APPENDIX
Considering Residual Faults of Burr Type XII
Software Reliability Growth Model

B.Rama Devi 1, Dr.R. Satya Prasad 2 and Dr.G.Sridevi 3
1Research Scholor, Department of CSE, Acharya Nagarjuna University, Guntur, India.
2Assoc.Prof, Department of CSE, Acharya Nagarjuna University, Guntur, India.
3Professor, Dept. of CSE, KL University, Vaddeswaram, Guntur, India

Abstract- Software Reliability Growth model (SRGM) is a mathematical model of how the software reliability improves as faults are detected and repaired. A large number of software reliability growth models have been proposed to analyze the reliability of software application during the testing phase, with the increasing demand to deliver high-quality software, more accurate software reliability models are required to estimate the optimal software release time and the cost of testing efforts. This paper proposes Burr type XII based Software Reliability growth model with Interval domain data. The unknown parameters of the model are estimated using the maximum likelihood (ML) estimation method. Reliability of a software system using Burr type XII distribution, which is based on Non-Homogenous Poisson process (NHPP), is presented through estimation procedures. The performance of the SRGM is judged by its ability to fit the software failure data. How good does a mathematical model fit to the data is also being calculated. To access the performance of the considered SRGM, we have carried out the parameter estimation on the real software failure datasets.

Keywords – Software Reliability, Burr type XII distribution, NHPP, ML Estimation, Fault detection rate.

I. INTRODUCTION

Software reliability is defined as the probability of failure free software operation for a specified period of time in a specified environment (Lyu, 1996) (Musa et al., 1987)[3][4]. SRGM is a mathematical model of how the software reliability improves as faults are detected and required (Quadri and Ahmad, 2010) [12]. Among all SRGMs developed so far a large family of stochastic reliability models based on a Non-Homogeneous Poisson Process known as NHPP reliability model has been widely used. Software Reliability is the most dynamic quality characteristic which can measure and predict the operational quality of the software system during its intended life cycle. If the selected model does not fit the collected software testing data relatively well. We would expect a low prediction ability of this model and the decision makings based on the analysis of this model would be far from what is considered to be optimal decision (xie et al., 2001) [13]. This paper presents a method for model validation.

II. RELATED RESEARCH

This section presents the theory that underlies the proposed distributions and maximum likelihood estimation for complete data. If ‘t’ is a continuous random variable with pdf: \( f(t; \theta_1, \theta_2, ..., \theta_k) \). Where \( \theta_1, \theta_2, ..., \theta_k \) are k unknown constant parameters which need to be estimated, and cdf: \( F(t) \). Where, the mathematical relationship between the pdf and cdf is given by: \( f(t) = \frac{d(F(t))}{dt} \). Let ‘a’ denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function of the finite failure NHPP models can be written as: \( m(t) = aF(t) \). Where, \( F(t) \) is a cumulative distributive function. The failure intensity function \( \lambda(t) \) in case of the finite failure NHPP models is given by: \( \lambda(t) = aF'(t) \) [8].
A. NHPP Model

There are numerous software reliability growth models available for use according to probabilistic assumptions. The Non Homogenous Poisson Process (NHPP) based software reliability growth models are proved to be quite successful in practical software reliability engineering [4]. Model parameters can be estimated by using maximum Likelihood Estimate (MLE). NHPP model formulation is described in the following lines.

A software system is subjected to failures at random times caused by errors present in the system. Let \( \{N(t), t \geq 0\} \) be a counting process representing the cumulative number of failures by time ‘t’, where \( t \) is the failure intensity function, which is proportional to the residual fault content.

Let \( m(t) \) represent the expected number of software failures by time ‘s’. The mean value function \( m(t) \) is finite valued, non-decreasing, non-negative and bounded with the boundary conditions.

\[
m(t) = \begin{cases} 
0, & t = 0 \\
\alpha, & t \to \infty
\end{cases}
\]

Where ‘\( \alpha \)’ is the expected number of software errors to be eventually detected.

Suppose \( N(t) \) is known to have a Poisson probability mass function with parameters \( m(t) \) i.e.,

\[
P\{N(t) = n\} = \frac{[m(t)]^n e^{-m(t)}}{n!}, n = 0, 1, 2, \ldots, \infty
\]

Then \( N(t) \) is called an NHPP. Thus the stochastic behaviour of software failure phenomena can be described through the \( N(t) \) process. Various time domain models have appeared in the literature that describes the stochastic failure process by an NHPP which differ in the mean value function \( m(t) \).

B. Proposed Model Description –

In this paper, we propose to access the software reliability based on Burr Type XII distribution model. The Burr distribution has a flexible shape and controllable scale and location which makes it appealing to fit to data. It is frequently used to model insurance claim sizes [5]. The mean value function and intensity function of Burr Type XII NHPP model are as follows.

The Cumulative distributive function (CDF) is given by

\[
m(t) = \int_0^t \lambda(t) dt = a \left[ 1 - \left(1 + t^c \right)^{-b} \right]
\]

\[= a \cdot F(t)\]

The Probability Density Function (PDF) of Burr XII distribution are given, respectively by

\[
\lambda(t) = a \left\{ \frac{cbt^{c-1}}{(1+t^c)^{b+1}} \right\} = a \cdot f(t)
\]

Where \( t > 0, a > 0, b > 0 \) and \( c > 0 \) denote the expected number of faults that would be detached given infinite testing time in case of finite failure NHPP models. In order to have an assessment of the software reliability, \( a, b \) and \( c \) are unknown parameters and estimated by using Newton Raphson method. Expressions are now delivered for estimating ‘\( a \)’, ‘\( b \)’ and ‘\( c \)’ for the Burr type XII model.
\[
p\{N(t) = n\} = \frac{[m(t)]^n e^{-m(t)}}{n!}
\]

\[
\lim_{n \to \infty} p\{N(t) = n\} = \frac{e^{-a} \cdot a^n}{n!}
\]

This is also a Poisson model with mean ‘a’.

Let \(S_k\) be the time between \((k - 1)th\) and \(kth\) failure of the software product. Let \(X_k\) be the time up to the \(kth\) failure. Let us find out the probability that time between \((k - 1)th\) and \(kth\) failures, i.e., \(S_k\) exceeds a real number ‘s’ given that the total time up to the \((k - 1)th\) failure is equal to \(x\).

\[i.e., \ P\left[S_k > \frac{s}{X_{k-1}} = x\right]\]

\[RS_k/X_{k-1}(s/x) = e^{-[m(x+s)−m(s)]}\]

This Expression is called Software Reliability.

III. ILLUSTRATING THE MLE

In this section we develop expressions to estimate the parameters of the Burr type XII model based on Interval domain data. Parameter estimation is of primary importance in software reliability prediction.

