Mobile Software Reliability Measurement Using Growth Model in Testing

Haeng-Kon Kim[1], Eun-Ju Park[1]

[1]Dept. of Computer information & Communication Engineering
Catholic Univ. Of Daegu, Korea
{hangkon, ejpark}@cu.ac.kr

Abstract. In this paper, we study the software reliability measurement method of reliability testing metrics. Software reliability is very important. But, it is very difficult to test for software reliability measurement. So we describes the software reliability metrics for ISO/IEC 9126, and we introduce Gamma-Lomax software reliability model for multiple error debugging. And we calculate the software reliability measure(reliability metrics, parameter estimation etc). We introduce the measurement method of software reliability quality of software product.

Keywords: Software reliability quality metrics, ISO/IEC 9126, Gamma-Lomax Software Reliability Growth Model, Quality Metrics

1 Introduction

Software system have been applied into the various fields of science, business, society, our life and its importance and necessary has been raised and recognized. The software system environments have been enlarged, that is the range of applied fields become wide and the technology changed rapidly. As a result software system seem more complex. Also, software reliability is most important part of business in the late 20th century. It is a time to study on testing the software developing techniques and research on the software reliability for the security of the software products. We have to measure the software reliability quality of the product produced and use that information to improve the process producing it. Software reliability quality assurance for software development needs to be bound up with good measurement.

The rest of this paper is arranged as follows: Chapter 2 describes the commonly used software reliability attributes and currently available metrics for measuring software product quality.

Chapter 3, we introduce a Gamma-Lomax software reliability growth model(SRGM) for the multiple software errors debugging in the software testing stage. This model is the modification of Jelinski-Moranda model incorporated

"This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by IITA(Institute of Information Technology Advancement)" (IITA-2008-(C1090-0801-0032))

dependence among the interfailure times for multiple software errors. And we evaluate the software reliability measures by the method of maximum likelihood estimation.

Chapter 4, we introduce the concepts of the software reliability testing method. Also, we introduce the result of the simulation study for software reliability measure.

2 Software Reliability Metrics

Software reliability is important part of the chief industries in the late 20th century. Integral to the development of software is the process of detecting, locating, and correcting bugs. In spite of these effective more common types of software nonperformance include the failure to conform to specifications or standards. Software testing tools are available that incorporate proprietary testing algorithms and metrics that can be used to measure the performance and conformance of software. But, development of standard testing tools and metrics for software testing could go a long way toward addressing some of the software testing problems that plague the software industry.

Improved tools for software testing could increase the value of software in a number of ways:

(1) reduce the cost of software development and testing
(2) reduce the time required to develop new software products
(3) improve the performance, interoperability, and conformance of software.

MaCall, Richards, and Walters(1977) attempted to assess quality attributes for software. Boehm(1978) introduced several additional quality attributes. As software changed and improved and the demands on software increased, a new metrics of software quality attributes was needed.

In 1991, the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) adopted ISO/IEC 9126 as the standard for software quality. The ISO/IEC 9126 is now widely accepted.

ISO/IEC 9126[1] consists of the external metrics, internal metrics. It categorises software quality attributes into six characters (functionality, reliability, usability, efficiency, maintainability, and portability), which are further subdivided into subcharacteristics. The subcharacteristics can be measured by internal or external metrics. An reliability character in external metrics should be able to measure related to the software reliability model. Software reliability models enable to make estimations and predictions for the reliabilities of software systems by applying the software failure data and experiences obtained from the previous experiments for the similar software systems to the developing target software system. The reliability measurement consist of the maturity metrics, fault tolerance metrics, recoverability metrics, reasbility compliance metrics. The maturity measurement of the software product to avoid failure as a result of faults in the software. Fault tolerence measurement of the software product to maintain a specified level of performance in cases of software faults of its specified interface. The recoverability measurement of the software product to re-establish a specified level of performance and recover the

data directly affected in the case of a failure. The reliability compliance capability of the software product to adhere to standards, conventions or regulations relating to reliability. Testing was seen as a necessary process to prove to the final user that the product worked. An improved instrastructure for software testing has the potential to affect software developers and users by removing bugs before the software product is released and detecting bugs earlier in the software development process and locating the source of bugs faster and with more precision.

