신뢰성 성장모형에 대한 소프트웨어 신뢰성 메트릭 추정량의 민감도 분석

김대경[†]

전북 대학교 통계학과

Sensitivity analysis of software reliability metric estimator for Software Reliability Growth Models

Kim Dae Kyung[†]

Chonbuk National University, Department of Statistics

Key Words : software reliability metric, MLE, SRGM, NHPP, Taylor approximation

Abstract

When we estimate the parameters of software reliability models, we usually use maximum liklihood estimator(MLE). But this method is required a large data set. In particular, when we want to estimate it with small observed data such as early stages of testing, we give rise to the non-existence of MLE. Therefore, it is interesting to look into the influence of parameter estimators obtained using MLE. In this paper, we use two non-homogenous poisson process software reliability growth model : delayed S-shaped model and log power model. In this paper, we calculate the sensitivity of estimators about failure intensity function for two SRGMs respectively.

1. Introduction

Today, software products are rapidly increasing but users are often not satisfied with software quality because of its failure. Software faults become the cause of many failures. The consequences and implications of failures can range from minor inconveniences to catastrophic loss of life and property. Software plays a critical part in not only scientific and business related enterprises, in daily life where it runs devices such as cars, phones and television sets. We get cash from an ATM, make a phone call, and drive our cars. A typical cell phone now contains 2 million lines of software code; by 2010 it will likely have 10 times as many. General Motors Corporation estimates that by then its cars will each have 100 million lines of code [2]. In the future, software will be increased in size and be complicated in respects of functions and structures. Since software is being used to monitor and control both systems and human life, there is a great demand for high -quality software products.

Reliability, failure rate and the residual number of faults of software are the three most important metrics that provide a quantitative assessment of the failure characteristics of devices. These are primary concerns for both software developers and software users. Failures are the result of a fault in the software code. In practice, fault detection and correction activity consumes a non negligible amount of time

† 교신저자 dkkim@chonbuk.ac.kr

※ This paper was supported by research funds of Chonbuk National University in 2007

and resources (costs, human etc). The process of finding and removing faults to improve the software reliability can be described by a mathematical relationship. Software reliability growth models (SRGMs) have been used to estimate above software reliability metrics. The most wisely used SRGM is a exponential model based on a non-homogenous Poisson process(NHPP). These numerous SRGMs have been developed by software developers for the estimation of reliability growth of products during different types of software projects since the early 1970s [11,13]. Each such model is based on a different set of assumptions and hence takes a different functional form. Also many papers have discussed the optimal software release time problem using SRGMs[9]. But various authors have discussed SRGMs' applications, limitations, and underlying assumptions [5].

The popular software reliability models may generally be classified as perfect(finite) debugging models and imperfect(infinite) debugging models. The former assumes that each time a failure occurs, the errors which caused it is immediately removed and no new errors are introduced[6]. The latter consider that faults are not always fully repaired and new ones can be introduced as part of the fault repair process [15]. But both models are often inaccurate. Also models that are good overall are not always the best choice for a particular data set.

When we estimate the parameters for reliability metrics, we first must estimate the point estimates of SRGMs. Generally, the parameters are estimated from the observed failure data using maximum likelihood estimator(MLE) [10]. We use this method because it seems more stable than other methods. But the stability of parameter estimates using this method depends on a large number of failures data. In particular, when we want to estimate it with small observed data such as early stages of testing, we give rise to the non-existence of MLE [7, 10].

In this paper, we first look into variances of parameter estimates for nonhomogenous Poisson process(NHPP) class of SRGMs. Then we investigate the effects of variances in the failure intensity function estimate of these models using ones of parameter estimates. In this paper, we selected the following finite and infinite failure models from the NHPP model class: Log power model [15] and Delayed S-Shaped model [14]. Some field research demonstrates the superiority for these models [1]. And we will use the real data TV data sets. Section 2 we review the NHPP SRGMs. Section 3 obtains the variance of failure intensity function for models. Numerical example is presented in Section 4. Finally, Section 5 concludes the paper.

