Ratio mathematica 18
(2008), 62 - 90

Effect of Introduction of Fault and Imperfect Debugging on Release Time

P. K. Kapur[*], Deepali Gupta[@], Anshu Gupta[*], P. C. Jha*

Abstract
One of the most important decisions related to the efficient management of testing phase of software development life cycle is to determine when to stop testing and release the software in the market. Most of the testing processes are imperfect once. In this paper first we have discussed an optimal release time problem for an imperfect fault-debugging model due to Kapur et al considering effect of perfect and imperfect debugging separately on the total expected software cost. Next, we proposed a SRGM incorporating the effect of imperfect fault debugging and error generation. The proposed model is validated on a data set cited in literature and a release time problem is formulated minimizing the expected cost subject to a minimum reliability level to be achieved by the release time using the proposed model. Solution method is discussed to solve such class of problem. A numerical illustration is given for both type of release problem and finally a sensitivity analysis is performed.

Keywords: Software Reliability, Non-Homogeneous Poisson Process, Imperfect Debugging, Error Generation, Release Time.

1. INTRODUCTION

Last decade of the twentieth century is marked in history for the incredible growth in the information technology. Consequently computers and computer-based systems have entered in every walk

[*] Department of Operational Research, University of Delhi, Delhi –07, India.
[@] Department of Mathematics, Jaypee Institute of Information Technology, Noida

and talk of our lives. We have become heavily dependent on automated tools and intelligent systems for almost every activity. A mere delay in the operation of these systems can led to big financial loses. Our lives depend critically on the correct functioning of these systems. There are already numerous instances where failures of computer-controlled systems have led to colossal loss of human lives and economy. With the increased dependence of human kind on software systems, software systems are also becoming complex and large and a major concern for the software developers is to deliver more reliable software in smaller development time.

It is the testing stage of the software development in which attempts are made to remove most of the faults lying dormant in software. A successful test strategy begins by considering the requirement specification and continues by specifying test cases based on this requirement specification, to be executed later to find the corresponding faults, which might have been introduced during the various stages of the SDLC. The growing field of Software Reliability Engineering deals in building mathematical models that describe the failure\removal phenomenon with respect to time\testing efforts and consequent enhancement in reliability of the software due to fault removal known as Software Reliability Growth Modeling (SRGMs). Several SRGMS have been discussed and validated by the various researchers under the varying set of assumptions. Most of these models depict either exponential or S-shaped relationship between the testing time\effort and the corresponding number of faults removed [2,9].

Most of the earlier software reliability models assume the fault removal process (fault debugging) to be perfect i.e. when an attempt is made to remove a fault, it is removed with certainty and no new faults are introduced. But this assumption is not realistic due to the complexity of the software system and incomplete understanding of the user's requirements or specifications by the testing team. The software testing team may not be able to fix the cause of the failure properly or they may introduce new faults during removal. Therefore it is necessary to incorporate the effect of imperfect debugging into the software reliability growth modeling. In recent years, several imperfect debugging SRGMs have been proposed and studied (Pham [10], Kapur and Younes [5], Slud [12], and Obha and Chou [8], etc.)

There are two type of imperfect debugging possibilities-first, on a failure the corresponding fault is identified, but just because of incomplete understanding of the software, the detected fault is not removed completely and hence the fault content of the software remains unchanged on the removal action, proposed by Kapur [3] known as imperfect fault debugging, - second, when on a failure the corresponding fault is identified and removed with certainty but some new faults are added to the software during the removal process, proposed by Obha and Chou[8]. This type of imperfect debugging led to an increase in the fault content of the software known as error generation.

No software can be tested indefinitely in order to make it bug free since users of the software want faster deliveries and constraint on development cost. As discussed above an important objective of developing SRGM is to predict software performance using the measure of software reliability and use the information for decision-making. An important decision problem of practical concern is to determine when to stop testing and release the software system to the user known as “Release Time Problem”. This decision depends on the model used for describing the failure phenomenon and the criterion used for determining system readiness. The optimization problem of determining the optimal time of software release can be formulated based on goals set by the management. Firstly the management may wish to determine the optimal release time such that total expected cost of testing in the testing and operation phase is minimum. Secondly they may set a reliability level to be achieved by the release time. Thirdly they may wish to determine the release time such that the total expected cost of the software is minimum and reliability of the software is achieved to a certain desired level. Such a problem is known as a Bi-criteria release time problem. For Bi-criteria release time problem release time is determined by carrying a trade off between cost and reliability. Many researchers in literature have studied various release time problems for different SRGMs [2,3,6,7,9]. Min Xie [13] attempted to determine the optimal release time of software using the SRGM proposed by Obha and Chou [8] incorporating the second type of imperfect debugging i.e. error generation. Whereas the author is referring to imperfect fault debugging that is due to the fault not fixed properly, in his cost model, which creates confusion between two types of imperfect debugging. The cost model used by the author is incomplete, as he considered the cost of fixing an error to be same for due to perfect and imperfect fault debugging during testing and operation phase. The mathematical form of SRGM by Obha and Chou [8] is equivalent to the Kapur [3] model of imperfect fault debugging but the two models are based on different set of assumptions, Obha and Chou model incorporate the effect of error generation whereas Kapur model incorporate the effect of imperfect fault debugging. In this paper we have determined the optimal time when software is ready to be release for use using the imperfect fault-debugging model due to Kapur [3] modifying the cost model of Min Xie. We incorporated separate cost of fixing an error due to perfect and imperfect fault debugging during testing and operation phase in the cost model and determined the release time in the way as determined by Min Xie, which gives the optimal values of the release time and the level of perfect debugging p. However it is imperative to estimate the level of perfect fault debugging i.e. p from the SRGM used to describe the failure phenomenon using the collected failure data, and not as a decision to be obtained from release time problem. At the optimal release time of software determined minimizing the cost, we may not obtain the desired reliability level. Hence if we have a reliability level to be achieved by the optimal time of software release we should incorporate the desired reliability level either as a constraint of a release time problem or as an objective of Bi-criteria release time problem. However we may not obtain a minimum cost at the desired reliability level, therefore release time is determined by a trade-off between reliability and cost. In this paper we have proposed a SRGM incorporating two types of imperfect debugging simultaneously. The proposed model is validated on software failure data sets used in literature. We then determined the release time for the proposed model minimizing the total expected software cost subject to minimum level of reliability to be achieved by the release time incorporating the effect of imperfect debugging and error generation on cost model.

