Journal of Engineering and Applied Sciences 14 (23): 8779-8785, 2019
ISSN: 1816-949X

The Analysis and Prediction of Software Development Cost Based on NHPP Software Reliability Model

Tae-Jin Yang
Department of Electronic Engineering, Namseoul University, Cheonan, Republic of Korea

Abstract: It is very important to find the optimal software development cost along with release timeduring the software development process. Therefore, in this study, we analyze and predict software development cost and release time using NHPP reliability models. As a result first, the cost curves of all models proposed in this study decrease at the initial stage but gradually increase at the latter stage when the release time passes. Second, prior to software release, if the cost of testing per unit time and the cost of removing one defect found during the testing process increased, the development cost also increased but the release time remained unchanged. Third, after the software release, if the cost of correcting the faults found by the operator increases at the operating stage, the development cost is reduced but the release time is delayed. Fourth as a result of comparing the models applied in this study, Lindley Model is the best model because it has high reliability for mission time in the future, low cost of software development and quick release time. Through this study, we were able to present software developers and operators with the necessary prior information to predict the economic software development cost and the optimal software release time.

Key words: Erlang life time distribution, failure time data, Lindley distribution, NHPP reliability model, software development cost, software release time

INTRODUCTION

With the rapid development of information technology, the necessity of computer software development is also increasing. If we can predict the optimal software development cost and release time, we will be able to perform efficiently in the software development process.

Therefore, based on software reliability, software development process considering software development cost and release time is very important issue. To date, many software reliability models have been proposed. In particular, a Non-Homogeneous Poisson Process (NHPP) reliability model using an intensity function and an average value function has been proposed in order to estimate software reliability attributes within a given software failure time (Song *et al*., 2017). Yamada and Osaki (1985) suggested that the mean value function can be predicted using the maximum likelihood method and Huang (2005) presented the confidence interval of the mean value function. Pham and Zhang (2003) proposed a stability model to evaluate software stability. Recently, Yang (2016) analyzed the software development cost model, according to the change of shape parameter of Erlang life distribution and Lomax distribution. However, most of the existing studies have suggested efficient prediction methods based on reliability models or most of the cost studies have been based on changes in shape parameters of single models. In particular, there is no study on the analysis and prediction of software development costs based on reliability models (Song *et al*., 2017; Yang, 2017).

Therefore, in this study, we will analyze and predict the software development cost using NHPP reliability models such as Goel-Okumoto basic model, Erlang Model, and Lindley Model.

Literature review

NHPP Goel-Okumoto basic model: The Goel and Okumoto (1979) model is the most well-known basic model in the field of software reliability. Let f(t) and F(t) for the Goel-Okumoto Model be a probability density function and a cumulative density function, respectively. Assuming that the expected value of the defect that can be observed up to the observation point [0, t] is •, the finite fault strength function •(t) and the mean value function m(t) are as follows (Yamada and Osaki, 1985).

$$\lambda(t|\theta, b) = \theta f(t) = \theta b e^{-bt} \quad (1)$$

$$m(t|\theta, b) = \theta F(t) = \theta (1-e^{-bt}) \quad (2)$$

Note that •>, b>0. The likelihood function of the finite-fault NHPP model using the Eq. 1 and 2 is derived as follows:

$$L(\theta,b|\underline{x}) = \left(\prod_{i=1}^{n}\theta b e^{-b x_i}\right)\exp\left[-\theta\left(1-e^{-b x_n}\right)\right] \quad (3)$$

Note that $\underline{x}=(0\le x_1\le x_2\le,...,\le x_n)$. The likelihood function using the Eq. 3 is simplified to the following log conditional expression.

$$\ln L_{NHPP}\left(\Theta|\underline{x}\right)= n\ln\theta+ n\ln b-b\sum_{k=1}^{n}x_k-\theta\left(1-e^{-bx_n}\right) \quad (4)$$

When Eq. 4 is partially differentiated with respect to • and b, the result is as follows:

$$\frac{n}{\theta}=1-\exp(-\hat{b}x_n) \quad (5)$$

$$\frac{n}{b}=\sum_{i=1}^{n}x_n+\hat{\theta}x_n\exp\left(-\hat{b}x_n\right) \quad (6)$$

Therefore, the maximum likelihood estimator $\hat{\theta}_{MLE}$ and $\hat{b}_{MLE}$ satisfying the following the Eq. 5 and 6 can be estimated by a numerical method.