A set of failure data is usually collected in one of two common ways, Interval domain data and time domain data. In this paper parameters are estimated from the Interval domain data.

The mean value function of Burr type XII model is given by

\[
m(t) = a \left[1 - \left(1 + t^c\right)^{-b}\right], \ t \geq 0
\]

\[
LogL = \sum_{i=1}^{k} (n_i - n_{i-1}) \log \left[ m(t_i) - m(t_{i-1}) \right] - m(t_k)
\]

\[
LogL = \sum_{i=1}^{k} (n_i - n_{i-1}) \log \left[ a \left[1 - \left(1 + t_i^c\right)^{-b}\right] - a \left[1 - \left(1 + t_{i-1}^c\right)^{-b}\right] \right] - a \left[1 - \left(1 + t_k^c\right)^{-b}\right]
\]

\[
LogL = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ Loga + \log \left[ \left(1 + t_i^c\right)^{-b} - \left(1 + t_{i-1}^c\right)^{-b} \right] \right] - a + a(1 + t_k^c)^{-b}
\]

Taking the Partial derivative with respect to ‘a’ and equating to ‘0’.

\[
(i.e., \ \frac{\partial \text{Log } L}{\partial a} = 0)
\]

\[
\therefore a = \sum_{i=1}^{k} (n_i - n_{i-1}) \frac{(1 + t_i^c)^b}{(1 + t_k^c)^b - 1}
\]
The parameter 'b' is estimated by iterative Newton Raphson Method using
\[ b_{n+1} = b_n - \frac{g(b)}{g'(b)} \]
Where \( g(b) \) and \( g'(b) \) are expressed as follows.
\[ g(b) = \frac{\partial \log L}{\partial b} = 0 \]
\[ \frac{\partial \log L}{\partial b} = g(b) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ - \log(t_{i-1} + 1) - \log(t_i + 1) + \frac{(t_i + 1)^b \log(t_i + 1) - (t_{i-1} + 1)^b \log(t_{i-1} + 1)}{(t_i + 1)^b - (t_{i-1} + 1)^b} \right] \]
\[ \quad + \frac{1}{(t_k + 1)^b - 1} \log \left( \frac{1}{1 + t_k} \right) \]
(5)
Again partial differentiating with respect to 'b' and equate to 0 , we get
\[ g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0 \]
\[ \frac{\partial^2 \log L}{\partial b^2} = g'(b) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{2(t_{i-1} + 1)^b(t_i + 1)^b \log(t_i + 1) \log \left( \frac{t_i + 1}{t_{i-1} + 1} \right)}{(t_i + 1)^b - (t_{i-1} + 1)^b} \right] \]
\[ \quad + \sum_{i=1}^{k} (n_i - n_{i-1}) \log(1 + t_i) \left( \frac{(t_i + 1)^b \log(t_i + 1)}{(t_i + 1)^b - 1} \right) \]
(6)
The parameter 'c' is estimated by iterative Newton Raphson Method using
\[ c_{n+1} = c_n - \frac{g(c_n)}{g'(c_n)} \]
Where \( g(c) \) and \( g'(c) \) are expressed as follows.
\[ g(c) = \frac{\partial \log L}{\partial c} = 0 \]
\[ \frac{\partial \log L}{\partial c} = g(c) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ - \log(t_{i-1}) \frac{t_{i-1}^c}{(1 + t_{i-1})} - \log(t_i) \frac{t_i^c}{(1 + t_i)} + \frac{t_i^c \log(t_i) - t_{i-1}^c \log(t_{i-1})}{(t_i - t_{i-1})} \right] - \sum_{i=1}^{k} (n_i - n_{i-1}) \frac{\log(t_i - t_{i-1})}{(1 + t_i)} \]
(7)
\[ g'(c) = \frac{\partial^2 \log L}{\partial c^2} = 0 \]

\[ \frac{\partial^2 \log L}{\partial c^2} = g'(c) = \sum_{i=1}^{k}(n_i - n_{i-1}) \left[ \left( \log \left( \frac{t_{i-1}^c}{t_i^c} \right) \frac{t_i^c - t_{i-1}^c}{(t_i^c - t_{i-1}^c)^2} \right) \{\log t_i - \log t_{i-1}\} \right] + \sum_{i=1}^{k}(n_i - n_{i-1}) (\log t_i^c)^2 \frac{t_i^c}{(1+t_i^c)^2} \]

\[ -(\log t_{i-1})^2 \frac{t_{i-1}^c}{(1+t_{i-1}^c)^2} - (\log t_i)^2 \frac{t_i^c}{(1+t_i^c)^2} \]

IV. DATA ANALYSIS

A set of failure data Phase 1 and Phase 2 taken from Pham (2005) and Release #1, #2, #3 and #4 datasets taken from Wood (1996) consists of the observation time(week), CPU Hours and the number of failures detected per week :defects found [7][11].

Solving equations in Section III by Newton Raphson Method (N-R) method for all the data sets, the iterative solutions for MLEs of a, b, c of given software failure datasets are shown in Table-1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of samples</th>
<th>Estimated Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>Phase 1</td>
<td>21</td>
<td>25.994042</td>
</tr>
<tr>
<td>Phase 2</td>
<td>21</td>
<td>41.590454</td>
</tr>
<tr>
<td>Release #1</td>
<td>20</td>
<td>87.533224</td>
</tr>
<tr>
<td>Release #2</td>
<td>19</td>
<td>111.77847</td>
</tr>
<tr>
<td>Release #3</td>
<td>12</td>
<td>59.376054</td>
</tr>
<tr>
<td>Release #4</td>
<td>19</td>
<td>42.831021</td>
</tr>
</tbody>
</table>

V. METHOD OF PERFORMANCE ANALYSIS

The performance of SRGM is judged by its ability to fit the software failure data. The term goodness of fit denotes the question of “How good does a mathematical model fit to the data?”. In order to validate the model under study and to assess its performance, experiments on a set of actual software failure data have been performed. The performance evaluation of software reliability growth model is generally measured with sum of square errors (SSE) and correlation...
Considering Residual Faults of Burr Type XII Software Reliability Growth Model

index of regression curve equation (R-square). Among them, the model performance is better when SSE is smaller and R-square is close to 1.

SSE is used to describe the distance between actual and estimated number of faults detected totally, which is defined as

$$SSE = \sum_{i=1}^{n} (y_i - m(t_i))^2$$

Where \(n\) denotes the number of failure samples in failure data set, \(y_i\) denotes the number of faults observed to the moment \(t_i\), and \(m(t_i)\) denotes the estimated number of faults detected to the time \(t_i\) according to the proposed model. The model can provide a better goodness-of-fit when the value of SSE is smaller.

