A study of the sensitivity of software release time

M. Xie *, G.Y. Hong

Industrial and Systems Engineering Department, National University of Singapore, Kent Ridge, Singapore 119260, Singapore

Received 30 March 1997; received in revised form 16 June 1997; accepted 11 July 1997

Abstract

Software release time determination is of great importance to software developers. In order to predict the amount of testing needed and to estimate the release time at which a certain reliability target is met, probabilistic models can be used to predict the reliability level achieved during the software testing. However, most of the results assume that the model parameters are known which is not true since they have to be estimated based on the information available. In many cases, they are inaccurate because of the small amount of data available. Hence, it is useful to know the sensitivity of software release time with respect to the estimated parameters so that attention can be paid to those parameters that affect the release time significantly. In this paper, the sensitivity issue is studied for a commonly used software reliability model. The study can be used, for example, if an overestimation of a parameter implies an underestimation of the release time which can be costly as more failures are experienced by the consumers, we should try not to overestimate that parameter. Also, if a parameter affects the release time more than others, it is important to have this parameter estimated as accurately as possible. The interval estimation of release time is recommended to avoid further excessive adjustment of release time. © 1998 Elsevier Science Inc. All rights reserved.

Keywords: Software reliability; Release time determination; Sensitivity analysis; Goel–Okumoto model; Confidence interval

1. Introduction

Determination of software release time is critical to the success of the developers with regard to both mission and marketing aspects. Software may fail to execute its function during the mission. The number of failures expected depends on the number of faults remaining in the software and the probability for them to be encountered. Before the release, faults in the software are detected and removed during the testing phase and it is known that the testing consumes a large portion of software development expenses. Probabilistic models can be employed to predict the reliability level achieved in the software so that the developer can predict the amount of testing needed and to estimate the release time at which the reliability target is met. A lot of studies have been carried out related to software release time determination, see, e.g. Kapur and Garg (1989), Dalal and Mallows (1988), Leung (1992), Yamada et al. (1984), Xia et al. (1993) and May et al. (1995).

Software release time determination is essentially an optimization problem in practice. Factors like reliability, testing cost, market influence, release of competitors products will all affect management decision regarding when to release the product. However, reliability which is the most important aspect of product quality has to be ensured. We will only consider reliability criteria in this paper. Most of the existing studies assume that the parameters of the underlying software reliability models are known. This is seldom the case since the estimation of the model parameters are based on the number of faults discovered and the actual failure time data collected during the testing. These parameters are usually estimated using the so-called the method of maximum likelihood or least square technique. Hence, the estimation errors always exist and it is important to know the sensitivity of software release time with respect to the estimated parameters.

In this paper, the sensitivity issue of software release time is studied. The variation of the optimum release time, due to the variation of the estimated parameters, is investigated. If an overestimation of a parameter implies an underestimation of the release time which can be costly as more failures are experienced by the consumers, we should try not to overestimate the parameter. Also, if a parameter affects the release time more than others, it is important to have this parameter estimated accurately.
as accurately as possible. However, the sensitivity issue is a very complex and it varies among different release time models. We restrict ourselves in this paper to the case of given a reliability requirement which is more important as it is closely related to meeting the customer requirements.

A problem when using the traditional method to determine the release time is that the optimum release time increases as more failures are observed. This is not a desirable property. Some procedures for obtaining the optimum release time is also introduced based on the conclusions drawn from the sensitivity study. In order not to get an optimistic release time and later to delay the release, we could use a more conservative estimate of the most sensitive parameter and determine the release time based on that. We could also use the confidence interval estimation of the model parameters, but the procedure is much more complicated.

This paper is organized as follows. Section 2 discusses the problem with the error of the release time because of the errors with the estimated model parameters. Section 3 defines and investigate the sensitivity of the release time with respect to the model parameters. Some interesting results are obtained. Some procedures are discussed in Section 4 to obtain conservative but reasonable estimate of the release time.

2. Problems associated with release time determination

Software release time can be determined based on the software reliability model assuming that the model parameters are known. However, these parameters have to be estimated based on the available information, which might be very limited. Hence the estimates are not accurate and they change from time to time as more data are made available. This makes the release time different and we might not have any confidence with the release time obtained using this method.