This paper is organized as follows: In the section 2.1 we have discussed a release time problem for perfect debugging SRGM due to Goel Okumoto. In section 2.2.1 we have discussed we have reviewed imperfect fault debugging SRGM due to kapur et al. Then in section 2.2.2 we have discussed the effect of imperfect debugging on total expected software cost and then finally in section 2.2.3 we formulated a release time problem for imperfect fault debugging SRGM due to kapur et al and derived the optimal release time of the software minimizing the total expected software cost. In section 3 first we proposed a SRGM incorporating the effect of imperfect debugging and fault generation in section 3.1. Parameters of the proposed model are estimated in section 3.2. Further we discuss the effect of imperfect fault debugging and error generation on total expected software cost in section 3.3 and finally a release time problem is formulated and solved minimizing the total expected software testing cost subject to minimum reliability level constraint. In section 4.1 a numerical illustration is given for both type of release problem and finally a sensitivity analysis is performed to determine the effect of variations in minimum reliability level to be achieved, in cost of fixing an error perfectly and imperfectly in operation phase and in level of perfect debugging..

2. Release Time Problem for Imperfect Fault Debugging SRGM

2.1 Determination of Release Time for Perfect Debugging SRGM

Among all SRGMs developed so far a large family of stochastic reliability models based on a non-homogeneous Poisson process known as NHPP reliability models, has been widely used. Some of them depict exponential growth while others show S-shaped growth depending on nature of growth phenomenon during testing. Most commonly cost model seen in literature for determination of release time for perfect debugging NHPP models is [2,13]

$$C = C_1 m(T) + C_3(m(\infty) - m(T)) + CT \qquad \ldots(2.1)$$

Using Goel Okumoto NHPP [2] model, for which the mean value function is

$$m(t) = a(1 - e^{-bt}) \qquad \ldots(2.2)$$

The optimal release time minimizing the total expected software cost defined as (1) is given by

$$T^* = \frac{1}{b}\ln\left(\frac{ab(C_3 - C_1)}{C}\right) \qquad \ldots(2.3)$$

Maximum likelihood estimates (MLE) of a and b for the software failure data cited in Zhang and Pham [10], are obtained as a = 142.32 and b = 0.1246. Assuming C_1 = \$200, C_3 = \$1500, and C = \$5, from (3), the optimal release time is calculated as 67.70556 and the minimum expected software cost is found to be \$28,842. However, the model assumes a perfect testing process. It would be of interest to study the effect of imperfect debugging on total expected software testing cost. In the next section we have discussed the effect of imperfect fault debugging on expected software testing cost, briefly discussing the imperfect fault debugging SRGM due to Kapur [3].

2.2 Release Time Problem for Imperfect Fault Debugging SRGM (Kapur [])

2.2.1 Imperfect Fault Debugging SRGM

A simple imperfect fault debugging model proposed by Kapur [3] assume on a failure the corresponding fault is identified and when an attempt is made to remove the fault it is not fixed properly, which does not lead to any change in the initial fault content of the software. The model is formulated as follows

Model Assumptions

1. Software system is subject to failures at random times caused by faults remaining in the software.
2. Failure rate of the software is equally affected by errors remaining in the software.
3. At any time the failure rate of the software is proportional to the faults remaining in the software.
4. On a instantaneous repair effort starts and the following may occur:
   (a) Fault contents are reduced by one, with probability p
   (b) Fault contents are unchanged with probability 1-p.
5. The error removal phenomenon in the software is modeled by NHPP.

Notations

a	:	initial error content.
b	:	proportionality constant(fault removal rate per remaining fault).
p	:	probability of perfect debugging.
$m_f(t)$	:	mean number of failures detected in (0,t].
$m_r(t)$	:	mean number of faults removed in the software till time t.
$\lambda(t)$	:	intensity function or fault detection rate per unit time.

The differential equation describing the rate of change of $m_r(t)$ with respect to time under the assumptions specified above and following the notations is given by

$$m_r'(t) = bp\left(a\text{-}m_r(t)\right) \qquad \ldots(2.4)$$

Solving equation (2.4)under the initial condition $m_r(0) = 0$ is given by

$$m_r(t) = a\left[1 - e^{-bpt}\right] \qquad \ldots(2.5)$$

Corresponding mean number of failures in (0,t] is given by

$$m_f(t) = \int_0^t b\left(a - m_r(t)\right)dt \qquad \ldots(2.6)$$

$$m_f(t) = \frac{a}{p}\left[1 - e^{-bpt}\right] \qquad \ldots(2.7)$$

The NHPP intensity function is given by

$$\lambda(t) = ab\exp(-bpt) \qquad \ldots(2.8)$$

It can be seen that $\lambda(t)$ is a decreasing function in t with $\lambda(0) = ab$ and $\lambda(\infty)=0$.

In the next section we have proposed the cost model incorporating the effect of imperfect debugging.

2.2.2 Effect of Imperfect Debugging on the Cost Model

A major concern in software development is the cost. It is well known that the development of a software system is time-consuming and costly. Since most software testing processes are imperfect debugging ones, it is of great importance for the management to know the effect of the imperfect debugging on software cost (Ammann et al. [1], Shanthikumar [11], and Pham [10]). On the other hand, if the release time of the software is determined by the minimum cost criterion, the imperfect debugging will affect the release time as well.