2. NHPP models

Goel and Okumoto[6] assumed that the number of software failures during non-overlapped time intervals is s-independent and the software failure intensity $\lambda(t)$ is proportional to the residual fault content. Let $N(t)$ denote cumulative number of random events(faults) in the time interval $[0,t]$. Then $\{N(t), t \geq 0\}$ is called a counting process. For each point of time t, we have a random number $N(t)$. $m(t)$ denotes its expectation, i.e. $m(t) = E[N(t)]$. The function $m(t)$ is called the mean value function. Then software failure intensity function is related as follows:

$$m(t) = \int_0^t \lambda(s)ds$$

and

$$\frac{dm(t)}{dt} = \lambda(t).$$

$N(t)$ is known to have a Poisson probability mass function with parameter $m(t)$, that is:

$$P\{N(t) = t\} = \frac{[m(t)]^n e^{-m(t)}}{n!}, \; n = 0,1,2,\ldots.$$

If $\lambda(t) = c$ is a constant, we have $m(t) = \lambda t$ and $N(t)$ is called a homogenous Poisson process. We generally represent the mean value function as a function of two functions, the error content $a(t)$ and the error detection rate $b(t)$. The mean value function of the infinite failure NHPP models can be written as follows:

$$m(t) = e^{-B(t)}\left[m_0 + \int_{t_0}^{t} a(s)b(s)e^{B(s)}ds\right]$$

where $m(t_0) = m_0$ and $B(t) = \int_{t_0}^{t} b(s)ds$, in which $a(t)$ and $b(t)$ are both functions of time. The NHPP models can be classified into finite failure and infinite failure categories. If $a(t) = a$ and $b(t)$ is time-dependent error detection rate, we call its model perfect debugging model. Otherwise, we call it imperfect debugging model. If we know mean value function $m(t)$, we estimate the parameters using m.l.e method.

We are interest in infinite NHPP model because it is realistic model. But we will treat the finite NHPP model to compare with infinite one. Almering et.al[1] use the following SRGMs for test his real data in their paper. They demonstrate the superiority for these models. Therefore, we also want to use these models in my research.

3. Expression for estimator of software reliability metric

We want to express variation of the failure intensity function estimate in according to the variances in parameter estimates for proposed models. Parameters of these models include unknown parameters and estimate them using failure data. Therefore, the software reliability metric estimator included estimates which are a function of random variables. Using the

Taylor approximation for functions of several random variables and independence of random variables[4], we obtain the variance of the failure intensity function estimator.

$$V[\lambda(t_m)] = \left[\frac{\partial\lambda(t_m)}{\partial a}\Big|_{\hat a,\hat b}\right]^2 V(a) + \sum_{i=1}^{n}\left[\frac{\partial\lambda(t_m)}{\partial b_i}\Big|_{\hat a,\hat b}\right]^2 V(b_i)$$

where t_m is given mission time and a and b_i are parameters .

We want to obtain the variances of failure intensity function estimator for proposed models using the above expression. For the delayed S-shaped model,

$$V[\hat\lambda(t_m)] = \hat b^2 e^{-\hat b t_m} t_m V(\hat a) - \hat a \hat b e^{-\hat b t_m} t_m(-2 + \hat b t_m) V(\hat b) \tag{1}$$

and for the log power model,

$$V[\hat\lambda(t_m)] = \log^{\hat\beta}(1+t_m) V(\hat\alpha) + \hat\alpha log^{\hat\beta}(1+t_m)\log[\log(1+t_m)] V(\hat\beta) \tag{2}$$

4. Numerical analysis

Here we use a set of real test data [TV 2005] introduced in [1]. These data show the occurrence times of errors. They execute stability testing and model selection using real test data from three software development projects(TV2003, TV2004, and TV2005). All the projects concerned developing software for high-end TV-sets containing several million lines of code. Their analysis focused on the expected remaining number of faults in the software during stability tests in maturity phase.

They selected the two finite NHPP SRGMs(Goel and Okumoto model, Delayed S-Shaped model) and two infinite NHPP SRGMs(log power model, log Poisson execution time model). They concluded that the log-like infinite NHPP SRGMs outperformed both the finite models. We obtain the maximum likelihood estimator of the parameters based on 103 error data of the TV 2005. Here we assume the mission time $t_m = 1342$ test hours which is the stopping testing time of the TV2005 project in their paper. If we want to release the software devices at $t_m = 1342$, it is important to see an estimate of failure intensity function prior to the release of the software devices. In particular, when $\beta > 1$, the failure intensity function of the log-power model is increasing and decreasing at $t^* = e^{\beta - 1}$.