MATERIALS AND METHODS

NHPP Erlang Model: The Erlang distribution is the life distribution of the gamma family widely used in the software reliability. The probability density function and the cumulative density function considering the shape parameter (a) and the scale parameter (b) are as follows:

$$f(t)=\frac{b^a}{\Gamma(a)}t^{a-1}e^{-bt} \quad (7)$$

$$F(t)=\left(1-e^{-bt}\sum_{i=0}^{a-1}\frac{(bt)^i}{i!}\right) \quad (8)$$

Note that, b>0, a = 1, 2, 3, ..., t•[0, •]. The log-likelihood function of the finite-fault NHPP modelby using the Eq. 7 and 8 is derived as follows.

$$\begin{aligned}\ln L_{NHPP}(\Theta|\underline{x})&= n\ln\theta-n\ln\Gamma(a)+na\ln b+(a-1)\\&\sum_{i=1}^{n}\ln x_i-b\sum_{i=1}^{n}x_i-\theta+\theta e^{-bx_n}\left(\sum_{i=0}^{a-1}\frac{(bx_n)^i}{i!}\right)\end{aligned} \quad (9)$$

Note that $\underline{x}=(0\le x_1\le x_2\le,...,\le x_n),\Theta$ is parameter.

When the Eq. 9 is partially differentiated with respect to • and b, the result is as follows:

$$\frac{\partial \ln L_{NHPP}(\Theta|x)}{\partial\theta}=\frac{n}{\theta}-1+e^{-b x_n}\left(\sum_{i=0}^{a-1}\frac{(bx_n)^i}{i!}\right)=0 \quad (10)$$

$$\begin{aligned}&\frac{\partial \ln L_{NHPP}(\Theta|x)}{\partial b}=\frac{an}{b}-\sum_{i=1}^{n}x_i+\\&\frac{\partial[\theta e^{-bx_n}\left(\sum_{i=0}^{a-1}\frac{(bx_n)^i}{i!}\right)}{\partial b}=0\end{aligned} \quad (11)$$

In this study, we consider a case where the shape parameter (a) which means the type of software failure distribution is 2.

$$\frac{\partial \ln L_{NHPP}(\Theta|\underline{x})}{\partial\theta}=\frac{n}{\theta}-1+e^{-bx_n}(1+bx_n)=0 \quad (12)$$

$$\frac{\partial \ln L_{NHPP}(\Theta|\underline{x})}{\partial b}=\frac{2n}{b}-\theta b x_n^2 e^{-bx_n}-\sum_{i=1}^{n}x_i=0 \quad (13)$$

Therefore, the maximum likelihood estimator $\hat{\theta}_{MLE}$ and $\hat{b}_{MLE}$ satisfying the following the Eq. 12 and 13 can be estimated by a numerical method.

NHPP Lindley Model: The Lindley distribution is a mixture type of exponential distributions and gamma distributions. Let f(t) and F(t) for the Lindely Model be a probability density function and a cumulative density function, respectively. The probability density function and the cumulative density function are as follows.

$$f(t)=\frac{b^2}{b+1}(1+t)e^{-bt} \quad (14)$$

$$F(t)=1-\frac{b+1+bt}{b+1}e^{-bt} \quad (15)$$

Note that b>0, t>0. Ghitany Adhikari and Srivastava explains the characteristics of the Lindley distribution and confirmed that it is a more efficient model than the exponential distribution. Assuming that the expected value of the defect that can be observed up to the observation point [0, t] is •, the finite fault strength function •(t) and the mean value function m(t) are as follows:

$$\lambda(t|\theta,b)=\theta f(t)=\theta\left(\frac{b^2}{b+1}(1+t)e^{-bt}\right) \quad (16)$$

$$m(t|\theta,b)=\theta F(t)=0\left(1-\frac{b+1+bt}{b+1}e^{-bt}\right) \quad (17)$$