For example, a software was tested before the release, the failure information available is assumed to be given in Table 1. The data set is taken from a paper by Currit et al. (1986). For illustration, we apply the commonly used Goel–Okumoto model proposed by Goel and Okumoto (1979), for which the mean value function is given by

$$m(t) = a(1 - e^{-bt}).$$

(1)

The optimum release time based on the reliability criteria can be determined as follows. Note that the software reliability $R(x|t)$ is defined as the probability of a failure-free operation of a computer software for a specified time interval $(t, t + x]$, see e.g., Musa et al. (1987) and Xie (1991). Given the mean value function $m(t)$, we have

$$R(x|t) = \exp[-(m(t + x) - m(t))].$$

(2)

If it is required that the software is to have a minimum reliability of $R_0$ for an operation of $t + x$ time units, then it can be shown that the minimum testing time $T_R$ using the Goel–Okumoto (GO) model of is given by Yamada and Osaki (1985) as

$$T_R = \frac{\ln[a(1 - \exp(-bx))] - \ln[-\ln(R_0)]}{b}.$$  

(3)

Here $T_R$ is a function of parameters $a$ and $b$ which are traditionally obtained by the maximum likelihood method. Because of the random nature of testing data, there are some problems with the stability of the maximum likelihood estimates. This was discussed in detail in Knafl (1992) for a broad collection of software reliability models and Knafl and Morgan (1996) for ungrouped data sets. Consequently, the instability of the maximum likelihood estimates and the lack of sufficient amount of data make the traditional procedures for determination of the optimum release time not very useful.

Using the data set in Table 1 for example, we can estimate the parameters in the GO-model which fits the data quite well. We can then apply the conventional procedure mentioned above to determine the release time, the estimated parameters and the release time for three phases are given in Table 2. The reliability requirement used is $R(80) = 99.9\%$. That is, for a continuous execution of 80 CPU seconds, the probability of

<table>
<thead>
<tr>
<th>Failure number</th>
<th>Failure time</th>
<th>Failure number</th>
<th>Failure time</th>
<th>Failure number</th>
<th>Failure time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85</td>
<td>11</td>
<td>5213</td>
<td>21</td>
<td>19021</td>
</tr>
<tr>
<td>2</td>
<td>170</td>
<td>12</td>
<td>5363</td>
<td>22</td>
<td>22819</td>
</tr>
<tr>
<td>3</td>
<td>649</td>
<td>13</td>
<td>5640</td>
<td>23</td>
<td>25594</td>
</tr>
<tr>
<td>4</td>
<td>1614</td>
<td>14</td>
<td>6143</td>
<td>24</td>
<td>27987</td>
</tr>
<tr>
<td>5</td>
<td>2083</td>
<td>15</td>
<td>6639</td>
<td>25</td>
<td>28896</td>
</tr>
<tr>
<td>6</td>
<td>2468</td>
<td>16</td>
<td>7259</td>
<td>26</td>
<td>29890</td>
</tr>
<tr>
<td>7</td>
<td>3264</td>
<td>17</td>
<td>11309</td>
<td>27</td>
<td>58102</td>
</tr>
<tr>
<td>8</td>
<td>3541</td>
<td>18</td>
<td>14373</td>
<td>28</td>
<td>73058</td>
</tr>
<tr>
<td>9</td>
<td>4468</td>
<td>19</td>
<td>14895</td>
<td>29</td>
<td>75029</td>
</tr>
<tr>
<td>10</td>
<td>4808</td>
<td>20</td>
<td>16692</td>
<td>30</td>
<td>80337</td>
</tr>
</tbody>
</table>

For illustration, we apply the commonly used Goel–Okumoto model proposed by Goel and Okumoto (1979), for which the mean value function is given by

$$m(t) = a(1 - e^{-bt}).$$

(1)
Table 2
Estimate of the optimum release time using the current maximum likelihood estimates (MLE) of the model parameter

<table>
<thead>
<tr>
<th># failures</th>
<th>Failure time</th>
<th>a (MLE)</th>
<th>b (MLE)</th>
<th>(T_R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>16 692</td>
<td>23.46</td>
<td>0.0001146</td>
<td>46 813</td>
</tr>
<tr>
<td>25</td>
<td>28 896</td>
<td>28.31</td>
<td>0.0000742</td>
<td>68 988</td>
</tr>
<tr>
<td>30</td>
<td>80 337</td>
<td>30.56</td>
<td>0.0000498</td>
<td>96 362</td>
</tr>
</tbody>
</table>

having a failure should be less than 0.1%. It can be seen that the obtained optimum release time varies as the testing progresses.

It can be seen from Table 2 that at \(T = 16 692\), 20 failures have occurred. Based on the estimated value of the model parameters, the release time is estimated at \(T = 46 812\). However, after observing 25 failures at \(T = 28 896\), the revised estimate of the release time is \(T = 68 988\) which is significantly larger than the previous estimate. In this case, after the last failure which is observed at \(T = 80 337\), using the latest estimates of the model parameters, the software has still not met the reliability requirement and it has to be tested for another 16 000 CPU seconds.