The parameter p representing the probability of perfect debugging can also represent the testing level, indicating "how perfect" the testing process is. Testing level parameter p is usually influenced by a number of factors, such as the experience of the testing personnel, the testing strategy adopted, and the number of reviews in debugging. When the testing level is low, it is possible to increase it to a certain extent, but usually this has to be achieved at a higher testing cost.

Total expected software cost includes cost of testing and the cost of fixing a fault during testing and operation phase for perfect and imperfect debugging. Cost of fixing an error is different for both perfect and imperfect debugging. Also the cost of testing is a function of perfect debugging probability p. Since the testing cost parameter C depends on the testing team composition and testing strategy used, If the probability of perfect debugging is to be increased, it is expected that extra financial resources will be needed to engage more experienced testing personnel, and this will result in an increase of C. In other words, C should be a function of the testing level, denoted by C(p) and hence this function should possess the following two properties:

1. C(p) is a monotonous increasing function of p.
2. When $p \rightarrow 1$, $C(p) \rightarrow \infty$.

The second property implies that perfect debugging is impossible in practice or the cost of achieving it is extremely high.

Notations:

$C_1(C_2)$:	cost incurred on a prefect (imperfect) debugging effort before release of the software system.
$C_3(C_4)$:	cost incurred on a prefect (imperfect) debugging effort after release of the software system. ($C_3 > C_1$, $C_4 > C_2$).
C :	testing cost per unit time.
T :	release time of the software.
T^* :	optimal release time.

R_0 : desired level of software reliability at the release time($0 < R_0 < 1$).

Although there are many cost functions that can satisfy these conditions, a simple, but reasonable function that meets the two properties above is given by:

$$C(p) = \frac{C}{(1-p)} \qquad \text{…(2.9)}$$

Hence, the cost model (2.1) can be modified as

$$\text{Min } C(T,p) = \left(C_1 p + C_2(1-p)\right) m_f(T) + \left(C_3 p + C_4(1-p)\right)\left(m_f(\infty) - m_f(T)\right) + \frac{CT}{(1-p)} \qquad \text{...(2.10)}$$

If the release time remains at 67.70556, the software cost under different probabilities of perfect debugging or testing levels is calculated and summarized in Table 1. It is clear that the software cost changes significantly as the testing level, p, changes. Obviously, if the management has not taken into consideration the effect of imperfect debugging on software cost, the model may give a wrong estimate of the system reliability and\or cost at the release time.

Table 1.

p	Cost($\$10^3$)
0.7	37037
0.75	35485.03
0.8	34345.28
0.85	33652.61
0.9	33693.12
0.95	36123.14
1	∞

In the next section we will determine optimal release time and optimal testing level such that the total expected software cost is minimized.

2.2.3 Optimal Release Policy

The optimization problem minimizing the total expected software cost in order to determine optimal release time T^* and optimal testing level p^* can be formulated as follows

$$\text{Min } C(T,p) = \left[C_1 p + C_2(1-p)\right].m_f(T) + \left[C_3 p + C_4(1-p)\right].\left(m_f(\infty) - m_f(T)\right) + \frac{CT}{(1-p)}$$

Subject to $0 < p < 1$ and $T > 0$ …(2.11)

Using the principles of calculus the above optimization problem can be solved as follows:

Taking partial derivates of C(p,T) with respect to p and T and equate them to zero, we have that

$$\frac{\partial C}{\partial T} = \left[C_1 p + C_2(1-p)\right].abe^{-bpT} + \left[C_3 p + C_4(1-p)\right].\left(-abe^{-bpT}\right) + \frac{C}{(1-p)} = 0$$

…(2.12)

And

$$\frac{\partial C}{\partial p} = \left(C_1 - C_2\right)\frac{a}{p}\left(1 - e^{-bpT}\right) + \left(C_1 p + C_2(1-p)\right)\left(-\frac{a}{p^2} + \frac{a}{p^2}e^{-bpT} + \frac{abT}{p}e^{-bpT}\right)$$

$$(C_3 - C_4)\left\{\frac{a}{p}\left(e^{-bpT}\right)\right\} + \left(C_3 p + C_4(1-p)\right)\left(-\frac{a}{p^2}e^{-bpT} - \frac{abT}{p}e^{-bpT}\right) + \frac{CT}{(1-p)^2} = 0$$

…(2.13)

From (2.12) T can be expressed in terms of p as

$$T = g(p) = \frac{1}{bp}\ln\left(\frac{ab(D_2 - D_1)(1-p)}{C}\right) \qquad \text{…(2.14)}$$

Where $D_1 = C_1 p + C_2(1-p)$, $D_2 = C_3 p + C_4(1-p)$
It is clear that, when p takes values between (0,1), the condition T > 0 is always satisfied.

Substituting the value of T from (2.14) into (2.13), we get

$$\frac{\partial C}{\partial p}=\frac{(2p-1)C(D_2-D_1)\ln\left\{\dfrac{ab(D_2-D_1)(1-p)}{C}\right\}-C_2ab(D_2-D_1)(1-p)^2-C(C_4-C_2)(1-p)}{bp^2(1-p)^2(D_2-D_1)}=0$$

…(2.15)

or, equivalently h(p) = 0.

$$h(p)=(2p-1)C(D_2-D_1)\ln\left\{\frac{ab(D_2-D_1)(1-p)}{C}\right\}-C_2ab(D_2-D_1)(1-p)^2-C(C_4-C_2)(1-p)=0$$

…(2.16)

h(p) is a continuous function of p on(0,1) and

$$\lim_{p\to 0^+} h(p)=-K \qquad \lim_{p\to 1^-} h(p)=-\infty \qquad \ldots(2.17)$$

$$\text{where } K=\left(C\ln\left(\frac{ab(C_4-C_2)}{C}\right)+abC_2+C\right)\left(C_4-C_2\right) \qquad \ldots(2.18)$$

Now, taking the derivative of h(p) with respect to p, we have that

$$h'(p)=2C(D_2-D_1)\ln\left\{\frac{ab(D_2-D_1)(1-p)}{C}\right\}+(2p-1)C(D_2-D_1)\left(\frac{-1}{1-p}\right)+C(C_4-C_2)$$
$$+\left(2C_2ab(D_2-D_1)(1-p)-C_2ab(1-p)^2+(2p-1)C\ln\left(\frac{ab(D_2-D_1)(1-p)}{C}\right)\right)(C_3-C_1-C_4+C_2)=0$$

It can be seen that h′(p) is a continuous and strictly decreasing function on (0,1) and

$$\lim_{p\to 0^+} h'(p)=2K+\left(C\ln\left(\frac{ab(C_4-C_2)}{C}\right)+C_2ab\right)(C_4-C_3-C_2+C_1) \qquad \lim_{p\to 1^-} h'(p)=-\infty$$

The following Theorem summarizes some analytical results regarding the existence and uniqueness of the optimal solution.