The likelihood function of the finite-fault NHPP model by using the Eq. 16 and 17 is derived as follows:

$$L_{NHPP}(\Theta|\underline{x}) = \left(\prod_{i=1}^{n}\lambda(x_i)\right)\exp[-m(x_n)] = [\prod_{i=1}^{n}\theta\frac{b^2}{b+1}(1+t)e^{-bt}]\cdot\exp[-\theta(1-\frac{b+1+bt}{b+1}e^{-bt})] \quad (18)$$

Note that $x_n = \sum_{i=1}^{n} t_i$ $(i=1, 2, \ldots, n; 0 \le x_1 \le x_2 \le \ldots, \le x_n)$

When the Eq. 18 is partially differentiated with respect to • and b, the result is as follows:

$$\frac{\partial \ln L_{NHPP}(\Theta|\underline{x})}{\partial\theta} = \frac{n}{\theta} - 1 + \frac{b+1+bx_n}{b+1}e^{-bx_n} = 0 \quad (19)$$

$$\frac{\partial \ln L_{NHPP}(\Theta|\underline{x})}{\partial b} = \frac{2n}{b} - \frac{n}{b+1}\sum_{i=1}^{n}x_i + \theta e^{-bx_n}\left(x_n - b^2x_n + b - b^3x_n - b^3\right) = 0 \quad (20)$$

Therefore, the maximum likelihood estimator $\hat{\theta}_{MLE}$ and $\hat{b}_{MLE}$ satisfying the following the Eq. 19 and 20 can be estimated by a numerical method.

RESULTS AND DISCUSSION

Software development cost model: The estimated total cost of development based on software development cost model is as follows (Zhang and Wu, 2012).

$$E = E_1 + E_2 + E_3 + E_4 = E_1 + C_2 \times t + C_3 \times m(t) + C_4 \times [m(t+t') - m(t)] \quad (21)$$

Note that E is Estimated total cost of software development. E_1 stands for software design and initial software development costs and is considered a constant. E_2 represents the software testing cost per unit time and is expressed by the following Eq. 22.

$$E_2 = C_2 \times t \quad (22)$$

Where:

C_2 = The testing cost per unit time
t = The testingtime point

E_3 represents the cost of removing a defect by detecting a basic defect and is expressed by the following Eq. 23.

$$E_3 = C_3 \times m(t) \quad (23)$$

Note that C_3 is the cost of removing one defect found in the testing process and m(t) is the expected value of the defect that can be detected at time t.

E_4 represents the cost of eliminating all remaining defects in the operating software system and is expressed by the following Eq. 24.

$$E_4 = C_4 \times [m(t+t') - m(t)] \quad (24)$$

Note that C_4 is the defect correction cost discovered by the operator at the software operation stage after the software is released and t• is the time for operating and maintaining the software after launching the software system. In reality, C_4 has a higher cost than C_2 and C_3.

Therefore, the optimal software release time for the software development cost can be derived as follows (Yang, 2016).

$$\frac{\partial E}{\partial t} = E' = (E_1 + E_2 + E_3 + E_4)' = 0 \quad (25)$$

In other words, it can be seen that the timing point at which the smallest development cost is satisfied is the optimal software releasetime.

The analysis and prediction algorithm: In this study, the proposed algorithm 1 for analyzing and predicting software development cost is as follows:

Software development cost algorithm

Step 1: Validate the software failure data collected through the Laplace trend test analysis.

Step 2: Calculate the parameters ($\hat{\theta}_{MLE}$, $\hat{b}_{MLE}$) for the proposed model using the MaximumLikelihood Estimation.

Step 3: Analyze the results of changing the costs (C_1, C_2, C_3) that make up the total cost of software development.

Step 4: Determine an efficient cost model that meets both optimal software development cost and release time.

Step 5: Provides analysis and prediction information about software development cost and release time along with reliability analysis.

Let analyze and predict the software development cost of the proposed reliability models using the software failure time data (Prasad *et al*., 2011) as shown in Table 1.

