ABSTRACT
The fault detection and correction process is the most important phase in software testing and development cycle. A number of software reliability growth models (SRGMs) have been proposed in recent years to capture the time lag between detected and corrected fault numbers. But unfortunately most of the models were discussed under static environment. The main purpose of this paper is to investigate an optimal resource allocation plan for fault detection and correction model to minimize the cost of software during the testing and operational phase under dynamic condition. An elaborate optimization policy based on the optimal control theory is proposed and numerical examples are illustrated. The paper also studies the optimal resource allocation problems for various conditions by examining the behavior of the model parameters. The experimental results greatly help us to identify the contributions of each selected parameter and its weight.

KEYWORDS

1. INTRODUCTION
In the recent years computer systems have been extensively used to manage the complex and vital systems. A fault in the software may lead to huge losses in terms of money and time. There are several reported and unreported instances where failure of computer controlled systems have resulted in tremendous harm of social life. Therefore, when an organization invests a large amount of money in making a software product it ensures that the software product must be acceptable to the users in the prevalent market conditions. Software development and testing is typically a multistage process, where each stage performs some predefined activities, aimed to deliver the end product as per the commitment made to the customer. This is where testing plays an important role. Software testing is not only a process of debugging the software but it also evaluates the software quality. Software testing interacts with every phase of SDLC. Software reliability growth has been a key issue in the software industry since it can provide significant information for the management during the software testing life cycle (STLC). Mathematical models known as Software reliability growth models (SRGM) establish the relation among failure observation, fault removal process and software reliability. For the last three decades, many SRGM have been proposed to study the probability that software will not cause the failure of a product for specified time under specified conditions. This probability is a function of the inputs from the users and use of the product, as well as a function of the existence of faults which is in software. Resources such as manpower and CPU time are consumed during testing process. The fault detection and correction processes depend upon the nature and amount of resources consumed. Many SRGMs have been proposed to minimize the amount of testing efforts utilized during the testing cycle but mostly under the assumptions that the relationship between the testing effort consumed and testing time (the calendar time) follows Exponential and Rayleigh distribution. Basili et al [1] Huang et al [4], Putman[11], Kapur and Garg [7] and Yamada et al. [16] discussed the time dependent behavior of testing efforts in their pioneer work. Usually, exponential curve is used to describe the behavior of testing resources whenever consumed uniformly, otherwise Rayleigh curve is used. Logistic and Weibull type functions have also been used to describe testing efforts. Musa et al. [8] assumed that the resource consumption as an explicit function of number of faults removed and calendar time. Kapur et al. [6] in their work have discussed the optimization problem of allocating testing resources in software having modular structure and proposed that allocation of efforts should depend upon the size and severity of faults. Various authors have recommended that for different resource constraints one can develop a trade-off between maximum number of faults to be removed in each module and the efforts required.

Various proposed SRGMs in four decades only consider the fault detection process and are based on the assumption that a fault is removed immediately as soon as it is detected. However, this assumption may not be reasonable as the detected faults are rarely corrected immediately (Gokhale et al [2], Schneidewind [12], Schneidewind [13], Ohba [9] and Xie and Zhao [15]). Generally, whenever a fault is detected, first it is reported, next diagnosed, verified and finally it gets corrected. Therefore, the time lag between detection and correction should not be neglected during the modeling process. Hence, it is essential to have different growth models for modeling the detection-correction process. Many researchers emphasized the significance of the fault detection-correction modeling. Schneidewind [12] first modeled the
fault-correction process by using a constant delayed fault-detection process. Later, Xie and Zhao [15] extended the Schneidewind model to a continuous version by substituting a time-dependent delay function for the constant delay which measures the expected time lag to correct a detected fault. Kapur et. al. [5] proposed a remarkable model by incorporating time dependent lag function during modeling of dependent fault detection process. Yamada et al [17] in their research work discussed the time lag between two stage process (failure and removal) by taking constant removal rate. Later Huang and Lin [3] in their research paper discussed the dependency of faults and time lag between failure observation and fault removal.

Though, over last thirty years many SRGMs have been proposed to minimize the total efforts expenditures during testing phase of SDLC but mostly under the static assumption. In this article we have tried to investigate an optimal resource allocation plan to minimize the cost of software during the testing phase using fault detection and correction model under dynamic conditions. Experimental results show that the proposed framework to incorporate debugging time lag for SRGM has a fairly accurate optimal distribution of effort policy. The paper also studies the optimal resource allocation problems for various conditions by examining the behavior of the model parameters.

