

Software Reliability - Different Model Analyses

Vladimir Zeljković¹, Nada Begenišić²

¹ LOLA Institute, Belgrade

² Mihailo Pupin Institute, Belgrade

I INTRODUCTION

We are witnessing our increasing dependence on software systems, as they are becoming more and more complex, thus harder to develop and maintain. Software systems are present in many safety-critical applications such as power plants, health care systems, air-traffic, etc. They all require high quality, reliability and safety.

The software system problems come from presence of the faults. The two basic activities can be performed in order to decrease the number of remaining faults in the system after entering into the operation:

- Fault prevention
- Fault detection and removal through inspections/reviews and testing

There are numerous methods and techniques dealing with fault prevention. There are also many software-testing methods and tools and they are valuable for the process of reliability growth, and for achieving the desired software reliability.

In this paper we are going to present and compare two approaches to reliability estimation based on the failure data on detected faults in the system during development and/or maintenance. The first approach applies the nonhomogeneous Poisson process (NHPP) methodology on selected mathematical model to estimate the model parameters and reliability. The second approach is classical reliability growth modeling (RGT).

Mathematical model is illustrated by numerical data of software for process control, developed in LOLA Institute. During development, the data are collected and parameters of modified G-O and Duane models are estimated. It is calculated the total number of errors, remaining number after given testing time, error rate and MTTF, and reliability.

II SOFTWARE RELIABILITY ANALYSES

Software reliability is defined as *the probability of failure-free software operation for a specified period of time in a specified environment* [1]. So, the main concern is centered around software faults, their effect on the system and the remaining number of faults; system failures, the way of detecting failures, time between failures and failure rates, as well as the confidence in the performed estimates.

This reliability growth method leads to decrease of system failure rate and increase of system reliability with time. The

cumulative number of errors, detected and corrected, increases with time.

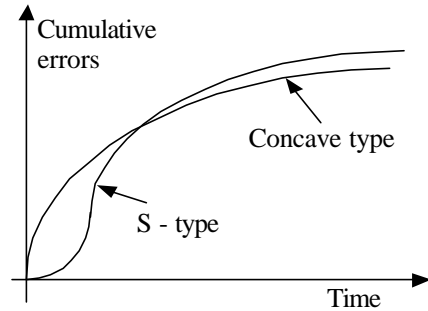


Figure 1. Concave and S – type shape of the cumulative error growth curve

Two types of software reliability growth models are typical: concave and S - type, shown in Figure 1. The cumulative number of errors is a function that increases with time and asymptotically approaches the total number of errors N . By systematically detecting failures and removing faults, it is possible to achieve the required system $MTTF$ and failure rate $\theta_R=1/\lambda_R$, before the software is shipped to the customer. For a specific software system it is necessary to find out the mathematical model and the parameters that best fit the software reliability growth.

III NONHOMOGENEOUS POISSON PROCESS

For analytical modeling, it is convenient to apply the nonhomogeneous Poisson process (NHPP) for cumulative number of failure $N(t)$ observed over a period of time t [2,3,4]. The Poisson mass function is

$$P_r \{N(t) = k\} = \frac{[m(t)]^k}{k!} e^{-m(t)}, \quad k = 0, 1, 2, \dots \quad (1)$$

where $m(t)$ is the expected number of failures observed over a period of time t . At the beginning of testing, $m(t=0) = 0$, and $m(t) \rightarrow N$ as the time $t \rightarrow \infty$.

Expected number of failures after elapsed time t is

$$n(t) = E\{m(\infty) - m(t)\} \quad (2)$$

The failure rate is determined as

$$\lambda(t) = \frac{dm(t)}{dt} \quad (3)$$

The reliability, defined as the probability that the software system will operate for a time interval T , after elapsed time t , is

$$R(T|t) = e^{-\int_t^{t+T} \lambda(\tau) d\tau} = e^{-\{m(t+T)-m(t)\}} \quad (4)$$

where T is defined as the mission interval that started at t . Many software reliability growth models are suggested [5]:

