IEMS Vol. 7, No. 1, pp. 51-56, June 2008.

Reliability Models for Application Software in Maintenance Phase

Yung-Chung Chen[†]
Department of Logistics Management, Shu-Te University 59,
Hun Shang Rd, Hun Shang Village, Yen Chao, Kaohsiung County, 82442, TAIWAN
Tel: +886-7-615-8000 ext 4513, Email: yjchen@mail.stu.edu.tw

Shih-Ying Tsai
Department of Logistics Management, Shu-Te University 59,
Hun Shang Rd, Hun Shang Village, Yen Chao, Kaohsiung County, 82442, TAIWAN
Tel: +886-7-615-8000 ext 4513

Peter Chen
Berlin Company Limited
43 Ta-Yeh South Road, Kaohsiung, 81261, Taiwan
Tel: +886-7-871-1101 ext 1300, Email: berlin.pc@gmail.com

Selected paper from APIEMS2005

Abstract. With growing demand for zero defects, predicting reliability of software systems is gaining importance. Software reliability models are used to estimate the reliability or the number of latent defects in a software product. Most reliability models to estimate the reliability of software in the literature are based on the development lifecycle stages. However, in the maintenance phase, the software needs to be corrected for errors and to be enhanced for the requests from users. These decrease the reliability of software. Software Reliability Growth Models (SRGMs) have been applied successfully to model software reliability in development phase. The software reliability in maintenance phase exhibits many types of systematic or irregular behaviors. These may include cyclic behavior as well as long-term evolutionary trends. The cyclic behavior may involve multiple periodicities and may be asymmetric in nature. In this paper, SGRM has been adapted to develop a reliability prediction model for the software in maintenance phase. The model is established using maintenance data from a commercial shop floor control system. The model is accepted to be used for resource planning and assuring the quality of the maintenance work to the user.

Keywords: Software reliability, Maintenance phase, Reliability growth model, Non homogeneous Poisson process.

1. INTRODUCTION

No piece of software is free of faults. As software is written by humans, errors are inevitable. Due to a wrong or incomplete specification, large problem complexity, lack of time, and other factors, mistakes are made; when this happens during developing software, it is known as an "error". The result of a human error being made is a software "fault", i.e. an incorrect piece of software. The software fault is usually called a software "bug". When the faulty software is executed, it can exhibit an unexpected behavior and produce an incorrect result. A software "failure" thus occurs.

Software reliability is one of the most important characteristics of software quality. It is defined as the probability of failure-free software operation in a specified environment for a specified period of time (Michael, 1996). Its measurement and management technologies during the software life-cycle are essential to produce and maintain quality/reliable software systems.

The software life cycle can be divided into two distinct phases (Moriguchi 1996): a) the initial development of software; and b) the maintenance of the software. Large and long term software applications will have frequent requirement changes in maintenance. The requirement changes are caused mainly by government regulations, market competitors, and changes in agreement between stakeholders, business environment and regulations in partner companies. In maintenance environment, new requirements are added to release, or existing requirements are deleted or modified. This will affect the reliability and quality of the software.

† : Corresponding Author

Software reliability models are used to estimate the reliability or the number of latent defects in a software product. However, there are large software systems which are maintained by the system users themselves. Such systems have been used for a considerably long period of time. The various details concerning the development stages are usually not known to the users who are responsible for the maintenance of these systems. In such a situation, the reliability models available for the development phase cannot be applied to the maintenance phase. Nevertheless, the defects predicting in the maintenance for such systems is of crucial importance so as to provide confidence to the system user and also for resource planning.