Theorem 1. The optimal values of p and T, denoted by p* and T*, which minimize the expected software cost given by (9) are as follows:

Case 1. If $K \le 0$, then $p^* = \inf\{p : h(p) < 0\}$and $T^* = g(p^*)$.
Case 2. If $K > 0$, then define $p' = \inf(p : dh/dp < 0\}$and

1. If $h(p') > 0$, then $p^* = \min[C(p_1, T_1), C(p_2, T_2)]$ and $T^* = g(p^*)$,where p_1 and p_2 are the solutions to the equation of $h(p) = 0$ and $T_1 = g(p_1)$, $T_2 = g(p_2)$.
2. If $h(p') = 0$, then p* equals the unique solution to the equation of $h(p) = 0$ and $T^* = g(p^*)$.
3. If $h(p') < 0$, then p* and T* does not exist within $0<p<1$ and $T>0$.

Using the above procedure to find the optimal release time first we need to determine the value $\inf\{p : h(p) < 0$ or $p' = \inf(p : dh/dp < 0\}$ what ever is the case assuming a perfect debugging environment i.e p=1 as both h(p) and h′(p) function of p in order to determine the optimal value of p and then using this optimal value of p we estimate the other parameters of the SRGM based on the collected failure data and then determine the optimal release time. The procedure if repeated for this optimal value and more dense data we will obtain another set of optimal values and hence it is a iterative approach. Hence the solution procedure adopted by Min Xie does not terminate in one step to give the optimal values. However it is imperative to estimate the level of perfect fault debugging i.e. p from the SRGM used to describe the failure phenomenon using the collected failure data over a period of time, and not as a decision to be obtained from release time problem by minimizing cost function. The effect of level of perfect debugging on release time can be obtained by carrying a sensitivity analysis on the release problem.

Whenever a decision is made to release the software the management evaluate the reliability of the software as quality metric at the release time. In the numerical example given in this paper we found that the reliability level at the optimal release time is 0.9398, where as for the problem discussed by Min Xie it is 0.9117. However if the management desires to obtain a reliability level 0.95 by the release time, the approach followed above to find the optimal solution couldn't be used. Therefore we must consider the level of reliability to be achieved while formulating such class of problem.

Before we discuss the release time problem by minimizing the cost under reliability constraint we propose and validate a SRGM incorporating the effect of both imperfect fault debugging and error generation in the next section.

3. Release Time Problem for an SRGM Incorporating Two Types of Imperfect Debugging

3.1 SRGM with Two types of Imperfect Debugging

During the testing process when a fault in encountered, corresponding fault is identified and an attempt is made to remove the fault, there are three possibilities, first the fault is removed perfectly, secondly the fault is not removed perfectly due to which the fault content remains unchanged known as imperfect fault debugging, third the fault is removed perfectly, but when the test case that led to the failure is re-executed some other fault is encountered, known as error generation. In fact while removing the fault the programmer has introduced a new fault leading to an increase in total fault content of the software. Newly introduced fault leads to a failure only when the original fault is removed perfectly.

Model Assumptions

1. Software system is subject to failures at random times caused by faults remaining in the software.
2. Failure rate of the software is equally affected by errors remaining in the software.
3. At any time the failure rate of the software is proportional to the faults remaining in the software.

4. On a instantaneous repair effort starts and the following may occur:
   (a) fault contents are reduced by one with probability p
   (b) fault contents are unchanged with probability 1-p.
5. The error removal phenomenon in the software is modeled by NHPP.
6. During the fault removal process faults are generated with a constant probability α.

Under the assumptions specified above the differential equation for the proposed model is given by

$$m_r^{'}(t) = bp\left(a(t)\text{-}m_r(t)\right) \qquad \ldots(3.1)$$

Where a(t) can be expressed as

$$a(t) = a + \alpha m_r(t) \qquad \ldots(3.2)$$

Substituting (3.2) in (3.1) we have

$$m_r^{'}(t) = bp\left(a + \alpha m_r(t) - m_r(t)\right) \qquad \ldots(3.3)$$

Solving equation (3.3) under the initial condition $m_r^{'}(0) = 0$ we get

$$m_r(t) = \frac{a}{1-\alpha}\left[1 - e^{-bp(1-\alpha)t}\right] \qquad \ldots(3.4)$$

Corresponding Mean number of failures in (0,t] is given by

$$m_f(t) = \int_0^t b\left(a(t) - m_r(t)\right)dt \qquad \ldots(3.5)$$

$$m_f(t) = \frac{a}{p(1-\alpha)}\left[1 - e^{-bp(1-\alpha)t}\right] \qquad \ldots(3.6)$$

The NHPP intensity function is given by $\lambda(t) = ab\exp(-bpt)$ …(3.7)

It can be seen that $\lambda(t)$ is a decreasing function in t with $\lambda(0) = ab$ and $\lambda(\infty) = 0$.

In the next section we validate and compare the model with some existing models.

3.2 Parameter Estimation

Method of least squares or maximum likelihood has been suggested and widely used for estimation of parameter of mathematical models. The model proposed in this paper is a non-linear and it is difficult to find solution for nonlinear models using Least Square method and require numerical algorithms to solve it.