This problem is common in practice and if this type of revisions is frequently shown to the management or the customer, there will certainly be no confidence of this type of procedure and the company’s reputation might suffer. It is hence important to determine the release time so that no excessive increase is required while the testing progresses. On the other hand, we do not want to increase the release time by certain amount without justification as that would mean an unnecessary delay of the release which affects the overall time to market.

3. A sensitivity analysis of optimum release time

In this section, we study the sensitivity of the optimum release time with respect to the model parameter in the GO model. The partial derivatives of the optimum release time for different model parameters are studied and compared. Some useful properties are derived.

3.1. The trend of the release time with respect to model parameters

By taking the partial derivative of \(T_R\) with respect to \(a\) and \(b\), we have that

\[
\frac{\partial T_R}{\partial a} = \frac{1}{ab}, \quad \frac{\partial T_R}{\partial b} = \frac{\ln a - \ln b - \ln e^{-b R}}{b^2} - \ln[a(1 - e^{-b R})] + \ln[-\ln(R_0)]
\]

From the equations, we can see that the derivative of \(T_R\) with respect to \(a\) is always greater than zero which means that any overestimation of \(a\) would lead to an overestimation of the release time. If a conservative estimate of the release time is needed, it is important not to underestimate the parameter \(a\). This is expected as the parameter \(a\) is related to the number of faults in the software and when this is underestimated, the release time tends to be shorter.

It would be useful to know whether \(T_R\) is a decreasing or increasing function of \(b\). Because \(b\) is related to fault detection rate, it could be expected that when it is underestimated, the testing time would be longer and so is the release time. However, it is not as clear as the case for the parameter \(a\). In fact, it can be decreasing as well as increasing.

A condition for which it is a decreasing function is given in the following.

\[
\frac{\partial T_R}{\partial b} \leq 0 \quad \text{if} \quad b(x + T_R) \geq 1.
\]

The proof of this is given in Appendix A.

Because \(T_R\) has to be larger than \(t\), the time since the testing started, if \(bt > 1\), then we are sure that \(\partial T_R/\partial b\) is a decreasing function of \(b\). In fact, when \(t\) is small and \(bt \geq 1\) is not satisfied, we usually have that \(T_R\) is very large and the condition \(b(x + T_R) \geq 0\) can still be met.

The result implies that it is important not to overestimate the parameter \(b\), in which case it would underestimate the release time and the customers might experience a higher failure intensity than what is promised. It is of course possible to revise the release time when noting that the reliability requirement is not met at the time before release, but a delay of the schedule and release could cost much more.

3.2. The most sensitive parameter

Comparing the parameters \(a\) and \(b\), it would be helpful to know which of the parameters affects the release time more than the other, so that more accurate estimates can be obtained for the most important one. That is, we are interested in the condition at which the following is satisfied

\[
\left| \frac{\partial T_R}{\partial a} \right| \leq \left| \frac{\partial T_R}{\partial b} \right|
\]

or

\[
-\frac{\partial T_R}{\partial b} = -\frac{\ln a - \ln b - \ln e^{-b R}}{b^2} - \ln[a(1 - e^{-b R})] + \ln[-\ln(R_0)] \geq \frac{\partial T_R}{\partial a} = \frac{1}{ab}.
\]

Note that the minus sign before \(\partial T_R/\partial b\) is added because for most practical cases, it will be negative and especially when \(-\partial T_R/\partial b\) is greater than \(1/ab\), \(\partial T_R/\partial b\) is certainly negative. A condition for \(\left| \frac{\partial T_R}{\partial a} \right| \leq \left| \frac{\partial T_R}{\partial b} \right|\) is that

\[
b(x + T_R - 1/a) \leq 1.
\]

The proof of Eq. (7) is given in Appendix B. Note that, similar to the condition in Eq. (5), the condition \(b(x + T_R - 1/a) \leq 1\) for which implies \(\partial T_R/\partial a\)

\[ \frac{\partial T_R}{\partial b} \] is usually satisfied. This is because \( a \) is usually a large number. Note that usually the value of \( T_R \) is large and \( b \) is not very small.

### 3.3. Some numerical illustrations

In fact, the sensitivity of parameter \( b \) can be much higher than that of \( a \). For example, the derivative of \( T_R \) with respect to parameter \( a \) and \( b \) at the end of the testing is plotted in Fig. 1, noting that the scale for \( b \) is much larger.

