Executive Summary of the Minor Research Project
On

“STATISTICAL ANALYSIS OF SOFTWARE MODELS FOR
MULTI-TYPES OF ERRORS”
(Vide Ref. No. MRP(S) 483/09-10KAGU024/UGC-SWRO Dated 27.01.2010)

Principal Investigator       Co-Investigator
Dr. M. R. Huggi              Dr. B. V. Dhandra

GODUTAI DODDAPPA APPA ARTS & COMMERCE COLLEGE FOR WOMEN GULBARGA

2012
INDEX

1. Introduction --- 1-5

   1.1 Software Reliability Growth Models
   1.2 Software Testing
   1.3 Problem Definition
   1.4 Motivation

2. Optimal Release Policy for Multi-Stage Superposed NHPP Model --- 6-30

   2.1 Introduction
   2.2 Assumption
   2.3 Notation
   2.4 Mean value functions of a NHPP for different stages of Error removal process
   2.5 Mean value functions and instantaneous failure rates for different stages of superposed NHPP
   2.6 Cost functions-I for Multi-Types of Errors
      2.6.1 Cost functions and optimal release policy for two-stage process
      2.6.2 Cost function an optimal release policy for 3-stage process
      2.6.3 Cost function and optimum release policy for n-stage process
   2.7 Cost function-II for Multi-Types of Errors
   2.8 Discussion and Further Scope

Acknowledgment --- 31
References --- 32-35
Publications --- 36
Executive Summary of the Project Work

“STATISTICAL ANALYSIS OF SOFTWARE MODELS FOR MULTI-TYPES OF ERRORS”

Chapter 1. Introduction

Computers are being widely used for variety of data processing to generate the desired information. The data processing is the most important component of any industry. The rapid advancement of technology made the cost of computer hardware to decrease steadily on the contrary cost of developing computer software is increasing. The development and growth in software production has expended and continues to expand at a rapid rate. Hence, the development of computer software is seen to be most prominent industry in 21st century. Developing software product is generally a quite complex and time consuming process. The software customers are more aware of the products and service available to them. The three most significant components of any customers’ expectations are quality of software, delivery time of software and cost of software. The quality of software can be described in terms of complexity, maintainability, portability, reliability etc. The software reliability quantifies the quality of software. It is defined as the probability that the software will work without causing any failure for a specified time under a given usage environment and is measured by observing the failure phenomenon. Hence, the software reliability is important to evaluate and predict the reliability and performance of software for quality assurance. Models applicable to the assessment of software reliability are called Software Reliability Growth Models (SRGMs). The SRGM is defined as the phenomenon that the number of errors remained in the software system decrease or the time internal between software failures become longer with progress of software testing. Hence, the software testing objective demands to model a software errors detection process in which the software errors remained in software system are detected and corrected by software testing process. Several SRGMs are available in the literature to monitor the error removal process.
Software development process consists of a sequence of activities, where perfection is yet to be achieved. Hence, there is every possibility that fault can be introduced and can remain in a software. A fault occurs whenever a person commits a logical or other mistake during software development activity. These faults can lead to failures with catastrophic results. Therefore, an emphasis is made on avoiding introduction of fault, during software development and to remove dormant fault before the product is released in the market for use. The testing phase is prominent one and nearly half of the development resources are spent during this phase. Hence, the study of software testing and the release policy for software are important components. The testing phase aims at detecting and removing faults which are introduced during software development thereby increasing the reliability of the software.

1.1 Software Reliability Growth Models

Models that describe the failure phenomenon and consequent enhancement in reliability due to fault removal are termed as software Reliability Growth Models (SRGMs). Jelinski and Moronda (JM) 1972 have proposed Non-Homogeneous Poisson Process (NHPP) based SRGM assuming that (i) the initial number of errors in a software is finite (ii) rectification of errors is perfect and (iii) inter-occurrence time is proportional to the number of errors remained in the software (Failure rate is constant). Moronda (1975) has obtained the release time of a software by relaxing the assumption of equal failure intensity proposed in the JM model. Goel and Okumoto (1979) Generalized the JM model by considering the random number of errors in the software and the error detection rate is proportional to the expected number of errors remained in the software, further it is assumed that the inter occurrence times are dependent.

Some models due to Goel and Okumoto (1979), Yamada et al. (1983), Hossain and Ram (1993) and many others assume that each time a failure occurs and errors causing failures are rectified with perfect rectification (perfect debugging). Later Yamada et al. (1992) Phom et al. (1999), Rattihalli and Zachariah (2002), Kapur et al. (2002), Phom and Zhany (2003) have developed the models with imperfect
rectification of errors. Many researchers have considered the test effort and learning factors in their cost models and some are due to Yamada and ohtera (1990) Yamada et al. (1986). Chatterje et al. (1997), Kapur and Bardhan (2001) Rattihalli and Zacharizh. (2002) etc. It has been observed that the relationship between the testing time and the corresponding number of faults removed is either exponential or S-Shaped or the mixture of the two. Many SRGMs like Bittanti et al (1988), Goel et al (1979), Kapur et al, (1999) Kapur and Garg (1992) and Ohba (1984) describe one of the said curves. Most of the SRGMs do not distinguish between software failure and isolation/removal processes. But in reality the actual removal of a fault is done after a failure is reported and corresponding fault is isolated. Hence, the fault removal is carried out in two phases. In the first phase the failure identification team (testing team) isolates a failure. In the second phase development team consisting of programmers rectifies the fault causing that failure. Goel and Okumoto (1979) developed SRGM by considering the time lag between failure and fault removal. Farther, Yamada (1985), kapur and Garg (1992), Kapur et al. (1995), kapur et al. (2000), Kapur et al. (2002) and many others have studied the models by categorizing the errors based on severity.

1.2 Software Testing

In order to detect and rectify the errors, the software test is performed. During the test phase, failures will be observed and errors will be corrected until such a time the programmer believes that the software is ready for use. In way, it is important to know when to release software for use to attract the customers and is the crucial aspect. This motivates to find the optimum release time of a software product.

The only way to verify and validate the software is by testing. The software testing involves running the software and checking for unexpected behavior of the software output. In the literature one can find various test approaches viz testing methods can be classified based on the way in which inputs are selected. Singpurwalla and Wilson (1994) call this as testing strategies. There are different types of partition testing as described by Weyuker and Jeng (1991) and Fronkal and Weyaker (1993).
1.3 Problem Definition

It is necessary that SRGMs should explicitly take into account the errors of different severity with respect to the stages of testing such a modeling approach is adopted by Kapur et al. (1995). A simple error removal process is termed as 1-stage process. For harder types of errors more test effort is required. Testing team personals have to spend more time to analyze the cause of the failure and more efforts are required to rectify the errors. Thus, analyzing the cause of the failure is the 1-stage and rectifying the errors is the 2-stage and such a process is called 2-stage process. Similarly in 3-Stage process, failure observation is 1-stage, fault isolation is 2-stage and fault removal is 3-stage. Thus, there may be n-stages error removal process in general. Here 2-stage and 3-stage superposed NHPP cost models for software product with a life time warranty and discount rates are developed. These cost models reduce to William et al. (2005) if all the errors are of same severity say simple. Further a general cost model for n-stage superposed NHPP is developed for n= 1, 2, 3 the cost model reduces to the 1-stage, 2-stage, 3-stage superposed NHPP cost models respectively. The optimum release time by minimizing the total expected cost is obtained for different cost functions.