Laplace trend test was used to verify the reliability of the software failure time dataas shown in Fig. 1.

Table 1: Software failure time data

Failure No.	Failure time (h)	Failure interval-time	Failure time (h)×10^{-2}
1	30.02	30.02	0.30
2	31.46	1.44	0.31
3	53.93	22.47	0.53
4	55.29	1.36	0.55
5	58.72	3.43	0.58
6	71.92	13.20	0.71
7	77.07	5.15	0.77
8	80.90	3.83	0.80
9	101.90	21.00	1.01
10	114.87	12.97	1.14
11	115.34	0.47	1.15
12	121.57	6.23	1.21
13	124.97	3.40	1.24
14	134.07	9.10	1.34
15	136.25	2.18	1.36
16	151.78	15.53	1.51
17	177.50	25.72	1.77
18	180.29	2.79	1.80
19	182.21	1.92	1.82
20	186.34	4.13	1.86
21	256.81	70.47	2.56
22	273.88	17.07	2.73
23	277.87	3.99	2.77
24	453.93	176.06	4.53
25	535.00	81.07	5.35
26	537.27	2.27	5.37
27	552.90	15.63	5.52
28	673.68	120.78	6.73
29	704.49	30.81	7.04
30	738.68	34.19	7.38

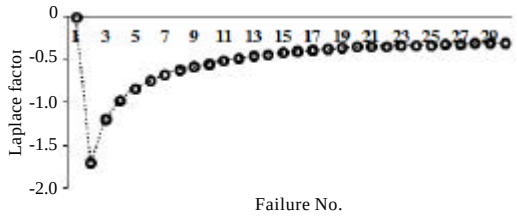

Fig. 1: Estimation results of Laplace trend test (Laplace trend test vs. Failure number)

As aresult of this test in Fig. 1, the estimated value of the Laplace factor was distributed between -2 and 2 as shown in Fig. 1. Therefore, it is efficient to apply this data because there is no extreme value.

In this study, the Maximum Likelihood Estimation (MLE) was used to perform parameter estimation. And numerical conversion data (Failure time [h]×10^{-2}) in order to facilitate the parameter estimation was used.

The calculation method of the nonlinear equations is solved using the bisection method and the results are shown in Table 2.

In this study, we assumed the cost of software development as [Supposition 1] ~ [Supposition 4] in order to operate same as actual software development environment.

Table 2: Parameter estimation of each model

	MLE	
Model	$\hat{\theta}_{MLE}$	$\hat{b}_{MLE}$
Goel-Okumoto basic	33.4092	0.3089
Erlang (at a = 2)	30.5978	0.7922
Lindley	30.4691	1.3460

MLE = Maximum Likelihood Estimation

To do this, we analyze and predict software development cost and release time with changing each cost component (C_2, C_3, C_4) that constitutes the total development cost of software, such as the situation that may occur during the actual software development process.

[**Supposition 1:** Basic conditions]

$$E_1 = 5\$,\ C_2 = 5\$,\ C_3 = 1.5\ \$,\ C_4 = 10\$,\ t' = 200 \quad (26)$$

The result of the cost curve using [Supposition 1] is as shown in Fig. 2. In this figure, the transition of the development cost curve shows a constant pattern for a short period of time after showing a decrease pattern in the initial stage but it shows an increasing pattern as the discharge time passes. The reason is that in the process of eliminating defects during the initial stage, the development cost is decreased because the number of defects inherent in the software is reduced but the development cost is increased because the probability of finding the remaining defects during the latter stage is gradually lowered. As a result, the pattern of the development cost curve gradually increases as the release time passes.

Therefore, by using this prior information, the software development cost trends as well as the optimal software release time can be predicted together.

As shown in Fig. 2, although, all of the proposed models show similar pattern, Lindley Model is relatively efficient model because it has lower development cost and faster release time than Erlang Model or Go-Okumoto basic model.