The equation of calculating the value R-square is written as:

$$R-square = 1 - \frac{\sum_{i=1}^{n} (y_i - m(t_i))^2}{\sum_{i=1}^{n} (\bar{y} - y_i)^2}$$

Where \(\bar{y}\) denotes the mean value of faults detected. The model can provide a better goodness-of-fit when the value of R-square is close to 1.

Table -2 The results on different datasets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Reliability ((t_n+50))</th>
<th>SSE</th>
<th>R-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>0.999985708</td>
<td>2603.5148</td>
<td>1.2859</td>
</tr>
<tr>
<td>Phase 2</td>
<td>0.999983394</td>
<td>10015.7216</td>
<td>1.3703</td>
</tr>
<tr>
<td>Release #1</td>
<td>0.999973211</td>
<td>20913.2148</td>
<td>0.3464</td>
</tr>
<tr>
<td>Release #2</td>
<td>0.999967139</td>
<td>42882.6562</td>
<td>0.9822</td>
</tr>
<tr>
<td>Release #3</td>
<td>0.999902327</td>
<td>8491.8095</td>
<td>0.8617</td>
</tr>
<tr>
<td>Release #4</td>
<td>0.999989656</td>
<td>5918.2451</td>
<td>1.1753</td>
</tr>
</tbody>
</table>

From the Table -2 it can be seen that the value of SSE is smaller and the value of R-square is more close to 1. The results indicate that our NHPP Burr type XII model based on fault detection rate fits the data in the given datasets, best and predicts the number of residual faults in software most accurately.

VI. CONCLUSION

Software reliability growth model can estimate the optimal software release time and the cost of testing efforts [14]. And SRGM can help project managers to determine the testing resources and manpower needed to achieve desired reliability requirements. So more accurate model is needed to decrease the testing cost and increase the profit of releasing software [15][16][17]. In this paper the fault detection rate is calculated with the number of faults remaining in the software. Considering the two factors jointly the fault detection rate is more realistic and accurate. Moreover, we have discussed the
performances of 6 datasets by using our new Burr type XII SRGM. The experiment result shows that the Phase 1 data set can provide a better goodness-of-fit compared with other datasets are given in Table 2. The reliability of the model over Release #4 data is high among the data sets which were considered.

REFERENCE

Assessing Burr Type XII software reliability for interval domain data using SPC

R. Satya Prasad¹, B. Rama Devi² and G. Sridevi³

¹Department of CSE, Acharya Nagarjuna University, Guntur, India.
²Department of CSE, Acharya Nagarjuna University, Guntur, India.
³Department of CSE, Nimra Women’s College of Engineering, Vijayawada, India.

ABSTRACT
Statistical Process Control (SPC) is the best choice to monitor software reliability process. It assists software development team to identify and actions to be taken during software failure process and hence, assures better software reliability. In this paper we propose a control mechanism based on the cumulative observations of Interval domain data using the mean value function of Burr type XII model, which is Non-Homogenous Poisson Process (NHPP) based. The Maximum Likelihood Estimation (MLE) approach is used to estimate the unknown parameters of the model.

Introduction
Many software reliability models have been proposed in last 40 years to compute the reliability growth of products during software development phase. These models can be of two types i.e. static and dynamic. A static model uses software metrics to estimate the number of defects in the software. A dynamic model uses the past failure discovery rate during software execution over time to estimate the number of failures. Various software reliability growth models (SRGMs) exist to estimate the expected number of total defects (or failures) or the expected number of remaining defects (or failures).

The goal of software engineering is to produce high quality software at low cost. As, human beings are involved in the development of software, there is a possibility of errors in the software. To identify and eliminate human errors in software development process and also to improve software reliability, the Statistical Process Control concepts and methods are the best choice. SPC concepts and methods are used to monitor the performance of a software process over time in order to verify that the process remains in the state of statistical control. It helps in finding assignable causes, long term improvements in the software process. Software quality and reliability can be achieved by eliminating the causes or improving the software process or its operating procedures [1].

The most popular technique for maintaining process control is control charting. The control chart is one of the seven tools for quality control. Software process control is used to secure, that the quality of the final product will conform to predefined standards. In any process, regardless of how carefully it is maintained, a certain amount of natural variability will always exist. A process is said to be statistically “in-control” when it operates with only chance causes of variation. On the other hand, when assignable causes are present, then we say that the process is statistically “out-of-control”. Control charts should be capable to create an alarm when a shift in the level of one or more parameters of the underlying distribution occurs or a non-random behavior comes into. Normally, such a situation will be reflected in the control chart by points plotted outside the control limits or by the presence of specific patterns. The most common non-random patterns are cycles, trends, mixtures and stratification [2]. For a process to be in control the control chart should not have any trend or non random pattern. The selection of proper SPC charts is essential to effective statistical process control implementation and use. The SPC chart selection is based on data, situation and need [3].

Chan et al.,[4] proposed a procedure based on the monitoring of cumulative quantity. This approach has shown to have a number of advantages: it does not involve the choice of a sample size; it raises fewer false alarms; it can be used in any environment; and it can detect further process improvement. Xie et al.,[5] proposed t-chart for reliability monitoring where the control limits are defined in such a manner that the process is considered to be out of control when one failure is less than LCL or greater than UCL. Assuming an acceptable false alarm =0.0027 the control limits were defined. In section 5 of present paper, a method is proposed to estimate the parameters and defining the limits. The process control is decided by taking the successive differences of mean values.

Background Theory
This section presents the theory that underlies NHPP models, the SRGMs under consideration and maximum likelihood estimation for complete data. If ‘t’ is a continuous random variable with pdf: 

\[ f(t; \theta_1, \theta_2, \ldots, \theta_k) \]

where \( \theta_1, \theta_2, \ldots, \theta_k \) are k unknown constant parameters which need to be estimated, and cdf: \( F(t) \). Where, the mathematical relationship between the pdf and cdf is given by:

\[ f(t) = \frac{d(F(t))}{dt} \]

Let ‘a’ denote the expected number of faults that would be detected given infinite testing time in case of...
finite failure NHPP models. Then, the mean value function of the finite failure NHPP models can be written as: \( m(t) = a F(t) \).

Where, \( F(t) \) is a cumulative distributive function. The failure intensity function \( \lambda(t) \) in case of the finite failure NHPP models is given by: \( \lambda(t) = a f(t) \) [6].

**NHPP model**

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proved to be quite successful in practical software reliability engineering [7]. The main issue in the NHPP model is to determine an appropriate mean value function to denote the expected number of failures experienced up to a certain time point. Model parameters can be estimated by using Maximum Likelihood Estimate (MLE). Various NHPP SRGMs have been built upon various assumptions. Many of the SRGMs assume that each time a failure occurs, the fault that caused it can be immediately removed and no new faults are introduced. Which is usually called perfect debugging. Imperfect debugging models have proposed a relaxation of the above assumption [8][9].

