NHPP SOFTWARE RELIABILITY

In a non-homogeneous Poisson process (NHPP) development (Kuo & Yang, 1996; Yoo, 2015), the mean assessment property $m(t)$ and power property $\lambda(t)$ were known to following relationship.

$$m(t) = \int_0^t \lambda(\omega)\, d\omega, \quad \frac{dm(t)}{dt} = \lambda(t) \tag{1}$$

The $N(t)$ was achieved using Probability Density Function (PDF) (Kim, 2015) of Poisson distribution using the factor $m(t)$.

$$p(N(t) = n) = \frac{[m(t)]^n}{n!} e^{-m(t)}, \quad n = 0, 1, ..., \infty. \tag{2}$$

Therefore, let θ represents the initial number of faults that can be noticed assumed limited failure NHPP models, the mean value meaning $m(t)$ and intensity meaning $\lambda(t)$ were known to following situation (Yoo, 2015; Kim, 2015)

$$m(t) = \theta F(t), \lambda(t) = \theta\, F'(t) = \theta\, f(t) \tag{3}$$

From Equation (3), $f(t)$ denotes PDF (Probability Density Function) and $F(t)$ is CDF (Cumulative Distribution Function).

The interval failure time $\{t_i, i = 1, 2, ...\}$ indicate the times among successive software failure. Consequently, the failure last time x_n denotes following relationship.

$$xn = \sum_{i=1}^{n} t_i \; (i = 1, 2, ..., n; \quad 0 \le x_1 \le x_2 \le ... \le x_n) \tag{4}$$

The probability property of $x_1, x_2, ..., x_n$ was stated to next equation (Yoo, 2015; Kim, 2015; Kuo and Yang, 1996).

$$f_{X_1, X_2, ..., X_n}(x_1, x_2, ..., x_n) = L(\Theta | \underline{x}) = e^{-m(x_n)} \prod_{i=1}^{n} \lambda(x_i) \tag{5}$$

In these situations, the reliability for restricted process $\hat{R}(\xi | x_n)$ can be expressed next estimating equation using the work time (or mission time) ζ.

$$\hat{R}(\xi | x_n) = e^{-\int_{x_n}^{x_n+\xi} \lambda(t)\, dt} = \exp[-\{m(\xi + x_n) - m(x_n)\}] \tag{6}$$

3. SOFTWARE DEVELOPMENT COST MODEL

The estimated whole charge of software development was achieved as next special features (Kim, 2015; Zhang & Wu, 2012).

$$\mathrm{E} = \mathrm{E}_1 + \mathrm{E}_2 + \mathrm{E}_3 + \mathrm{E}_4 = \mathrm{E}_1 + \mathrm{C}_2 \times t + \mathrm{C}_3 \times m(t) + \mathrm{C}_4 \times [m(t+t') - m(t)] \tag{7}$$

The based on equation (7), E mean estimated total cost of software development. Also, E_1 denotes initial software plan and software growth costs (data to analyze, the number of software development professionals, CPU time, etc.). The software testing costs per unit time (constant) is E_2. $\mathrm{E}_2 = \mathrm{C}_2 \times t$, using C_2 (means cost per unit time) and t(represents the failure period). Also, E_3 denotes cost of the removal of a defect of the activity, such as to detect the fault and remove the defective. $\mathrm{E}_3 = \mathrm{C}_3 \times m(t)$, using C_3 (represents the price of the eliminating a error in the difficult stage) and $m(t)$ (indicates the predictable amount of errors noticed at the period t). And E_4 denotes cost of the fixative failure, in the working stage from similarly connected to the software dependability forming; it must satisfy $\mathrm{E}_4 = \mathrm{C}_4 \times [m(t+t') - m(t)]$ in this situation. Thus C_4 represents the price of the setting a fault which is detected by operators in the software working stage after the software announcement and t' represents the time of functioning and keeping the software after releasing the software organization. From special feature, a price of C_4 is abundant larger than payment of C_2 & C_3.