2. PROBLEM DESCRIPTION

The main objective of software testing is detection and correction of faults before the release of the software in the market. Generally whenever a failure is identified the fault correction team requires a period of time to locate the fault and modify some codes accordingly to remove it. Thus, the time lag between detection and correction is a common experience in software testing. In general, this time lag is the time delay between the fault detection and correction processes. The removal time of a fault depends on various factors such as the number of the detected faults, the complexity of the faults, structural complexity of the software, the skill of the correction personnel, etc. Thus the correction lag can’t be ignored. Some faults which are detected but not corrected still remain in the software. These latent faults are caused by the correction lag and reflect the relationship between the fault detection and correction processes. In this research paper we recommend that testing, identification and correction of a fault should be viewed as simultaneous activities. Our main objective is to construct a simple, structured model of concurrent testing, identification and debugging with a view to gain insights. To develop the mathematical model we have assumed that during the Software Testing Life Cycle (STLC), there are separate teams for detecting, locating and correcting the faults present in the software. A team identifies the number of faults causing failure of software, second team locates or detects the faults and the third team is removes the detected faults. Therefore, the fixed total resources (W) at any point of time ‘t’ can be divided into three portions ‘w₁(t)’, ‘w₂(t)’ and ‘w₃(t)’ (where, w₁(t) is the current effort consumption for testing purpose at time ‘t’, and w₂(t) is the efforts consumed for detection of the faults at any point of time ‘t’ and w₃(t) is the current effort consumption due to fix fault at time ‘t’). The resource allocation model is depicted in the figure 1.

![Allocation of Total resources](image)

**Figure 1: Allocation of Total resources**

To solve the dynamic optimization problems for resource allocation problem, we have used optimal control theory approach and assumed that software testing and debugging can run concomitantly. Optimal control theory has been accepted widely as a technique to solve the dynamic optimization problems in many areas of engineering, management, economics, etc. Optimal control theory decouples a dynamical system over the time into a simple static optimization problems at a time instance ‘t’ and the economic insights can be viewed from the analytical results [14]. The control variables here ‘w₁(t)’ and ‘w₂(t)’ manage the evolution of a system in such a way that an optimal outcome (here, minimum cost) is achieved by the end of the time horizon. In the paper the relation between failure rates of software and cost to decrease this rate is modeled by various types of learning curves effect. We begin our analysis by stating a general model with a very few assumptions. The notations used in the analysis are as follows:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Planning period.</td>
</tr>
<tr>
<td>W</td>
<td>Total resources utilized during the SDLC at any point of time ‘t’.</td>
</tr>
<tr>
<td>w₁(t)</td>
<td>Resources utilized during the SDLC for testing purpose at any point of time ‘t’.</td>
</tr>
<tr>
<td>w₂(t)</td>
<td>Resources utilized during the SDLC to fix a bug at any point of time ‘t’.</td>
</tr>
</tbody>
</table>

Copy Right © INDIACom-2011 ISSN 0973-7529 ISBN 978-93-80544-00-7
\[ w_3(t) \]: Resources utilized during the SDLC for detection purpose at any point of time \( t \).

\[ c_1(m_f(t), w_3(t)) \]: Total cost per unit at time \( t \) for cumulative fault detected is \( m_f(t) \) with detection efforts \( w_3(t) \).

\[ c_2(m_r(t), w_2(t)) \]: Total cost per unit at time \( t \) for cumulative fault corrected is \( m_r(t) \) debugging efforts \( w_2(t) \).

\[ c_3 \]: Total cost of testing per unit time \( t \).

\[ a \]: Initial fault content present in the software.

### 3. MODEL DEVELOPMENT

#### 3.1 FAULT DETECTION AND CORRECTION MODELING

Under the ideal assumption of instantaneous and perfect fault removal, the expected number of faults corrected is the same as the expected number of faults detected. However, if we take into consideration the time required for correction, the expected number of faults corrected at any given time is less than the expected number of faults detected. To propose fault detection-correction model we assume that the removal of a fault is done after a fault is detected. Hence, the removal is done in two stages. In the first stage, the testing team identifies the faults. In the second stage, another debugging team removes the fault causing that failure. To begin with, let \( a \) be the amount of fault present in the software, \( m_f(t) \) denotes the expected number of faults detected till time \( t \), and \( m_r(t) \) denotes the expected number of faults removed till time \( t \). Therefore it is reasonable to assume the following differential equations for detection and correction process.

\[
x(t) = \frac{dm_f(t)}{dt} = b_1 w_3(t)(a - m_f(t))
\]

and

\[
y(t) = \frac{dm_r(t)}{dt} = b_2 w_2(t)(m_f(t) - m_r(t))
\]

where

\[ m_f(0) = 0, m_r(0) = 0 \]

Where \( b_1 \) and \( b_2 \) are the fault detection and correction rate respectively.