Let \( S_{p,a} \) be defined as the relative change of the release time when \( a \) is changed by 100\%\%. That is

\[
S_{p,a} = \frac{T_R(a + pa, b) - T_R(a, b)}{T_R(a, b)} \times 100\%.
\]

Using the GO model for illustration with the parameters estimated at the end of the testing which was shown in Table 2, some numerical values of \( S_{p,a} \) are given in Table 3.

Similarly, we can define \( S_{p,b} \) as the relative change of the release time when \( b \) is changed by 100\%\% which is

\[
S_{p,b} = \frac{T_R(a, b + pb) - T_R(a, b)}{T_R(a, b)} \times 100\%.
\]

Again, using the parameters estimated at the end of the testing which was shown in Table 2. In Table 4, some numerical values of \( S_{p,b} \) are given.

From Table 3, we can see that \( T_R \) is increasing with the increase of \( a \) and \( S_{p,b} \) is increasing from negative to positive. From Table 4, we can see that \( T_R \) is decreasing with the increase of \( b \) and \( S_{p,b} \) is decreasing from positive to negative. On the other hand, a change of a parameter by 100\%\% will normally not imply a change of the release time by 100\%\%.

### 4. Some methods for conservative estimation

It was noticed in the previous section that the optimum release time is much more influenced by the value of \( b \) than the value of \( a \). It is thus much more important to obtain an accurate estimate of \( b \). However, it is not always possible because the parameters have to be estimated based on limited amount of information. The problem is very severe especially when the release time has to be estimated at an early stage for which little failure data is available.

In order not to be too optimistic, we could provide a more conservative estimate of the release time by, for example, in the formula for the optimum release time, using \( b - 10\%b \) instead of the estimated \( b \) value. Note that this is not the same as increasing the release time by 10\%. Table 5 shows some numerical value of the release time based on the more conservative estimate of the parameter \( b \).

In practice, if \( b \) is estimated from similar projects or earlier testing information, we could obtain an estimate of its variance, and use the lower 90\% confidence limit in the calculation of the release time. For the GO model,
assuming independent and identically distributed samples, it has been shown that the maximum likelihood estimators have the asymptotic properties (Zhao and Xie, 1993). Based on the results in Section 3, we can make use of the asymptotic variance for the estimates of the parameters and obtain a conservative estimates in order not to release the software too early.

Denote by \( \hat{a} \) and \( \hat{b} \), the maximum likelihood estimates for \( a \) and \( b \), respectively, we have that by standard method in statistical analysis, see e.g. Zhao and Xie (1993), that

\[
\sigma_a^2 = \text{Var}(\hat{a}) = \frac{a^2[n - nb^2\hat{\lambda}(t_n)]}{n^2 - nbt^2\hat{\lambda}(t_n) - t_n^2\hat{\lambda}^2(t_n)},
\]

\[
\sigma_b^2 = \text{Var}(\hat{b}) = \frac{nb^2}{n^2 - nbt^2\hat{\lambda}(t_n) - t_n^2\hat{\lambda}^2(t_n)},
\]

where \( \hat{\lambda}(t) \) is the failure intensity function defined as

\[
\hat{\lambda}(t) = \frac{dm(t)}{dt} = ab e^{-bt}
\]

for the GO model and \( n \) is the number of failure observed with the \( i \)th failure time denoted by \( t_i, i = 1, 2, \ldots, n \), see e.g., Xie (1991) and Lyu (1996).

Compared with that of Table 2, we can notice the following.

1. If the traditional approach is used, the release time has to be more than doubled (an increase of 100%) from 20th to 30th failure, while using the proposed method, the increase is only 30%. When an adjustment is needed, the smaller the better as excessive adjustment will affect customer confidence and the management planning.

2. From a practical point of view, we may not be able to make any adjustment of the schedule, so it is important to “do it right the first time”. Using the confidence estimate, the first estimate in Table 6 is 92 472. This is very close to the latest estimate in Table 2 which is 96 362.

In practice, it is better to use the 90% confidence limit of \( b \) as it might give a more robust estimate of the release time, see Zhao and Xie (1993). On the other hand, this is based on the asymptotic results for maximum likelihood estimates and when a small sample is given, it may not be accurate. The simple method to get a reasonably conservative estimate of the release time is worth using in practice.

### 5. Discussion

In this paper, we have studied the sensitivity of the software release time using Goel-Okumoto model. Some interesting results are obtained. The results are very useful in practice because we can better allocate resources for a more accurate estimation for the most important parameters. It also provides a way to obtain reasonably conservative estimate of the release time.