1.4 Motivation

It is important part of the software development to know when to stop testing the software and release for use. If the testing is prolonged for longer duration. The software will achieve high reliability but the cost of the software will increase further, it may lose the market attraction due to change in technology, environment and languages. On the other hand, if the testing period is too short the cost may reduce, software can avail market opportunity but it may fail to achieve the consumers’ satisfaction with respect to the quality of software. In view of this, there is a trade off between delivery schedule and reliability of a software. In other words, it is important to determine the optimal length of software testing time subject to reliability and cost. Such an optimization need is called an optimal software release problem. This motivated us to study the optimal release policies.
In the literature there are many evaluation criteria to obtain the optimum release policies for a SRGMs based on NHPP. One is, when to release software so that the cost incurred during the life cycle of the software is minimized or the reliability is maximized.

Other approach is when to stop testing so that the failure intensity/reliability reaches a desired level irrespective of the cost incurred. Formon and Singapurwalla (1991) have studied the stopping rules for testing of the software. The optimal release policies can be classified into two groups (i) constant life cycle model chatterjee et al. (1997) and (ii) Random life cycle model Yun and Bai (1990). Thus, the optimal release policies are obtained by considering the following: (i) cost criteria (ii) reliability criteria (iii) cost and reliability criteria (iv) cost and reliability criteria under penalty cost (v) test effort (vi) discount rate and (vii) imperfect rectification of errors.
Chapter 2 Optimal Release Policy for Multi-stage Superposed NHPP Model

Here, we have proposed cost models based on SRGMs for 2-stage, 3-stage and n-stage superposed NHPP, which exhibits the exponential reliability growth. For these models cost function is developed and the optimal release polices are proposed by minimizing the total expected cost.

2.1 Introduction:

The quality of the product is important factor to attract the consumers. To bring quality software, the software is tested for its quality in terms of error detection and removal. As and when a failure is reported, effort is made to first isolate the cause of the failure and then to remove the errors involved in failure. The software quality in terms of reliability can be measured by observing the failure phenomenon. Many SRGMs have been developed in the literature to monitor the error removal process and predict the reliability of software. It has been observed that the relationship between the testing time and the number of errors removed is either exponential or S-shaped or the mixture of two. Here, the exponential growth model is considered. The number of errors detected gives an idea about the type of errors being detected and the nature of errors remained in the software. In practice, the large number of errors can be observed at the beginning of the software testing phase, while error detection and removal may become difficult at the later stages. The errors detected at the early stages are relatively easy and termed as simple errors and the errors that are detected at the later stages are relatively difficult and are termed as hard errors. A very few additional errors may be detected during last phase of the software testing and a decision is taken for termination of software testing. These errors are termed as complex errors. This leads to the classification of errors such as simple, hard and complex. Hence, testing effort can be streamlined to achieve the better error detection. An appropriate testing control can be initiated for a particular error type. It is assumed that testing is done uniformly. Simple, hard and complex faults manifest themselves at any time during software testing phase. It is necessary that SRGMs should explicitly take into account the errors of different severity with respect to the stages of testing, such a modeling approach is adopted by Kapur et al. (1995a) and Kapur et al. (1995b). Here, software models are considered by assuming that
they have different errors at various stages of testing. Cost functions denoting the total expected cost are constructed for these models and the optimal release time of software is determined with respect to the reliability levels.

Mean value functions and instantaneous failure rates for 2-stage, 3-stage and so on for n-stage process are obtained and then we focus on cost functions for multi types of errors and decision rule.

2.2 Assumption:

Following are the assumptions made in these models.

A1. Software system is subject to failure during execution caused by errors / faults.

A2. Failure rate of the software is equally affected by errors / faults remained in the software.

A3. The number of fault detected at any time is proportional to the number of errors / faults remained in the software.

A4. On a failure, repair effort start and error / fault causing failure is removed with certainty.

A5. All errors / faults are mutually independent from failure detection point of view.

A6. The proportion of failure detection / faults isolation / fault removal is constant.

A7. Corresponding to the error detection / removal phenomenon at the practitioners / user end, there exist an equivalent error detection / error removal at the user / practitioners end.

A8. The error detection / removal phenomenon modeled by NHPP.

2.3 Notation:

\( m(t) \) : Representing mean value function of a NHPP

\( a_i \) : Initial number of type i errors.

\( b_i \) : Error removal rate per remaining error of type i in a software.

\( m_{if}(t) \) : Expected number of failure in \((0, t]\) due to type i errors.

\( m_{II}(t) \) : Expected number of type i errors identified in \((0, t]\).

\( m_{ir}(t) \) : Expected number of type i errors removed in \((0, t]\).
C(t) : Total expected software testing cost, which includes cost of testing and removal of errors.

2.4 Mean value functions of a NHPP for different stages of error removal process

In this section, mean value functions for 2-stage, 3-stage and n-stage processes are obtained, based on errors of different severity. Kapur and Garg (1992) have proposed a software reliability growth model for an error removal phenomenon, by assuming that the software contains three types of errors / faults simple, hard and complex, same can be extended to n-types of faults. A simple error removal process is modeled as NHPP and termed as 1-stage process. By using A3, its intensity rate is given by

$$\frac{d}{dt}(m_{1r}(t)) = b_1 (a_1 - m_{1r}(t)) \tag{2.4.1}$$

solving the above differential equation by using the boundary condition $m_{1r}(0) = 0$, we get

$$m_{1r}(t) = a_1 \left(1 - e^{-b_1 t}\right). \tag{2.4.2}$$

For harder types of errors more testing effort is required. Testing team personnel have to spend more time to analyze the cause of the failure and hence they need more effort to remove them. It is further assumed that the number of faults isolated at any time is proportional to the current number of fault not isolated. Failure rate and isolation rate per error assumed to be same and equal to $b_2$. The error removal process for such faults is modeled as 2-stage process, and their intensity rates are given by

$$\frac{d}{dt}(m_{2f}(t)) = b_2 (a_2 - m_{2f}(t)) \tag{2.4.3}$$

$$\frac{d}{dt}(m_{2r}(t)) = b_2 (m_{2f}(t) - m_{2r}(t)). \tag{2.4.4}$$
The first stage is given by Equation (2.4.3) and describes the failure observation and fault detection phase. The second stage is given by (2.4.4) and it describes the fault removal phase.

