Modern complex software systems are often developed with components supplied by independent teams under various environments. A component based software development approach has become a trend in integrating modern software system. To ensure the overall reliability of a software system, its software components have to meet certain reliability requirements. For whole systems with its components, the system-testing problem can be formulated as a combination of optimization problem with known cost, reliability, effort, maintenance and other attributes of the system components. In the present investigation, we address the issue of determining the optimal testing time so that the total maintenance cost of the software could be minimized and a desired level of reliability could be achieved. Since software maintenance is an ongoing process required to keep software useful, a modified approach is discussed to determine the delivery cost of software, when warranty is provided during the cost incurred in the maintenance phase and is paid by the developer. A technique for estimating the parameter of the software reliability growth models is also described. Numerical illustrations support the validity of the analytic results.

2.1 Introduction

The software reliability models play a key role in modeling the growth of the software, tested over a period of time. The increasing role of software in every sphere of life creates both requirements for being able to trust it more than before, and for more people to know how much they can trust their software. The importance of research in software reliability and testing is not surprising because the users have new and updated versions of the software due to continuing growth in the development of software. Reliability growth testing is performed to quantify the current reliability, isolating and eliminating faults and to predict future reliability. The testing of modern software systems for isolating and removing defects is a difficult and costly business and should be dealt with scientifically. The applications of software reliability growth modeling are increasing exponentially due to development...
of complex softwares. The goal of software testing is the production of software which should be error-free as much as possible.

However, it is difficult for developers to produce highly reliable software systems. It is essential to control the software in terms of reliability, cost and release time. The quality of the software system depends upon the length of testing and on what type of testing methods is used. From longer testing time we expect the more reliable software, but it may be too expensive and may results in a late delivery. On the other hand, the short testing gives the unreliable software and as a result it may increase the cost during the operational phase. Therefore, it is very important to ensure when to stop testing or when to release the software so that the total system cost is minimized with respect to some testing schedule and resource constraints.

Many researchers have discussed the software release problems and have presented different policies (cf. Okumoto and Goel 1980; Yamada and Osaki, 1986). Kapur and Bhalla (1992) suggested optimal release policies for a flexible reliability growth model. Kapur et al. (1993) developed an exponential software reliability growth model (SRGM) with a bound on the number of failures. An optimal release problem with warranty period based on a software maintenance cost model has been discussed by many researchers and the results have been applied to many practical situations (cf. Yamada, 1994; Pham and Zhang, 1999; Kimura et al., 1999). Jain and Handa (2001) suggested cost analysis for repairable units under hybrid warranty. Williams and Vivekanandan (2005) developed life-time warranty cost model for software reliability with discount rate. Williams (2007) studied the warranty cost model for software reliability with an imperfect debugging phenomenon.

Some studies of the effect of imperfect debugging on the software cost model were made by Xia et al., (1993), Pham, (1996); Xie and Yang (2003) and others. Chatterjee et al. (1997) estimated the test effort and learning factor on the software reliability and optimal release policies. Popstojanova and Trivedi (2001) presented architecture based approach for quantitatively assessment the optimal software testing time by assuming that the SRGM follows mixed distribution i.e., exponential and Rayleigh for different modules. Optimal release policies for cost and reliability of software systems in the testing efficiency have been discussed from time to time by
many researchers (cf. Tal et al., 2002; Jain and Priya 2002; Huang, 2005a; Huang and Lo, 2006). Boland and Chuiv (2007) considered optimal times for the software release when repair is imperfect.

In this chapter, we study the optimal testing time of the software. The software development cost, warranty cost, discount rate, imperfect debugging, and reliability requirement are taken into consideration while formulating the software reliability growth model (SRGM). The mixed distribution which is a combination of exponential and Rayleigh distributions is used for fault detection for different modules. A technique for estimation of the parameter of the SRGM is also presented. The rest of the chapter is organized as follows. In section 2.2, the SRGM is described along with requisite assumptions and nomenclature. In section 2.3, the maintenance cost model is described. Optimal software release policies are elaborated in section 2.4. In section 2.5, the estimation of parameters by the maximum likelihood method is suggested. In section 2.6, numerical illustrations are provided to examine the optimal testing policies. The concluding remarks are given in the last section 2.7.