The product analyzed in this paper is a data through the survey. Respondents were asked to answer the satifaction of the software product quality on the seven categories rating scale: “Very Very Dissatified”, “Very Dissatified”, “Dissatisfied”, “Neutral”, “Satisfied”, “Very Satisfied”, “Very Very Satisfied”

Also, Respondents were asked to answer the importance of the software product quality on the same method.

Characteristics and subcharacteristics of software product quality have been measured with the same rating scale. In order to anaslysis the difference influence of characteristics, and subcharacteristics on overall satisfication and importance due to user difference was collected including application domain, training, age, similar software experience, and respondent’s job type(developer, user). We analysis the result using the SPSS/PC statistical package program.

3 GAMMA-LOMAX Software Reliability Model

In 1970’s, researchers have studied the modeling of software reliability. The Jelinski-Moranda(1972)[9] model for software reliability growth model(SRGM) is one of the most commonly cited models. The main property of the model is that the intensity between two consecutive failure is constant. And the distribution of interfailure time T_i at the i-th testing stage is given by

$$f(t_i)= (N-i+1)\ \phi\ \exp((-N-i+1)\phi t_i) \tag{3.1}$$

where the parameter $\lambda_I=(N-i+1)\phi$ is a failure rate, and N is the number of latent software errors before the testing starts, and ϕ is the failure intensity contributed by each failure. The parameter ϕ and N in Jelinski-Moranda model was estimated by maximizing the likelihood function.

The software reliability growth model suggested by Littlewood and Verrall(1973)[10] is perhaps the most well known Bayesian model. The Littlewood-Verrall model assumes that interfailure times are exponential distributed random variable with the density function.

$$f(t_i|\ \lambda_i)= \lambda_i \quad \exp(-\lambda_i\ t_i) \tag{3.2}$$

where λ_i is an unknown parameter whose uncertainty is due to the randomness of the testing and the random location of the software errors. And the probability density

function of λ_i is

$$(\varphi(i)^{\alpha} \lambda_i^{\alpha-1} \exp(-\varphi(i)\lambda_i) f(\lambda_i \mid \alpha, \varphi(i)) = (\Gamma(\alpha)) \quad (3.3)$$

where $\varphi(i)$ is depending on the number of detected error. Usually, $\varphi(i)$ describes the quality of the test and it is a monotone increasing function of i.
Langberg and Singpurwalla(1985)[12] presented a shock model interpretation of software failure and justified the Jelinski-Moranda model.

In the Langberg and Singpurwalla Bayesian model the parameters in the Jelinski-Moranda model are treated as random variables.

Nayak(1986)[13] proposed a model for incorporating dependence among the detection times of software reliability measures. Many software reliability model have been studied . But they treated with the case of software reliability growth model for single error debugging at each testing stage until now. Jung[20] studied the software reliability model with the multiple errors debugging.

In this section, we introduce the software reliability growth modeling and the parameters in the introduced software reliability growth model for the multiple errors debugging at each testing stage. This model is condition under the incorporated dependence for multiple errors.

The model introduced by Jung(1994)[20] for describing multiple errors at each testing stage assumes that

(1) The multiple software errors at the i-th testing stage occur n_i times for i=1,2,…,m , and the last testing stage, m is unknown and fixed constant, and $N=\sum_{i=1}^{m} n_i$
(2) All of the detected software errors in each testing stage are perfectly debugged at once before the next testing stage begins
(3) The multiple software interfailure times, $T_i = X_{(i)} - X_{(i-1)}$ i=1,2,….,m have independent gamma distribution with parameters n_i and $\lambda_i = (m-i+1)\phi$, where $0= X_{(0)} \leq X_{(1)} \leq ,\ldots, \leq X_{(m)}$ are the ordered failure times and the software failure intensity ϕ is unknown

The test environments change the software failure rates of all testing stages by a common parameter η, so that the random multiple software interfailure times T_1, T_2, …, T_m.