The Equation (2.4.3) can be written as

\[- \frac{1}{(a_2 - m_{2f}(t))} \frac{d}{dt}(m_{2f}(t)) = -b_2.\]  

(2.4.5)

By integrating Equation (2.4.5) both sides with respect to \(t\), yields

\[\log (a_2 - m_{2f}(t)) = -b_2 \ t + K_1, \]  

Where \(K_1\) is a constant and is found by using boundary condition \(m_{2f}(0) = 0\). It gives

\[K_1 = \log a_2.\]

Hence,

\[a_2 - m_{2f}(t) = e^{-b_2 t} + \log a_2.\]

That is

\[m_{2f}(t) = a_2 \left(1 - e^{-b_2 t}\right).\]  

(2.4.6)

Now solve the differential Equation (2.4.4) by substituting the Equation (2.4.6) in (2.4.4) to obtain,

\[\frac{d}{dt}(m_{2r}(t)) = b_2 \left( a_2 \left(1 - e^{-b_2 t}\right) - m_{2r}(t)\right).\]

Re-arranging the terms and multiplying both sides of the above equation by \(e^{b_2 t}\), yields

\[\frac{d}{dt} \left( m_{2r}(t) e^{b_2 t} \right) = a_2 \ b_2 \ \left(e^{b_2 t} - 1\right).\]

By integrating above equation with respect to \(t\), to get

\[m_{2r}(t) e^{b_2 t} = a_2 \ e^{b_2 t} - a_2 \ b_2 \ t + K_2.\]  

Where \(K_2\) is a constant and determined by using boundary condition \(m_{2r}(0) = 0\). It gives
\[ K_2 = a_2. \]

Thus,

\[ m_{2r}(t) = a_2 \left( 1 - (1 + b_2 t) e^{-b_2 t} \right). \]  \( (2.4.7) \)

The complex error removal process can be modeled as a 3-stage process and is described below

\[
\frac{d}{dt} (m_{3f}(t)) = b_3 \left( a_3 - m_{3f}(t) \right)  
\]  \( (2.4.8) \)

\[
\frac{d}{dt} (m_{3I}(t)) = b_3 \left( m_{3f}(t) - m_{3I}(t) \right).  
\]  \( (2.4.9) \)

\[
\frac{d}{dt} (m_{3r}(t)) = b_3 \left( m_{3I}(t) - m_{3r}(t) \right).  
\]  \( (2.4.10) \)

The first-stage is given by Equation (2.4.8), which describes the failure observation and fault detection. Note that the complex faults need more efforts for their removal, the removal phase is modeled as a two-stage process, one for isolation and second for removal and they are given by Equations (2.4.9) and (2.4.10) respectively.

Solving Equation (2.4.8) by using boundary conditions, \( m_{3f}(0) = 0 \), yields

\[ m_{3f}(t) = a_3 \left( 1 - e^{-b_3 t} \right). \]  \( (2.4.11) \)

Solving Equation (2.4.9), subject to the boundary condition \( m_{3I}(0) = 0 \) and equation (2.4.11), one gets

\[ m_{3I}(t) = a_3 \left( 1 - (1 + b_3 t) e^{-b_3 t} \right). \]  \( (2.4.12) \)

Substituting the Equation (2.4.12) in Equation (2.4.10), yields,

\[
\frac{d}{dt} (m_{3r}(t)) = b_3 \left[ a_3 \left( 1 - (1 + b_3 t) e^{-b_3 t} \right) - m_{3r}(t) \right].
\]
That is,
\[
\frac{d}{dt} (m_{3r}(t)) e^{b_3t} + b_3 m_{3r}(t) e^{b_3t} = a_3 b_3 \left( e^{b_3t} - (1+b_3t) \right).
\]

Hence,
\[
\frac{d}{dt} \left( m_{3r}(t)e^{b_3t} \right) = a_3 b_3 \left( e^{b_3t} - (1+b_3t) \right).
\]

Integrating the above equation w.r.t both sides to obtain,
\[
m_{3r}(t) e^{b_3t} = a_3 e^{b_3t} - a_3 b_3 t - a_3 b_3^2 \frac{t^2}{2} + K_3,
\]
where \(K_3\) is a constant and is found subject to the boundary condition \(m_{3r}(0) = 0\), that is,
\[
K_3 = a_3.
\]

Thus,
\[
m_{3r}(t) = a_3 \left[ 1 - \left( 1 + b_3 t + b_3^2 \frac{t^2}{2} \right) e^{-b_3t} \right]. \tag{2.4.13}
\]

In the following section mean value functions and instantaneous failure functions are obtained for superposed NHPP.

### 2.5 Mean value functions and instantaneous failure rates for different stages of superposed NHPP

If a software contains two types of faults in the proportion \(p_1\) and \(p_2\) respectively, where \(p_1 + p_2 = 1\), then the mean value function of the superposed NHPP is
\[
m(t) = p_1 m_{1r}(t) + p_2 m_{2r}(t).
\]

Substituting the expressions for \(m_{1r}(t)\) and \(m_{2r}(t)\) from (2.4.2) and (2.4.7) in the above equation, we get
\[
m(t) = p_1 a_1 \left( 1 - e^{-b_1t} \right) + p_2 a_2 \left( 1 - (l + b_2t)e^{-b_2t} \right).
\]
The instantaneous failure intensity rate at time \( t \), for 2-stage superposed NHPP is

\[
\lambda(t) = m'(t) = p_1 a_1 b_1 e^{-b_1 t} + p_2 a_2 b_2^2 t e^{-b_2 t}.
\]  

(2.5.1)

If we consider a software product consisting of all three types of faults in the proportion \( p_1, p_2 \) and \( p_3 \) respectively, where \( p_1 + p_2 + p_3 = 1 \). Then the mean value function of superposed NHPP for 3-stage process is given by

\[
m(t) = p_1 m_{1f}(t) + p_2 m_{2f}(t) + p_3 m_{3f}(t)
\]

That is

\[
m(t) = p_1 a_1 \left(1 - e^{-b_1 t}\right) + p_2 a_2 \left(1 - (1 + b_2 t) e^{-b_2 t}\right) + p_3 a_3 \left(1 - \left(1 + b_3 t + b_3^2 \frac{t^2}{2}\right) e^{-b_3 t}\right).
\]  

(2.5.2)

The instantaneous failure rate at time \( t \) for 3-stage superposed NHPP is given by

\[
\lambda(t) = m'(t) = p_1 a_1 b_1 e^{-b_1 t} + p_2 a_2 b_2^2 t e^{-b_2 t} + p_3 a_3 b_3^3 \frac{t^2}{2} e^{-b_3 t}.
\]  

(2.5.3)