2.2 Model Description

Consider a SRGM in which fault detection is characterized by a mixture of Exponential and Rayleigh distributions. This specifies that some modules of the software follow the exponential distribution and some other modules follow the Rayleigh distribution with their respective probabilities.

Following assumptions are made for describing the SRGM:

- The software consists of two types of modules, which follow different failure patterns.
- The failure detection rate is different for different types of modules.
- All faults in the software are mutually independent from the failure detection point of view.
- The number of failures that are detected at any time is proportional to the current faults in the software.
- Whenever an error occurs, it is immediately removed and at that moment, no other new errors are introduced in the software.
There is a warranty period in the operational phase of the software where the maintenance cost is paid by the developer.

The discount maintenance cost is considered to take care of the present value of the money.

The notations used for the mathematical formulation of the model are as follows:

- \( a_1 (a_2) \): Initial number of errors in the software for exponential (Rayleigh) distribution.
- \( b_1 (b_2) \): Fault detection rate for exponential (Rayleigh) distribution.
- \( d_1 (d_2) \): The probability of perfect debugging for exponential (Rayleigh) distribution.
- \( C_o \): Initial testing cost.
- \( C_t \): Testing cost per unit time.
- \( C_w \): Maintenance cost per fault during the warranty period.
- \( T \): Release time of the software.
- \( T^* \): Optimum time of software release.
- \( T_w \): Warranty period.
- \( \alpha \): Discount rate of the cost.
- \( EC(T) \): Expected total maintenance cost of software.
- \( C_w(T) \): Maintenance cost during the warranty period.

The expected number of failures detected at time \( t \) is given by

\[
m(t) = \frac{a_1}{d_1} p (1 - e^{-b_1d_1t}) + \frac{a_2}{d_2} (1 - p) \left( 1 - e^{-b_2d_2t} \right) \quad \ldots (2.1)
\]

where \( p \) is the proportion of first module having exponential distribution for failures and \( (1-p) \) is the proportion of second module having Rayleigh distribution for failures.

The failure intensity function is obtained as follows:

\[
\lambda(t) = \frac{d}{dt} [m(t)] = a_1 b_1 p e^{-b_1d_1t} + 2a_2 b_2 t (1 - p) e^{-b_2d_2t} \quad \ldots (2.2)
\]
2.3 Cost Model

In this section, we discuss the optimal testing time by considering the developmental cost and warranty cost with the discount rate. First of all we construct cost model for the software by assuming that there are three types of costs i.e., (i) an initial testing cost, (ii) testing cost per unit time, and (iii) the maintenance cost during the warranty period.

When the cost of software development is estimated, the software developers also have to consider the cost of after-sales support. This is known as the warranty cost. The computation in this warranty cost is dependent on the release time of the software.

Hence, the total expected software maintenance cost is given by

\[ EC(T) = C_0 + C_t \int_0^T e^{-\alpha t} dt + C_w(T) \] … (2.3)

In order to determine the maintenance cost \( C_w(T) \) during the warranty period, we consider the following two cases depending on the warranty period:

**Case 1:** In this case, it is assumed that during the warranty period, the software reliability growth does not occur. Here we assume that we correct only minor errors that will not affect the reliability of the software. Then \( C_w(T) \) is defined as

\[ C_w(T) = C_w \int_T^{T+T_w} \lambda(T) e^{-\alpha t} dt \] … (2.4)

**Case 2:** We assume that during the warranty period, the software reliability growth occurs, even after the testing phase. In this case we correct only major errors that will improve the reliability of the software. Now \( C_w(T) \) is given by:

\[ C_w(T) = C_w \int_T^{T+T_w} \lambda(t) e^{-\alpha t} dt \] … (2.5)
Substituting the value of $C_w(T)$ from equations (2.4) and (2.5) in equation (2.3), we get the expected total maintenance cost as follows:

$$EC(T) = \begin{cases} 
C_o + C_i \int_0^T e^{-\alpha t} dt + C_w \int_{T+T_w}^{T+T_w} \lambda(T) e^{-\alpha t} dt, & \text{for case 1} \\
C_o + C_i \int_0^T e^{-\alpha t} dt + C_w \int_T^{T+T_w} \lambda(t) e^{-\alpha t} dt, & \text{for case 2} 
\end{cases} \quad \ldots (2.6)$$