A software system is subjected to failures at random times caused by errors present in the system. Let \( F(t) \) be a counting process representing the cumulative number of failures by time ‘\( t \)’, where \( t \) is the failure intensity function, which is proportional to the residual fault content. Let \( m(t) \) represent the expected number of software failures by time ‘\( t \)’. The mean value function \( m(t) \) is finite valued, non-decreasing, non-negative and bounded with the boundary conditions.

\[
m(t) = \begin{cases} 
0, & t = 0 \\
a, & t \to \infty 
\end{cases}
\]

Where ‘\( a \)’ is the expected number of software errors to be eventually detected.

Suppose \( N(t) \) is known to have a Poisson probability mass function with parameters \( m(t) \) i.e.,

\[
P[N(t) = n] = \frac{[m(t)]^n \cdot e^{-m(t)}}{n!}, n = 0, 1, 2, \ldots, \infty
\]

Then \( N(t) \) is called an NHPP. Thus the stochastic behaviour of software failure phenomena can be described through the \( N(t) \) process. Various time domain models have appeared in the literature that describes the stochastic failure process by an NHPP which differ in the mean value function \( m(t) \).

**Model under consideration: Burr Type XII model**

In this paper, we propose to monitor software quality using SPC based on Burr Type XII distribution model. The Burr distribution has a flexible shape and controllable scale and location which makes it appealing to fit to data. It is frequently used to model insurance claim sizes [16].

The mean value function and intensity function of Burr Type XII NHPP model are as follows.

The Cumulative distribution function (CDF) is given by

\[
m(t) = \int_0^t \lambda(t) \, dt = a \left[1 - \left(1 + t^\gamma\right)^{-\beta}\right] = a F(t)
\]

The Probability Density Function (PDF) of Burr XII distribution are given, respectively by

\[
\lambda(t) = a \frac{c b t^{c-1}}{(1+t^\gamma)^{b+1}} = a f(t)
\]

Where \( b > 0, a > 0, b > 0 \) and \( c > 0 \) denote the expected number of faults that would be detached given infinite testing time in case of finite failure NHPP models.

**Maximum Likelihood Estimation (MLE)**

In this section we develop expressions to estimate the parameters of the Burr type XII model based on interval domain data. Parameter estimation is of primary importance in software reliability prediction.

A set of failure data is usually collected in one of two common ways, time domain data and interval domain data. In this paper parameters are estimated from the interval domain data.

The mean value function of Burr type XII model is given by

\[
m(t) = a \left[1 - \left(1 + t^\gamma\right)^{-\beta}\right]
\]

(3.1)

In order to have an assessment of the software reliability, \( a, b \) and \( c \) are to be known or they are to be estimated from software failure data. Expressions are now delivered for estimating ‘\( a \)’, ‘\( b \)’ and ‘\( c \)’ for the Burr type XII model.

Assuming the given data are given for the cumulative number of detected errors \( n_i \) in a given time interval \((0, t_i)\) where \( i=1,2, \ldots, n \) and \( 0 < t_1 < t_2 < \ldots < t_n \), then the logarithmic likelihood function (LLF) for interval domain data [10] is given by

\[
\log L = \sum_{i=1}^k (n_i - n_{i-1}) \log \left[m(t_i) - m(t_{i-1})\right] - m(t_i)
\]

(3.2)
\[
\log L = \sum_{i=1}^{k} (n_i - n_{i-1}) \log \left[ a \left[ 1 - \left(1 + t_i^c \right)^{-b} \right] - a \left[ 1 - \left(1 + t_{i-1}^c \right)^{-b} \right] - a \left[ 1 - \left(1 + t_k^c \right)^{-b} \right] \right] - a + a(1 + t_k^c)^{-b}
\]

Taking the Partial derivative with respect to 'a' and equating to '0',
\[
\frac{\partial \log L}{\partial a} = 0
\]

\[
\therefore a = \sum_{i=1}^{k} (n_i - n_{i-1}) \left(1 + t_i^c\right)^b \left(1 + t_k^c\right)^{-b} - 1
\]

The parameter 'b' is estimated by iterative Newton Raphson Method using
\[
g(b) = \frac{\partial \log L}{\partial b} = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ - \log \left( t_{i-1} + 1 \right) - \log \left( t_i + 1 \right) + \frac{(t_i + 1)^b \log \left( t_i + 1 \right) - \left( t_{i-1} + 1 \right)^b \log \left( t_{i-1} + 1 \right)}{(t_i + 1)^b - \left( t_{i-1} + 1 \right)^b} \right]
\]

Again partial differentiating with respect to 'b' and equate to 0, we get
\[
g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0
\]

\[
\frac{\partial^2 \log L}{\partial b^2} = g'(b) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{2(t_i + 1)^b \log \left( t_i + 1 \right) \log \left( \frac{t_i + 1}{t_{i-1} + 1} \right)}{(t_i + 1)^b - \left( t_{i-1} + 1 \right)^b} \right] + \sum_{i=1}^{k} (n_i - n_{i-1}) \log(1 + t_i^c) \left( \log(t_i + 1) \right)^2
\]

The parameter 'c' is estimated by iterative Newton Raphson Method using
\[
g(c) = \frac{\partial \log L}{\partial c} = 0
\]

\[
\frac{\partial \log L}{\partial c} = g(c) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ - \log t_{i-1} - \frac{t_i^c}{(1 + t_i^c)} - \log t_i^c - \frac{t_i^c - t_{i-1}^c \log t_{i-1}}{(t_i^c - t_{i-1}^c)^2} \right] - \sum_{i=1}^{k} (n_i - n_{i-1}) \frac{\log t_k^c}{(1 + t_k^c)}
\]

\[
g'(c) = \frac{\partial^2 \log L}{\partial c^2} = 0
\]

\[
\frac{\partial^2 \log L}{\partial c^2} = g'(c) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \log \left( \frac{t_{i-1} + 1}{t_i^c} \right) \frac{t_i^c - t_{i-1}^c}{(t_i^c - t_{i-1}^c)^2} \left( \log t_i - \log t_{i-1} \right) \right] + \sum_{i=1}^{k} (n_i - n_{i-1}) \left( \log t_k^c \right)^2 \frac{t_i^c}{(1 + t_i^c)^2} + \sum_{i=1}^{k} (n_i - n_{i-1}) \left( \log t_i^c \right)^2 \frac{t_i^c}{(1 + t_i^c)^2}
\]
Interval Domain Datasets

**DS #1: Telecommunication System Data**

The dataset was reported by Zhang *et al.* (2002) based on system test data for a telecommunication system [17] are shown in Table 4.1.