### 3.2 COST OPTIMIZATION MODELING

Now suppose the software firm wants to minimize the total expenditure over the finite planning horizon \( T \). Thus, the total cost at any point of time \( t \) during the testing phase of SDLC is the sum of testing cost \( c_3 w_1(t) \), detection cost \( c_1(t)x(t) \) and debugging cost \( c_2(t)y(t) \). Therefore, mathematically the model can be represented as

\[
\min_{0}^{T} \int [c_1(t)x(t) + c_2(t)y(t) + c_3 w_1(t)] dt
\]

subject to

\[
x(t) = \frac{dm_f(t)}{dt} = b_1 w_3(t)(a - m_f(t))
\]

\[
y(t) = \frac{dm_r(t)}{dt} = b_2 w_2(t)(m_f(t) - m_r(t))
\]

where

\[ m_f(0) = 0, m_r(0) = 0, c_1(t) = c_1, c_2(t) = c_2, c_3 = c_3 \]

and \( w_1(t) + w_2(t) + w_3(t) = w \)

\[ (w_1(t), w_2(t), w_3(t)) \geq 0 \]  \( \text{(2)} \)

### 4. OPTIMAL SOLUTION

To solve the problem, Maximum principle can be applied. The current value Hamiltonian is as follows [14]:

The Hamiltonian for the system stated by equation (2), is defined as

\[
H(m_f(t), m_r(t), \lambda(t), w_1(t), w_2(t), w_3(t), t) = -[c_1(t)x(t) + c_2(t)y(t) + c_3 w_1(t)] + \lambda(t)x(t) + \mu(t)y(t) \quad \text{(3)}
\]

where, \( \lambda(t) \) and \( \mu(t) \) are the adjoint variables (shadow cost of \( x(t) \) and \( y(t) \) respectively) which satisfy the following differential equation.

\[
\frac{d}{dt} \lambda(t) = \lambda = -H_{m_f} \quad \text{(4)}
\]

with the transversality condition at \( t = T \), \( \lambda(T) = 0 \)

Similarly

\[
\frac{d}{dt} \mu(t) = \mu = -H_{m_r} \quad \text{(5)}
\]

The transversality condition at \( t = T \), \( \mu(T) = 0 \)

The adjoint variable \( \lambda(t) \) represents the per unit change in the objective function for a small change in \( m_f(t) \) i.e. \( \lambda(t) \) can be interpreted as marginal value of faults detected at time \( t \). Similarly \( \mu(t) \) can be interpreted as marginal value of fault removed at time \( t \). Thus the Hamiltonian is the sum of
current cost \((c_1x + c_2y)\) and the future cost \((\lambda x + \mu y)\). In short, \(H\) represents the instantaneous total cost of the firm at time \(t\).

The following are the necessary conditions hold for an optimal solution:

\[
H_{w_1} = 0; \quad H_{w_2} = 0
\]

\[
\Rightarrow -c_{w_1}(t)x(t) - (c_1(t) - \lambda(t))x_{w_1}(t) - c_3 = 0
\]

\[
\Rightarrow -c_{w_2}(t)x(t) - c_{2w_2}(t)y(t) - (c_2(t) - \mu(t))y_{w_2}(t) = 0
\]

The other optimality conditions are

\(H_{w_1} < 0\) and \(H_{w_1}H_{w_2} - H_{w_2} > 0\)

From the above optimality conditions, we have

\[
w_1 = -\frac{(c_1(t) - \lambda(t))b_1(a - m_f(t)) + c_3}{c_{1w_1}(t)b_1(a - m_f(t))}
\]

And

\[
w_2 = -\frac{(c_2(t) - \mu(t))b_2(m_f(t) - m_r(t)) + c_{2w_2}(t)x(t)}{c_{2w_2}(t)b_2(m_f(t) - m_r(t))}
\]