Differentiating equation (2.6) with respect to $T$ and equating it to zero, we get the optimal testing time $T^*$ as follows:

For case 1, $T^* = T_1$.

$$\frac{C_i \alpha}{C_w(1-e^{-\alpha T_1})} = a_1 b_1 p e^{-b_1 d_1 T} \left[ \alpha + b_1 d_1 \right] + 2 a_2 b_2 (1-p) e^{-b_2 d_2 T^2} \left[ T(a + 2 b_2 d_2 T) - 1 \right]$$

$\ldots (2.7)$

For case 2, $T^* = T_2$.

$$e^{-\alpha T} \left[ C_i + C_w a_1 b_1 p e^{-b_1 d_1 T} \left[ e^{-b_1 d_1 T} - 1 \right] \right] + 2 C_w a_2 b_2 (1-p) \frac{d}{dT} \int_T^{T+T_w} t e^{-b_2 d_2 t + \alpha} dt = 0$$

$\ldots (2.8)$
We can see that the second derivative of $EC(T)$ is greater than zero for both cases. i.e.,

$$\left[ \frac{d^2 EC(T)}{dT^2} \right]_{T=T_i} > 0 \text{ and } \left[ \frac{d^2 EC(T)}{dT^2} \right]_{T=T_i} > 0$$

Hence, $EC(T)$ gives a minimum value at $T^*=T_1$ and $T^*=T_2$ for both cases, respectively.

**Optimal Software Release Policies**

Now we suggest the optimal release time policies by minimizing the total expected software cost $EC(T)$ as follows:

(i) **Optimal Testing Policy 1- for Case 1.**

- **P1.1** $T^* = T_1$ when $\lambda_>(0) > \lambda_>(T_1)$
- **P1.2** $T^* = 0$ when $\lambda_>(0) \leq \lambda_>(T_1)$

(ii) **Optimal testing Policy 2- for case 2,**

- **P2.1** $T^* = T_2$ when $\lambda_>(0) > \lambda_>(T_2)$
- **P2.2** $T^* = 0$ when $\lambda_>(0) \leq \lambda_>(T_2)$

### 2.4 Warranty Cost Model with Reliability Constraint

The software reliability of the NHPP model is defined as the probability that a software failure will not occur during the testing time interval $(T, T+x]$ and is defined as

$$R\left(\frac{X}{T}\right) = \exp[-\{m(T+x) - m(T)]$$

... (2.9)

Using eq. (2.1) in eq. (2.9), we get

$$R(X/T) = \exp\left[-\left\{\frac{a_1}{d_1}pe^{-b_1d_1T}(1-e^{-b_1d_1X}) + \frac{a_2}{d_2}(1-p)(e^{-b_2d_2T} - e^{-b_2d_2(T+X)})\right\}\right]$$

... (2.10)
Let $R_0 \ (0 < R_0 \leq 1)$ be the desired level of reliability. Then the optimal release problem is defined as

Minimize $EC(T)$

Subject to $R(x/T) \geq R_0$ \hspace{1cm} \ldots (2.11)

Let $T_R$ denote the optimal time satisfying the above constraint. Thus using the relation $R(x/T) = R_0$ in equation (2.9), we get

$$R_0 = \exp [- \{m(T+x) - m(T) \}] \quad \ldots (2.12)$$

The optimal release policies, including minimization of the total expected software cost can be stated for the two cases identified earlier as follows:

(i) \hspace{1cm} \textbf{For case 1: Optimum Release Policy 3.}

\begin{align*}
\text{P 3.1} & \quad \text{If } \lambda(0) > \lambda(T_1) \text{ and } R(x/0) < R_0, \\
& \quad \text{then } T^* = \max \{T_1, T_R\} \\
\text{P 3.2} & \quad \text{If } \lambda(0) > \lambda(T_1) \text{ and } R(x/0) \geq R_0, \\
& \quad \text{then } T^* = T_1 \\
\text{P 3.3} & \quad \text{If } \lambda(0) \leq \lambda(T_1) \text{ and } R(x/0) < R_0, \\
& \quad \text{then } T^* = T_R \\
\text{P 3.4} & \quad \text{If } \lambda(0) \leq \lambda(T_1) \text{ and } R(x/0) \geq R_0, \\
& \quad \text{then } T^* = 0 \\
\end{align*}

(ii) \hspace{1cm} \textbf{For Case 2: Optimum Release Policy 4.}

\begin{align*}
\text{P 4.1} & \quad \text{If } \lambda(0) > \lambda(T_2) \text{ and } R(x/0) < R_0, \\
& \quad \text{then } T^* = \max \{T_2, T_R\} \\
\text{P 4.2} & \quad \text{If } \lambda(0) > \lambda(T_2) \text{ and } R(x/0) \geq R_0, \\
& \quad \text{then } T^* = T_2 \\
\text{P 4.3} & \quad \text{If } \lambda(0) \leq \lambda(T_2) \text{ and } R(x/0) < R_0, \\
& \quad \text{then } T^* = T_R \\
\text{P 4.4} & \quad \text{If } \lambda(0) \leq \lambda(T_2) \text{ and } R(x/0) \geq R_0, \\
& \quad \text{then } T^* = 0 \\
\end{align*}
2.5 Estimation of Parameters

The estimation of parameters is an important aspect of the software reliability prediction. We suggest the maximum likelihood estimation (MLE) technique for estimating the unknown parameters i.e. $p$, $a_1$, $b_1$, $d_1$, $a_2$, $b_2$, and $d_2$ in the SRGM described in the previous section.

Let $t_1$, $t_2$, ..., $t_n$ be the random failure time of $n$ items where $0 < t_1 < t_2 < ... < t_n$. Then log likelihood function (LLF) is given by

$$\text{LLF} = \sum_{i=1}^{n} \left( y_i - y_{i-1} \right) \log \left[ m(t_i) - m(t_{i-1}) \right] - m(t_n) \quad \ldots (2.13)$$

where, $m(t) = \frac{a_1}{d_1} p \left( 1 - e^{-b_1 d_1 t} \right) + \frac{a_2}{d_2} (1 - p) \left( 1 - e^{-b_2 d_2 t} \right)$.

Here, $y_i$ is the cumulative number of detected errors. Now the maximum of the LLF is determined by using the following system of equations:

$$0 = \sum_{i=1}^{n} \frac{\partial}{\partial \psi} m(t_i) - \frac{\partial}{\partial \psi} m(t_{i-1}) \left( y_i - y_{i-1} \right) - \frac{\partial}{\partial \psi} m(t_n) \quad \ldots (2.14)$$

where all unknown parameters $p$, $a_1$, $b_1$, $d_1$, $a_2$, $b_2$, and $d_2$, can be put one by one in place of $\psi$. So we obtain

$$\sum_{i=1}^{n} \frac{\left[ \frac{a_1}{d_1} \left( e^{-b_1 d_1 t_i} - e^{-b_1 d_1 t_{i-1}} \right) + \frac{a_2}{d_2} \left( e^{-b_2 d_2 t_i^2} - e^{-b_2 d_2 t_{i-1}^2} \right) \right] [y_i - y_{i-1}]}{D}$$

$$- \left[ \frac{a_1}{d_1} \left( 1 - e^{-b_1 d_1 t_n} \right) + \frac{a_2}{d_2} \left( 1 - e^{-b_2 d_2 t_n^2} \right) \right] = 0 \quad \ldots (2.15)$$