<table>
<thead>
<tr>
<th>Week Index</th>
<th>Fault</th>
<th>Week Index</th>
<th>Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>14</td>
<td>2</td>
</tr>
</tbody>
</table>

**DS #2: Failure Data from Misra (1983)**

A set of failure data taken from Misra (1983), given in Table 4.2 consists of the observation time (week) and the number of failures detected per week are errors: major and minor[18].

<table>
<thead>
<tr>
<th>Week</th>
<th>Minor Errors</th>
<th>Week</th>
<th>Minor Errors</th>
<th>Week</th>
<th>Minor Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>13</td>
<td>5</td>
<td>25</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>14</td>
<td>3</td>
<td>26</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>15</td>
<td>3</td>
<td>27</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>16</td>
<td>3</td>
<td>28</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>17</td>
<td>4</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>18</td>
<td>10</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>19</td>
<td>3</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>20</td>
<td>1</td>
<td>32</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>21</td>
<td>2</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>22</td>
<td>4</td>
<td>34</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>23</td>
<td>5</td>
<td>35</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>24</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Results

The performance of the model under consideration is exemplified by applying on the datasets given in tables 4.1 and 4.2.

**Calculation of Control Limits**

The control limits for the chart are defined in such a manner that the process is considered to be out of control when the time to observe exactly one failure is less than LCL or greater than UCL. Our aim is to monitor the failure process and detect any change of the intensity parameter. When the process is normal, there is a chance for this to happen and it is commonly known as false alarm. The traditional false alarm probability is to set to be 0.27% although any other false alarm probability can be used. The actual acceptable false alarm probability should in fact depend on the actual product or process [13].

\[
T_U = \left[ 1 - \left(1 + t^\gamma \right)^{-b} \right] = 0.99865
\]

\[
T_C = \left[ 1 - \left(1 + t^\gamma \right)^{-b} \right] = 0.5
\]

\[
T_L = \left[ 1 - \left(1 + t^\gamma \right)^{-b} \right] = 0.00135
\]

The estimated parameters and the control limits are shown in Tables 4.3 and table 4.4.

**Table 4.3. Estimated Parameters for the datasets**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS #1</td>
<td>26.623982</td>
<td>0.973637</td>
<td>1.066462</td>
</tr>
<tr>
<td>DS #2</td>
<td>142.175009</td>
<td>0.985185</td>
<td>1.079388</td>
</tr>
</tbody>
</table>

**Table 4.4. Estimated Control Limits**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(m(t_U))</th>
<th>(m(t_C))</th>
<th>(m(t_L))</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS #1</td>
<td>26.588039</td>
<td>13.311991</td>
<td>0.035942</td>
</tr>
<tr>
<td>DS #2</td>
<td>141.983072</td>
<td>71.087504</td>
<td>0.191936</td>
</tr>
</tbody>
</table>

**Distribution of Failures**

The \(m(t)\) values are calculated at each cumulative value of ‘\(t\)’. The successive differences of these values are calculated to plot as a failure control chart along with the calculated control limits which vary with the considered data. The following tables 5.2.1, 5.2.2 and graphs given in figures 5.2.1, 5.2.2 shows the performance of the Burr type XII model in software process control.
### Table 5.2.1. Successive Differences of Mean Values DS #1

<table>
<thead>
<tr>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.06649898</td>
<td>4.693181825</td>
<td>6</td>
<td>24.13700545</td>
<td>0.434032852</td>
<td>11</td>
<td>25.58644296</td>
<td>0.045203949</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>17.75968081</td>
<td>2.322150723</td>
<td>7</td>
<td>24.5710383</td>
<td>0.306969711</td>
<td>12</td>
<td>25.63164691</td>
<td>0.041497386</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20.08183153</td>
<td>2.280123102</td>
<td>8</td>
<td>24.87800802</td>
<td>0.228336084</td>
<td>13</td>
<td>25.67314429</td>
<td>0.073550535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22.36195463</td>
<td>1.483230742</td>
<td>9</td>
<td>25.1063441</td>
<td>0.316567045</td>
<td>14</td>
<td>25.74669483</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23.84518537</td>
<td>0.29182008</td>
<td>10</td>
<td>25.42291114</td>
<td>0.163531812</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 5.2.2. Successive Differences of Mean Values DS #2

<table>
<thead>
<tr>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
<th>FN</th>
<th>( m(t) )</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>129.588386</td>
<td>3.83236519</td>
<td>13</td>
<td>140.458879</td>
<td>0.08193940</td>
<td>25</td>
<td>141.173419</td>
<td>0.02937191</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>133.420751</td>
<td>3.09485021</td>
<td>14</td>
<td>140.540818</td>
<td>0.07467703</td>
<td>26</td>
<td>141.202790</td>
<td>0.05399569</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>136.515601</td>
<td>1.33800818</td>
<td>15</td>
<td>140.615496</td>
<td>0.06833068</td>
<td>27</td>
<td>141.256786</td>
<td>0.02487059</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>137.853609</td>
<td>0.71949381</td>
<td>16</td>
<td>140.683826</td>
<td>0.08254680</td>
<td>28</td>
<td>141.281657</td>
<td>0.00800213</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>138.573103</td>
<td>0.32965740</td>
<td>17</td>
<td>140.766373</td>
<td>0.17210562</td>
<td>29</td>
<td>141.289639</td>
<td>0.00786424</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>138.902761</td>
<td>0.18091777</td>
<td>18</td>
<td>140.938479</td>
<td>0.04392962</td>
<td>30</td>
<td>141.297523</td>
<td>0.03014629</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>139.091778</td>
<td>0.39112151</td>
<td>19</td>
<td>140.982408</td>
<td>0.01397754</td>
<td>31</td>
<td>141.327669</td>
<td>0.02132215</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>139.482900</td>
<td>0.24962793</td>
<td>20</td>
<td>140.996386</td>
<td>0.02702252</td>
<td>32</td>
<td>141.348992</td>
<td>0.01364538</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>139.732528</td>
<td>0.10866962</td>
<td>21</td>
<td>141.023408</td>
<td>0.05059012</td>
<td>33</td>
<td>141.362637</td>
<td>0.06785800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>139.841917</td>
<td>0.19114361</td>
<td>22</td>
<td>141.073998</td>
<td>0.05747388</td>
<td>34</td>
<td>141.430495</td>
<td>0.04783265</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>140.052601</td>
<td>0.27048270</td>
<td>23</td>
<td>141.131472</td>
<td>0.02139033</td>
<td>35</td>
<td>141.478328</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>140.303084</td>
<td>0.15579537</td>
<td>24</td>
<td>141.152863</td>
<td>0.02055385</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.2.1. Failure Control Chart

Figure 5.2.2. Failure Control Chart
A point below the control limit $m(t_{ij})$ indicates an alarm signal. A point above the control limit $m(t_{ij})$ indicates better quality. If the points are falling within the control limits it indicates the software process is in stable. By placing the failure cumulative data shown in tables 5.2.1 and 5.2.2 on y axis and failure number on x axis and the values of the control limits are placed on Control chart, we obtained figures 5.2.1 and 5.2.2. The software quality is determined by detecting failures at an early stage.