Notations

$N(t)$	cumulative number of software faults (or the cumulative number of observed software failures) detected in the time-interval (0; t)
$H(t)$	mean value function which indicates the expectation of $N(t)$, i.e. the expected cumulative number of faults detected (or the expected cumulative number of software failures occurred) in the time-interval (0; t), $H(t) = E[N(t)]$.
$\lambda(t)$	failure intensity function which indicates the instantaneous fault-detection rate at time t, $\lambda(t) = H'(t)$.
$R(x\|t)$	software reliability, the probability that a software failure does not occur in the time-interval $(t\,;t+x](x \geqq 0)$
a	initial number of faults present in the software prior to testing
a_m	initial number of faults present in the software prior to operation
b	parameter representing the fault-detection rate in development phase
b_m	parameter representing the fault-detection rate in maintenance phase
$n(t)$	the expected number of faults remaining in the system at time t

During the past two decades, a range of probabilistic models to predict the occurrence time of defects in a system have been applied successfully to model software reliability in development phase (Musa, 1998). These are known as Software Reliability Growth models (SRGM's) and they have focused almost exclusively on defect data from the system development phase. The software reliability in maintenance phase exhibits many types of systematic or irregular behaviors. These may include cyclic behavior as well as long-term evolutionary trends. The cyclic behavior may involve multiple periodicities and may be asymmetric in nature.

This paper examines the application of a particular class of models, known as non homogeneous Poisson process, to predict reliability and defect occurrence of a commercial Shop Floor Control system (SFC). Based on the assumptions and constraints of the models and the maintenance circumstances, a NHPP model is proposed. This model is accepted by the users and has been used for resource planning and assuring the quality of the maintenance work to the user.

2. NHPP MODEL

Numerous probabilistic models are available in software reliability engineering to predict the occurrence time of defects in a system. Software Reliability Growth Model (SRGM) represents the relationship between the time span of software testing and the number of detected errors as a process of growth in software reliability. This relationship could be described by a counting process.

The SRGM based on non homogeneous Poisson process (NHPP) is most commonly used. Among these models, Goel and Okumoto considered an NHPP as the stochastic process to describe the fault process (Okumoto *et al*., 1980), it is known as G-O model. Yamada *et al*., (1983, 1985, 1986, 1993) mo-dified the G-O model and incorporated the concept of testing-effort in an NHPP model to get a better description of the software fault phenomenon.

The general NHPP model is based on the following assumptions:

1. The expected number of failures observed by time t follows a Poisson distribution with a bounded and non-decreasing mean value function $H(t)$.
2. The number of software failures that occur in (t, $t+\Delta t$) is proportional to the number of remaining faults in the software, $a\text{-}H(t)$.
3. The fault removal process when failures are detected is instantaneous and perfect.

In the NHPP models, a non homogeneous Poisson process (NHPP) is assumed for the random variable $N(t)$, the cumulative number of observed software failures at time t. The time-dependent failure rate is proposed to follow an exponential distribution. The model is given by:

$$\Pr\{N(t)=n\} = \frac{[H(t)]^n}{n!}\exp\{-H(t)\} \qquad (1)$$

$$n = 0,\ 1,\ 2,\ \cdots,\ n$$

, where

$$H(t) = a\big(1-\exp(-bt)\big) \qquad (2)$$

, and

$$\lambda(t) \equiv dH(t)/dt = ab\exp(-bt) \qquad (3)$$

In the model, $\Pr\{A\}$ means the probability of event A. $H(t)$ is a mean value function indicating the expectation of $N(t)$, i.e., the expected cumulative number of software failures occurred in the time-interval (0; t). $\lambda(t)$ in Eq. (1) is called a intensity function which indicates the instantaneous fault-occurring rate at time t. a is the initial number of faults presented in the software prior to testing; and b is the parameter representing the fault-detection rate.

Several extrapolations of the NHPP model exist incorporating different assumptions. Musa's basic model (Musa *et al*., 1987), the G-O model, the delayed S-shaped model (Yamada *et al*., 1983), the Gompertz model (Ke-

Table 1. Typical NHPP models

NHPP Model	Mean value function *H(t)*	Fault intensity function	Remark
Exponential SRGM	$H(t)=a\left(1-e^{-bt}\right)$ $(a>0,\ b>0)$	$\lambda(t)=abe^{-bt}$	Failure-occurrence with a constant fault-detection rate at an arbitrary time.
Delayed S-shaped SRGM	$H(t)=a[1-(1+bt)e^{-bt}]$ $(a>0,\ b>0)$	$\lambda(t)=ab^2te^{-bt}$	Fault-detection process is descrybed by successive two phenomena, failure-detection and faultisolation process.
Inflection S-shaped SRGM	$H(t)=\dfrac{a(1-e^{-bt})}{(1+ce^{-bt})}$ $(a>0,\ b>0,\ c>0)$	$\lambda(t)=\dfrac{ab(1+c)e^{-bt}}{(1+ce^{-bt})^2}$	Failure-occurrence with mutual dependency of detected faults
SRGM : Software Reliability Growth Model *a*: initial number of faults presented in the software prior to testing *b*: parameter representing the fault-detection rate in development phase *c*: parameter of inflection-rate of detected fault			

cecioglu, 1991), and the Yamada Exponential model (Yamada *et al.*, 1986) are all based on an NHPP. Table 1 shows the typical NHPP models.