[**Supposition 2:** Assumed that the cost C_2 is increased in supposition 1]

$$E_1 = 5\$,\ C_2 = 10\ \$,\ C_3 = 1.5\ \$,\ C_4 = 10\$,\ t' = 200 \quad (27)$$

[Supposition 2] is a case where the software testing cost (C_2) per unit time is doubled compared with [supposition 1]. The simulation result is as shown in Fig. 3.

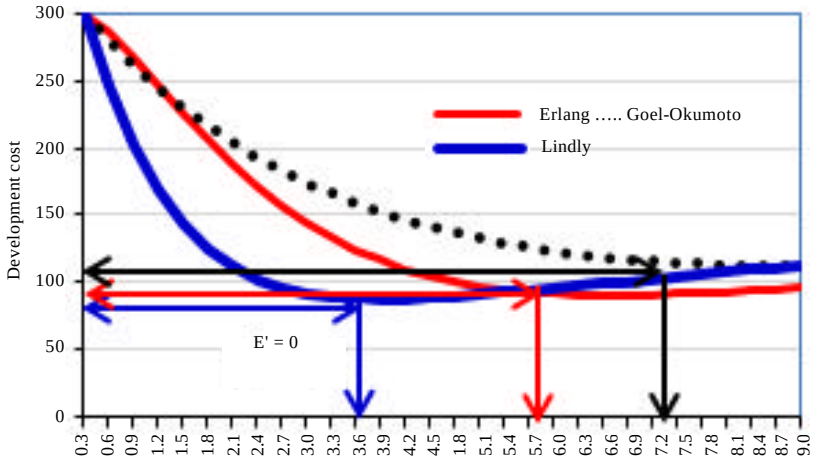

Fig. 2: The development curve applied to the condition of [Supposition 1] (cost vs. Time)

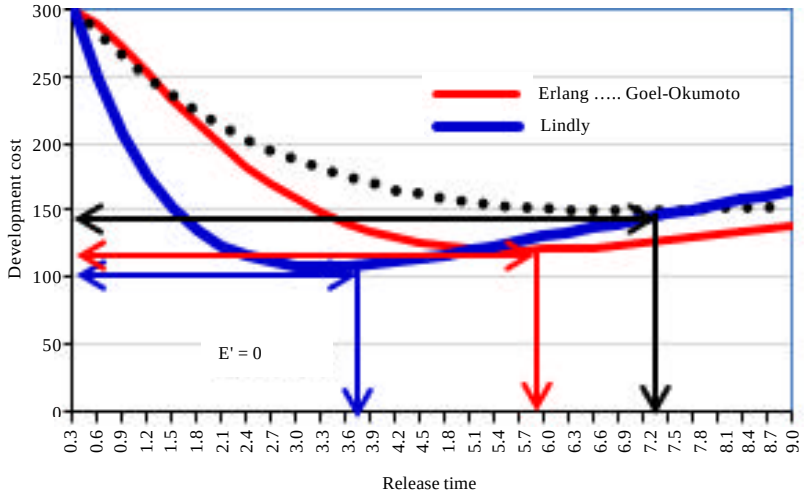

Fig. 3: The development curve applied to the condition of [Supposition 2] (cost vs. Time)

As shown in Fig. 3, the development cost has increased but the release time has not changed. Therefore, in this case, it can be seen that fast and precise testing is required, so that, the testing cost per unit time does not increase before the software is released. In addition, Lindley Model is relatively efficient because it has lower development cost and faster release time than Erlang Model or Go-Okumoto basic model.

[**Supposition 3:** Assumed that the Cost C_3 is increased in supposition 1]

$$E_1 = 5\$, C_2 = 5\$, C_3 = 3\$, C_4 = 10\$, t' = 200 \quad (28)$$

[Supposition 3] is a case where the Cost (C_3) of removing one defect found in the software testing process is doubled compared to [Supposition 1]. The simulation result is as shown in Fig. 4. As shown in Fig. 4, the development cost has increased but the release time has not changed. Therefore, in this case as many defects as possible should be removed at once, so that, the cost of removing one defect in the software testing step is not increased. In addition, Lindley Model is relatively efficient model because it has relatively less development cost and faster release time than Erlang Model or Go-Okumoto basic model.