- Gompertz $m(t) = a(b^{c^t})$
- Weibull $m(t) = a(1 - e^{-bt^c})$
- Pareto $m(t) = a[1 - (1 + t/\beta)^{1-\alpha}]$

For a specific software, it is necessary to estimate the parameters of the model $\delta_1, \delta_2, \dots, \delta_n$, from test pairs $\{i, t_i\}$ $i = 1, 2, \dots, j$. The likelihood function is expressed as

$$L = \prod_{i=0}^{j-1} \frac{[m(t_{i+1} - t_i)]^i}{i!} e^{-m(t_{i+1} - t_i)} \quad (5)$$

The maximum likelihood estimate $\hat{\delta}$ is obtained by solving the system of equations

$$\frac{\partial l(\delta|t, k)}{\partial \delta_j} = 0, (j = 1, 2, \dots, l), \quad (6)$$

As a measure of goodness -of -fit for different models, the least square is compared

$$LS = \sum_{i=1}^j [k_i(t_i) - \hat{m}(t_i)]^2 \quad (7)$$

The maximum likelihood estimates $\hat{\delta}$ follows the joint normal distribution if the number of sample size, test pairs $\{i, t_i\}$, increases [3]. The lower and upper confidence bounds on the expected number of failure is

$$m_{U/L}(t) = m(\hat{\delta}, t) \pm K_\gamma \sqrt{\text{Var}[m(\hat{\delta}, t)]} \quad (8)$$

where:

$m_{U/L}(t)$ - Upper and lower confidence bounds on the cumulative number of errors,

$m(\hat{\delta}, t)$ - Model estimation,

K_γ - The value of $100(1-\gamma)/2$ percent of standard normal distribution,

$\text{Var}[m(\hat{\delta}, t)]$ - Variance.

The variance is determined by

$$\text{Var}[m(\delta, t)] = \begin{bmatrix} \frac{\partial m(\delta, t)}{\partial \delta_1} & \dots & \frac{\partial m(\delta, t)}{\partial \delta_l} \end{bmatrix} \cdot \text{Cov}[m(\delta, t)] \cdot \begin{bmatrix} \frac{\partial m(\delta, t)}{\partial \delta_1} \\ \vdots \\ \frac{\partial m(\delta, t)}{\partial \delta_l} \end{bmatrix}$$

defined for $\hat{\delta}_1, \hat{\delta}_2, \dots, \hat{\delta}_l$. The covariance matrices is

$$\text{Cov}[m(\delta, t)] = I^{-1} \quad (9)$$

where the elements of matrices are

$$I_{ij} = E \left[-\frac{\partial^2 l(\delta)}{\partial \delta_i \partial \delta_j} \right], i, j = 1, 2, \dots, l \quad (10)$$

The above procedure gives the basic elements of reliability growth analysis for any software, like the estimate of total number of errors, remaining number of errors after test time t , $MTTF$ and $R(t)$.

IV DUANE'S MODEL

The Duane failure rate improvement model is:

$$\lambda_C = \frac{N_F}{t_T} = kt_T^{-\alpha}, \quad (11)$$

where:

λ_C - average estimate of cumulative failure rate,

N_F - total failures in time t_T ,

t_T - total accumulated operating hours,

k - constant representing cumulative failure rate at $t_T = 1$,

α - improvement rate constant.

Taking the log of above model gives the linear equation of the form:

$$\log \lambda_C = \log k - \alpha \log t_T. \quad (12)$$

with tow parameters k, α , easy to estimate by LS method.

V A CASE STUDY

The complex software system for process control is considered [5]. The data are obtained by monitoring and recording naturally occurring errors and failures in a running system under different workloads, Table 1.