Statistical software packages such as SPSS help to overcome this problem. SPSS is a statistical package for Social Sciences. It is a comprehensive and flexible package for statistical analysis and data management system. SPSS can take data from almost any type of file and use them to generate tabulated reports, charts and plots of distributions and trends, descriptive statistics, and conduct complex statistical analysis. SPSS Regression Models enables the user to apply more sophisticated models to the data using its wide range of nonlinear regression models. For the estimation of the parameters of the proposed model method of Least Square has been used. Non-linear regression is a method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike traditional linear regression, which is restricted to estimating linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables.

3.2.1 Comparison Criteria

1. Mean Square Error (MSE)**:**

The model under comparison is used to simulate the fault data, the difference between the expected values, N(t) and the observed data N_i is measured by MSE as follows.

$$MSE = \sum_{i=1}^{k} \frac{(N(t_i) - N_i)^2}{k} \qquad \ldots(3.8)$$

Where k is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit.

2. Coefficient of multiple determination (R^2):

We define this coefficient as the ratio of the sum of squares resulting from the trend model to that from constant model subtracted from 1.

$$R^2 = 1 - \frac{\text{residual SS}}{\text{corrected SS}} \qquad \ldots (3.9)$$

R^2 measures the percentage of the total variation about the mean accounted for the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well. The larger value of R^2 explains the better fit of the model.

3.2.2 Data Analysis and Model Comparison

To validate the proposed model we have carried out the parameter estimation on a data set from a real time command and control system, which represents 136 failures, observed during system testing for 25 hours of CPU time [9]. Parameters of the model are estimated by the nonlinear least squares method in SPSS using cumulative failure data against time. Estimated parameter values are given in table- 2. The MSE and R^2 values are also given. The Fitting of the models is illustrated graphically in figure 1 and figure 2.

Table 2:

Estimated parameter values for the proposed model					
Parameters				**Goodness of Fit Criteria**	
a	b	p	α	RMSPE	R2
134	0.140238	0.998417	0.0125628	30.64387	.96641

Figure 1:

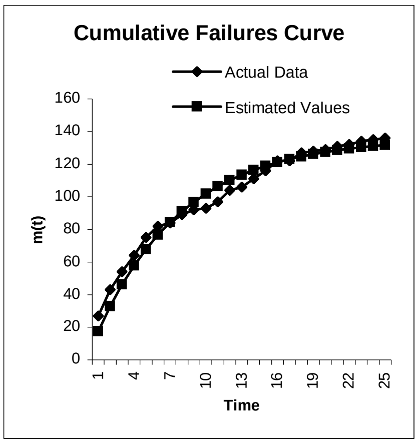
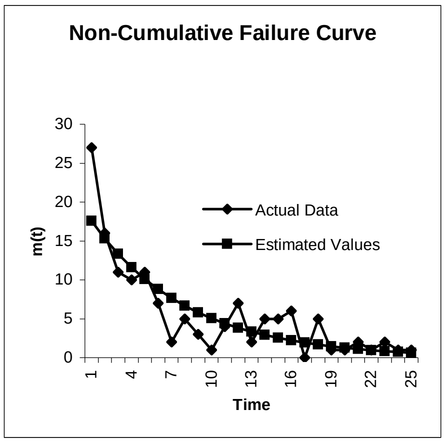


Figure 2:



In the next section we have proposed the cost model incorporating the effect of imperfect fault debugging and error generation.

3.3 Effect of Imperfect Fault Debugging and Error Generation on Cost Model

Like knowing the effect of the imperfect debugging on software cost it is also of great importance for the management to know the effect of fault generation on cost. Since due to fault generation amount of fault content of software increases, it has a direct effect on the reliability level of the software achieved by the release time.

The parameter α representing the probability of error generation is usually influenced by a number of factors, such as the experience of the testing personnel, the testing strategy adopted, and the number of reviews in debugging etc. It is possible to decrease the value of α to a certain extent, but usually this has to be achieved at a higher testing cost. As specified above total expected software cost includes cost of testing and the cost of fixing a fault during testing and operation phase for perfect and imperfect debugging. Cost of fixing an error is different for both perfect and imperfect debugging however it remains unchanged due to error generation. But the cost of testing is a function of both perfect debugging probability p and fault generation probability α. Since the testing cost parameter C depends on the testing team composition and testing strategy used, If the probability of perfect debugging is to be increased and probability of error generation is to be decreased, it is expected that extra financial resources will be needed to engage more experienced testing personnel, and this will result in an increase of C. In other words, C should be a function of the testing level and error generation, denoted by $C(p,\alpha)$ and hence this function should possess the following two properties:

1. $C(p,\alpha)$ is a monotonous increasing function of p and $(1-\alpha)$.
2. When $p\rightarrow 1$, and $\alpha \rightarrow 0$, $C(p,\alpha)\rightarrow\infty$.

The second property implies that perfect debugging is impossible in practice or the cost of achieving it is extremely high. Although there are many cost functions that can satisfy these conditions, a simple, but reasonable function that meets the two properties above is given by:

$$C(p) = \frac{C}{\left(1-p\left(1-\alpha\right)\right)} \qquad \ldots(3.10)$$

Hence, the cost model (2.9) can be modified as

$$\text{Min } C(T,p) = \left(C_1p + C_2(1-p)\right)m_f(t) + \left(C_3p + C_4(1-p)\right)\left(m_f(\infty) - m_f(t)\right) + \frac{CT}{\left(1-p\left(1-\alpha\right)\right)} \qquad \ldots(3.11)$$

In the next section we will determine optimal release time for software minimizing the total expected cost subject to the desired reliability constraint.

3.4 Optimal Release Policy

The optimization problem minimizing the total expected software cost in order to determine optimal release time T^* subject to the software reliability not less than a specified reliability objective can be formulated as follows

$$\text{Min } C(T) = \left[C_1 p + C_2(1-p)\right].m_f(T) + \left[C_3 p + C_4(1-p)\right].\left(m_f(\infty) - m_f(T)\right) + \frac{CT}{1-p(1-\alpha)}$$

Subject to $R(x \mid T) = \exp[-(m(T+x) - m(T))] \geq R_0$ Where $0 < R_0 < 1$ and $x > 0$.