The removal rate per fault for simple fault is a constant \( b_1 \), whereas for hard and complex faults are given by

\[
R_2(t) = \frac{d}{dt} \left( \frac{m_{2f}(t)}{a_2 - m_{2f}(t)} \right) = \frac{b_2^2 t}{(1 + b_2 t)}.
\]

\[
R_3(t) = \frac{d}{dt} \left( \frac{m_{3f}(t)}{a_3 - m_{3f}(t)} \right) = \frac{b_3^3 \frac{t^2}{2}}{2 \left(1 + b_3 t + b_3^2 \frac{t^2}{2}\right)}.
\]
By observing the process for a longer duration the fault removal rates for hard and complex faults become constant. That is, if \( t \to \infty \), then \( R_2(t) \to b_2 \) and \( R_3(t) \to b_3 \). Thus, for large value of \( t \), Equations (2.4.7) and (2.4.13) behave similar to exponential growth curve. Hence without loss of generality one can assume that the rates \( b_2 \) and \( b_3 \) to be equal to \( b_1 \), (i.e., \( b_3 = b_2 = b_1 = b \) (say)). The Equations (2.5.1) and (2.5.3) reduce to

\[ \lambda(t) = be^{-bt} \left( p_1 a_1 + p_2 a_2 bt \right). \]  

(2.5.4)

and

\[ \lambda(t) = be^{-bt} \left( p_1 a_1 + p_2 a_2 bt + p_3 a_3 b^2 \frac{t^2}{2} \right). \]  

(2.5.5)

Fault removal rates per fault for three types of faults are \( b, \frac{b^2 t}{1+bt} \) and \( \frac{b^3 t^2}{2 \left( 1+bt+b^2 \frac{t^2}{2} \right)} \) respectively. Also one can note that

\[ b > \frac{b^2 t}{1+bt} > \frac{b^3 t^2}{2 \left( 1+bt+b^2 \frac{t^2}{2} \right)}, \]

which is in accordance with the severity of a fault defined earlier.

In general, if software consists of \( n \)-types of faults in the proportion \( p_1, p_2, \ldots, p_n \), where \( p_1 + p_2 + \ldots + p_n = 1 \), then the mean value function of superposed NHPP can be written as

\[ m(t) = \sum_{i=1}^{n} a_i p_i \left( 1-e^{-bt} \sum_{j=0}^{i-1} \frac{(bt)^j}{j!} \right). \]  

(2.5.6)

The instantaneous failure intensity rate is given by
\[ \lambda(t) = m'(t) = \sum_{i=1}^{n} a_i p_i e^{-bt} \sum_{j=0}^{i-1} \frac{b^j j^{j-1}}{j!} \]

\[ + \sum_{i=1}^{n} a_i p_i b e^{-bt} \sum_{j=0}^{i-1} \frac{(bt)^j}{j!}. \]

That is,

\[ \lambda(t) = -\sum_{i=1}^{n} a_i p_i e^{-bt} \left[ \sum_{j=1}^{i-1} \frac{b^j j^{j-1}}{(j-1)!} + b \sum_{j=0}^{i-1} \frac{(bt)^j}{j!} \right]. \]

Therefore

\[ \lambda(t) = -\sum_{i=1}^{n} a_i p_i e^{-bt} \left[ \frac{b + b^2 t}{1!} + \frac{b^3 t^2}{2!} + \ldots + \frac{b^{i-1} t^{i-2}}{(i-2)!} \right. \]

\[ - b - \frac{b^2 t}{1!} - \frac{b^3 t^2}{2!} - \ldots - \frac{b^i t^{i-1}}{(i-1)!} \].

Hence,

\[ \lambda(t) = \sum_{i=1}^{n} a_i p_i e^{-bt} \frac{b^i t^{i-1}}{(i-1)!}. \quad (2.5.7) \]

One can easily verify that for \( n = 1 \), the Equation (2.5.7) reduces to failure intensity rate for 1-stage process, for \( n = 2 \) and \( n = 3 \), we get 2-stage and 3-stage instantaneous failure rates for superposed NHPP. In the following, expected total cost functions for multi-types of faults are obtained.

**2.6 Cost function-I for multi-types of errors**

In this section, the life-time warranty cost model of a software product with discount rate is considered and a cost function is derived for the multi-types of faults. This cost model ensures the errors of different severity through different instantaneous failure intensity functions \( \lambda(t) \).
Here factors like test effort and imperfect rectification of errors are not taken into account to get failure intensity rate $\lambda(t)$. The cost functions are developed for different stages like 2-stage, 3-stage and n-stage superposed NHPP and the optimal release time is obtained for different cases. The total expected cost of a software product is given by

$$C(T) = C_0 + C_t \int_0^T e^{-\alpha t} dt + C_w(T), \quad (2.6.1)$$

Where $C_0$ is initial testing cost which is the barest minimum requirement, $C_t$ is the testing cost per unit time and $C_w(T)$ is the maintenance cost during warranty period. Again we consider two cases (i) the software reliability growth does not occur after the testing phase, and (ii) the software reliability growth does occur after testing phase. That is,

$$C_w(T) = \begin{cases} 
C_w \int_0^\infty \lambda(T)e^{-\alpha t} dt, & \text{when software reliability is constant after testing phase} \\
C_w \int_0^T \lambda(t)e^{-\alpha t} dt, & \text{when software reliability changes after testing phase.}
\end{cases}$$

**2.6.1 Cost function and optimal release policy for two-stage process**

In the following, we obtain the total expected cost for 2-stage process when software reliability growth does change and does not change after testing phase. The decision rule for optimal release of a software product is presented.

**Case (i):** In this case we have

$$C_w(T) = C_w \int_0^T \lambda(t)e^{-\alpha t} dt,$$
Where \( \lambda(T) \) is the instantaneous failure intensity rate at time \( T \) for 2-stage superposed NHPP and it is given by equation (2.5.4). The total expected software product cost reduces to

\[
C(T) = C_0 + C_t \int_0^T e^{-\alpha t} dt + C_w \int_T^\infty b e^{-b T} \left[ p_1 a_1 + p_2 a_2 b T \right] e^{-\alpha t} dt.
\]

That is,

\[
C(T) = C_0 + C_t \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_w b e^{-b T} \left[ p_1 a_1 + p_2 a_2 b T \right] e^{-\alpha T} dt.
\]

Thus,

\[
C(T) = C_0 + C_t \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + \frac{C_w}{\alpha} b e^{-\alpha + b T} \left[ p_1 a_1 + p_2 a_2 b T \right]. \tag{2.6.2}
\]

Differentiating above equation w.r.to \( T \) we get

\[
C'(T) = C_t e^{-\alpha T} + \frac{C_w}{\alpha} p_2 a_2 b^2 e^{-\alpha + b T} - \frac{(\alpha + b) C_w b}{\alpha} e^{-\alpha + b T} \left[ p_1 a_1 + p_2 a_2 b T \right].
\]

\( C'(T) = 0 \), Implies that,

\[
\alpha C_t = b C_w e^{-b T} \left[ p_1 a_1 (\alpha + b) + p_2 a_2 b \{ (\alpha + b) T - 1 \} \right]. \tag{2.6.3}
\]

Where \( C'(T) \) is differentiation of \( C(T) \) with respect to \( T \).