It is similar to that the S-shaped NHPP models. Because of human learning process, dependent software faults, etc, the failure intensity may increase in the beginning before the testing results in reliability growth. So, we choose delayed S-Shaped model as finite NHPP SRGM and log power model as infinite NHPP SRGM. Table1 is m.l.e of parameters for two SRGMs. Also we express the variation of the software reliability metric, failure intensity function as variance. We calculate the values using the R 2.9 program.

4.1 Delayed S-shaped model

First of all, we want to evaluate $V[\hat{\lambda}(t_m)]$ on $V(\hat{a})$ irrespective of $V(\hat{b})$ using the expression(1). Let $V(\hat{a})$ change from 100 to 190 by 10. And $V(\hat{b})$ change from 0.001 to 0.01 by 0.001. All figures are round off in the fourth place.

Table 2. $V[\hat{\lambda}(t_m)]$ on $V(\hat{a})$

$V(\hat{a})$	$V[\hat{\lambda}(t_m)]$	$V(\hat{a})$	$V[\hat{\lambda}(t_m)]$
100	0.009	150	0.014
110	0.010	160	0.015
120	0.011	170	0.016
130	0.012	180	0.017
140	0.013	190	0.018

Secondly, we similarly want to evaluate $V[\hat{\lambda}(t_m)]$ on $V(\hat{b})$ irrespective of $V(\hat{a})$. Let and $V(\hat{a})=0$ and $V(\hat{b})$ change from 0.001 to 0.010 by 0.001.

Table 3. $V[\hat{\lambda}(t_m)]$ on $V(\hat{b})$

$V(\hat{b})$	$V[\hat{\lambda}(t_m)]$	$V(\hat{b})$	$V[\hat{\lambda}(t_m)]$
0.001	0.008	0.006	0.051
0.002	0.017	0.007	0.059
0.003	0.025	0.008	0.068
0.004	0.034	0.009	0.076
0.005	0.042	0.010	0.0844

Table 2 and 3 show that the larger $V(\hat{a})$ and $V(\hat{b})$ respectively, also the larger $V[\hat{\lambda}(t_m)]$ as we expect. Also we know that a rate of increase for $V(\hat{b})$ becomes larger than $V(\hat{a})$. Finally, we evaluate the effect of $\hat{a}$ on the variance in failure intensity function estimate. We set $V(\hat{a})=0.1$ and $V(\hat{b})=0.001$ for convenience' sake. For the effect of $\hat{a}$,we change $\hat{a}$ from 100 to 109 by 1. The results are as follows:

Table 4. $V[\hat{\lambda}(t_m)]$ on $\hat{a}$

$\hat{a}$	$V[\hat{\lambda}(t_m)]$	$\hat{a}$	$V[\hat{\lambda}(t_m)]$
100	0.002	105	0.002
101	0.002	106	0.002
102	0.002	107	0.002
103	0.002	108	0.002
104	0.002	109	0.002

Similarly, we evaluate the effect of $\hat{b}$ on the variance in failure intensity function estimate. We set $V(\hat{a})=0.1$ and $V(\hat{b})=0.000001$ for convenience' sake. For the effect of $\hat{b}$, we change $\hat{b}$ from 0.001 to 0.010 by 0.001. The results are as follows:

Table1. SRGMs and m.l.e

SRGM	Mean value function	Failure intensity function	m.l.e	Note
Delayed S-shaped model[14]	$m(t)=a(1-(1+bt)e^{-bt})$	$\lambda(t)=ab^2te^{-bt}$	$\hat{a}=105.987$ $\hat{b}=0.004$	Perfect debugging model(finite model)
Log power model[15]	$m(t)=\alpha\log^{\beta}(1+t)$	$\lambda(t)=\dfrac{\alpha\beta\ln^{\beta-1}(1+t)}{1+t}$	$\hat{\alpha}=0.563$ $\hat{\beta}=2.640$	Imperfect debugging model(infinite model)