$$\sum_{i=1}^{n} \frac{p \left[ \left( e^{-b_1 d_1 t_i} - e^{-b_1 d_1 t_{i-1}} \right) [y_i - y_{i-1}] \right]}{D} - \frac{1}{d_1} p \left[ 1 - e^{-b_1 d_1 t_n} \right] = 0 \quad \ldots (2.16)$$

$$\sum_{i=1}^{n} \frac{a_1 p \left[ \left( e^{-b_1 d_1 t_i} - e^{-b_1 d_1 t_{i-1}} \right) [y_i - y_{i-1}] \right]}{D} - a_1 p \left[ 1 - e^{-b_1 d_1 t_n} \right] = 0 \quad \ldots (2.17)$$
\[
\sum_{i=1}^{n} \frac{a_1 p}{d_1^2} \left[ b_1 d_1 (t_i e^{-b_1 d_1 t_i} - t_{i-1} e^{-b_1 d_1 t_{i-1}}) + \left( e^{-b_1 d_1 t_i} - e^{-b_1 d_1 t_{i-1}} \right) \right] [y_i - y_{i-1}] \\
- \frac{a_1 p}{d_1^2} e^{-b_1 d_1 t_n} \left( b_1 d_1 t_{n+1} \right) - 1 = 0 \quad \text{... (2.18)}
\]

\[
\sum_{i=1}^{n} \frac{1 - p}{d_2} \left[ e^{-b_2 d_2 t_{i-1}^2} - e^{-b_2 d_2 t_i^2} \right] [y_i - y_{i-1}] \\
- \frac{(1 - p)}{d_2} \left( 1 - e^{-b_2 d_2 t_n^2} \right) = 0 \quad \text{... (2.19)}
\]

\[
\sum_{i=1}^{n} \frac{a_2 (1 - p)}{d_2^2} \left[ t_i^2 e^{-b_2 d_2 t_i^2} - d_2 t_{i-1}^2 e^{-b_2 d_2 t_{i-1}^2} \right] [y_i - y_{i-1}] \\
- a_2 (1 - p) \left[ t_n^2 e^{-b_2 d_2 t_n^2} \right] = 0 \quad \text{... (2.20)}
\]

\[
\sum_{i=1}^{n} \frac{a_2 (1 - p) \left[ b_2 d_2 \left( t_i^2 e^{-b_2 d_2 t_i^2} - t_{i-1}^2 e^{-b_2 d_2 t_{i-1}^2} \right) + \left( e^{-b_2 d_2 t_i^2} - e^{-b_2 d_2 t_{i-1}^2} \right) \right] [y_i - y_{i-1}] \\
- \frac{a_2 (1 - b)}{d_2^2} e^{-b_2 d_2 t_n^2} \left( b_2 d_2 t_n^2 + 1 \right) - 1 = 0 \quad \text{... (2.21)}
\]

where, \( D = \frac{a_1}{d_1} \left[ e^{-b_1 d_1 t_{i-1}} - e^{-b_1 d_1 t_i} \right] + \frac{a_2}{d_2} (1 - p) \left[ e^{-b_2 d_2 t_{i-1}} - e^{-b_2 d_2 t_i} \right] \).

The above equations (2.16)-(2.21) are difficult to be solved and are typical. However, to obtain the estimated values of the unknown parameters \( p, a_1, b_1, d_1, a_2, b_2, \) and \( d_2 \), we can employ a numerical technique (e.g. Newton-Raphson method) by using softwares namely MATLAB, MAPPLE, MATHEMATICA, etc..

2.6 Numerical Illustration

In this section, numerical results have been provided by developing a program in software MATLAB to calculate the reliability \( R(t) \) and maintenance cost \( EC(T) \).
The effects of parameters namely initial number of faults $a_1$, $a_2$, and fault detection rates $b_1$ and $b_2$ on reliability for both the cases have been examined in figs (2.2)-(2.4) for default parameters fixed as $a_1=200$, $a_2=600$, $d_1=.4$, $d_2=.6$, $b_1=.05$, $b_2=.07$, $x=.008$, $p=0.1$, $a=.1$, $C_0=15$, $C_t=20$, $C_w=10$, $T_w=10$.