In decision, optimum software release period t was expressed next estimating equation.

$$\mathrm{E}' = (\mathrm{E}_1 + \mathrm{E}_2 + \mathrm{E}_3 + \mathrm{E}_4)' = \mathrm{C}_2 + \mathrm{C}_3 \times m'(t) + \mathrm{C}_4 \times [m'(t+t') - m'(t)] = 0 \tag{8}$$

4. THE PROPOSED INVERSE EXPONENTIAL FAILURE NHPP MODEL

The inverse Weibull distribution (Kersey, 2010) can be widely applied in the medical field ecology and reliability and distribution meaning of this distribution is recognized, as next form.

$$\mathrm{F}(t) = \exp[-(\beta t)^{-\gamma}] \tag{9}$$

The case of the shape parameter (γ) is 1 from inverse Weibull distribution was known to inverse exponential distribution. Thus, the distribution function and probability density function for the inverse exponential distribution is as follows.

$$\mathrm{F}(t) = \exp[-(\beta t)^{-1}],\ f(t) = F'(t) = \beta^{-1} t^{-2} \exp[-(\beta t)^{-1}] \tag{10}$$

Therefore, using an equation (3) for the finite failure NHPP model of the inverse exponential distribution function, the mean value function and intensity function were represented using next following situation.

$$m_{Ied}(t) = \theta\,\mathrm{F}(t) = \theta \exp[-(\beta t)^{-1}],\ \lambda_{Ied}(t) = \theta\, f(t) = \theta\beta^{-1} t^{-2} \exp[-(\beta t)^{-1}] \tag{11}$$

The likelihood function from equation (11) can be represented as next forms.

$$\mathrm{L}_{\mathrm{NHPP}}(\Theta|\underline{x}) = \left[\prod_{i=1}^{n} \theta\beta^{-1} x_i^{-2} \exp[-(\beta x_i)^{-1}]\right]\left(\theta \exp[-(\beta x_n)^{-1}]\right) \tag{12}$$

where, $\underline{x} = (x_1 \le x_2 \le x_3, ..., \le x_n)$ and Θ means parameter space.

So as to the parameter estimation, the log likelihood form can be represented as next situation in relation to equation (12).

$$\ln \mathrm{L}_{\mathrm{NHPP}}(\Theta|\underline{x}) = n \ln \theta - n \ln \beta - 2\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} (\beta x_i)^{-1} - \theta\, e^{-(\beta x_n)^{-1}} \tag{13}$$

Through equation (13), the estimation for $\hat{\theta}_{\mathrm{MLE}}$ and $\hat{\beta}_{\mathrm{MLE}}$ must be gratified the following circumstances.

$$\frac{\partial \ln \mathrm{L}_{\mathrm{NHPP}}(\Theta|\underline{x})}{\partial \theta} = \frac{n}{\theta} - e^{-(\beta x_n)^{-1}} = 0 \tag{14}$$

$$\frac{\partial \ln \mathrm{L}_{\mathrm{NHPP}}(\Theta|\underline{x})}{\partial \beta} = -\frac{n}{\beta} + \frac{1}{\beta^2}\sum_{i=1}^{n} \frac{1}{x_i} - \theta \frac{1}{\beta^2 x_n} e^{-(\beta x_n)^{-1}} = 0 \tag{15}$$

5. SOFTWARE FAILURE TIME AND DEVELOPMENT COST ANALYSIS

In this section, using a software failure time data, it should be evaluated the reliability of software cost model. Table 1 is information of the software failure time (Hayakawa & Telfar, 2000).

In addition, the trend analysis should be followed in order to ensure the reliability test on data (Kanoun, Laprie & Lyu, 1996; Kim, 2016). In this study, the trend analysis was performed using the Laplace trend test. Figure 1.1 is an outcome of Laplace trend test. In this figure, since estimation value of Laplace factors has the nature from 2 and −2, the extreme value does not exist. Thus, the estimation of the reliability using this data may be possible. The maximum likelihood estimation so as to the parameter approximation was applied. So as to enable the parameter estimation, in this research, a mathematical translation data (Failure time (hours) × 0.1) was used and calculation method of nonlinear equations was used bisection method.

Table 1.1
Data of software failure information

Failure Number	*Failure Time (Hours)*	*Failure Number*	*Failure Time (Hours)*
1	0.479	16	10.771
2	0.745	17	10.906
3	1.022	18	11.183
4	1.576	19	11.779
5	2.610	20	12.536
6	3.559	21	12.973
7	4.252	22	15.203
8	4.849	23	15.640
9	4.966	24	15.980
10	5.136	25	16.385
11	5.253	26	16.960
12	6.527	27	17.237
13	6.996	28	17.600
14	8.170	29	18.122
15	8.863	30	18.735

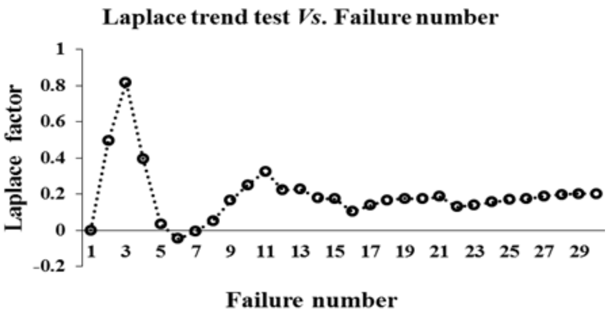

Figure 1.1: Result for test of Laplace Trend

These calculations were given the initial value of 0.01 and 3.50 and tolerance width of interval (10^{-5}), iterations of 100 times were performed for checking adequate point estimation value using C-language. The results of the maximum likelihood estimation were estimated $\hat{\theta}_{MLE} = 41.2881$ and $\hat{\beta}_{MLE} = 1.6922$.