If \( p_1 = 1 \) and \( p_2 = 0 \), then solution obtained by Equation (2.6.3) is same as Williams et al. (2005). The optimal release time \( T \) for software can be obtained by using bisection method. For this a general program in C is developed. The sample outputs of the program are presented in Table-1.
The optimum release policy for the said model can be stated as

$$T^* = \begin{cases} 
T^\circ, & \text{when } \lambda(0) > \lambda(T^\circ) \\
0, & \text{when } \lambda(0) \leq \lambda(T^\circ) 
\end{cases}$$

The following numerical values are considered for the parameters of the 2-stage superposed NHPP to study the behavior of the optimum release policies.

$$a_1 = 300, \quad a_2 = 400 \quad b = 0.05, \quad \alpha = 0.05, \quad p_1 = 0.25 \quad p_2 = 0.75$$

The optimal release times are then calculated for different values of $C_t$ and $C_w$. The tabulated values are shown in Table 1.

**Table 1:**
Optimum Release Times for 2-stage superposed NHPP for Case (i)

<table>
<thead>
<tr>
<th>$C_w$</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_t$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4.7422</td>
<td>4.8203</td>
<td>4.8672</td>
<td>4.8984</td>
<td>4.9141</td>
<td>4.9297</td>
</tr>
<tr>
<td>12</td>
<td>4.6797</td>
<td>4.7891</td>
<td>4.8359</td>
<td>4.8672</td>
<td>4.8984</td>
<td>4.9141</td>
</tr>
<tr>
<td>14</td>
<td>4.6328</td>
<td>4.7578</td>
<td>4.8203</td>
<td>4.8516</td>
<td>4.5828</td>
<td>4.9140</td>
</tr>
<tr>
<td>15</td>
<td>4.6016</td>
<td>4.7422</td>
<td>4.8047</td>
<td>4.8359</td>
<td>4.8672</td>
<td>4.8984</td>
</tr>
<tr>
<td>18</td>
<td>4.5234</td>
<td>4.6797</td>
<td>4.7578</td>
<td>4.8047</td>
<td>4.8359</td>
<td>4.8828</td>
</tr>
<tr>
<td>20</td>
<td>4.4609</td>
<td>4.6484</td>
<td>4.7422</td>
<td>4.7891</td>
<td>4.8203</td>
<td>4.8672</td>
</tr>
</tbody>
</table>

The behavior of optimum release time $T$ is reasonable in this case, because the heavy warranty cost forces the practitioners to rectify as many errors as possible by prolonging the test period. Whereas if testing cost/unit increases the practitioners have to release the software as early as possible to reduce the total cost of the software, the same can be observed in the Table 1.
Case (ii)

Here, \( C_w(T) = \int_{T}^{\infty} \lambda(t)e^{-\alpha t} \, dt \), where the failure intensity rate \( \lambda(t) \) at time \( t \), for 2-stage superposed NHPP is given by (2.5.4). Hence the total expected software cost reduces to,

\[
C(T) = C_0 + C_t \int_{0}^{T} e^{-\alpha t} \, dt + C_w \int_{T}^{\infty} e^{-bt} (p_1 a_1 + p_2 a_2 b t) e^{-\alpha t} \, dt.
\]

That is,

\[
C(T) = C_0 + C_t \left[ \frac{1 - e^{-\alpha T}}{\alpha} \right] + C_w p_1 a_1 b \int_{T}^{\infty} e^{-(\alpha+b)t} \, dt
\]

\[+ C_w p_2 a_2 b^2 \int_{T}^{\infty} t e^{-(\alpha+b)t} \, dt.\]

Hence,

\[
C(T) = C_0 + C_t \left[ \frac{1 - e^{-\alpha T}}{\alpha} \right] + C_w p_1 a_1 b \frac{e^{-(\alpha+b)}}{(\alpha+b)} \]

\[+ C_w p_2 a_2 b^2 \frac{1}{(\alpha+b)} \frac{1}{(\alpha+b)} \left[ T + \frac{1}{(\alpha+b)} \right]. \quad (2.6.4)
\]

Differentiating above equation with respect to \( T \), we get

\[
C'(T) = C_t e^{-\alpha T} - C_w b e^{-(\alpha+b)T} \left[ p_1 a_1 + p_2 a_2 b T \right]
\]

\[C'(T) = 0, \text{ implies that}
\]

\[
C_t = C_w b e^{-bT} \left[ p_1 a_1 + p_2 a_2 b T \right]. \quad (2.6.5)
\]

If \( p_1 = 1 \) and \( p_2 = 0 \), then solution of Equation (2.6.5) reduces to William et al. (2005) and it is called as one-stage process. The optimal release time
T° for software can be obtained by executing the program in C.

The optimum release policy for the said model can be stated as

\[ T^* = \begin{cases} 
T^°, & \text{when } \lambda(0) > \lambda(T^°) \\
0, & \text{when } \lambda(0) \leq \lambda(T^°) 
\end{cases} \]

2.6.2 Cost function and optimal release policy for 3-stage process

For the Case (i), the total expected software product cost is given by

\[ C(T) = C_0 + C_t \int_0^T e^{-\alpha t} dt + C_w \int_0^\infty \lambda(t) e^{-bt} dt, \]

where the intensity function \( \lambda(T) \) is given by (2.5.5).

\[ C(T) = C_0 + C_t \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_w b e^{-bT} \left[ p_1 a_1 + p_2 a_2 bT + p_3 a_3 b^2 \frac{T^2}{2} \right] \int_0^\infty e^{-at} dt. \]

Differentiating above equation with respect to \( T \) to obtain

\[ C'(T) = C_t e^{-\alpha T} + C_w \frac{b}{\alpha} \left\{ -p_1 a_1 (\alpha + b)e^{-(\alpha + b)T} + p_2 a_2 b e^{-(\alpha + b)T} - p_2 a_2 b(\alpha + b)T e^{-(\alpha + b)T} + p_3 a_3 b^2 T e^{-(\alpha + b)T} - p_3 a_3 b^2 \frac{T^2}{2} (\alpha + b)e^{-(\alpha + b)T} \right\}. \]

If \( C'(T) = 0 \) then, we get

\[ \alpha C_t = b C_w e^{-bT} \left[ p_1 a_1 (\alpha + b) + p_2 a_2 b \{(\alpha + b)T - 1\} \right]. \]
If \( p_3 = 0 \), then Equation (2.6.6) reduces to one obtained for 2-stage process and if \( p_1 = 1, p_2 = 0 \) and \( p_3 = 0 \), then solution obtained by above equation is same as William’s model. The optimum release time \( T^* \) for software can be obtained on the lines obtained in earlier cases. Thus one can state the optimum release policy as given in 2-stage process.