**Conclusion**

The given Interval domain failures data are plotted through the estimated mean value function against the failure serial order. The graphs have shown out of control signals i.e., below the LCL. Hence we conclude that our method of estimation and the control chart are giving a positive recommendation for their use in finding out preferable control processor desirable out of control signal. By observing the control chart it is identified that, for DS #1 the failure process out of UCL. For DS #2 the failure situation is detected at 6th point below LCL. Hence our proposed control chart detects out of control situation.

**References**


**Authors Profile**

**Dr. R. Satya Prasad**, received Ph.D. degree in computer science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Guntur, Andhra Pradesh. He have a satisfactory consistent academic track of record and received Gold medal from Acharya Nagarjuna University for his outstanding performance in master’s degree. He is currently working as Associate Professor in the department of Computer Science & Engg., Acharya Nagarjuna University. He has occupied various academic responsibilities like practical examiner, project adjudicator, external member of board of Examiners for various Universities and colleges in and around in Andhra Pradesh. His current research is focused on Software engineering. He has published several papers in National & International Journals.

**Mrs. B.Ramadevi**, received M.Sc. degree from Acharya Nagarjuna University and M.Tech from Vinayaka Missions University. She is currently pursing Ph.D at Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. She is currently working as Asst. professor in the Department of Computer Science, Acharya Nagarjuna University, Andhra Pradesh. Her research interests lies in Software Engineering.

**Mrs. G. Sridevi**, received M.Sc. and M.Tech degree from Acharya Nagarjuna University. She is currently pursing Ph.D at Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. She is currently working as a Vice-Principal and Associate professor in the Department of Computer Science, Nimra Women’s College of Engineering, Jupudi, Ibrahimpatnam, Vijayawada, Andhra Pradesh. Her research interests lies in Data Mining and Software Engineering.
Detection of Burr type XII Reliable Software Using SPRT on Interval Domain Data

Dr. R. Satya Prasad¹, B. Ramadevi², Dr. G. Sridevi³

¹Assoc. Prof, Department of CSE, Acharya Nagarjuna University, Guntur, India.
²Research Scholar, Department of CSE, Acharya Nagarjuna University, Guntur, India.
³Professor, Dept. of CSE, KL University, Vaddeswaram, Guntur, India.

Abstract—As the volumes of data/software is getting increased in the internet day by day, there is a need for the people to have the tools/machinery to assess the software reliability as it takes more time to come to conclusion. In Classical Hypothesis first of all testing volumes of data is to be collected and later the conclusions are to be drawn which may take more time. In this paper a well known test procedure of statistical science called as Sequential Probability Ratio Test (SPRT) is adopted for Burr Type XII model in assessing the reliability of developed software. It requires considerably less number of observations when compared with the other existing testing procedures. Hence Sequential Analysis of Statistical Science could be adopted to decide upon the reliable/unreliable of the developed software very quickly. Besides the present paper proposes the performance of SPRT on interval domain data using Burr type XII model and analyzed the results by applying on 6 data sets. The Maximum Likelihood Estimation is used for estimation of parameters.

Keywords: Burr Type XII model, Sequential Probability Ratio Test, MLE, Software Reliability, NHPP.

I. INTRODUCTION

The SPRT was initially developed by Wald (1947) for quality control problems during World War II. It has many extensions and applications: such as in clinical trial and in quality control. The original development of the SPRT is used as a statistical device to decide which of two simple hypotheses is more correct. Wald’s SPRT is currently the only Bayesian Statistical procedure in SISA. What is required in Bayesian statistics is quite a detailed description of the expectations of the outcome under the model prior to executing the data collection. In Wald’s SPRT, if certain conditions are met during the data collection decisions are taken with regard to continuing the data collection and the interpretation of the gathered data. Wald's procedure is particularly relevant if the data is collected sequentially. Sequential Analysis is different from Classical Hypothesis Testing were the number of cases tested or collected is fixed at the beginning of the experiment. In Classical Hypothesis Testing the data collection is executed without analysis and Consideration of the data. After all data is collected the analysis is done and conclusions are drawn. However, in Sequential Analysis every case is analyzed directly after being collected, the data collected up to that moment is then compared with certain threshold values, incorporating the new information obtained from the freshly collected case. This approach allows one to draw conclusions during the data collection, and a final conclusion can possibly be reached at a much earlier stage as is the case in Classical Hypothesis Testing. The advantages of Sequential Analysis are easy to see. As data collection can be terminated after fewer cases and decisions taken earlier, the savings in terms of human life and misery, and financial savings, might be considerable. In the analysis of software failure data we often deal with either Time between Failures or failure count in a given time interval. If it is further assumed that the average number of recorded failures in a given time interval is directly proportional to the length of the interval and the random number of failure occurrences in the interval is explained by a Poisson process then we know that the probability equation of the stochastic process representing the failure occurrences is given by a homogeneous Poisson process with the expression

\[ P[N(t) = n] = \frac{e^{-\lambda t} \lambda^n}{n!} \]  

(1.1)

(Sieber 1997) observes that if classical testing strategies are used, the application of software reliability growth models may be difficult and reliability predictions can be misleading. However, he observes that statistical methods can be successfully applied to the failure data. He demonstrated his observation by applying the well-known sequential probability ratio test of (Wald 1947) for a software failure data to detect unreliable software components and compare the reliability of different software versions. In this paper we consider popular SRGM Burr Type XII model and adopt the principle of Stieber in detecting unreliable software components in order to accept or reject the developed software. The theory proposed by Stieber is presented in Section 2 for a ready reference. Extension of this theory to the SRGM – Burr Type XII is presented in Section 3. Maximum Likelihood parameter estimation method is presented in Section 4. Application of the decision rule to detect unreliable software components with respect to the proposed SRGM is given in Section 5.