Of a software system are independently gamma-distributed with multiple software failure rates $\eta\lambda_1$, $\eta\lambda_2$,…, $\eta\lambda_m$ respectly, under the software test environments in assumptions (1), (2), and (3). Assuming that η is a gamma random variable as G(a,b) then the unconditional joint probability density function $f(t_1, t_2, \ldots, t_m)$ of interfailure times $T_1, T_2, \ldots, T_m$ occurring $n_1, n_2, \ldots, n_m$ times respectively is given by

$$\begin{aligned} GL_m,(a, b; \lambda_1, \lambda_2, \ldots, \lambda_m ; n_1, \ldots, n_m) \\ = (\prod_{i=1}^{m} t_i^{ni-1} \lambda_i^{ni} / \Gamma(n_i)) \\ \times (b\, \Gamma(\sum_{i=1}^{m} n_i + a)/\Gamma(a)) \end{aligned} \quad (3.4)$$

$$\times(1 / \sum_{i=1}^{m} \lambda_I t_i+b)^{\sum i=1m \; ni+a}$$

and Jung[20] introduce this distribution as a Gamma-Lomax Software Reliability Growth Model which is an extension case of the multivariate Lomax distribution of Nayak[13].

The marginal probability density function of the multiple software interfailure times $T_1, T_2, \ldots, T_m$ until r(≤m)-th testing stage is $GL_r(a, b; \lambda_1, \lambda_2, \ldots, \lambda_r ; n_1, \ldots, n_r)$ where λ_i is the failure rate of the i-th testing stage.

We will use the method of maximum likelihood in order to estimate the unknown parameters m, ϕ, n_{r+1} of the software reliability. The number of multiple software errors detected until r-th testing stage is denoted by $\sum_{i=1}^{r} n_i$, which will be called sample size. Suppose that $\Theta=\{ t_1, t_2, \ldots, t_r \}$ is a tested data set of the multiple software interfailure times.

The parameters m, ϕ, n_{r+1} of the software reliability can be estimated by miximizing the likelihood function. Let the observed r ordered multiple software failure times be $x_{(1)}, x_{(2)}, \ldots, x_{(r)}$ and let τ be the specified time $\tau \in (x_{(r)}, x_{(r+1)},)$. And, we let $t_i,= x_{(i)} -x_{(r-1)}$, i=1,2,….r and $T_{r+1},> \tau - x_{(r)},$. We estimate the parameters as follows procedures;

[ML1]

We calculate the likelihood function of m and ϕ given Θ from equation (3.4).

$$L_1(m, \phi \mid \Theta)=(\prod_{i=1}^{m} ((m-i+1)\phi)^{ni} t_i^{ni-1})/ \Gamma(n_i)) \quad (3.5)$$
$$\times(b^a \Gamma(\sum_{i=1}^{r} n_i +a))/\Gamma(a)$$
$$\times(1 / \sum_{i=1}^{m}(m-i+1)\phi t_i +b)^{(\sum i=1r \; ni+a)}$$

[ML2]

We obtain the conditional probability density function of t given Θ.

[ML3]

We combine (3.5) and conditional probability density function in [ML2] and get the likelihood function of m, ϕ, and n_{r+1}.

$$L_2(m, \phi, n_{r+1} \mid \Theta)=(\prod_{i=1}^{m} ((m-i+1)\phi)^{ni} t_i^{ni-1})/ \Gamma(n_i)) \quad (3.6)$$
$$\times((b^a /\Gamma(a)) \times(((m-r)\phi)^{nr+1} /\Gamma (n_{r+1}))$$
$$\times((t^{nr+1-1} \Gamma (\sum_{i=1}^{r+1} n_i +a))/ ((\sum_{i=1}^{r}$$
$$(m-i+1)\phi t_i +b+(m-r)\phi t)^{(\sum i=1r+1 \; ni+a)}$$

Hence we have the logarithm in the above likelihood function of (3.6) by

$$L(m, \phi, n_{r+1})=\text{lm } L(m, \phi, n_{r+1})$$

[ML4]

We take the partial derivative of the log likelihood function with respect to m, ϕ, n_{r+1}, respectively, and equate them to zeros. Then we have the normal equation (3.7).

$$\partial L(m, \phi, n_{r+1})/\partial m=0 \tag{3.7}$$

$$\partial L(m, \phi, n_{r+1})/\partial \phi=0$$

$$\partial L(m, \phi, n_{r+1})/\partial\, n_{r+1}=0$$

[ML5]