3. MODELS FOR MAINTENANCE PHASE

In 1989, Yamada *et al.* (1989 [1]) proposed the reliability assessment method for software products in the maintenance phase based on the empirical knowledge. They supposed that the software fault detection process in the maintenance phase followed an NHPP with a mean value function $a-Z_o(t)(t>\tau)$, where Z_o denotes the expected number of remaining faults in the maintenance phase.

In the other papers by Yamada *et al.* (1989 [2], 1991), they describe the fault detection process in the maintenance phase which is completely different from the stochastic process in the testing phase. They assumed that the probabilistic law in the maintenance phase was different from that in the testing phase.

Okamura *et al.* (2001) propose an accelerated life testing model for reliability assessment of software products in maintenance phase. In their papers, they described the difference of the software failure-occurrence phenomena between the maintenance and the testing phases based on the idea of the accelerated life testing model in hardware products.

4. PRESENTED MODEL FOR MAINTENANCE PHASE

In this paper, the NHPP model is used to predict the reliability of software system in maintenance phase from the view of users.

In the research conducted by Yamada *et al.* (1989 [2], 1991) and Okamura *et al.* (2001), an environmental coefficient *k* is introduced, which represents the differences between the environment in the testing phase (effort, test method, and so on) and the software maintenance environment, such as the frequency of use.

However, in most situations, the software systems are maintained by the system users themselves, and such systems may have been used for a considerably long period of time. Various details concerning the development stages are usually not known to the users who are responsible for the maintenance of these systems. In such a situation, the statistical information collected in the testing phase is not available to estimate the fault detection process in the maintenance phase.

In the presented model for maintenance environment is based on the following assumptions:

a: The software failure-occurrence phenomenon in maintenance phase can be described by an exponential SRGM.
b: The statistical information collected in the testing phase is not available to estimate the fault detection process in the maintenance phase.
c: A parameter b_m represents the fault detection rate per unit time in maintenance phase, regardless of the different environment between testing and maintenance phases.

Because an NHPP model is characterized by its mean value function, the proposed model is described by the mean value function defined in the following.

$$H(t)=a_m\left(1-e^{-b_m t}\right) \tag{4}$$

and

$$\lambda(t) \equiv dH(t)/dt = a_m b_m \cdot e^{-b_m t} \tag{5}$$

In the two equations above, a_m is the expected number of initial inherent faults before operation, b_m is the software failure occurrence rate per inherent fault in maintenance phase.

The expected number of faults remaining at the system testing time t which is obtained by taking expectation of random variable $\{N(\infty)\text{- } N(t)\}$ is

$$n(t) = E[N(\infty) - N(t)] = a_m - H(t) \tag{6}$$

Given that the software operation has been going on up to time t, the software reliability in the time-interval (t; $t+x$] ($x \geqq 0$) is expressed as:

$$R(x \mid t) = \exp[-\{H(t+x) - H(t)\}] \quad (t \geq 0,\ x \geq 0) \tag{7}$$

In the equation above, t is the total operating time of the last failure and x is the operating time measured from the last failure.