[**Supposition 4:** Assumed that the cost C_4 is increased in supposition 1]

$$E_1 = 5\$, C_2 = 5\$, C_3 = 1.5\$, C_4 = 20\$, t' = 200 \quad (29)$$

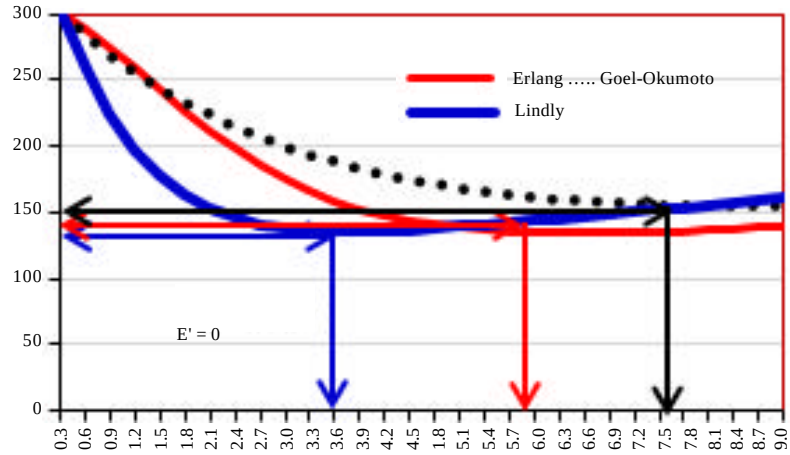

Fig. 4: The development cost curve applied to the condition of [Supposition 3] (cost vs. Time)

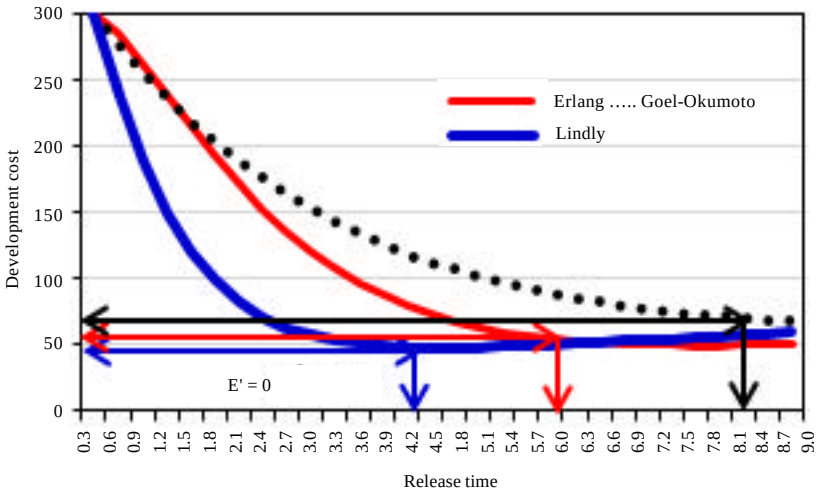

Fig. 5: The development cost curve applied to the condition of [Supposition 4] (cost vs. Time)

[Supposition 4] is a case where the cost C_4 of correcting defects discovered by the software operator during the operation stage after the release of the software is doubled compared to [Supposition 1]. And the simulation result is as shown in Fig. 5.

As shown in Fig. 5, the development cost is reduced, but the release time is delayed. Therefore, in this case, we must eliminate all possible defects at the testing stage rather than the operational stage to reduce the all defects before the software release.

In addition, the Lindley Model is a relatively excellent model because it has a low development cost and a faster release time than the Erlang Model and the Goel-Okumoto basic model.

In this study, the reliability of the mission time in the future is analyzed together to secure the reliability of the proposed models. Reliability indicates the probability that a software failure will occur at the testing time $x_n = 7.3868$ of the proposed model and no software failure will occur during the confidence interval $[x_n, x_n+\bullet]$ (where, $\bullet$ is the mission time at future). Reliability is derived from the Eq. 30 as follows (Yang and Park, 2015).

$$\hat{R}(\tau|x_n) = e^{-\int_{x_n}^{x_n+\tau}\lambda(\tau)\,d\tau} = \exp[-\{m(x_n+\tau)-m(x_n)\}] \quad (30)$$

As shown in Fig.6, the Lindley Model and the Goel-Okumoto basic model show a higher reliability trend than the Erlang Model in which the reliability decreases with the mission time in the future. Therefore, the Lindley Model is more efficient than the Erlang model because it has high and stable reliability.