Table 1. Failure Data for the Case Study

T_i [months]	No of errors in the interval
T_0	
$T_1 = 1$	2
$T_2 = 2$	4
$T_3 = 3$	5
$T_4 = 5$	8

$T_5 = 6$	11
$T_6 = 7$	12
$T_7 = 8$	15
$T_8 = 31$	16
$T_9 = 48$	16

First, we applied the modified G-O model:

$$m(T) = \alpha(1 - (1 + \beta t)e^{-\beta T}) \quad (13)$$

The estimated parameters based on $(T_i, k_i(T_i))$ pairs and maximum likelihood function are

$$\hat{\alpha} = 16.36 \text{ and } \hat{\beta} = 0.3880 \text{ with } L_{min} = 7.2758.$$

The model for the expected number of errors, shown in Figure 2, is

$$m(T) = 16.36(1 - (1 + 0.388t)e^{-0.388T})$$

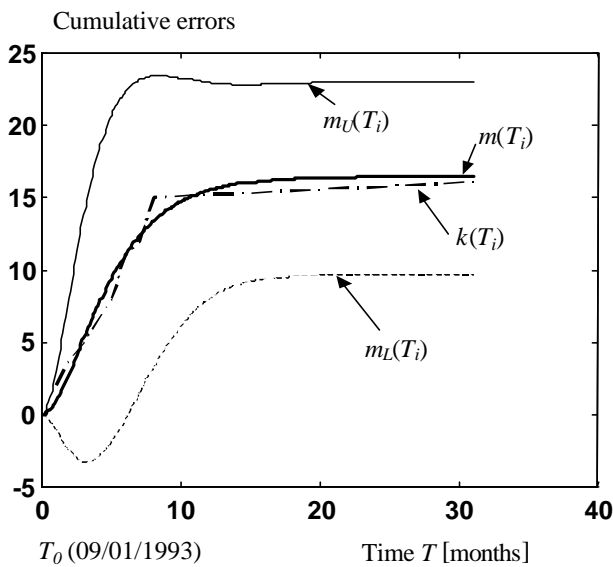


Figure 2 Failure Data, G-O model and confidence bounds
Upper and lower bounds were determined for $\gamma=0.9$.

The failure intensity is presented in Figure 3.

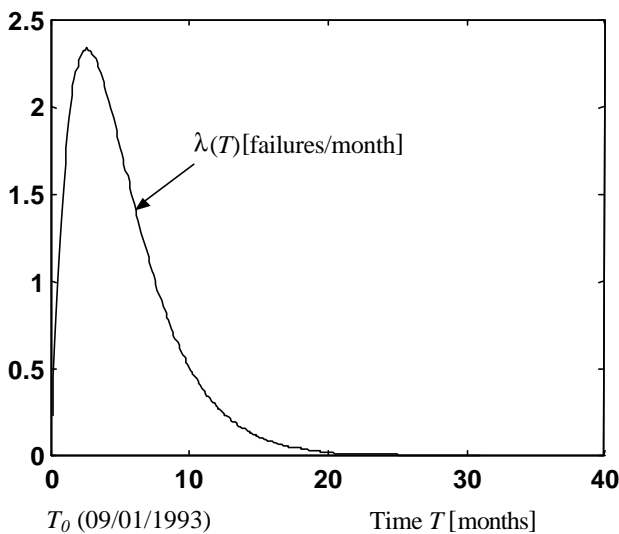


Figure 3 Failure rate – Modified G-O model

The least square estimate of the Duane model parameters give the values:

improvement rate $\alpha = -0.4697$ and $\log k = 1.1268$.

The recorded data and regression line are presented at fig. 3.

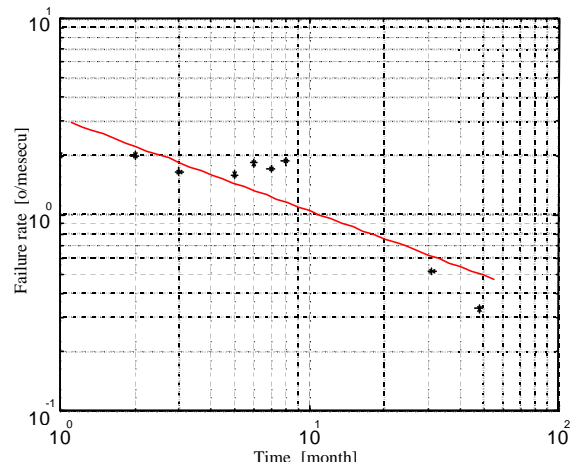


Figure 3. Failure rate - recorded data and regression line

The failure rate plot of recorded data, Duane model and modified G-O model is given at fig 4.