**Case (ii):**

The total expected software product cost when the software reliability growth does changes after the testing phase is given by

\[
C(T) = C_0 + C_t \int_0^T e^{-\alpha t} \, dt + C_w \int_0^\infty \lambda(t) e^{-\alpha t} \, dt ,
\]

Where \( \lambda(t) \) is failure intensity rate at time \( t \) and it is given by Equation (2.5.5). Thus, we have

\[
C(T) = C_0 + C_t \int_0^T e^{-\alpha t} \, dt + C_w \int_0^\infty \lambda(t) e^{-\alpha t} \, dt \left( p_1 a_1 + p_2 a_2 b + p_3 a_3 b^2 \frac{t^2}{2} \right) e^{-\alpha t} \, dt
\]

That is,

\[
C(T) = C_0 + C_t \left[ \frac{1 - e^{-\alpha T}}{\alpha} \right] + C_w p_1 a_1 b \int_0^T e^{-(\alpha+b)t} \, dt
\]

\[
+ C_w p_2 a_2 b^2 \int_0^T t e^{-(\alpha+b)t} \, dt + C_w p_3 a_3 b^3 \int_0^T t^2 e^{-(\alpha+b)t} \, dt.
\]

Thus,
\[ C(T) = C_0 + C_T \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_W p_1 a_1 b e^{-(\alpha+b)T} + C_W p_2 a_2 b^2 \left[ \frac{T+1}{\alpha+b} \right] \]
\[ + C_W p_3 a_3 b^3 e^{-(\alpha+b)T} \left[ \frac{T^2}{2} + \frac{T}{\alpha+b} + \frac{1}{(\alpha+b)^2} \right]. \]

Differentiating above equation w.r.to \( T \), we get

\[ C'(T) = C_T e^{-\alpha T} - C_W p_1 a_1 b e^{-(\alpha+b)T} - C_W p_2 a_2 b^2 e^{-(\alpha+b)T} \left[ \frac{T+1}{\alpha+b} \right] \]
\[ + C_W p_2 a_2 b^2 e^{-(\alpha+b)T} \left[ \frac{T+1}{\alpha+b} \right] \]
\[ - C_W p_3 a_3 b^3 e^{-(\alpha+b)T} \left[ \frac{T^2}{2} + \frac{T}{\alpha+b} + \frac{1}{(\alpha+b)^2} \right]. \]

If \( C'(T) = 0 \), then we have

\[ C_T = C_W e^{-bT} b \left[ p_1 a_1 + p_2 a_2 bT + p_3 a_3 b^2 \frac{T^2}{2} \right]. \tag{2.6.7} \]

If \( p_3 = 0 \), then \( C'(T) = 0 \), reduces to that of 2-stage process of case (ii). If \( p_1 = 1, p_2 = 0 \) and \( p_3 = 0 \), then it is same as William et al. (2005). The optimum release time \( T^* \) for software can be obtained on the lines discussed in earlier cases. Thus one can also state the optimum release policy.

### 2.6.3 Cost function and optimum release policy for n-stage process

**Case (i):**

The total expected software product cost under the assumption that the software reliability does not change after the testing phase is given by

\[ C(T) = C_0 + C_T \int_0^T e^{-\alpha t} dt + C_W \int_T^\infty \lambda(T)e^{-\alpha t} dt, \]
where \( \lambda(T) \) is failure intensity rate at time \( T \) and it is given by Equation (2.5.7). Thus substituting the expression for \( \lambda(T) \), we get

\[
C(T) = C_0 + C_t \int_0^T e^{-\alpha t} \, dt + C_w \sum_{i=1}^{\infty} \frac{n \sum a_i p_i b^i}{(i-1)!} e^{-bT} e^{-\alpha t} \, dt.
\]

That is,

\[
C(T) = C_0 + C_t \left[ \frac{1 - e^{-\alpha T}}{\alpha} \right] + C_w \sum_{i=1}^{n} a_i p_i b^i \frac{T^{i-1}}{(i-1)!} e^{-(\alpha+b)T}.
\]

Differentiating above Equation with respect to \( T \), we obtain,

\[
C'(T) = C_t e^{-\alpha T} + \frac{C_w}{\alpha} \sum_{i=1}^{n} a_i p_i b^i \frac{T^{i-2}}{(i-2)!} e^{-(\alpha+b)T} - \frac{T^{i-1}}{(i-1)!} (\alpha+b) e^{-(\alpha+b)T}.
\]

Thus,

\[
C'(T) = C_t e^{-\alpha T} + \frac{C_w}{\alpha} \sum_{i=1}^{n} a_i p_i b^i e^{-(\alpha+b)T} \frac{T^{i-1}}{(i-2)!} \left[ \frac{1}{T} - \frac{(\alpha+b)}{(i-1)!} \right].
\]

\[
C'(T) = 0, \quad \text{implies that,}
\]

\[
\alpha C_t = C_w e^{-bT} \sum_{i=1}^{n} a_i p_i b^i \frac{T^{i-1}}{(i-2)!} \left[ \frac{\alpha+b}{(i-1)!} - \frac{1}{T} \right].
\]

If \( n = 1 \) and \( p_1 = 1 \), then it reduces to William et al. (2005). If \( n = 2 \) and \( n = 3 \), then equation (2.6.9) reduces to (2.6.3) and (2.6.6) respectively for 2-stage and 3-stage superposed NHPP. The optimum release time \( T^* \) and optimum release policy can be obtained on the same lines as carried out in the 2-stage process.

**Case (ii):**

The total expected software product cost under the assumption that the software reliability changes after the testing phase is
\[ C(T) = C_0 + C_t \int_0^T e^{-\alpha t} \, dt + C_w \sum_{i=1}^\infty \frac{a_i p_i b^i}{(i-1)!} e^{-bt} e^{-\alpha t} \, dt. \]

That is,
\[ C(T) = C_0 + C_t \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_w \sum_{i=1}^\infty \frac{a_i p_i b^i}{(i-1)!} \int_0^T e^{-(\alpha+b)t} \, dt. \quad (2.6.10) \]

Let \[ I = \int_0^T t^{i-1} e^{-(\alpha+b)t} \, dt. \quad (2.6.11) \]

By using the result
\[ \int u(x) v(x) \, dx = u v_1 - u' v_2 + u'' v_3 \ldots. \]

Where \( v_1 = \int v(x) \, dx, \ v_2 = \int v(x) \, dx \) and so on
\[ u' = \frac{du(x)}{dx}, \ u'' = \frac{d^2u(x)}{dx^2} \] and so on

With \( u(x) = t^{i-1} \) and \( v(x) = e^{-(\alpha+b)t} \), Equation (2.6.11) can be simplified as follows
\[ I = \int_0^T t^{i-1} e^{-(\alpha+b)t} \, dt = \left[ t^{i-1} e^{-(\alpha+b)t} \frac{t^{i-2} e^{-(\alpha+b)t}}{(\alpha+b)^2} \right]_{t=0}^{t=T} + \frac{(i-1)(i-2)t^{i-3} e^{-(\alpha+b)t}}{-(\alpha+b)^3} \ldots + \ldots + (-1)^{i-1} \frac{t^{i-1}}{(-1)^i (\alpha+b)^i}. \]

That is,
\[ I = \left[ -e^{-(\alpha+b)t} \left\{ \frac{t^{i-1}}{(\alpha+b)} + \frac{(i-1)t^{i-2}}{(\alpha+b)^2} + \ldots + \frac{(i-1)! t^0}{(\alpha+b)^i} \right\} \right]_0^T. \quad (2.6.12) \]
Therefore,
\[ I = e^{-(\alpha+b)T} \left[ \frac{T^{i-1}}{(\alpha+b)} + \frac{(i-1)T^{i-2}}{(\alpha+b)^2} + \cdots + \frac{(i-1)!}{(\alpha+b)^i} \right]. \]