Using the principles of calculus and assuming that the values of all the parameters of the proposed SRGM have been estimated including p and α form the past failure data, the above optimization problem can be solved as follows:

Taking partial derivates of C(T) with respect to T and equating it to zero, we have

$$\frac{\partial C}{\partial T} = \left[C_1 p + C_2(1-p)\right] abe^{-bp(1-\alpha)T} + \left[C_3 p + C_4(1-p)\right]\left(-abe^{-bp(1-\alpha)T}\right) + \frac{C}{1-p(1-\alpha)} = 0$$

…(3.12)

From (3.12) we observe that

$$\lambda(t) = \frac{C}{(D_2 - D_1)(1-p(1-\alpha))} \qquad \text{…(3.13)}$$

Where $D_1 = C_1 p + C_2(1-p)$, $D_2 = C_3 p + C_4(1-p)$ …(3.14)

$$\lambda(t) = ab\exp(-bp(1-\alpha)t) \quad \lambda(0) = ab \qquad \lambda(\infty) = 0 \qquad \text{…(3.15)}$$

From (3.15) it can be seen that $\lambda(t)$ is a decreasing function in time.

Result 1:

If $ab > \frac{C}{(D_2 - D_1)(1 - p(1-\alpha))}$ then C(T) is decreasing for $T < T_0$ and increasing for $T > T_0$ thus, there exist a finite and unique $T=T_0$ (>0) minimizing the total expected cost. And if $ab \le \frac{C}{(D_2 - D_1)(1 - p(1-\alpha))}$ then $C'(T) > 0 \text{ for } T > 0$ and hence C(T) is minimum for T = 0.

Further reliability of software defined as "given that the testing has continued up to time T, the probability that a software failure does not occur in time interval $(T, T+x)(x \ge 0)$". Hence the reliability of software is represented mathematically as

$$R(x|T) \equiv R(T+x|T) = \exp^{-(m(T+x)-m(T))} \qquad \ldots(3.16)$$

Using (3.16) we obtain

$$R(x|0) = e^{-m(x)}, R(x|\infty) = 1 \qquad \ldots(3.17)$$

Result 2:

From (3.17) it is observed that $R(x|t), t > 0$ is a increasing function of time. Thus $R(x|0) < R_0$ there exist $T=T_1(>0)$ such that $R(x|T) = R_0$ and if $R(x|0) \ge R_0$ then $R(x|t) \ge R_0 \quad \forall \quad t \ge 0$ and $T=T_1=0$.

Combining the cost and reliability requirements we state the following theorem for optimal release policy for the proposed SRGM of imperfect fault debugging and error generation.

Theorem 2: Assuming

$C_3 > C_1 > 0,\ C_4 > C_2 > 0, C > 0, x > 0, \text{and } 0 < R_0 \le 1$

(a) if $ab > \frac{C}{(D_2 - D_1)(1 - p(1-\alpha))}$ & $R(x|0) < R_0 < 1,\ T^* = \max(T_0, T_1)$

(b) if $ab > \frac{C}{(D_2 - D_1)(1 - p(1-\alpha))}$ & $R(x|0) \ge R_0 > 0,\ T^* = T_0$

(c) if $ab \le \dfrac{C}{(D_2 - D_1)(1 - p(1-\alpha))}$ $\& R(x\,|\,0) < R_0 < 1,\; T^* = T_1$

(d) if $ab \le \dfrac{C}{(D_2 - D_1)(1 - p(1-\alpha))}$ $\& 0 < R_0 \le R(x\,|\,0),\; T^* = 0$

Using the above theorem we can determine the optimal release time minimizing the total expected software cost under a desired reliability level constraint.

4. Numerical Examples and Sensitivity Analysis

4.1 Numerical Example of Release Time Problem for Imperfect Fault Debugging SRGM

Assuming that the parameters a and b of Imperfect Fault Debugging SRGM due to Kapur et al the SRGM have already been estimated using the collected failure data and estimated values of a and b are 142.32 and 0.1246 respectively. Further assuming that cost of perfect fault debugging during testing and operation phase i.e. C_1 and C_2 to be \$200 and \$110 respectively, cost of imperfect fault debugging during testing and operation phase to be same i.e. $C_3 = C_4$ =\$1500 and cost of per unit testing C=\$10. Following the theorem 1 we obtain the optimal release time T* = 56.28, optimal level of perfect debugging p* = 0.8897 and optimal total expected software cost C(T*) = 33365.047 and achieved level of reliability R(T*) = 0.9398. Where as for the release time problem discussed by Min Xie T* = 55.196, optimal level of perfect debugging p* = 0.85 and optimal total expected software cost C(T*) = 37931.44 and achieved level of reliability R(T*) = 0.9117. Thus we can see that if we include separate cost of fixing faults perfectly and imperfectly it has significant effect on optimal release time and cost depending upon the values of the various costs associated with the cost model. Note that the above release time problem is solved in way as done by Min Xie which gives optimal values of p and T* however it is imperative to estimate the level of perfect fault debugging i.e. p from the SRGM used to describe the failure phenomenon using the collected failure data, and not as a decision to be obtained from release time problem to be obtained from release time problem. In the next numerical example we have determined the optimal release time minimizing the cost function subject to reliability constraint assuming that value of perfect debugging and error generation parameters are estimated using collected failure data.