Table 5. $V[\hat{\lambda}(t_m)]$ on $\hat{b}$

$\hat{b}$	$V[\hat{\lambda}(t_m)]$ $\times 10^{-6}$	$\hat{b}$	$V[\hat{\lambda}(t_m)]$ $\times 10^{-6}$
0.001	3.509	0.006	0.137
0.002	3.533	0.007	0.049
0.003	2.001	0.008	0.017
0.004	0.912	0.009	0.005
0.005	0.368	0.010	0.002

In the Table 4, we can see that as $\hat{a}$ is increasing $V[\hat{\lambda}(t_m)]$ is decreasing. But its amount of decreasing is very small. In the table 5, as $\hat{b}$ is increasing, $V[\hat{\lambda}(t_m)]$ is increasing and decreasing. It is observed that, because of human learning process, the failure intensity may increase in the beginning before the testing results in a reliability growth.

4.2 Log power model

Log power model is the one of imperfect SRGMs. It is interesting to see the variation of failure intensity function when all faults in the software are infinite. Similarly, we investigate to $V[\hat{\lambda}(t_m)]$ on $V(\hat{\alpha})$ irrespective of $V(\hat{\beta})$ using the expression(2). Let $V(\hat{\beta})=0$ and $V(\hat{\alpha})$ change from 0.001 to 0.010 by 0.001.

Table 6. $V[\hat{\lambda}(t_m)]$ on $V(\hat{\alpha})$

$V(\hat{\alpha})$	$V[\hat{\lambda}(t_m)]$	$V(\hat{\alpha})$	$V[\hat{\lambda}(t_m)]$
0.001	0.020	0.006	0.122
0.002	0.041	0.007	0.142
0.003	0.061	0.008	0.162
0.004	0.081	0.009	0.183
0.005	0.102	0.010	0.203

Secondly, we want to evaluate $V[\hat{\lambda}(t_m)]$ on $V(\hat{\beta})$ irrespective of $V(\hat{\alpha})$. Let $V(\hat{\alpha})=0$ and $V(\hat{\beta})$ change from 0.001 to 0.010 by 0.001.

Table 7. $V[\hat{\lambda}(t_m)]$ on $V(\hat{\beta})$

$V(\hat{\beta})$	$V[\hat{\lambda}(t_m)]$	$V(\hat{\beta})$	$V[\hat{\lambda}(t_m)]$
0.001	0.006	0.006	0.034
0.002	0.011	0.007	0.040
0.003	0.017	0.008	0.045
0.004	0.023	0.009	0.051
0.005	0.028	0.010	0.057

Table 6 and 7 show that the larger $V(\hat{\alpha})$ and $V(\hat{\beta})$, also the larger $V[\hat{\lambda}(t_m)]$ as we expect. Also we know that a rate of increase $V(\hat{\beta})$ for becomes larger than $V(\hat{\alpha})$. Finally, we evaluate the effect of $\hat{a}$ on the variance in failure intensity function estimate. We set $V(\hat{\alpha})=0.1$ and $V(\hat{\beta})=0.001$ for convenience' sake. For the effect of $\hat{a}$, we change $\hat{a}$ from 0.1 to 1 by 0.1. The results are as follows:

Table 8. $V[\hat{\lambda}(t_m)]$ on $\hat{a}$

$\hat{a}$	$V[\hat{\lambda}(t_m)]$	$\hat{a}$	$V[\hat{\lambda}(t_m)]$
0.1	2.034	0.6	2.055
0.2	2.038	0.7	2.058
0.3	2.042	0.8	2.062
0.4	2.046	0.9	2.066
0.5	2.050	1	2.070

We evaluate the effect of $\hat{\beta}$ on the variance in failure intensity function estimate. Let $V(\hat{\alpha})=0.1$ and $V(\hat{\beta})=0.000001$ for convenience' sake. For the effect of $\hat{\beta}$, we change $\hat{\beta}$ from 2.1 to 3.1 by 0.1. The results are as follows:

Table 9. $V[\hat{\lambda}(t_m)]$ on $\hat{\beta}$

$\hat{\beta}$	$V[\hat{\lambda}(t_m)]$	$\hat{\beta}$	$V[\hat{\lambda}(t_m)]$
2.1	1.097	2.6	1.940
2.2	1.229	2.7	2.174
2.3	1.378	2.8	2.437
2.4	1.544	2.9	2.731
2.5	1.731	3.0	3.061

In the Table 8 and 9, we can see that as $\hat{\alpha}$ is increasing $V[\hat{\lambda}(t_m)]$ is increasing. Also, as $\hat{\beta}$ is increasing $V[\hat{\lambda}(t_m)]$ is increasing. The reason is that the new errors are introduced in the software when we debugged a fault.