We arrange the simultaneous linear equation induced from (3.7) with respect to to m, ϕ, n_{r+1}, respectively.

$$(\sum_{i=1}^{r+1} (n_{r+1})/(m-i+1))=((\sum_{i=1}^{r+1} n_i(\sum_{i=1}^{r} t_i+t)) /((\sum_{i=1}^{r} (m-i+1)\, t_i +(m-r)t)) \tag{3.8}$$

$$\phi=\exp(\Gamma'(\sum_{i=1}^{r+1} n_i +a)/ \Gamma(\sum_{i=1}^{r+1} n_i +a) - \Gamma'(n_{r+1})/ \Gamma(n_{r+1})) \tag{3.9}$$

where

$$\Gamma'(x)/\Gamma(x) = -\gamma +(1-(1/x)) +((1/2) –(1/(x+1))) +((1/3) – (1/(x+2))) + \ldots + ((1/n) – (1/(x+n-1))) + \ldots,$$

and

$$-\gamma =\gamma'(1)=\int \exp(-x) \ln (x)\, dx\ .$$

Where the range of intergral is 0 to infinity.
According to the maximum likelihood estimation procedure [ML1] to [ML5], the approximations of estimates m, ϕ, n_{r+1}, can be obtained by solving the numerical algorithms, respectively.

Therefore substituting these estimates into the multiple software reliability (3.7),

(3.8), and (3.9), we can obtain the maximum likelihood estimates of the multiple software reliability by the result of Zehna.

4 Reliability Metrics

An external software reliability metric of ISO/IEC 9126 should be able to measure attributes related to the software failure times. Software reliability consists of the maturity metrics, fault tolerence metrics, recoverability metrics, reliability compliance metrics. Also, Maturity metrics consist of the estimated latent failure density, estimated latent fault density, failure density etc.

But, we can calculate this measure by using the software failure time distribution . It is difficult to get the software failure time data. So we try to the survey for measuring of software reliability metrics.

We present our result the number of failure during one month. For example, the questions as follows;

The product analyzed in this paper is a data through the survey. Respondents were asked to answer the satifaction of the software product quality on the seven categories rating scale: “Very Very Dissatified”, “Very Dissatified”, “Dissatisfied”, “Neutral”, “Satisfied”, “Very Satisfied”, “Very Very Satisfied”

How many failures were detected during one month?
How many fault were detected during one month?
How many failure condition are resolved?
How often the software product cause the break down ?
Also, we predict the quality of software product by using the satisfaction survey.

If the causes of failure occur according to a Poisson distribution. The times between failures are independent exponential variables, and the data constitutes a random sample of sample size from exponential with parameter $1/\phi$.

A variable or distribution would arise if one discusses the number of occurrences in some time interval. For each interval the number of failures follows a poisson distribution with parameter $\mu=\phi t$.

For example , specifically the Goel & Okumoto model assumes that the failure process is modeled by Nonhomogeneous Poisson Process model with mean value function m(t) given by

$$m(t)=a(1-\exp(-\phi t)),\ a>0,\ b>0 \tag{4.1}$$

where a and b are parameters to be determined using collected failure data.
Note that for Goel & Okumoto model[4], we have $m(\infty)=a$, and $m(0)=0$.

Since $m(\infty)$ is the expected number of faults which will eventually be detected, the parameter a is the final number of faults that can be detected by the testing process.

A company examine the as follows.

Table 1. Frequency Table

Number of failure	Frequency	Probability
0	8	0.21052632
1	15	0.39473684
2	7	0.18421053
3	7	0.18421053
5	1	0.02631579

The value of ϕ is unknown, but sample mean is a good estimate of $E(T)=1/\phi$. We estimate the mean, parameter a, and ϕ after the one month test period.

Table 2. Mean and Estimation of parameters after the thirty test day.

Mean	1.447368421
ϕ	0.048245614
a	71.91313439

We estimated the mean, parameter a, and ϕ using the homogeneous poisson distribution. So we can calculate the value of mean value function m(t) and hazard rates r(t) on based true mean value M(t).