Figs 2.2(i-iv)-2.4(i-iv) depict the variation of the total maintenance cost $EC(T)$ with the testing time $T$ for cases 1 and 2, respectively. Figs 2.4(i-iv) illustrate the pattern of the reliability of the software by varying parameters $a_1$, $b_1$, $a_2$ and $b_2$, respectively.

It can be inferred that $EC(T)$ first increases and then shows the decreasing trend with $T$ for all the parameters. Figs 2.2(i) and 2.2(ii) show that expected maintenance cost $EC(T)$ first increases and after some time it decreases and finally it becomes linear with respect to $T$ for case 1. We examine that $EC(T)$ remains constant for values $a_1$ and $b_1$. Fig. 2.2(iii) illustrates that $EC(T)$ increases as $a_2$ increases up to $T=10$; after that it becomes constant. Fig. 2.2(iv) shows the reverse pattern of fig. 2.2(iii). In this fig., $EC(T)$ decreases as $b_2$ increases.

From figs 2.3(i)-2.3(iv), we notice that for both the cases, expected maintenance cost $EC(T)$ decreases with the initial testing time ‘$T$’ and therefore shows the almost linear increments in the values of $EC(T)$. Also $EC(T)$ is higher for smaller number of errors $a_1$ and $a_2$ but on the other hand, it decreases with the failure detection rates $b_1$ and $b_2$.

In figs 2.4(i) and 2.4(ii), initially reliability remains slightly linearly decreasing and after some time (i.e. $T=5$) it increases and finally becomes constant for parameters $a_1$ and $b_1$. From fig. 2.4(iii), we notice that reliability decreases with the increase of $a_2$ up to $T=5$; after that it increases up to $T=10$ and then it becomes constant for higher values of $T$. Fig. 2.4(iv) depicts the reverse pattern as noted in fig. 2.4(iii) up to $T=10$. It is seen that the reliability increases with the increase in $b_2$. Over all we see from figs 2.4(i-iv), that for lower values of testing time $T$, reliability first decreases, then it increases for higher values of $T$ and finally the reliability reaches to the satisfactory level and becomes almost constant.
Chapter-2: Optimal Testing Time For Software Reliability ….

Over all we conclude that

- In case 1, the total expected maintenance cost calculated first increases for lower values of T, and then it goes on increasing for the further increment in T followed by the almost constant trend for the increased values of T. This depicts the fact that the expected maintenance cost is greatly affected by the testing time T.

- In case 2, we see that there is a steep decreasing trend in the expected maintenance cost for the lower values of T followed by a steep decrease. Again the expected maintenance cost shows the linear increment in expected cost for the further increased values of T.

2.7 Conclusion

In this chapter, we have proposed optimal release time policies to predict the optimal release time of the software in imperfect debugging environment along with the warranty period. The study will be helpful to a software vender to calculate the total software product cost by considering the warranty period. The optimal testing time of module-based software is determined by minimizing the total maintenance cost of the software subject to the reliability constraint. The software reliability growth model developed can be successfully employed for analyzing the real time software systems, which consist of two types of modules following different failure patterns. The first is that the warranty is provided to only keep the reliability level promised at the time of software release. The second is that the warranty is provided to increase the reliability level from that stated at the time of software release.

It can be concluded from the analysis of the maintenance cost of the software that more time is required for testing the software if the reliability constraint is considered along with the minimization of the maintenance cost of the software. Moreover, the optimal time required for testing is comparatively lesser if the software reliability growth is assumed to occur in the warranty period also.
Fig. 2.2: Expected maintenance cost vs T for different values of (i) $a_1$ (ii) $b_1$ (iii) $a_2$ (iv) $b_2$ for case 1.
Fig 2.3: Expected maintenance cost Vs T for different values of (i) $a_1$ (ii) $b_1$ (iii) $a_2$ (iv) $b_2$ for case 2.
Fig. 2.4: Software reliability Vs T for different values of (i) $a_1$ (ii) $b_1$ (iii) $a_2$ (iv) $b_2$. 

---

Chapter-2: Optimal Testing Time For Software Reliability 

---

63