In this study, the following cost curve form attempts to analyze using equation (5) and next assumptions.

(i) (Assumption) A, $E_1 = 10\$, C_2 = 0.5\$, C_3 = 0.8\$, C_4 = 0.9\$, t' = 3.5$

(ii) (Assumption) B, $E_1 = 10\$, C_2 = 0.5\$, C_3 = 0.8\$, C_4 = 1.0\$, t' = 3.5$

(iii) (Assumption) C, $E_1 = 10\$, C_2 = 0.5\$, C_3 = 0.8\$, C_4 = 0.9\$, t' = 4.0$

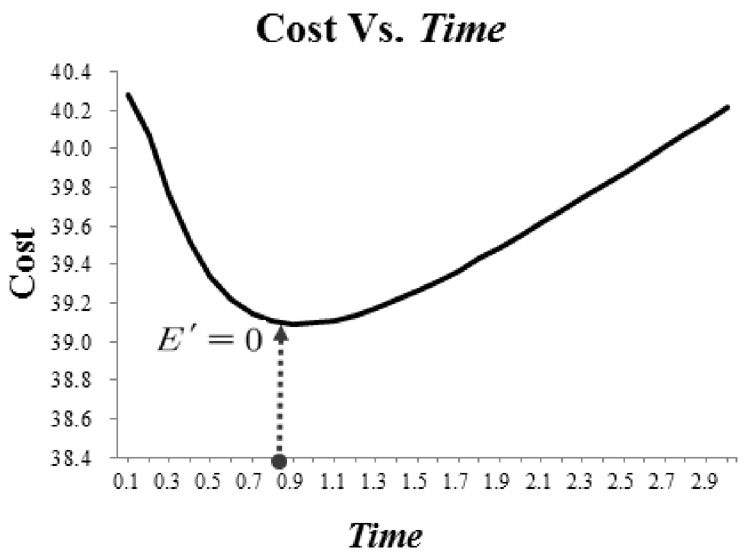

Figure 1.2: The cost curve underneath the situation of (Assumption) A

The cost curve of the model presented that in the early stages while showing a tendency to decrease gradually increase after a certain time to failure as shown in Figure 1.2. The reason is that the amount of defects remaining in the software structure has been reduced more and more in the course of removal of the defectives. Thus, the observed probability from remaining defects is lowered.

In the initial step of difficult, there are unmoving numerous errors in software which are simply noticed. The cost to remove an error in the step is far lesser than that of eliminating a error in the procedure period. Accordingly, the price of the software decreases throughout the procedure of faults correcting. But from the later step the amount of remaining faults in software is previously less, and in this testing stage the time of noticing a fault is very long and the cost of removing a fault become advanced than that in the task step, therefore the cost curve increases continuously with time. From the tendency of the cost curve, can be predictable the optimal software release period and it is also the most faithful phenomenon. The most circumstances are reliable in a progression from the real software development (Zhang & Wu, 2012).

In compared to the (Assumption) A, other assumptions are the same and the case of from $C_4 = 0.9\$$ to $C_4 = 1.0\$$, (Assumption) B is shown in the Figure 1.3.

When comparing curves from (Assumption) A and (Assumption) B, in the software operation steps after the software is released, realized that the optimal software release period delay. Therefore, in this case, it should be removed for possible faults in the testing phase, than the operating phase so as to reduce the defects in time after the software is released.

Also, in compared to the (Assumption) A, other assumptions are the same and the case of from $t' = 3.5$ to $t' = 4.0$, (Assumption) C is shown in the Figure 1.3.

When comparing curves from (Assumption) A and (Assumption) C, after launching the software system, Time to operate and maintain the software was increased, thus, the software also can be seen that the optimum release period postponement. Also in this case, it should be removed for possible faults in the testing phase, than the operating phase so as to reduce the defects in time after the software is released.