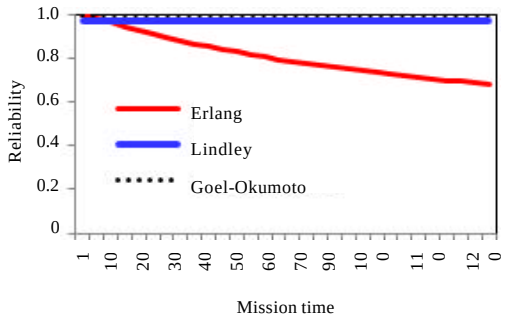

Fig. 6: Transition of reiliability (reliability vs. Mission time)

CONCLUSION

If software development costs can be quantitatively modeled along with release time during the software development process, the attributes of development costs can be efficiently analyzed and predicted.

Therefore, this study analyzes and predicts software development cost along with software release time through the proposed reliability models.

The results of this study can be summarized as follows: first, under the given basic conditions (Supposition 1), the software development cost curve shows a constant pattern for a short time after a significant decrease in the initial stage but shows a pattern of increasing again in the latter stage when the release time passes.

The reason is that the number of remaining defects is gradually reduced in the course of removing defects, so, the probability of finding residual defects is getting lower. For this reason, it can be seen that the cost eventually increases.

Second, prior to software release, if the cost of software testing per unit time increases or if the cost of removing one defect found in the testing process increases, the development cost has increased as well but the release time has not changed.

Third, after the software release, if the defect correction costs discovered by the software operator increase during the software operation phase, the development cost decreases but conversely the release time is delayed.

Fourth as a result of comprehensive analysis of the proposed model used in this study, Lindley Model is a relatively most efficient model because it has high reliability for future mission timeand low software development cost and fast release time compared to Erlang Model and Go-Okumoto basic model. Using the results of this study, it is possible to provide software developers and operators with the necessary prior information for predicting the most economical software development costs and the optimal software release time.

In addition, further studies will be needed to find out optimal software development cost model through analysis with other models having the same type of failure time distribution.

ACKNOWLEDGEMENT

Funding for this study was provided by Namseoul University.

REFERENCES

Goel, A.L. and K. Okumoto, 1979. Time-dependent error-detection rate model for software reliability and other performance measures. IEEE. Trans. Reliab., 28: 206-211.

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

Pham, H. and X. Zhang, 2003. NHPP software reliability and cost models with testing coverage. Eur. J. Opert. Res., 145: 443-454.

Prasad, R.S., K.R.H. Rao and R.R.L. Kantha, 2011. Software reliability measuring using modified maximum likelihood estimation and SPC. Int. J. Comput. Applic., 21: 1-5.

Song, K.Y., I.H. Chang and H. Pham, 2017. A software reliability model with a Weibull fault detection rate function subject to operating environments. Appl. Sci., 7: 1-16.

Yamada, S. and S. Osaki, 1985. Software reliability growth modelling: Models and applications. IEEE Trans. Software Eng., 11: 1431-1437.

Yang, T.J. and J.G. Park, 2015. A study on software reliability model based on mixture Weibull NHPP property. Intl. J. Appl. Eng. Res., 10: 548-552.

Yang, T.J., 2016. A software reliability cost model based on the shape parameter of Lomax distribution. J. Korea Instit. Inf. Electron. Commun. Technol., 9: 171-177.

Yang, T.J., 2017. The performance analysis comparative study depend on software reliability model and curve regression model. Intl. J. Soft Comput., 12: 313-317.

Zhang, Y. and K. Wu, 2012. Software cost model considering reliability and time of software in use. J. Convergence Inform. Technol., 7: 135-142.