Therefore

\[
w_3 = w + \frac{(c_1(t) - \lambda(t))b_1(a - m_f(t)) + c_3}{c_{1w_1}(t)b_1(a - m_f(t))}
\]

\[
+ \frac{(c_2(t) - \mu(t))b_2(m_f(t) - m_r(t)) + c_{2w_2}(t)x(t)}{c_{2w_2}(t)b_2(m_f(t) - m_r(t))}
\]

Integrating (4) with the transversality condition, we have the future cost of detecting one more fault from the software

\[
\lambda(t) = -\int_{t}^{T} \frac{\partial c_1}{\partial m_f} x(t) + (c_1(t) - \lambda(t)) \frac{\partial \lambda}{\partial m_f} + (c_2(t) - \mu(t)) \frac{\partial \mu}{\partial m_r} dt
\]

Similarly integrating (5) with the transversality condition, we have the future cost of removing one more fault from the software

\[
\mu(t) = -\int_{t}^{T} \frac{\partial c_2}{\partial m_r} y(t) + (c_2(t) - \mu(t)) \frac{\partial \mu}{\partial m_r} dt
\]

Now taking time derivative of (6), we have

\[
\dot{w}_1 H_{w_1} + \dot{w}_2 H_{w_2} + x(t)\dot{H}_{w_1} + y(t)\dot{H}_{w_1} = -\lambda x_{w_1}
\]

(13)
\[
\min_0^T \left[ c_1x(t) + c_2y(t) + c_3w_1(t) \right] dt
\]
subject to
\[
x(t) = \frac{dm_f(t)}{dt} = b_1w_1(t)(a - m_f(t))
\]
\[
y(t) = \frac{dm_f(t)}{dt} = b_2w_2(t)(m_f(t) - m_f(t))
\]
where
\[
m_f(0) = 0, m_r(0) = 0 \quad \text{and} \quad w_1(t) + w_2(t) + w_3(t) = w
\]
where \( H = -[c_1x(t) + c_2y(t)] + \lambda(t)x(t) + \mu(t)y(t) - c_3w_1(t) \) \hspace{1cm} (17)

where \( \lambda(t) \) is the adjoint variable, which satisfy the following differential equations.
\[
d\lambda(t) = -H_{m_f} = -(c_1 - \lambda)b_1w_1(t) + (c_2 - \mu)b_2w_2(t)
\]

Similarly \( \mu(t) \) is the adjoint variable, which satisfy following differential equations.
\[
d\mu(t) = -H_{m_r} = -(c_2 - \mu)b_2w_2(t)
\]
Solving the differential equation (20) together with the terminal condition, we get
\[
\lambda(t) = \int_t^T [(c_1 - \lambda)b_1w_1(t) - (c_2 - \mu)b_2w_2(t)] d\tau \hspace{1cm} (19)
\]

We get,
\[
\mu(t) = \int_t^T [(c_2 - \mu)b_2w_2(t)] d\tau
\]

Since Hamiltonian (17) is linear in control variable \( w_1(t) \) and \( w_2(t) \). Therefore we have the following optimal policy for \( w_1(t) \) and \( w_2(t) \) which maximize the objective function:
\[
w_1^*(t) = \begin{cases} w & \text{if} \quad \lambda(t)b_1(a - m_f(t)) > c_1 + c_2b_1(a - m_f(t)) \\ \text{undefined} & \text{if} \quad \lambda(t)b_1(a - m_f(t)) = c_1 + c_2b_1(a - m_f(t)) \\ 0 & \text{if} \quad \lambda(t)b_1(a - m_f(t)) < c_1 + c_2b_1(a - m_f(t)) \end{cases}
\]
\[
w_2^*(t) = \begin{cases} w & \text{if} \quad \mu(t) > c_2 \\ \text{undefined} & \text{if} \quad \mu(t) = c_2 \\ 0 & \text{if} \quad \mu(t) < c_2 \end{cases}
\]

5.2. CASE 2
In this section, we introduce the concept of learning curve phenomenon in the cost functions. Learning curve phenomenon in software fault correction is concerned with the idea that when a new fault has to be debugged for the first time it is likely that the debugging team involved will not achieve maximum efficiency immediately. Repetition of the task will make the people more confident and knowledgeable and will eventually result in a more efficient and rapid operation. Eventually the learning process will stop after continuous imitation. It states that more often a task is performed; the lower will be the cost of doing it. In this paper, it is assumed that each time cumulative volume of fault detected increases, value added costs (including administration, debugging etc.) fall by a constant and predictable percentage. In this section we have considered detection cost and debugging cost follows learning effect phenomenon (Pegels [10]), which are of the form
\[
c_1(w_3(t), m_f(t)) = c_0w_3(t)^m_r(t)
\]

where, \( c_0 \) is the base detection cost.