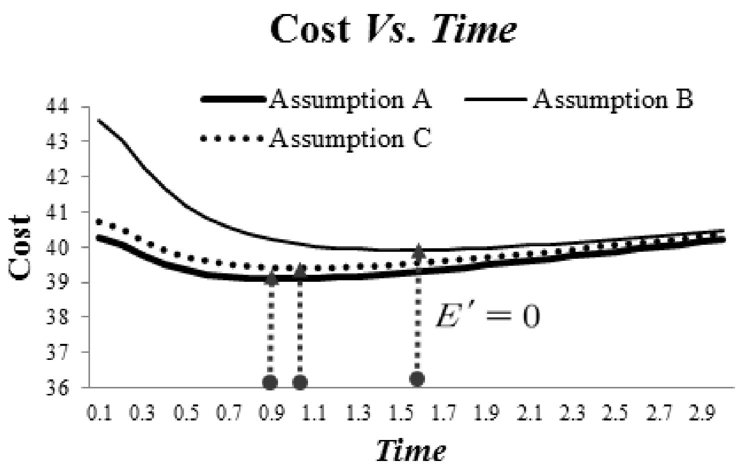

Figure 1.3: Comparative of the cost curve underneath the situation (Assumption)

6. CONCLUSIONS

In this research, the cost of the optimal release period can be predicted (Almering, Genuchten & Cloudt & Sonnemans, 2007). Therefore, more efficient models to reduce the cost of testing, it should ensure that profits can increase the release software. Software cost model using finite failure NHPP model based on an inverse exponential distribution used in this study is predicted to be the best software release time more efficiently. The total number of defect for the proposed model is detected while maintaining the software after the software release period and operational software and these defects are the software assumes that the user and all of it are not found. It's a real step in the detection and removal errors repair costs are lesser than the cost of removing altogether the errors that remain in the operating phase, this cost with the increase of the operating time can be seen increasing pattern.

Therefore, the optimal software release time can realistically anticipate preview, large software was modified and in the process of changing this situation can scarcely avoid the occurrence of defects it is reality.

The optimal release period can be situation satisfying the required reliability and minimize the total cost. Through this research, software developers is thought you have a somewhat help identify the optimal release time and economic development costs.

Acknowledgment

Funding for this paper was provided by Namseoul University.

References

Almering, V., Genuchten M.V., & Cloudt, G., & Sonnemans, P.J.M. (2007), Software Reliability Growth Models in Practice. *IEEE SOFTWARE*, 82-88.

Goel, A.L., & Okumoto, K. (1979), Time Dependent Error Detection Rate Model for software Reliability and other performance measure. *IEEE Trans. Reliability, R-28(3),* 206-211.

Gokhale, S.S., & Trivedi, K.S. (1999), A time/structure based software reliability model. *Annals of Software Engineering*, *8*, 5-12.

Hayakawa, Y., & Telfar, G. (2000), Mixed PoissonType Processes with Application in Software Reliability. *Mathematical and Computer Modeling, 31*, 151-156.

Huang, C.Y. (2005), Performance analysis of software reliability growth models with testing-effort and change-point. *J. Syst. Software*, *76*, 181-194.

Kanoun, K., Laprie, J.C., & Lyu, M.R. (1996), Handbook of software reliability engineering, Chapter Trend Analysis. *McGraw-Hill, New York*, 401-137.

Kersey, J.X. (2010), Weighted Inverse Weibull and Beta-Inverse Weibull Distribution. *Electronic Theses & Dissertations Paper 661*, 1-53.

Kim, H.C. (2015), The Property of Learning effect based on Delayed Software S-Shaped Reliability Model using Finite NHPP Software Cost Model. *Indian Journal of Science and Technology, 8(34),* 1-7.

Kim, H.C. (2016), A Performance Analysis of Software Reliability Model using Lomax and Gompertz Distribution Property. *Indian Journal of Science and Technology, 9(20),* 1-6.

Kuei-Chen, C., Yeu-Shiang, H., & Tzai-Zang, L. (2008), A study of software reliability growth from the perspective of learning effects. *Reliability Engineering and System Safety*, *93*, 1410-1421.

Kuo, L., & Yang, T.Y. (1996), Bayesian Computation of Software Reliability. *Journal of the American Statistical Association, 91*, 763-773.

Yoo, T.H. (2015), The Infinite NHPP software reliability model based on Monotonic Intensity Function. *Indian Journal of Science and Technology, 8(14),* 1-7.

Zhang, Y., & Wu, K. (2012) Software Cost Model Considering Reliability and Time of Software in Use. *Journal of Convergence Information Technology, 7(13),* 135-142.