M. Xie, G. Y. Hong and C. Wohlin, "A Practical Method for the Estimation of Software Reliability Growth in the Early Stage of Testing", Proceedings IEEE 7th International Symposium on Software Reliability Engineering, pp. 116-123, Albuquerque, USA, 1997.

A Practical Method for the Estimation of Software Reliability Growth in the Early Stage of Testing

M. Xie, G.Y. Hong
Dept of Industrial and Systems Engineering
National University of Singapore,
Kent Ridge, Singapore 119 260

C. Wohlin
Dept of Communication Systems
Lund Institute of Technology, Box 118
S-221 00, Lund, Sweden

Abstract

The traditional approach of reliability prediction using software reliability growth models requires a large number of failures which might not be available at the beginning of the testing. The commonly used maximum likelihood estimates may not even exist or converge to a reasonable value. In this paper, an approach of making use of information from similar projects in order to obtain an early estimation of one model parameter for a current project is studied. As most of the two-parameter reliability growth models contains one parameter related to the number of faults in the software and a reliability growth rate parameter related to the testing efficiency, information from a similar project can used to estimate the reliability growth rate parameter and the limited failure data from initial testing is used to estimate the other parameter. Our case study shows that this approach is very easy to use as the estimation does not require a numerical algorithm and it always exists. It is also very stable and when the maximum likelihood estimates exist and are reasonable, our approach gives values very close to that, and the approximate confidence interval is overlapping for most cases.

1. Introduction

Reliability of software is of growing importance in systems engineering and analysis today, see e.g., Musa et al. [10] and Lyu [7]. For complex system containing software and hardware, most system engineers know what to do with the hardware, but are not aware of means to predict the reliability of software. In fact, estimating software reliability accurately is not easy although there are a number of software reliability growth models that can be used for the analysis of software failure data in testing. They usually require a large number of failures to get a reasonably accurate estimate of the reliability. It is useful to be able to estimate the reliability growth in the testing phase at an early stage, so that appropriate planing regarding the release time, price and additional resource can be made.

In this paper, we present an approach for an early estimation of software reliability in the testing phase. Particularly, we focus on the problem of the non-existence of maximum likelihood estimates which makes the traditional approach not applicable in many cases. This problem has been addressed in a number of papers, see, e.g., Knafl [3], Knafl and Morgan [4] and Hossain and Dahiya [5], but there is no applicable solutions to it except that we are warned of the possibility of non-existence and advised not to estimate the reliability until we have a sufficient number of failure data making the estimation possible.

Our method relies on the use of information from similar projects that have gone through the testing, so that some information about the testing efficiency and reliability growth rate can be obtained. The Goel-Okumoto model is used in this paper as an example, although the approach is equally applicable to other two-parameter software reliability growth models. Our approach gives similar results as the traditional estimation approach using the maximum likelihood method without using information from previous projects, except that our approach is much easier to use and no numerical technique is required. Furthermore, our approach gives a much more stable

1071-9458/97 $10.00 © 1997 IEEE

estimate of the parameters, and last but not least, it can be used to estimate the reliability growth at a very early stage of the testing phase. That is, there is no problem with the existence of the solution and we do not have to wait until we have a large number of failures for the estimation.

This paper is organized as follows. In Section 2 we present a detailed discussion of the problem with the maximum likelihood estimation when used to software reliability growth models. In Section 3, an approach for the estimation of parameters based on their physical interpretation is presented and discussed. The approach is very easy to use in practice. A numerical example is shown in Section 4 to highlight the applicability of the approach. Furthermore, in Section 5, we derive the statistical confidence interval and present a comparative study with the traditional approach.

2. Problem with the traditional approach

The most commonly used approach for the estimation of software reliability is by using a software reliability growth model, or a nonhomogenuous Poisson process (NHPP) model with the mean value function $\mu(t)$, see, e.g., Xie [13] and Musa [10]. For a good reference on NHPP software reliability growth models, see Yamada and Osaki [14]. These models usually contain two or more parameters. The parameters in the model can be estimated using the maximum likelihood method based on the number of failures per interval or the time between failure data.

Suppose that an observation interval $(0, t_k]$ is divided into a set of subintervals $(0, t_1], (t_1, t_2], ..., (t_{k-1}, t_k]$, the number of failures per subinterval is recorded as $n_i (i = 1,2,..., k)$ with respect to the number of failures in $(t_{i-1}, t_i]$.