4.2 Numerical Example of Release Time Problem for an SRGM Incorporating Two Types of Imperfect Debugging

Assuming that the parameters a, b, p and α of proposed SRGM have already been estimated using the collected failure data and estimated values of a, b, p and α are 134, 0.14024, 0.99842 and 0.01256 respectively. Further assuming that cost of perfect and imperfect fault debugging during testing i.e. C_1 and C_2 to be \$200 and \$110 respectively, cost of perfect and imperfect fault debugging during operation phase to be same i.e. $C_3 = C_4 =\$1500$ and cost of per unit testing C=\$10. If minimum reliability requirement by the release time is 0.85, following result 1 and 2 we obtain $T_0 = 25.6162$ and $T_1 = 38.3983$. Then finally following theorem 2 we obtain $T^* = 38.3983$. The minimum total expected software cost at T^* i.e. $C(T^*) = \$55235.55$ and number of faults removed by the release time $m(T^*) = 135$.

4.3 Sensitivity Analysis

We have conducted, a sensitivity analysis of the release time problem formulated for the proposed model to study the effect of variations in minimum reliability requirement by the release time, most sensitive costs involved in cost function and level of perfect debugging, on the optimal release time and total expected software testing cost. Although we can analyze the sensitivity of all the parameters of the SRGM and Cost model but due to the limitation on size of paper we still can evaluate the optimal release time problem for various conditions by examining about the behavior of some parameters and costs that have the most significant influence.

We define

$$\text{Relative Change}\,(\text{RC}) = \frac{\text{MOV} - \text{OOV}}{\text{OOV}} \qquad \ldots(4.1)$$

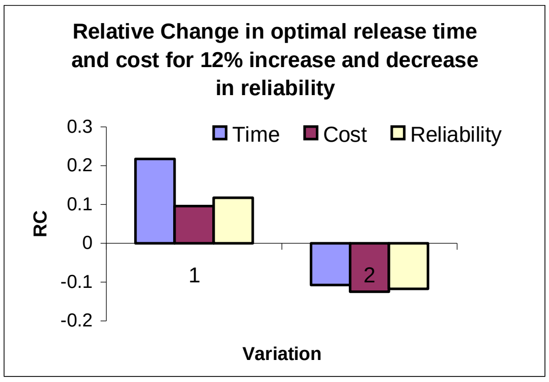
Where OOV is the original optimal values and MOV is the modified optimal values obtained when there is a variation is some attribute of the release time problem.

4.3.1 Effect of Variations in Minimum Reliability Requirement by the Release Time

The optimal value of the release time obtained for the desired reliability level may be too late as compared to the scheduled delivery time, in such a case the management and/or the user of a project based software may agree to release the software at some lower reliability level with some warranty on the failures, which in turn will change the optimal release time to an earlier time and consequently lower the cost. On the other hand if the scheduled delivery is later than the optimal release time the management may wish to increase the desired reliability level at some addition testing cost.

Assuming the values of parameters and various costs associated with cost model to be same as in section 4.2. If minimum reliability requirement by the release time increased to 0.95 (about 12% increase) then we obtain T* = 46.73 (about 21.7% increase) and its RC is 0.217229. The minimum total expected software cost at T* i.e. C(T*) = $60542.43 (about 9.6% increase), its RC is 0.096077 and number of faults removed by the release time m(T*) = 136 and if minimum reliability requirement by the release time decreased to 0.75 (about 12% decrease) then we obtain T* = 34.27 (about 10.7% decrease) and its RC is -0.10757. The minimum total expected software testing cost at T* i.e. C(T*) = $52984.85 (about 12.48% decrease), its RC is -0.12483 and number of faults removed by the release time m(T*) = 134. Figure 2 plots the relative change in the optimal release time and cost for the case of 12% increase and decrease in reliability objective.

Figure 2:



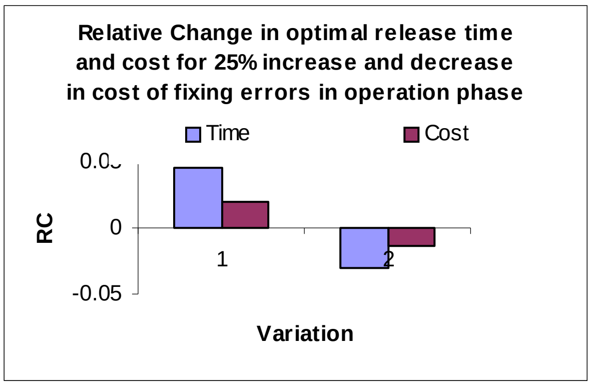
4.3.2 Effect of Variations in Costs Involved in the Cost Model

Here we investigate the sensitivity of variations in various costs involved in the cost model. If any of the cost fixing an error in testing phase or operation phase for perfect and \or imperfect debugging and cost of per unit testing time varies during the testing process, it will have significant changes in optimal testing cost and release time. We have studied the sensitivity of cost of perfectly fixing an error in testing and operation phase. Sensitivity for the rest of the costs can be carried in a similar manner.

If we assume that the values of parameters of the SRGM to be same given in section 4.2 and assuming that cost of perfect and imperfect fault debugging during testing i.e. C_1 and C_2 to be \$200 and \$110 respectively, cost of perfect and imperfect fault debugging during operation phase to be same i.e. $C_3 = C_4$ =\$2000 and cost of per unit testing C=\$2. If minimum reliability requirement by the release time is 0.85, following result 1 and 2 we obtain $T_0 = 39.61$ and $T_1 = 38.3983$. Then finally following theorem 2 we obtain T* = 39.61. The minimum total expected software cost at T* i.e. C(T*) = \$55958.87 and number of faults removed by the release time m(T*) = 134.

Now if cost of fixing a fault perfectly and imperfectly in operation phase i.e. C_3 and C_4 is increased by 25% i.e. from \$2000 to \$2500, then we obtain T* = 41.38 (about 0.4% increase) and its RC is 0.0447562. The minimum total expected software cost at T* i.e. C(T*) = \$57053.21 (about 1.9% increase), its RC is 0.019556 and number of faults removed by the release time m(T*) = 136 and if C_3 and C_4 is decreased by 25% i.e. from \$2000 to \$1500, then we obtain T* = 38.398 (about 3% decrease) and its RC is –0.030605. The minimum total expected software cost at T* i.e. C(T*) = \$55235.55 (about 1.2% decrease), its RC is –0.01293 and number of faults removed by the release time m(T*) = 135. Figure 3 plots the relative change in the optimal release time and cost for the case of 25% increase and decrease in cost of fixing an error in operation phase.