As the release time is more sensitive to some model parameters, it is important to estimate these parameters more accurately, for example by gathering more information from similar systems. On the other hand, for those parameters that do not affect the software release time very much, a rough estimate is needed. The approach in our study can be extended to any other model.

It should be mentioned that there are other ways to obtain conservative estimates. We have used the interval estimation which is based on the maximum likelihood estimation of the model parameters because it is theoretically sound and acceptable. In practice, such estimates may not exist in the earlier phase of testing, see, e.g., Knafl and Morgan (1996). However, when the testing process is stabilized and there is a fair amount of data, the estimates are usually reasonable.

### Acknowledgements

This research is partly supported by a research grant RP950643 at National University of Singapore. The authors would like to thank two referees for their constructive comments.

### Appendix A. The proof of Eq. (5)

Because

\[
T_R = \ln[a(1 - e^{-bt})] - \ln[-\ln(R_0)]
\]

for fix values of \( a, b \), and \( T_R \), the condition that \( \partial T_R/\partial b \leq 0 \) is equivalent to

\[
\frac{x e^{-bx}}{b(1 - e^{-bx})} \leq \frac{T_R}{b} \leq 0
\]

which in turn can be simplified to

\[
\frac{x e^{-bx}}{(1 - e^{-bx})} \leq T_R.
\]
This condition is the same as
\[ x e^{-bx} \leq T_R (1 - e^{-bx}) \]
and it is equivalent to
\[ (x + T_R) e^{-bx} \leq T_R. \]
It is clear that for \( x = 0 \), the condition is satisfied. If we can show that \((x + T_R) e^{-bx}\) is a decreasing function of \( x \), the condition will always be satisfied.

In fact, we have that
\[
\frac{\partial}{\partial x} (x + T_R) e^{-bx} = -b(x + T_R) e^{-bx} + [1 - b(x + T_R)] e^{-bx}
\]
and \((x + T_R) e^{-bx}\) is a decreasing function of \( x \) if
\[ 1 - b(x + T_R) \leq 0 \]
which is the condition stated.

Hence, when \( b(x + T_R) \geq 0 \), \((x + T_R) e^{-bx} \leq T_R\) and we have that \( \partial T_R / \partial b \leq 0 \). The proof is thus complete.

Appendix B. The proof of Eq. (7)

Similar to the proof of Eq. (5), we have that a condition for \( \partial T_R / \partial a \leq \partial T_R / \partial b \)
\[
b e^{-bx} / (1 - e^{-bx}) - \ln[a(1 - e^{-bx})] + \ln[-\ln(R_0)] \leq -1 / ab
\]
which implies to
\[
\frac{x e^{-bx}}{b(1 - e^{-bx})} \frac{T_R}{b} \leq - \frac{1}{ab}
\]
can be rewritten as
\[ x e^{-bx} \leq (T_R - 1/a)(1 - e^{-bx}) \]
or equivalent to
\[ (x + T_R - 1/a) e^{-bx} \leq T_R - 1/a. \]
Similar to the proof of Eq. (5), \((x + T_R - 1/a) e^{-bx}\) is a decreasing function of \( x \) if \( b(x + T_R - 1/a) \leq 1 \) which is the condition stated.

References


Min Xie obtained his Licentiate and Ph.D. in Quality Technology from Linkoping University in Sweden in 1986 and 1987, respectively. He graduated from the Royal Institute of Technology in Sweden and received his M.Sc. in Engineering in 1984. After working for four years as a research fellow at Linkoping University, he became one of the first recipients of Lee Kuan Yew fellowship, the most prestigious award for researchers in Singapore, tenable at National University of Singapore in 1991. Currently, he is a senior lecturer at Department of Industrial and Systems Engineering. His research interests include quality engineering, system reliability, software reliability, quality improvement and engineering statistics. Dr. Xie has published about 30 journal papers and he is also the author of a book Software Reliability Modelling published by World Scientific Publisher in 1991. Dr. Xie is also an editor of Int. J. of Reliability, Quality, and Safety Engineering. He is a senior member of IEEE.

Guan-Yue Hong graduated from Harbine Institute of Technology, China in 1992. She continued her postgraduate study in Beijing Institute of Aerospace and Aeronautics and received her M.Sc. in 1995. She then joined the National University of Singapore as a research scholar at Dept of Industrial and Systems Engineering. Currently she is with Motorola Singapore as a software quality engineer. Her areas of research include software reliability modelling, software quality and quality engineering. She has authored a number of papers in these areas.