Note that because the available data that we are using in our case study is inform of failure per period data, the approach will be presented based on this type of data although it can be easily modified for the case of exact failure time data such as the execution time data.

The likelihood function for a NHPP model with mean value function $\mu(t)$ given the number of failure per period data is

$$L(n_1,...,n_k) =$$

$$= \prod_{i=1}^{k} \frac{\{\mu(t_i) - \mu(t_{i-1})\}^{n_i}}{n_i!} \exp\{-[\mu(t_i) - \mu(t_{i-1})]\}$$

By taking the natural logarithm of both sides, we have

$$\ln L(n_1,...,n_k) =$$

$$= \ln \prod_{i=1}^{k} \frac{\{\mu(t_i) - \mu(t_{i-1})\}^{n_i}}{n_i!} \exp\{-[\mu(t_i) - \mu(t_{i-1})]\}$$

$$= \sum_{i=1}^{k} \ln \frac{\{\mu(t_i) - \mu(t_{i-1})\}^{n_i}}{n_i!} \exp\{-[\mu(t_i) - \mu(t_{i-1})]\}$$

$$= \sum_{i=1}^{k} \{n_i \ln[\mu(t_i) - \mu(t_{i-1})] - [\mu(t_i) - \mu(t_{i-1})] - \ln n_i!\}$$

The partial derivative of this with respect to each model parameter can be taken and set to be zero, so that a set of likelihood equations are obtained. Solving the equations, we can obtain the so-called maximum likelihood estimates of the model parameters.

However, although this is a sound statistical approach, a problem with this approach when it is applied in practice is that there might be no solution to the maximum likelihood equations, especially at the early stage of software testing. This problem has been noticed in Knafl [3] and Morgan and Knafl [9], among other. It also happens frequently that when the solution exists, it is not stable in a sense that when new data is collected, the revised estimates are totally different from previous estimate, making the use of it in planning very difficult.

A common procedure recommended or used in practice is to wait until we have a large number of failures and the estimation should then be carried out. On the other hand, we may not know for how long we should wait and when an estimate can be obtained, it is possible that after the next time interval, there is again no solution to the likelihood equations. This, in reality, implies that we have to wait until the end of testing before an estimation is possible although in practice, we need earlier

prediction for better planning of the redevelopment process, see, e.g., Smidts et al. [12].

In a number of papers, Knafl [3] and Knafl and Morgan [4] have tackled this problem from a statistical viewpoint. They have derived conditions for the existence of the maximum likelihood estimates and it can be used to determine whether the parameter should be estimated. However, this does not address the problem of non-existence of the maximum likelihood estimates and some alternative solutions are needed for practical applications.

The problems can be illustrated with an actual data set using the Goel-Okumoto model. The Goel-Okumoto model is a simple nonhomogenuous Poisson process (NHPP) model, see e.g., Goel and Okumoto [2], Nara et al. [11] and Yamada and Osaki [14]) with the following mean value function

$$\mu(t) = a(1 - e^{-bt})$$

In using this model, the parameter a is interpreted as the number of initial faults in the software and the parameter b is the fault detection rate which is related to the reliability growth rate in the testing process. The corresponding failure intensity function is given by

$$\lambda(t) = abe^{-bt}$$

For the Goel-Okumoto model, the likelihood equations are

$$\begin{cases} \dfrac{\partial \ln L}{\partial a} = \sum_{i=1}^{k} \{\dfrac{n_i}{a} + e^{-bt_i} - e^{-bt_{i-1}}\} = 0 \\ \dfrac{\partial \ln L}{\partial b} = \sum_{i=1}^{k} (\dfrac{n_i}{e^{-bt_{i-1}} - e^{-bt_i}} - a)(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}}) = 0 \end{cases}$$

Solving the above equation, we get

$$\begin{cases} a = \dfrac{\sum_{i=1}^{k} n_i}{1 - e^{-bt_k}} \\ \sum_{i=1}^{k} (\dfrac{n_i}{e^{-bt_{i-1}} - e^{-bt_i}} - \dfrac{\sum_{i=1}^{k} n_i}{1 - e^{-bt_k}})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}}) = 0 \end{cases}$$

Since the second equation above is nonlinear, we cannot find an analytic solution and it must solved numerically.

In fact, for the simple Goel-Okumoto model, there might be no solution at all for the likelihood equations. The solution, when it exists, is often not stable in a sense that when another failure is observed, new estimates can be totally different from the previous one. These problems are also common for other software reliability growth models. In fact, the likelihood function is very flat in the region of an optimal solution. Furthermore, there are convergence problem depending on the search algorithm we use.