Moreover, the instantaneous mean time between software failures (instantaneous MTBF) is a useful measure for the frequency of software failure occurrence, and is given by:

$$MTBF_I(t) = \frac{1}{\lambda(t)} \tag{8}$$

5. METHOD FOR PARAMETER ESTIMATION

Suppose that k data pairs (t_k, y_k), $(k = 1, 2, \cdots, n)$ are observed during the system maintenance phase, where the total number of software failures observed in the time interval $(0, t_k]$ is y_k $(k = 1, 2, \cdots, n)$. The logarithmic likelihood function of the NHPP model with mean value function $H(t)$ of Eq. (4) is

$$\ln L = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln\{H(t_k) - H(t_{k-1})\} - H(t_n) - \sum_{k=1}^{n} \ln\{(y_k - y_{k-1})!\} \tag{9}$$

where, $H(t) = a_m \left(1 - e^{-b_m t}\right)$

The log likelihood in the case of the logarithmic likelihood function of the proposed model for maintenance phase with mean value function $H(t)$ of Eq. (4) can be written as:

$$\ln L = y_n \ln a_m + \sum_{k=1}^{n} (y_k \text{ - } y_{k-1}) \times \ln\{e^{-b_m t_{k-1}} \text{ - } e^{-b_m t_k}\} - k a_m (1 - e^{-b_m t_n}) - \sum_{k=1}^{n} \ln\{(y_k - y_{k-1})!\} \tag{10}$$

Maximizing Eq. (9) with respect to a_m and b_m, we have

$$a_m = \frac{y_n}{1 - e^{-b_m t_n}} \tag{11}$$

and

$$\frac{y_n t_n e^{-b_m t_n}}{1 - e^{-b_m t_n}} = \sum_{k=1}^{n} \frac{(y_k \text{ - } y_{k-1})(t_k e^{-b_m t_k} \text{ - } t_{k-1} e^{-b_m t_{k-1}})}{e^{-b_m t_{k-1}} - e^{-b_m t_k}}$$

The point estimates of a_m and b_m can be solved by using numerical methods such as Newton-Raphson method.

6. NUMERICAL EXAMPLE

A numerical example is used to illustrate the application of the proposed software reliability growth model for maintenance environment.

6.1 Analytic steps of model evaluation

In the process of model evaluation, it is necessary to build a series of steps. These steps can systematically analyze collected fault. The related steps and description are as follows:

1. Collect and analyze the fault data in a fixed period of time.
2. Selecting suitable software reliability model. In this paper, exponential SRGM is assumed.
3. Eq. (11) and Eq. (12) are used to determine model parameter a_m and parameter b_m.
4. Computing the mean value of faults experienced by using Eq. (4).
5. Suitability of model is examined by means of a Kolmogorov-Smirnov test.

6.2 Explanation of data

Figure1 shows the fault-detection count data which is obtained from real tracking database records of a shop floor control system (SFC) utilized in an electronics factory.

733 faults were detected in the first 701 days after system operation began. We randomly chose two different subsets of fault-detection count data, subset I and subset II from the data.

161 faults were detected in the subset I within 90 days (146th day to 235th day after operation), and 139 further faults were detected in the subset II within another 108 days (326th day to 433th day after operation).

6.3 Analysis of numerical example

The model parameters of each subset have been estimated by solving the likelihood equations given in Eq. (9):

Subset I: $\hat{a}_{m1} = 242.03,\ \hat{b}_{m1} = 0.012$

Subset II: $\hat{a}_{m2} = 189.46,\ \hat{b}_{m2} = 0.012$

The same value of $\hat{b}_{m1}$ and $\hat{b}_{m2}$ indicateds that the fault-detection rate in these two subsets of fault-detection count data is similar. It is because there is no apparent difference in the organization, environment and personnel factors between these two subsets of fault-detection count data.

The mean value function $\hat{H}_1$ and $\hat{H}_2$ can be estimated respectively as:

$$\hat{H}_1(t) = \hat{a}_1(1-e^{-\hat{b}_{m1}t}) = 242.03 \cdot (1-e^{-0.012t}) \tag{13}$$

$$\hat{H}_2(t) = \hat{a}_2(1-e^{-\hat{b}_{m2}t}) = 189.46 \cdot (1-e^{-0.012t}) \tag{14}$$

Figure 2 shows the estimated mean value function $\hat{H}_1$ and $\hat{H}_2$ respectively and the corresponding 90% upper and lower confidence limits. The estimated remaining software faults are shown in Figure 3.

The results from this investigation show that such a model for software maintenance data are promising.