Table 3. mean value and hazard rate.

M(t)	55
m(t)	54.1640112
r(t)	0.433445678

We apply the simple homogeneous poisson software reliability model. By using the mean ϕ, we can calculate the software reliability by software GOMMA-LOMAX model.

5 Concluding Remarks

In this paper, we present the software reliability growth model and calculate the reliability and estimate the parameters in consideration of the multiple errors debugging at each testing stage. We introduce a Gamma-Lomax software reliability growth model for multiple software error debugging at each testing stage on the modification of Jelinski-Moranda model to have incorporated dependence between the failure times. And we calculate the parameters by maximum likelihood estimation method

Recently, software quality assurance activities for the development procedure concerned with the software development project have been highly concerned as well

as the software product itself. However the measurement of the software reliability is very difficult. We suggest that the software reliability model should be applied for the standardization and the software quality measurement activities of the software product.

We are going to study parameter estimation of software reliability models who propose in research that see forward.

References

1. ISO/IEC 9126.: Information Technology -Software Quality Characteristics and metrics-Part 1,2,3.
2. ISO/IEC 14598.: Information Technology -Software Product Evaluation-Part 1~6.
3. ISO/IEC12119.: Information Technology -Software Package-Quality requirement and testing".
4. Goel, A.L., Okumoto, K.: Time-dependent error detection rate model for software reliability and other performance measures. In: IEEE Trans. Reliability, R-28, pp. 206--211. (1979)
5. Goel, A.L.: Software reliability model : assumptions, limitations, and applicability. In: IEEE Trans Software Eng, SE-11, pp. 1411--1423. (1985)
6. Bain, L.J.: Statistical Analysis of Reliability and Life-Testing Models. MARCEL, DEKKER (1991)
7. Catuneanu, V.M. et al.: Optimal software release policies using SLUMT. In: Microelectronics and Reliability, 28, pp. 547--549. (1988)
8. Catuneanu, V.M. et al.: Optimal software release time with learning rate and testing effort. In: IEEE Trans. Reliability. (1991)
9. Jelinski, Z., Moranda, P.B.: Software reliability research, in Statistical Computer Performance Evaluation. In: Ed. W. Freiberger, pp. 465--497. Academic Press, New York (1972)
10. Littlewood, B., Verrall, J.L.: A Bayesian reliability growth model for computer software. In: Applied Statistics, 22, pp. 332--346. (1973)
11. Okumoto, K., Goel, A.L.: Optimum release time for software systems based on reliability and cost criteria. In: J. Systems and Software, 1, pp. 315--318. (1980)
12. Langberg, N., Singpurwalla, N.D.: A Unification of some software reliability models. In: SIAM J. Scientific and Statistical Computation, 6, pp. 781--790. (1985)
13. Nayak, T. K.: Software Reliability:Statistical Modeling and Estimation. In: IEEE Trans. On Reliability, 35, pp. 566--570. (1986)
14. Shooman, M. L.: Software Reliability : measurement and models. In: Proc. Ann. Reliability and Maintainability Symp, pp. 485--491. (1975)
15. Yamada, S., Osaki, S.: Cost-reliability optimal release policies for software systems. In: IEEE Trans. Reliability, R-34, pp. 422--424. (1985)
16. Yamada, S., Osaki, S.: Optimal software release polices for nonhomogeneous software error detection rate model. In: Microelectronics and Reliability, 26, pp. 691--702. (1986)
17. Yamada, S., Osaki, S.: Cost-reliability optimal release policies for software systems. In: IEEE Trans. Reliability, R-34, pp. 422--424. (1985)
18. Yamada, S., Osaki, S.: Optimal software release polices for nonhomogeneous software error detection rate model. In: Microelectronics and Reliability, 26, pp.691--702. (1986)
19. Elegbede, A.O.C., Chu, C., Adjallah, K.H., Yalaoui, F.: Reliability Allocation Through Cost Minimization. In: IEEE Trans on Reliability, 52, 1, pp. 96--105. (2003)
20. Jung, H. J.: Software Reliability Growth Modeling and Estimation for Multiple Errors Debugging. Kyungpook National University. (1994)