As an illustration, we apply the method to a set of software failure data in Table 1 which is collected from the testing of a large telecommunication software system.

Table 1. Number of failures per week for a telecommunication software.

Week	Failures	Week	Failures
1	3	15	7
2	3	16	0
3	38	17	2
4	19	18	3
5	12	19	2
6	13	20	5
7	26	21	2
8	32	22	3
9	8	23	4
10	8	24	1
11	11	25	2
12	14	26	1
13	7	27	0
14	7	28	1

The maximum likelihood estimates are given in Table 2. It can be noted that the estimates have both instability and non-existence problems and it is not until 24th week, the estimates start to stabilize. This is very late in the testing phase and decision-making is difficult as probability the release date has already be decided by that time.

Table 2. The ML estimates of the model parameter for the data set in Table 1. Note that prior to the 12th week, the ML estimates do not exist.

Week	a_2	b_2
11	NA	NA
12	211.3	0.180
13	NA	NA
14	NA	NA
15	NA	NA
16	269.4	0.0924
17	483.5	0.0335
18	NA	NA
19	NA	NA
20	NA	NA
21	276.8	0.0771
22	269.5	0.0819
23	NA	NA
24	260.22	0.0878
25	257.51	0.0896
26	257.39	0.0923
27	251.28	0.0991
28	249.22	0.0999

3. An alternative approach for estimation of the model parameters

In this section, we present an approach for early estimation of the model parameters, and hence enable us to estimate software reliability earlier. This approach and its background are discussed here and a numerical example to illustrate its application is presented in Section 4 using an actual case study. Furthermore, we also derive the confidence interval for the estimates and compare with the traditional maximum likelihood estimates in Section 5.

Software reliability estimation could be made much easier if there is a way to obtain an estimate for b. From Section 2, we know that the two parameters a and b of the Goel-Okumoto model can be estimated by solving the likelihood equations. Parameter a is a simple function of parameter b, but the equation for solving the parameter b is nonlinear and it can only be solved numerically.

It can be noted that the parameter b of the Goel-Okumoto model can be interpreted as the testing efficiency and it is related to the reliability growth rate in the testing. Hence, if the same well-defined process, and test methods and tools are used, it can be expected that the value of b remains the same. If we have earlier similar projects or earlier versions of the software tested in an similar environment, we could probably assume that the value of b is stable across software projects.

It should be noted here that this is the assumption we are using in this preliminary study and it has to be validated in practice. However, because of the interpretation of b as error detection rate, which depends on how the testing is carried out, for similar projects which means that the software systems are developed and testing during similar environment, we can expect the value of this parameter to be close to each other.

Denote the parameters in the Goel-Okumoto model by (a_1,b_1) and (a_2,b_2) for system one and two, respectively. We assume that the information about system one is available, and system two is developed in a similar environment as system one. In this case, the maximum likelihood estimate for parameter a_2 can be easily determined as follows:

$$a_2 = \frac{\sum_{i=1}^{k} n_i}{1 - e^{-b_2 t_k}}$$

given that $b_1=b_2$.

It can be noted that when the reliability growth rate parameter is estimated using earlier information, the estimate of the other parameter is more straightforward. No numerical procedure is needed and whenever we have some failure data, the reliability growth curve can be plotted. Although it might not be very accurate at the beginning, it is much more useful than the traditional maximum likelihood estimation without using prior information for which the numerical solution may not converge or there might be no solution at all.

It can be expected that when more data are available, we may still have to revise our estimate. However, as no numerical solution is needed, the revision can be carried out easily.

Furthermore, as can be seen in the example later, our estimation is also much more stable at a later

stage when we have sufficient amount of data for the estimation using the maximum likelihood method. In practice, estimates that vary greatly might cause a lot of practical problems when planning has to be constantly changed and a stable estimate can be more trustworthy.

4. A case study

Here we present a case study for which we have used the information from an earlier project for the estimation of the reliability of a current project. The current system is a telecommunication software and it has been tested for 28 weeks.

The actual failure data from the second project is given in Table 1. In Table 2, we have given the maximum likelihood estimates of the parameters and we have noticed that the estimates are not stable at all and in many instances, they do not even exist. Even in cases when they exist, they are not stable in a sense that totally different values can be obtained and this makes the use of the estimation technique and the result not convincing.