Figure 3.



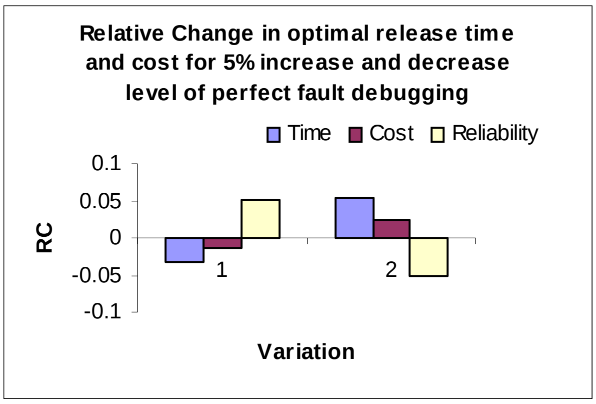
4.3.3 Effect of Variations in level of perfect fault debugging

Finally we investigate the sensitivity of variations in level of perfect fault debugging parameter p. If the testing personals were skilled personal the level of perfect fault debugging would be more or vice versa. Variations in level of perfect debugging have significant effect on the optimal time of software release. If the level of perfect debugging increases for a testing process it is expected that the software can be released earlier as compared to the optimal release time determined otherwise and vice versa

If we assume that the values of parameters a, b and α of the SRGM and the cost involved in cost function to be same given in section 4.2 and a reliability level of 0.85 is desired to be achieved and assuming value of perfect fault debugging parameter p is 0.9. Following result 1 and 2 we obtain T_0 = 39.369 and T_1 = 40.737. Then finally following theorem 2 we obtain T* = 40.737. The minimum total expected software cost at T* i.e. C(T*) = $56649.53 and number of faults removed by the release time m(T*) = 135.

Now if p is increase by 5%, then we obtain T* = 39.369 (about 0.03% decrease) and its RC is –0.03357. The minimum total expected software cost at T* i.e. C(T*) = $55812.96 (about 1.4% decrease), its RC is –0.01477 and number of faults removed by the release time m(T*) = 135 and if p is decrease by 5%, then we obtain T* = 42.93 (about 5.3 % increase) and its RC is 0.053869. The minimum total expected software cost at T* i.e. C(T*) = $58037.50 (about 2.4% increase), its RC is 0.024501 and number of faults removed by the release time m(T*) = 136. Figure 4 plots the relative change in the optimal release time and cost for the case of 5% increase and decrease perfect fault debugging parameter p.

Figure 4.



A similar conclusion can be obtained for the other costs and parameters of the SRGM such as C_1, C_2, C, a, b and α taking the simultaneous changes in two or more costs and SRGM parameters.

5. Conclusion

In this paper first we have formulated and derived optimal release time minimizing the expected software cost subject for an imperfect fault-debugging model due to Kapur et al considering effect of perfect and imperfect debugging separately on the total expected software cost. Next, we proposed a SRGM incorporating the effect of imperfect fault debugging and error generation. The proposed model is validated a data set cited in literature. Then a release time problem is formulated and solved minimizing the expected software cost subject to a minimum reliability level to be achieved by the release time for the proposed model. A numerical illustration is given for both type of release problem and finally a sensitivity analysis is performed to determine the effect of variations in minimum reliability level to be achieved by release time and various costs involved in cost model on optimal release time and cost.

Reference:

1. Ammann P.E., Brilliant S.S., and Knight J.C, "The Effect of Imperfect Error Detection on Reliability," IEEE Trans. Software Eng., vol. 20, pp. 142-148, 1994.
2. Kapur P.K., Garg R.B., and S. Kumar, "Contributions to Hardware and Software Reliability", World Scientific, Singapore.**1999**.
3. Kapur P.K., Garg R.B., "Optimal Software Release Policies for Software Reliability Growth Models under Imperfect Debugging", Recherché operationanelle/Operations Research, vol 24, pp. 295-305,1990.
4. Kapur P.K., Agarwal S., Garg R.B., " Bi-criterion Release Policy for Exponential Software Reliability Growth Models ", Recherche operationanelle/Operations Research, vol 28, pp. 165-180,1994.
5. Kapur P.K. and Younes S., "Modeling an Imperfect Debugging Phenomenon in Software Reliability," Microelectronics and Reliability, vol. 36, pp. 645-650, 1996.

6. Kapur PK, Bhalla VK. “Optimal release policy for a flexible software reliability growth model” *Reliability Engineering and System safet*y **1992**; 35: 49-54.
7. Kapur PK, Garg RB, Bahlla VK. “Release policies with random software life cycle and penalty cost” *Microelectronics Reliability* **1993**; 33 (1): 7-12.
8. Ohba M. and Chou X.M., “Does Imperfect Debugging Affect Software Reliability Growth?” Proc. 11th Int’l Conf. Software Eng., pp. 237-244, 1989.
9. Pham H., “Software Reliability”, Springer-Verlang Singapore Pte. Ltd. 2000.
10. Pham H, “A Software Cost Model with Imperfect Debugging, Random Life Cycle and Penalty Cost,” Int’l J. Systems Science, vol. 27, pp. 455-463, 1996.
11. Shanthikumar J.G., “A State and Time-Dependent Occurrence Rate Software Reliability Model with Imperfect Debugging,” Proc. Nat’l Computer Conf., pp. 311-315, 1981.
12. Slud E., “Testing for Imperfect Debugging in Software Reliability,” Scandinavian J. Statistics, vol. 24, pp. 555-572, 1997.
13. Xie M., “ A Study of the Effect of Imperfect debugging on Software Development Cost”, IEEE Transactions on Software Engineering, vol 29, No 5, May 2003.