5. Conclusions and further research

We calculate the sensitivity of estimators about failure intensity function for two SRGMs respectively. The one is perfect(finite) debugging model, delayed S-Shaped model and the other is imperfect(infinite) debugging model, log power model. Failure intensity functions include the unknown parameters. Parameters are estimated by maximum likelihood estimate. In our paper, we investigate the variation of failure intensity function for parameters. In the two SRGM models, as the variations of $\hat{a}$ and $\hat{b}$ are larger respectively, also the variation of failure intensity function is larger as we expect. But we know that a rate of increase for the variation of $\hat{b}$ becomes larger than $\hat{a}$. In the delayed S-shaped model, as $\hat{a}$ is increasing, the variation of failure intensity function is decreasing. But its amount of decreasing is very small. On the other hand, as $\hat{b}$ is increasing, the variation of failure intensity function is increasing and decreasing. But in the log power model, as $\hat{a}$ and $\hat{b}$ are increasing, the variation of failure intensity function is also increasing.

We want to apply to other reliability data and another reliability metrics(reliability and the residual number of faults etc) in the future. We can see which metric is sensitive to variation of estimator.

References

[1] Almering, V., Genuchten, M.V., Cloudt, G. and Sonnemans, P.J.M., "Using Software Reliability Growth Models in Practice," Software, IEEE, Vol. 24, Issue 6, pp. 82-88, 2007.
[2] Charette, Robert N., "Why Software Fails?", Spectrum, IEEE, Vol.42, Issue9, pp. 42-49, 2005.
[3] Cotte, D. W., "System-reliability confidence-intervals for complex-systems with estimated component-reliability.", IEEE Transactions on Reliability, Vol.46, No.4, pp. 487-493. 1997.
[4] Dadkhah, Foundations of Mathematical And Computational Economics, 1st Edition, South-Western Publisher 2007.
[5] Goel, A.L, "Software Relaibility Models: Assumptions, Limitations, and Applicability," IEEE Transactions Software Engineering, Vol.11, No,12, pp. 1411-1423, 1985.
[6] Goel, A.L. and Okumoto, K., "Time-dependent error-detection rate model for software and other performance measures," IEEE Transactions on Reliability, Vol. R-28, pp. 206-211, 1979.
[7] Hossain, S.A. and Dahiya, R.C., "Estimating the parameters of a non-homogenous poisson process model for software reliability, "IEEE Transactions on Reliability, R-42, No.4, pp. 604-612, 1993.
[8] Huang, C.Y. Kuo, S.Y. and. Lyu, M.R., "Optimal Software Release Policy Based on Cost, Reliability and Testing Efficiency,"
[9] Proceedings of the 23rd Annual International Computer Software and Applications Conference, pp. 468-473, 1999.
[10] Joe, H., "Statistical inference for general-order-statistics and non homogenous Poisson process software reliability models," IEEE Transactions on software Engineering, Vol.15, No.11, pp. 1485-1490. 1989.
[11] Knafl, G.J. and Morgan, J., "Solving ML equations for 2-paremater Piosson-process models for ungrouped software-failure data," IEEE Transactions on Reliability, R-45, No.1, pp. 42-53, 1996.
[12] Musa, J.D., Software Reliability Engineering: More Reliable Software, Faster Development and Testing. McGraw-Hill, 1998.
[13] Pham, H., "Software reliability assessment: Imperfect debugging and multiple failure types in software development," EG&G-RAAM-10737, Idaho National Engineering Laboratory.
[14] Pham, H, Software reliability, Singapore: Springer-Verlag, 2000.
[15] Yamada, S. Ohba, M. and Osaki, S., "S-Shaped Reliability Growth Modeling for Software Error Detection," IEEE Transaction on Reliability, Vol. 32, No.5, pp. 475-478, 1983.
[16] Zhao, M. and Xie, M. "On the log-power software reliability model," Proceeding of the 3rd International Symposium on Software Reliability Engineering, pp. 14-22, 1992.