Also \( c_2(w_2(t), m_r(t)) = b_0w_2(t)^m_r(t), \) where \( b_0 \) is the base cost for debugging.

The Hamiltonian for the above case can be written as
\[
H = -[c_1x(t) + c_2y(t)] + \lambda(t)x(t) + \mu(t)y(t) - c_3w_1(t)
\]

The following are the necessary condition hold for an optimal solution:
\[
H_{w_1} = 0 \quad ; \quad H_{w_2} = 0
\]
The adjoint variables given by the following differential equation
\[
\lambda(t) = \int_t^T (c_1(t) - \lambda(t)b_1w_1(t) - (c_2(t) - \mu(t))b_2w_2(t)) dt \hspace{1cm} (26)
\]
And
\[
\mu(t) = \int_t^T (c_2(t) - \mu(t))b_2w_2(t) - c_2(t)y(t)log(w_2(t))) dt \hspace{1cm} (27)
\]
Solving (25) , we get
$w_2^*(t) = \left[ \frac{m_f(t)c_1(t)x(t)}{b_2w_3(t)(m_f(t) - m_r(t))} - \frac{(c_2(t) - \mu(t))}{b_0m_r(t)} \right]^{1/m_3}$

Therefore

$w_1^*(t) = w - w_2^*(t) - w_3^*(t)$

The physical interpretation of the above equation (28) is that the optimal policy for detecting effort is equal to the $(m_f(t) - 1)$th root of the ratio of total cost of detecting and fixing per unit bugs to per unit base cost of detection multiplied by the number of errors removed at time ‘$t$’.

6. NUMERICAL ANALYSIS

In this section, the various optimal policies have been described on the proposed model using numerical example. The purpose of this study is to get some insight into the result and also to study the impact of change in efforts on the proposed cost model and the corresponding optimal cost model. Several simulation runs were conducted using various parameter values; results converged quickly and were stable. To do so, we have considered the Pegels [10] form of learning curve to define the detection and correction cost functions. First some base values were considered and then different model parameters were varied individually. The base values are as follows:

$a = 100, b_1 = 0.3, b_2 = 0.3, w_1 = 0.30, w_2 = 0.4, w = 1, \lambda(0) = 100, \mu(0) = 100, m_f(0) = 0, m_r(0) = 0, c_3 = 500, c_0 = 1000, b_0 = 1000$

The expected number of fault detected and corrected is shown in Figure 2(a) and Figure 2(b). Initially, we obtain the number of fault detected for various values of $w_1$. The expected number of faults detected decreases as $w_1$ decreases. Finally, the value of $w_1$ was set to be .38. The cumulative number of faults corrected increases as the debugging efforts increases and follows s-shaped curve. From the analysis it has been seen that as we keep on increasing the debugging efforts, the time duration of achieving the desired goal i.e. (maximizing the number of corrected faults) reduces fast. The result also indicated that increasing the debugging efforts reduces the total debugging cost and hence total cost. This situation may arise, due to the experience curve phenomenon.

CONCLUSION

In this paper we have studied an optimal resource allocation plan to minimize the cost of software when fault detection/correction process is a two stage process. Using optimal control theoretic approach we have obtained number of policies for detection effort and correction effort for the different cost function. An analysis of different fault removal strategies is also discussed. We describe a method to compute the cost of the software during testing phase taking into consideration. The results were verified using simulation technique.

FUTURE SCOPE

The model is based upon the strict assumption that at any point of time ‘$t$’ the total resources is fixed as a result one variable is dependent on the other two variables. Hence controlling the two variables will automatically control the third one. Also, it has been considered that the per unit testing cost is constant throughout the testing period functions, but in practice this may be function of time. Hence it is very important to consider a time dependent testing cost function for optimization purpose. Finally the model can be extended in several ways i.e. incorporating the other imperfect debugging software reliability growth models.
REFERENCES