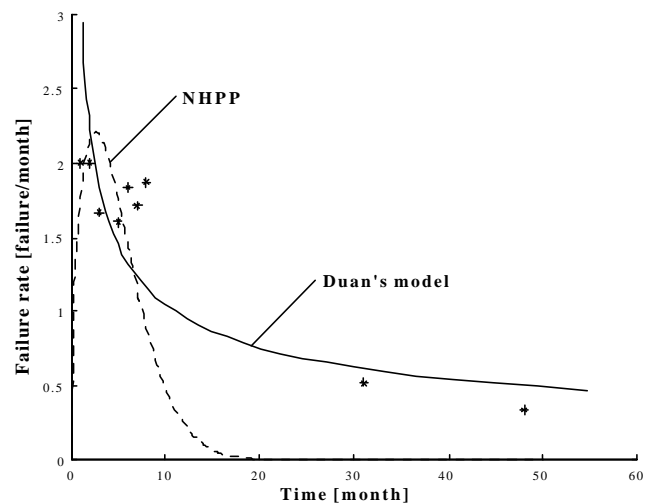


Figure 4 The failure rate plot of recorded data, Duane model and modified G-O model

Covering this tow models we can see the (significant) difference.

The modified G-O model follows pretty well the cumulative number of failures (errors in the software) in the time. It gives the estimates of total software error and possibility of calculation the MTBF, reliability as probability that software system will be functioning with no error for given missom after certain age of operation, confidence interval on the cumulative errors, the instant of time when next error will happened, and the other analysis. But, it should be noted that NHPP approach is complex analysis requiring extensive mathematical/statistical background.

On the other hand, the RGT procedure (in this case the application of Duane model) is simple for application. The

cumulative data and regression line estimation of failure (software errors) rate and MTBF is easy to obtain.

The straight compression of Duane model and modified G-O model, fig 4, shows the lower failure rate in the case of modified G-O model. This is to be expected because the Duane model take into account the history (total failures) in failure rate estimation for given instant of time, while the parameters estimation in modified G-O model takes only actual failure data for any instant of time. So, Duane model is more conservative than modified G-O model.

VI CONCLUSION

Software reliability can not be calculated during the design phase. Nevertheless, if adequate data on system failures (software error) is collected throughout the project especially during testing/ verification/ validation and maintenance (operational) phase, the same models for estimating reliability parameters, such as the expected number of failures in a certain period of time, failure intensity, the expected time of the next failure, etc., could be applied to software systems, as well. In this paper, we considered classical RGT approach and more complex NHPP approach to estimate model parameters and real data analysis. In the presented case study of a complex real-time software system we have shown that both, Duane and modified G-O, models give approximations, based on the failure data collected during the maintenance phase, that match the real data. But, this two model shows the difference in failure rate estimates in a way that the Duane model is more conservative.

In this way it is possible to supply valuable information and confidence in the system both to the customer and to management and software development team. This case study also showed the importance of reliability estimation both during the development/testing, proving the product readiness to be delivered to the customer, and during the operational phase, for validating the product reliability.

REFERENCES

- [1] ANSI/IEEE *Standard Glossary of Software Engineering terminology*, STD-729-1991.
- [2] Goel, L., A., *Software Reliability Models: Assumptions, Limitations, and Applicability*, IEEE Transaction on Software Engineering, Vol., SE-11. NO. 12, December 1985, pp 1411-1423.
- [3] Yamada, S., Ohba, M., and Osaki, S., : *S - Shaped Reliability Growth Modeling for Software Error Detection*, IEEE Transaction on Reliability, Vol. R-32. NO. 5, December 1983, pp 475-478.
- [4] Yamada, S., and Osaki, S.,: *Software Reliability Growth Modeling: Models and Applications*, IEEE Transaction on Software Engineering, Vol., SE-11. NO. 12, December 1985, pp 1431-1437.
- [5] V. Zeljkoviã; N. Radovanoviã : *Software Reliability - A Case Study*, Communications in dependability and quality management, An International Journal, Vol 1, 2000, pp 25-33