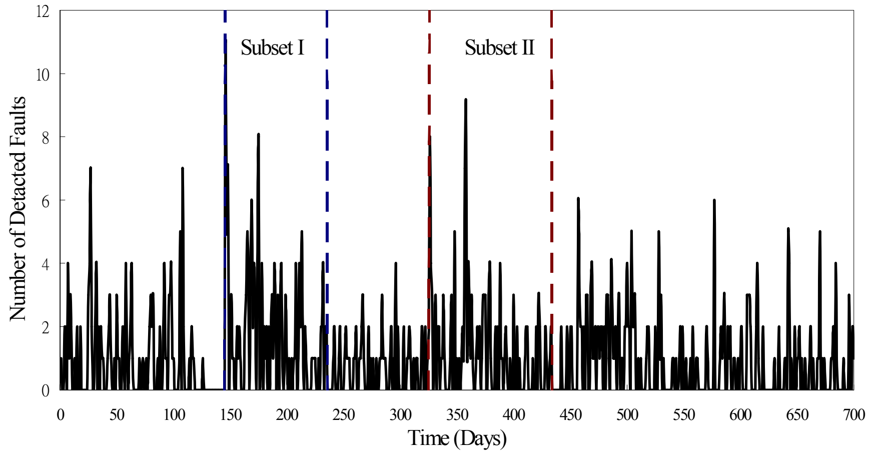
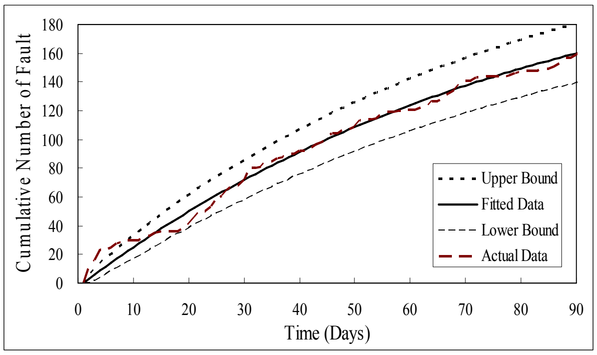
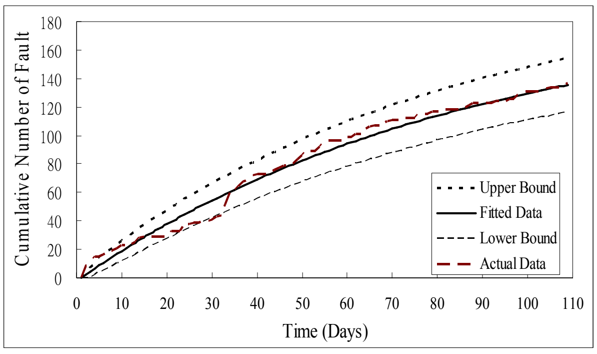


Figure 1. Fault-detection count data of a Shop Floor Control system

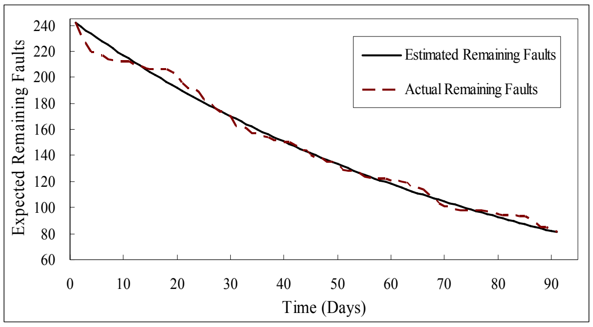


(a) Subset I (161 faults within 90 days)

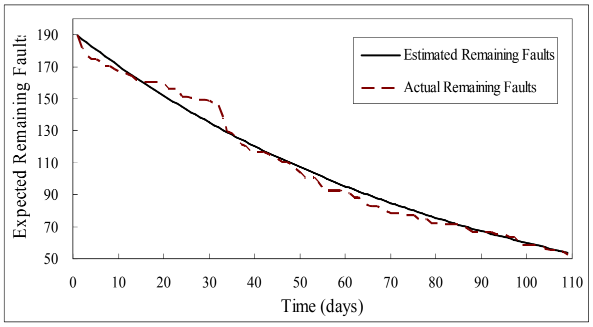


(b) Subset II (139 faults within 108 days)