Thus,
\[ \frac{I}{(i-1)!} = e^{-(\alpha+b)T} \left[ \frac{1}{(\alpha+b)^i} + \frac{T}{i!(\alpha+b)^{i-1}} + \frac{T^2}{2!(\alpha+b)^{i-2}} + \cdots + \frac{T^{i-1}}{(i-1)!(\alpha+b)} \right]. \]

Hence
\[ \frac{I}{(i-1)!} = e^{-(\alpha+b)T} \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha+b)^{i-j}}. \] (2.6.13)

By Equations (2.6.10) and (2.6.13), we get
\[ C(T) = C_0 + C_t \left[ 1 - e^{-\alpha T} \right] + C_w \sum_{i=1}^{n} a_i p_i b_i e^{-(\alpha+b)T} \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha+b)^{i-j}}. \] (2.6.14)

Differentiating above equation w.r. to \( T \), we have
\[ C'(T) = C_t e^{-\alpha T} + C_w e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b_i \sum_{j=0}^{i-1} \frac{jT^{j-1}}{j!(\alpha+b)^{i-j}} \]
\[ - C_w (\alpha+b)e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b_i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha+b)^{i-j}}. \]

That is,
\[ C'(T) = C_t e^{-\alpha T} + C_w e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b_i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha+b)^{i-j}} \left\{ \frac{j}{T} - (\alpha+b) \right\}. \]

For \( C'(T) = 0 \), we have
\[ C_t = C_w e^{-bT} \sum_{i=1}^{n} a_i p_i b_i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha+b)^{i-j}} \left\{ (\alpha+b) - \frac{j}{T} \right\}. \] (2.6.15)
If \( n = 1 \) then solution to the above equation reduces to William et al. (2005). For \( n = 2 \) and \( n = 3 \) the Equation (2.6.15) reduces to (2.6.5) and (2.6.7) respectively which are nothing but first derivatives of cost function for 2-stage and 3-stage superposed NHPP. The optimum release time \( T^o \) and optimum release policy can be obtained on the lines discussed earlier.

### 2.7 Cost function-II for multi-types of errors

In this section, we consider a cost function as stated below

\[
C(T) = C_1 \int_0^T e^{-\alpha t} \, dt + C_2 \int_0^T \lambda(t)e^{-\alpha t} \, dt + C_w(T) \tag{2.7.1}
\]

Where \( T \) is the release time of a software product, \( C_1 \) is the testing cost per unit time. Hence, \( C_1 \int_0^T e^{-\alpha t} \, dt \) represents the total testing cost with discount rate \( \alpha \), \( C_2 \) is the rectification cost of an error during testing phase. Thus, \( C_2 \int_0^T \lambda(t)e^{-\alpha t} \, dt \) represents the total rectification cost during testing phase, \( C_w(T) \) is the total warranty cost during operational phase. The warranty cost during operational phase depends on two factors: (i) the software reliability growth changes during warranty period and (ii) the software reliability growth does not change during warranty period and its diagrammatic presentation is shown in the following Figure 1.
Here \( C_w(T) \) is the total warranty cost, and life-cycle of the software is \( t_c \).

Now, we consider \( t_c \) to be infinite i.e., \( t_c \rightarrow \infty \) (i.e., life-time warranty cost). The instantaneous failure intensity rate is obtained for n-stage superposed NHPP, without considering the factors like test effort and imperfect rectification of errors when reliability growth occurs and does not occur during maintenance period. Hence optimum release time is obtained subject to the reliability condition during warranty period. From the general case one can easily obtain the optimum release time for 2-stage and 3-stage superposed NHPP.

**Case (i):**

Here, the total expected software product cost, when reliability does not change during operational and warranty periods is given by

\[
C(T) = C_1 \int_0^T e^{-\alpha t} \, dt + C_2 \int_0^T (T) e^{-\alpha t} \, dt + C_3 \int_T^\infty \lambda(T) e^{-\alpha t} \, dt \tag{2.7.2}
\]

Substituting the expression for \( \lambda(T) \) from the equation (2.5.7), we get

---

**Figure 1**

Diagram showing the relationship between reliability growth, constant failure intensity, and warranty periods. The graph illustrates the change in failure intensity rate over time, with notable segments labeled for reliability growth and constant phases. The diagram includes arrows indicating the transition from warranty period to life-cycles, with specific times marked for clarity.
\[ C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_2 \int_0^T \sum_{i=1}^n a_i p_i b^i T^{i-1} e^{-bT} e^{-\alpha t} \, dt \]
\[ + C_3 \int_{T=1}^\infty \sum_{i=1}^n a_i p_i b^i T^{i-1} e^{-bT} e^{-\alpha t} \, dt. \]

That is,
\[ C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_2 \sum_{i=1}^n a_i p_i b^i T^{i-1} e^{-bT} \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] \]
\[ + C_3 \sum_{i=1}^n a_i p_i b^i T^{i-1} (i-1)! e^{-bT} e^{-\alpha T} \alpha. \]

Thus,
\[ C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + \frac{e^{-bT}}{\alpha} \left[ C_2 - e^{-\alpha T} (C_2 - C_3) \right] \sum_{i=1}^n a_i p_i b^i T^{i-1} \frac{(i-1)!}{(i-1)!} \]
\[ \text{Therefore,} \]
\[ C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + \frac{e^{-(\alpha+b)T}}{\alpha} \left[ C_2 e^{\alpha T} - C_2 + C_3 \right] \sum_{i=1}^n a_i p_i b^i T^{i-1} \frac{(i-1)!(i-1)}{(i-1)!} \]
\[ (2.7.3) \]

Differentiating Equation (2.7.3) with respect to \( T \) to yield
\[ C(T) = C_1 e^{-\alpha T} + \frac{e^{-(\alpha+b)T}}{\alpha} \sum_{i=1}^n a_i p_i b^i T^{i-1} \left\{ \alpha C_2 e^{\alpha T} \right. \]
\[ + \left. \left( C_2 e^{\alpha T} - C_2 + C_3 \right) \left( \frac{(i-1)T}{T} - (\alpha+b) \right) \right\} \]
\[ C'(T) = 0, \quad \text{implies that} \]
\[ \alpha C_1 = e^{-bT} \sum_{i=1}^n a_i p_i b^i T^{i-1} \left\{ -\alpha C_2 e^{\alpha T} + \left( C_2 e^{\alpha T} - C_2 + C_3 \right) \right. \]
\[ \left. -\frac{(i-1)T}{T} + (\alpha+b) \right\}. \]
\[ (2.7.4) \]
For \( n = 1, n = 2 \) and \( n = 3 \), we get cost function for 1-stage, 2-stage and 3-stage superposed NHPP. The optimum release time \( T^o \) can be obtained on the same lines as discussed in the previous sections.