For this particular system, the information for an earlier system is available. The earlier system was developed in a similar environment and testing using the same methodology and hence, we could expect that the testing efficiency or reliability growth rate to be close to that of the current system. For the earlier system, the number of failures per week data is shown in Table 3. The maximum likelihood estimates of a and b are 199.48 and 0.098076, respectively. These values are based on the complete data set available by the end of testing and hence can be considered reasonably accurate.

After the release of the first system, the current system started to be developed and it has been tested for 28 weeks. The complete data set is given in Table 1. The traditional approach is to analyze this data set independently. However, since we have the information from the earlier system, we should make use of such information in making reliability prediction. The method presented in Section 3 is used here.

Note that in this case

$$b_2 = b_1 = 0.098076$$

and the estimate of parameter a_2 for the second project can be determined by

$$a_2 = \frac{\sum_{i=1}^{k} n_i}{1 - e^{-b_2 t_k}}$$

Table 3. Number of failures per week for a previous system.

Week	Failures	Week	Failures
1	2	26	1
2	11	27	0
3	18	28	0
4	10	29	0
5	12	30	1
6	4	31	0
7	28	32	0
8	6	33	0
9	7	34	0
10	6	35	1
11	17	36	0
12	31	37	1
13	8	38	0
14	7	39	0
15	10	40	0
16	2	41	0
17	2	42	0
18	0	43	1
19	3	44	1
20	2	45	0
21	1	46	0
22	1	47	1
23	1	48	0
24	0	49	0
25	1	50	1

Table 4 shows the estimate of parameter a_2 given that b_2=b_1 at different time points during the testing phase. It is clear that the values are much more reasonable and useful than that presented in Table 2.

Table 4. Parameter a_2 estimation assuming $b_2 = b_1$.

Week	a_2 (with $b_2=b_1$)
11	262.12
12	270.32
13	269.23
14	269.20
15	270.01
16	262.70
17	258.86
18	256.97
19	254.48
20	264.58
21	268.20
22	264.58
23	261.39
24	258.57
25	256.05
26	253.82
27	251.83
28	250.05

A statistical comparison with the case of no prior information will be made in the next section. It can be noted at this time that the estimation of a is very straightforward in this case and there it no need for solving a nonlinear equation for which the convergence might be a problem. Although there seems to be a decreasing trend in this case, we will show that the estimate of a is actually more stable compared with the case when we the information from the earlier project is not used.

5. A comparative study and the confidence interval

In order to compare the results with the case when no prior information from earlier projects are assumed, we estimate parameter a_2 and b_2 of the Goel-Okumoto model for system 2 for the latest eight weeks (week 21 to week 28). The results are presented in Table 4. Comparing Table 2 and Table 4, it can be seen that the estimate using prior information is more stable. Interestingly, our approach gives an estimate of a that is very close to the overall maximum likelihood estimate without using any earlier information.

For the sake of comparison, a 95% confidence interval for the prediction of parameter a can be constructed. To obtain the confidence limits, we can calculate the asymptotic variance of the maximum likelihood estimator of parameter a which is the inverse of the local Fisher information, see e.g., Lawless (1982),

$$Var(\hat{a}) = -1/\left[\frac{\partial^2 \ln L}{\partial a^2}\right]_{a=\hat{a}} = \hat{a}^2 / \sum_{i=1}^{k} n_i$$

For a given confidence level α, the two-sided confidence interval for parameter α is

$$a_L = \hat{a} - Z_{\alpha/2}\sqrt{Var(\hat{a})}$$

and

$$a_U = \hat{a} + Z_{\alpha/2}\sqrt{Var(\hat{a})}$$

where $Z_{\alpha/2}$ is the $[100(1+\alpha)/2]$th standard normal percentile.

For the given a =0.05, we have that

$$Z_{\alpha/2} = Z_{0.025} = 1.96$$

So the 95% confidence intervals for parameter a_2 using parameter $b_2 = b_1$ are listed in Table 5.

As can be seen from Table 5, because of the limited data available, the confidence intervals are usually very wide. This actually stresses the applicability of this simple earlier estimation approach: the maximum likelihood estimates based on limited information is not accurate and any reasonable alternatives will probably not be very bad. In reliability estimation, we should use confidence interval as often as possible because that will give a better indication of the level of reliability achieved.

5. Discussion

In this paper we have presented a practical approach for the estimation of software reliability at

the earlier stages of software testing. it overcomes the problem with the existence of the estimate when using the maximum likelihood method. As it makes use of the interpretation of the model parameter and information from similar projects, it is easy to accept and the results are very promising based on the case study in which we have used the approach and compared with the traditional approach.