II. SEQUENTIAL TEST FOR A POISSON PROCESS

The sequential probability ratio test was developed by A. Wald at Columbia University in 1943. Due to its
Usefulness in development work on military and naval equipment it was classified as ‘Restricted’ by the Espionage Act (Wald 1947). A big advantage of sequential tests is that they require fewer observations (time) on the average than fixed sample size tests. SPRTs are widely used for statistical quality control in manufacturing processes. An SPRT for homogeneous Poisson processes is described below. Let \{N(t), t \geq 0\} be a homogeneous Poisson process with rate \( \lambda \). In our case, \( N(t) \) = number of failures up to time \('t'\) and \( \lambda \) is the failure rate (failures per unit time ). Suppose that we put a system on test (for example a software system, where testing is done according to a usage profile and no faults are corrected) and that we want to estimate its failure rate \( \lambda \). We cannot expect to estimate \( \lambda \) precisely. But we want to reject the system with a high probability if \( \lambda \) is ‘smaller’ than \( \lambda_0 \). As always with statistical tests, there is some risk to get the wrong answers. So we have to specify two (small) numbers ‘\( \alpha \)’ and ‘\( \beta \)’, where ‘\( \alpha \)’ is the probability of falsely rejecting the system. That is rejecting the system even if \( \lambda < \lambda_0 \). This is the "producer’s” risk. \( \beta \) is the probability of falsely accepting the system. That is accepting the system even if \( \lambda > \lambda_1 \). This is the “consumer’s” risk. With specified choices of \( \lambda_0 \) and \( \lambda_1 \) such that \( 0 < \lambda_0 < \lambda_1 \), the probability of finding \( N(t) \) failures in the time span \((0,t)\) with \( \lambda_1, \lambda_0 \) as the failure rates are respectively given by

\[
P_1 = \frac{e^{-\lambda_1 t}[\lambda_1 t]^{N(t)}}{N(t)!} \quad (2.1)
\]

\[
P_0 = e^{-\lambda_0 t}[\lambda_0 t]^{N(t)}}{N(t)!} \quad (2.2)
\]

The ratio \( \frac{P_1}{P_0} \) at any time \( \text{‘}t\text{’} \) is considered as a measure of deciding the truth towards \( \lambda_0 \) or \( \lambda_1 \), given a sequence of time instants say \( t_1 < t_2 < \cdots < t_k \) and the corresponding realizations \( N(t_1), N(t_2) \ldots N(t_k) \) of \( N(t) \). Simplification of \( \frac{P_1}{P_0} \) gives

\[
\frac{P_1}{P_0} = \exp(\lambda_0 - \lambda_1) t + \frac{\lambda_1}{\lambda_0}^{N(t)} \quad (2.3)
\]

The decision rule of SPRT is to decide in favour of \( \lambda_1 \), in favor of \( \lambda_0 \) or to continue by observing the number of failures at a later time than \( \text{‘}t\text{’} \) according as \( \frac{P_1}{P_0} \) is greater than or equal to a constant say \( A \), less than or equal to a constant say \( B \) or in between the constants \( A \) and \( B \). That is, we decide the given software product as unreliable, reliable or continue (Satya Prasad 2007) the test process with one more observation in failure data, according as

\[
\frac{P_1}{P_0} \geq A \quad (2.3)
\]

\[
\frac{P_1}{P_0} \leq B \quad (2.4)
\]

\[
B < \frac{P_1}{P_0} < A \quad (2.5)
\]

The approximate values of the constants \( A \) and \( B \) are taken as \( A \equiv \frac{1-\beta}{\alpha}, \quad B \equiv \frac{\beta}{1-\alpha} \)

Where ‘\( \alpha \)’ and ‘\( \beta \)’ are the risk probabilities as defined earlier. A simplified version of the above decision processes is to reject the system as unreliable if \( N(t) \) falls for the first time above the line

\[
N_0(t) = a t + b2 \quad (2.6)
\]

To accept the system to be reliable if \( N(t) \) falls for the first time below the line

\[
N_1(t) = a t - b1 \quad (2.7)
\]

To continue the test with one more observation on \([t, N(t)]\) as the random graph of \([t, N(t)]\) is between the two linear boundaries given by Eq. (2.6) and (2.7) where

\[
a = \frac{\lambda_1 - \lambda_0}{\log[\lambda_1/\lambda_0]} \quad (2.8)
\]

\[
b_1 = \frac{\log[1-\alpha]}{\log[\lambda_0/\lambda_1]} \quad (2.9)
\]

\[
b_2 = \frac{\log[1-\beta]}{\log[\lambda_0/\lambda_1]} \quad (2.10)
\]

The parameters \( a, \beta, \lambda_0 \) and \( \lambda_1 \), can be chosen in several ways. One way suggested by Stieber (1997) is

\[
\lambda_0 = \frac{\lambda \log(q)}{q - 1} \quad \lambda_1 = \frac{q \lambda \log q}{q - 1} \quad \text{where} \quad q = \frac{\lambda_1}{\lambda_0}
\]

If \( \lambda_0 \) and \( \lambda_1 \) are chosen in this way, the slope of \( N_0(t) \) and \( N_1(t) \) equals \( \lambda \). The other two ways of choosing \( \lambda_0 \) and \( \lambda_1 \) are from past projects (for a comparison of the projects) and from part of the data to compare the reliability of different functional areas(components).

### III Sequential Test for Software Reliability Growth Models

In Section 2, for the Poisson process we know that the expected value of \( N(t) = \lambda t \) called the average number of failures experienced in time ‘\( t \)’. This is also called the mean value function of the Poisson process. On the other hand if we consider a Poisson process with a general function (not necessarily linear) \( m(t) \) as its mean value function the probability equation of such a process is

\[
P[N(t) = Y] = \frac{[m(t)]^Y}{Y!} e^{-m(t)}, \quad Y = 0, 1, 2 \ldots
\]

Depending on the forms of \( m(t) \) we get various Poisson processes called NHPP, for our Burr type XII model. The mean value function is given as

\[
m(t) = a \left[ 1 - (1 + t)^{-b} \right], \quad t \geq 0
\]
We may write
\[ P_1 = \frac{e^{-m_1 t}[m_1 t]^N(t)}{N(t)!} \]
\[ P_0 = \frac{e^{-m_0 t}[m_0 t]^N(t)}{N(t)!} \]

Where \( m_1(t), m_0(t) \) are values of the mean value function at specified sets of its parameters indicating reliable software and unreliable software respectively. The mean value function \( m(t) \) contains the parameters \( a, b \) and \( c \). Let \( P_0, P_1 \) be values of the NHPP at two specifications of \( b \) say \( b_0, b_1 \) where \( b_0 < b_1 \) and two specifications of \( c \) say \( c_0, c_1 \) where \( c_0 < c_1 \). It can be shown that for our model \( m(t) \) at \( b_1 \) is greater than that at \( b_0 \) and \( m(t) \) at \( c_1 \) is greater than that at \( c_0 \). Symbolically \( m_0(t) < m_1(t) \). Then the SPRT procedure is as follows:

Accept the system to be Reliable if \( \frac{P_1}{P_0} \leq B \)

\[ i.e., \quad \frac{e^{-m_1 t}[m_1 t]^N(t)}{e^{-m_0 t}[m_0 t]^N(t)} \leq B \]

\[ i.e., \quad N(t) \leq \frac{\log(\frac{\beta}{1-\alpha}) + m_1(t)-m_0(t)}{\log m_1(t)-\log m_0(t)} \]  
(3.1)

Decide the system to be unreliable and Reject if \( \frac{P_1}{P_0} \geq A \)

\[ i.e., \quad \frac{e^{-m_1 t}[m_1 t]^N(t)}{e^{-m_0 t}[m_0 t]^N(t)} \geq A \]

\[ i.e., \quad N(t) \geq \frac{\log(\frac{\beta}{1-\alpha}) + m_1(t)-m_0(t)}{\log m_1(t)-\log m_0(t)} \]  
(3.2)

Continue the test procedure as long as

\[ \frac{\log(\frac{\beta}{1-\alpha}) + m_1(t)-m_0(t)}{\log m_1(t)-\log m_0(t)} < N(t) < \]

\[ \frac{\log(\frac{1-\beta}{\alpha}) + m_1(t)-m_0(t)}{\log m_1(t)-\log m_0(t)} \]  
(3.3)

Substituting the appropriate expressions of the respective mean value function \( m(t) \), we get the respective decision rules and are given in followings lines.

Acceptance Region:

\[ \log \left( \frac{\beta}{1-\alpha} \right) + a \left[ (1+t^a)^{-h} - (1+t^a)^{-h} \right] \]

\[ \log a \left[ \frac{(1+t^a)^{-h}}{(1+t^a)^{-h}} \right] \]  
(3.4)

Rejection Region:

\[ \log \left( \frac{1-\beta}{\alpha} \right) + a \left[ (1+t^a)^{-h} - (1+t^a)^{-h} \right] \]

\[ \log a \left[ \frac{(1+t^a)^{-h}}{(1+t^a)^{-h}} \right] \]

Continuation Region:

\[ \log \left( \frac{\beta}{1-\alpha} \right) + a \left[ (1+t^a)^{-h} - (1+t^a)^{-h} \right] \]

\[ \log a \left[ \frac{(1+t^a)^{-h}}{(1+t^a)^{-h}} \right] \]

It may be noted that in the proposed model the decision rules are exclusively based on the strength of the sequential procedure \((\alpha, \beta)\) and the values of the respective mean value functions namely, \( m_0(t) \) , \( m_1(t) \). If the mean value function is linear in \( t \) passing through origin, that is, \( m(t) = at \) the decision rules become decision lines as described by (Stieber 1997). In that sense equations \( (3.1), (3.2) , (3.3) \) can be regarded as generalizations to the decision procedure of Stieber(1997).

The applications of these results for live software failure data are presented with analysis in Section 5.

IV MAXIMUM LIKELIHOOD ESTIMATION

In this section we develop expressions to estimate the parameters of the Burr type XII model based on interval domain data. Parameter estimation is of primary importance in software reliability prediction.

A set of failure data is usually collected in one of two common ways, time domain data and interval domain data. In this paper parameters are estimated from the interval domain data.

The mean value function of Burr type XII model is given by

\[ m(t) = a \left[ 1 - \left( 1 + t^c \right)^{-h} \right] \]  
(4.1)

In order to have an assessment of the software reliability, \( a, b \) and \( c \) are to be known or they are to be estimated from software failure data. Expressions are now delivered for estimating \( a \), \( b \) and \( c \) for the Burr type XII model.

Assuming the given data are given for the cumulative number of detected errors \( n_i \) in a given time interval \((0, t_i)\) where \( i=1,2, \ldots n \) and \( 0 < t_1 < t_2 < \ldots t_n \), then the logarithmic likelihood function (LLF) for interval domain data is given by
\[ \text{Log } L = \sum_{i=1}^{k} (n_i - n_{i-1}) \log \left[ m(t_i) - m(t_{i-1}) \right] - m(t_k) \] (4.2)

\[ \log L = \sum_{i=1}^{k} (n_i - n_{i-1}) \log \left\{ a \left[ 1 - \left( 1 + t_i^c \right)^{-b} \right] - a \left[ 1 - \left( 1 + t_{i-1}^c \right)^{-b} \right] \right\} \]

\[ \log L = \sum_{i=1}^{k} (n_i - n_{i-1}) \left\{ \log a + \log \left[ 1 + t_i^c \right]^{-b} - (1 + t_i^c)^{-b} \right\} - a + a(1 + t_i^c)^{-b} \] (4.3)

Taking the Partial derivative with respect to ‘a’ and equating to ‘0’.

\[ \frac{\partial \log L}{\partial a} = 0 \]

\[ a = \sum_{i=1}^{k} (n_i - n_{i-1}) \frac{(1 + t_i^c)^b}{(1 + t_k^c)^b - 1} \] (4.4)

The parameter ‘b’ is estimated by iterative Newton Raphson Method using

\[ b_{n+1} = b_n - \frac{g(b)}{g'(b)} \]

Where \( g(b) \) and \( g'(b) \) are expressed as follows.

\[ g(b) = \frac{\partial \log L}{\partial b} = 0 \]

\[ \frac{\partial \log L}{\partial b} = g(b) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ -\log(t_{i-1} + 1) - \log(t_i + 1) + \frac{(t_i + 1)^b \log(t_i + 1) - (t_{i-1} + 1)^b \log(t_{i-1} + 1)}{(t_i + 1)^b - (t_{i-1} + 1)^b} \right] \]

\[ + \left[ \frac{1}{(t_k + 1)^b - 1} \log \left( \frac{1}{1 + t_k} \right) \right] \] (4.5)

Again partial differentiating with respect to ‘b’ and equate to 0, we get

\[ g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0 \]

\[ \frac{\partial^2 \log L}{\partial b^2} = g'(b) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{2(t_{i-1} + 1)^b (t_i + 1)^b \log(t_i + 1) \log \left( \frac{t_{i-1} + 1}{t_i + 1} \right)}{(t_i + 1)^b - (t_{i-1} + 1)^b} \right] \]

\[ + \sum_{i=1}^{k} (n_i - n_{i-1}) \log(1 + t_i) \frac{(t_i + 1)^b \log(t_i + 1)}{[(t_i + 1)^b - 1]^2} \] (4.6)

The parameter ‘c’ is estimated by iterative Newton Raphson Method using

\[ c_{n+1} = c_n - \frac{g(c_n)}{g'(c_n)} \]

Where \( g(c) \) and \( g'(c) \) are expressed as follows.

\[ g(c) = \frac{\partial \log L}{\partial c} = 0 \]