Figure 2. Estimated mean value functions and the corresponding 90% upper and lower confidence limits



(a) Subset I (161 faults within 90 days)



(b) Subset II (139 faults within 108 days)

Figure 3. Estimated remaining software faults

7. DISCUSSION AND CONCLUSION

In this paper, a software reliability growth model based on an NHPP model have been proposed, in which the exponential software reliability growth model is incorporated, and the methods of quantitative reliability evaluation based on this model have been discussed. Numerical examples based on real data are also presented. However, there are several notorious points in this model. The assumption that a stable maintenance environment and perfect debugging in applying this model is unrealistic in many actual maintenance processes. Other environment factors exist. For example a time variant test effort should be considered for overcoming this deficiency. Furthermore, the analysis of the proposed model under imperfect debugging environment should be investigated.

ACKNOWLEDGEMENT

This work was supported in part by a research grant from Shu-Te University of Taiwan.

REFERENCES

Abran, A. and Moore, J. W. (2001), *Guide to the Software Engineering Body of Knowledge*, Trial Version, IEEE Computer Society Press.

ANSI/IEEE Standard STD-729 (1991), *Glossary of Software Engineering terminology*.

Kececioglu, D. (1991), *Reliability Engineering Handbook*, **2**, Prentice-Hall, Englewood Cliffs, N. J.

Lyu, Michael R. (1996), *Handbook of Software Reliability Engineering*, Los Alamitos, Calif.: IEEE Computer Society Press, New York: McGraw Hill.

Moriguchi, S. (1996), *Software Excellence*: *A Total Quality Management Guide*, Productivity Press.

Musa, J., Iannino A., and Okumoto K. (1987), *Software Reliability*: *Measurement, Prediction, Application*, McGraw-Hill, New York.

Musa J., Fuoco G., Irving N., Kropfl D, and Juhlin B. (1996), *The operational profile. In Lyu MR (ed), Handbook of software reliability engineering*, *(McGraw-Hill)*, chapter 5.

Musa, J. D. (1998), *Software Reliability Engineering*, McGraw-Hill, New York.

Okumoto, K. and Goel A. L. (1980), Optimum Release Time for Software Systems Based on Reliability and Cost Criteria, *Journal of Systems and Software*, **1**, 315-318.

Okamura, H., Tadashi, D., and Shunji, O. (2001), A reliability assessment method for software products in operational phase-Proposal of an accelerated life testing model, *Electronics and Communications in Japan*, *Part* 3, **84**(8), 25-33.

Yamada, S., Ohba, M., and S. Osaki (1983), S-Shaped Reliability Growth Modeling for Software Error Detection, *IEEE Transaction on Reliability*, **32**(5), 475-478.

Yamada, S. and Osaki, S. (1985), Cost-Reliability Optimal release policies for software systems, *IEEE Trans. on Reliability*, **34**(5), 422-424.

Yamada, S., Ohtera, H., and Narihisa, H. (1986), Software reliability growth models with testing effort, *IEEE Transaction on Reliability*, **35**(1), 19-23.

Yamada, S., Tanio Y., and Osaki, S. (1989), A software reliability evaluation method during operation phase, *Trans IEICE*, **J72-D-I**, 797-803, (in Japanese).

Yamada, S., Kimura, M., and Osaki, S. (1989), A note on software reliability evaluation during operation phase considering testing-effort expenditures, *Trans IEICE,* **J72-D-I**, 922-924, (in Japanese).

Yamada, S., Tanio, Y., and Osaki, S. (1991), Software reliability measurement and assessment methods during operation phase and their comparisons, *Trans IEICE*, **J74-D-I**, 240-248, (in Japanese).

Yamada, S., Hishitani, J., and Osaki, S. (1993), Software Reliability Growth Model with Weibull Testing Effort: A Model and Application, *IEEE Trans. on Reliability*, **R-42**, 100-105.

Yamada, S. (1993), Software reliability measurement during operational phase and its application, *Journal of Computer and Software Engineering*, **1**(4), 389-402.

Tamura, Y., Yamada, S., and Kimura, M. (2003), *Electronics and Communications in Japan*, Part 3, **86**(11), 13-20.