Table 5. The 95% confidence interval for parameter a_2 assuming $b_2 = b_1$.

Week	a ($b_2=b_1$)	a_U	a_L	a (raw data)
12	270.32	309.07	219.29	211.32
14	269.20	306.41	231.58	NA
16	262.70	298.40	231.98	269.42
18	256.97	291.49	226.99	NA
20	254.48	289.84	222.46	NA
22	264.58	298.49	230.68	NA
24	258.57	291.69	225.44	260.22
26	253.82	286.34	221.30	257.39
28	250.05	282.09	218.01	249.22

The Goel-Okomoto model, which is commonly used in practice, contains two parameters with clear physical interpretations. Our approach looked into ways to provide earlier estimation of some model parameter and the estimation is shown to be clearly simplified and more stable. The stability of the estimates is actually of great concern for many software reliability models, see e.g., Zhao and Xie [15] and Bondi and Simonetti [1].

There are different ways of using prior information. One possibility is to adopt a Bayesian approach, see Littlewood and Verrall [7] and Xie [13]. However, it requires the specification of a prior distribution and the numerical computation is much more involved. In comparison, our approach is based on the direct judgment of the similarity of the development approach and the approach is easier to apply for practicing engineers.

Although our numerical results are valid for the data sets used and more empirical studies are needed to determine the exact condition, such as the exact meaning of "similarity", the approach can be used, it is an indication that such an approach is feasible in practice. For practitioners, this approach opens a new way to make use of information from different, but similar projects. When the development process is stable, our approach can be adopted. As judgments are based on earlier experience, our approach provides an objective way of making use of available information.

Acknowledgments

Part of this research was funded by the Swedish National Board for Industrial Technical Development (NUTEK). The work is part of the "Software Engineering Research Triangle" initiative (grant: P7816-1). Part of this research is also supported by the National University of Singapore under research grant RP 3950643.

References

[1] P. Bondi and G. Simonetti, "Evaluating the reliability of the software of a switching system with a multi-variable model", *Software Testing, Verification and Reliability*, **5**(3), 181-202 (1995).

[2] A.L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures", *IEEE Transactions on Reliability*, **R-28**, 206-211 (1979).

[3] G.J. Knafl, "Solving maximum likelihood equations for two-parameter software reliability models using grouped data", *Proc. of the 3rd Int. Conf. on Software Reliability Engineering*, North Carolina, Research Triangle Park, IEEE Computer Press, pp.205-213, (1992).

[4] G.J. Knafl and J. Morgan, "Solving ML equations for 2-parameter Poisson-process models for ungrouped software-failure data", *IEEE Transactions on Reliability*, **R-45**(1), 42-53 (1996).

[5] S.A. Hossain and R.C. Dahiya, "Estimating the parameters of a non-homogeneous Poisson process model for software reliability", *IEEE Transactions on Reliability*, **R-42**(4), 604-612 (1993).

[6] Lawless, J.F., *Lifetime Data Analysis*, Wiley, New York, 1996.

[7] B. Littlewood and J.L. Verrall, "A Bayesian reliability growth model for computer software", *Applied Statistics*, **22**(3), 332-346 (1973).

[8] M. Lyu, (Editor), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996.

[9] J.A. Morgan and G.J. Knafl, "Residual fault density prediction using regression methods", *Proc. of 7th Int. Symp. on Software Reliability Engineering*, New York, USA, IEEE Computer Press, pp.87-92 (1996).

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

[11] T. Nara, M. Nakata and A. Ooishi, "Software reliability growth analysis - application of NHPP models and its evaluation", *Proc. of the 6th Int. Conf. on Software Reliability Engineering*, Toulouse, France, IEEE Computer Press, pp.250-255 (1995).

[12] C. Smidts, W. Stoddard and M. Stutzke, "Software reliability models: and approach to early reliability prediction", *Proc. of the 7th Int. Conf. on Software Reliability Engineering*, Toulouse, France, IEEE Computer Press, pp.132-141 (1996).

[13] M. Xie, *Software Reliability Modelling*, World Scientific Publisher, Singapore, 1991.

[14] S. Yamada and S. Osaki, "Software reliability growth modeling: models and applications", *IEEE Transactions on Software Engineering*, **SE-11**(12), 1431-1437 (1985).

[15] M. Zhao and M. Xie, "Robustness of software release time", *Proc. of the 4th Int. Conf. on Software Reliability Engineering*, Denver, Colorado, USA, IEEE Computer Press, pp.218-225 (1993).