**Case (ii):**

Here, the total expected software product cost under the assumption that the software reliability changes is given by

\[
C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_2 \int_0^T \lambda(t) e^{-\alpha t} \, dt + C_3 \int_0^\infty \lambda(t) e^{-\alpha t} \, dt.
\]

Where value of \( \lambda(t) \) is given by Equation (2.5.7). Thus,

\[
C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_2 \int_0^T \frac{n \sum a_i p_i b^i t^{i-1}}{(i-1)!} e^{-bt} e^{-\alpha t} \, dt
\]

\[
+ C_3 \int_0^\infty \frac{n \sum a_i p_i b^i t^{i-1}}{(i-1)!} e^{-bt} e^{-\alpha t} \, dt.
\]

That is,

\[
C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] + C_2 \sum_{i=1}^n \frac{a_i p_i b^i T}{(i-1)!} \int_0^{t^{i-1}} e^{-(\alpha+b)t} \, dt
\]

\[
+ C_3 \sum_{i=1}^\infty \frac{a_i p_i b^i T}{(i-1)!} \int_0^{t^{i-1}} e^{-(\alpha+b)t} \, dt.
\]

(2.7.5)

Let, \( I_1 = \int_0^{t^{i-1}} e^{-(\alpha+b)t} \, dt \)

By using the Equation (2.6.12) we have

\[
I_1 = \left[ -e^{-(\alpha+b)t} \left( \frac{t^{i-1}}{(\alpha+b)} + \frac{(i-1)t^{i-2}}{(\alpha+b)^2} + \ldots + \frac{(i-1)!t}{(\alpha+b)^{i-1}} + \frac{(i-1)!}{(\alpha+b)^i} \right) \right]_0^T
\]
Thus,

\[ I_1 = -e^{-(\alpha+b)T} \left\{ \frac{T^{i-1}}{(\alpha + b)} + \frac{(i-1)T^{i-2}}{(\alpha + b)^2} + \ldots + \frac{(i-1)!T}{(\alpha + b)^{i-1}} + \frac{(i-1)!}{(\alpha + b)^{i}} \right\} + \frac{1}{(\alpha + b)^{i}} \]

Thus,

\[ \frac{I_i}{(i-1)!} = -e^{-(\alpha+b)T} \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}} + \frac{1}{(\alpha + b)^{i}}. \quad (2.7.6) \]

By using (2.6.13) and (2.7.6) in (2.7.5), we get

\[ C(T) = C_1 \left[ \frac{1-e^{-\alpha T}}{\alpha} \right] - C_2 \sum_{i=1}^{n} \left( a_i p_i b^i \right) e^{-(\alpha+b)T} \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}} \]

\[ + \frac{1}{(\alpha + b)^{i}} \right\} + C_3 \sum_{i=1}^{n} a_i p_i b^i e^{-(\alpha+b)T} \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}}. \quad (2.7.7) \]

Differentiating above equation with respect to \( T \), to obtain

\[ C'(T) = C_1 e^{-\alpha T} - C_2 e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b^i \sum_{j=0}^{i-1} \frac{jT^{j-1}}{j!(\alpha + b)^{i-j}} \]

\[ + (\alpha + b) C_2 e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b^i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}} \]

\[ + C_3 e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b^i \sum_{j=0}^{i-1} \frac{jT^{j-1}}{j!(\alpha + b)^{i-j}} \]

\[ - C_3 (\alpha + b)e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b^i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}}. \]

Thus,

\[ C'(T) = C_1 e^{-\alpha T} + C_2 e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b^i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}} \left\{ -\frac{j}{T} + (\alpha + b) \right\} \]

\[ + C_3 e^{-(\alpha+b)T} \sum_{i=1}^{n} a_i p_i b^i \sum_{j=0}^{i-1} \frac{T^j}{j!(\alpha + b)^{i-j}} \left\{ \frac{j}{T} - (\alpha + b) \right\}. \]

\[ C'(T) = 0, \quad \text{implies that} \]
\[ C_1 = C_2 \ e^{-bT} \sum_{i=1}^{n} a_i p_i b^i \ i^{-1} \sum_{j=0}^{T^j} \ j!(\alpha + b)^{j-1} \left\{ \frac{j}{T} - (\alpha + b) \right\} \]

\[ + C_3 \ e^{-bT} \sum_{i=1}^{n} a_i p_i b^i \ i^{-1} \sum_{j=0}^{T^j} \ j!(\alpha + b)^{i-j} \left\{ (\alpha + b) - \frac{j}{T} \right\}. \quad (2.7.8) \]

For \( n = 1, n = 2 \) and \( n = 3 \), then cost function of \( n \)-stage process reduces to cost function of 1-stage, 2-stage and 3-stage superposed NHPP. The optimum release time \( T^\circ \) and optimum release policy can be obtained on the same lines as discussed in the earlier sections.

\section*{2.8 Discussion and Further Scope}

In the proposed cost functions we have taken the test effort \( w(t) = c \) for \( T < t < t_c \). But in reality it may not be so, because large number of customers may be using the software. Hence, one may consider the dynamic nature of test effort after the release time \( T \).

The cost models for 2-stage, 3-stage and \( n \)-stage superposed NHPPs for software product with life-time warranty and discount rate are developed. In these models we have not considered the test effort and imperfect rectification of errors due to the added notation and mathematical complexities. Hence, one may use these factors in the cost models and get \( n \)-stage superposed NHPP cost model in general. While developing 2-stage, 3-stage and \( n \)-stage superposed NHPP cost models, it is assumed that fault detection rate, fault isolation rate and fault rectification rate are the same. But in reality these rates are different. Hence, cost models for SRGMs based on NHPPs can be constructed by considering test effort, imperfect rectifications of errors and different rates.
of fault detection, isolation and rectification of errors. One can also
utilize the penalty cost and the life cycle of software product to be random
variable for constructing these cost models. These problems will be
tackled in the future study.

Acknowledgment
I convey my special gratitude to the President Poojya
Dr. Sharanabasawappa Appa, Sharnbasveshwar Vidya Vardhak Sangha,
Gulbarga for his support to carry out the Project work. I wish to
acknowledge with thanks manifold help rendered by Prof. B.V.Dhandra,
Chairman, Department of Computer Science, Gulbarga University,
Gulbarga. I am thankful to Prof. Sannabasanagouda Dollegoudar Patil
Principal, Godutai Doddappa Appa Arts and Commerce College for
Woman, Gulbarga for his co-operation during the project period. I am also
thankful to Mr.Gururaj R Mukarambi, Research Scholar, Department of
Computer Science, Gulbarga University, Gulbarga for his help in bringing
out the final report of the Minor Research Project. I record my thanks to
the authorities of the UGC for financial support extended under Minor
Research project Vide No.MRP(S)-483/09-10/KAGU 024/UGC-SWRO
dated 27th January 2010 without which it could have been difficult to
carry out this work.

Lastly but not least, I thank all my family members for their
constant encouragement and support.
REFERENCES


List of Publications under the Minor Research Project: