Burr Type XII Software Reliability Growth Model

R. Satya Prasad, Ph.D
Associate Professor
Department of CSE
Acharya Nagarjuna University
Guntur, India

K.V. Murali Mohan
Associate Professor
Department of ECE
H.M. Institute of Tech. and Science
Hyderabad, India

G. Sridevi
Associate Professor
Department of CSE
Nimra Women’s College of Engg
Vijayawada, India

ABSTRACT
Software Reliability Growth model (SRGM) is a mathematical model of how the software reliability improves as faults are detected and repaired. The development of many SRGMs over the last several decades have resulted in the improvement of software facilitating many engineers and managers in tracking and measuring the growth of reliability. This paper proposes Burr type XII based Software Reliability growth model with time 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.

General Terms
Software failure data, Mean value function.

Keywords
Software Reliability, Burr type XII distribution, NHPP, ML Estimation

1. 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). SRGM is a mathematical model of how the software reliability improves as faults are detected and required (Quadri and Ahmad, 2010). 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. 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. 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). This paper presents a method for model validation.

2. 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:
\[
\lambda(t) = \frac{d(F(t))}{dt}.
\]
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 the case of the finite failure NHPP models is given by:
\[
\lambda(t) = aF'(t) \quad [8].
\]

2.1 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.
\[
m(t) = \begin{cases} 
0, & t = 0 \\
a, & t \to \infty 
\end{cases}
\]
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.,
\[
P(N(t) = n) = \frac{(m(t))^n e^{-m(t)}}{n!}, \quad n = 0,1,2 \ldots \infty
\]

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 \( m(t) \).

### 2.2. Proposed Model Description

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

The Cumulative distribution function (CDF) is given by
\[
m(t) = \int_0^t \lambda(t) dt = a \left[ 1 - (1 + t^c)^{-b} \right] \quad (2.2.1)
\]

The Probability Density Function (PDF) of Burr XII distribution is given, respectively by
\[
\lambda(t) = a \left( \frac{c b t^{b-1}}{(1 + t^c)^{b+1}} \right) = a f(t)
\]

Where \( t \geq 0, a \geq 0, b \geq 0 \) 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.

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

We conduct an experiment and obtain \( N \) independent observations \( I_1, I_2, \ldots, I_n \). The likelihood function for time domain data [4] is given by

\[
L = \prod_{i=1}^{N} \frac{abct_i^{b-1}}{(1 + t_i^c)^{b+1}} e^{-a} \left[ 1 - (1 + t_i^c)^{-b} \right]
\]

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

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

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

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

Let \( N(t) \) be the number of errors remaining in the system at time ‘t’.

\[
N(t) = N(x) - N(t)
\]

\[
E[N(t)] = E[N(x)] - E[N(t)]
\]

\[
= a - m(t)
\]

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

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

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

\[
i.e., \quad P \left[ S_k > \frac{x}{X_{k-1}} \right] = \frac{x}{a m(x)}
\]

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

This Expression is called Software Reliability.

### 3. Illustrating the MLE

#### 3.1. Parameter Estimation based on Time Domain Data

In this section we develop expressions to estimate the parameters of the Burr type XII model based on time 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 time domain data. In this paper parameters are estimated from the time domain data.

The mean value function of Burr type XII model is given by
\[
m(t) = a \left[ 1 - (1 + t^c)^{-b} \right], \quad t \geq 0
\]

\[
(3.1.1)
\]
\[ \frac{\partial \log L}{\partial a} = 0 \]

\[ \therefore a = \frac{n(1 + t^c)^b}{(1 + t^c)^b - 1} \]  

(3.1.3)

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

\[ b_{n+1} = b_n - \frac{g(b)}{g'(b)} \quad \text{Where } g(b) \text{ and } g'(b) \text{ are expressed as follows.} \]

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

\[ \frac{\partial \log L}{\partial b} = g(b) = \frac{n \log \left( \frac{1}{t + 1} \right)}{(1 + t^c)^b - 1} + n - \sum_{i=1}^{n} \log(t_i + 1) \]  

(3.1.4)

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

\[ \frac{\partial^2 \log L}{\partial b^2} = g'(b) = -n \left[ \log \left( \frac{1}{t + 1} \right) \left( \log(t + 1) b \log(t + 1) \right) + \frac{1}{b^2} \right] \]  

(3.1.5)

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

\[ c_{n+1} = c_n - \frac{g(c)}{g'(c)} \quad \text{Where } g(c) \text{ and } g'(c) \text{ are expressed as follows.} \]

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

\[ \frac{\partial \log L}{\partial c} = g(c) = -n \frac{\log(t)}{(1 + t^c)} + n \frac{2 \log(t)}{c (1 + t^c)} + \sum_{i=1}^{n} \log(t_i) \]  

(3.1.6)

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

\[ \frac{\partial^2 \log L}{\partial c^2} = g'(c) = \frac{n t^c \log t}{(1 + t^c)^2} \log t - \frac{n}{c^2} - 2 \log t \sum_{i=1}^{n} t_i \left( \frac{1}{(1 + t_i^c)^2} \right) \]  

(3.1.7)

4. DATA ANALYSIS

The set of software errors analyzed here is borrowed from software development project as published in Pham (2006) [8].

Table 1: Naval Tactical Data System Software Dataset

<table>
<thead>
<tr>
<th>Failure Number (n)</th>
<th>Time Between Failures (S_n) days</th>
<th>Cumulative Time (S_{n})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Solving equations by Newton Raphson Method for the NTDS test data, the iterative solutions for MLEs of \( a \), \( b \) and \( c \) are

\[
\begin{align*}
\hat{a} &= 26.105273 \\
\hat{b} &= 0.998899 \\
\hat{c} &= 0.998903
\end{align*}
\]

Table 2: Parameters Estimated through MLE

<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>NTDS</td>
<td>26</td>
<td>26.105273</td>
</tr>
<tr>
<td>Xie</td>
<td>30</td>
<td>30.040800</td>
</tr>
<tr>
<td>AT &amp; T</td>
<td>22</td>
<td>22.032465</td>
</tr>
<tr>
<td>IBM</td>
<td>15</td>
<td>15.051045</td>
</tr>
<tr>
<td>SONATA</td>
<td>30</td>
<td>30.016391</td>
</tr>
</tbody>
</table>

Hence, we may accept these three values as MLSs of \( a, b, c \). The estimator of the reliability function from the equation (2.2.2) at any time \( x \) beyond 250 hours is given by

\[
R_S(x) / X_k = e^{-[m(x+s)-m(s)]} \\
R_S(x)/X_k(250/50) = e^{-[m(50+250)-m(250)]} \\
= e^{-m(300)-m(250)} \\
= 0.98269972
\]

5. 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 considered model fits more to the dataset whose Log Likelihood is most negative. The application of the considered distribution function and its Log Likelihood on different datasets collected from real world failure data is given below.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Log L (MLE)</th>
<th>Reliability (( t + x ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTDS</td>
<td>-38.477204</td>
<td>0.98269972</td>
</tr>
<tr>
<td>Xie</td>
<td>-44.402007</td>
<td>0.99616300</td>
</tr>
<tr>
<td>AT &amp; T</td>
<td>-18.949300</td>
<td>0.99573348</td>
</tr>
<tr>
<td>IBM</td>
<td>-27.365034</td>
<td>0.98841341</td>
</tr>
<tr>
<td>SONATA</td>
<td>-46.991596</td>
<td>0.99700207</td>
</tr>
</tbody>
</table>

6. CONCLUSION

To validate the proposed approach, the parameter estimation is carried out on the data sets collected from (Xie et al., 2002; Pham, 2006; Ashoka, 2010). Out of the data sets that were collected, the model under consideration best fits the data of SONATA using MLE approach. Since, it is having the highest negative value for the log likelihood. The reliability of all the data sets are given in Table 3. The reliability of the model over SONATA data is high among the data sets which were considered.

7. REFERENCES


Monitoring Burr Type XII Software Quality Using SPC

Dr.R.Satya Prasad¹, K.V.Murali Mohan² and G.Sridevi³

¹Assoc.Prof., Dept. of CSE, Acharya Nagarjuna University, India
²Assoc.Prof., Dept. of CSE, Holy Mary Institute of Technology and Science, India
³Assoc.Prof., Dept. of CSE, Nimra Women’s College of Engineering, India.
profrsp@gmail.com¹, kvmuralimohan22@gmail.com², sridevi.gutta2012@gmail.com³

Abstract

Software reliability in the software development process is an important issue. The quality of a Software Process can be monitored efficiently using Statistical Process Control (SPC). SPC is a method of process management through application of statistical analysis, which involves and includes the defining, measuring, controlling and improving of the process. With the help of SPC the software development team can identify software failure process and find out actions to be taken which assures better software reliability. This paper provides a control mechanism based on the cumulative quantity between observations of time domain failure data using mean value function of Burr type XII distribution, which is based on Non-Homogenous Poisson Process (NHPP). The Maximum Likelihood Estimation (MLE) method is used for parameter estimation of the proposed model.

Keywords: Software quality, SPC, NHPP, Burr type XII distribution, Time Domain data, ML Estimation, Control Charts.

Introduction

Software reliability assessment is important to evaluate and predict the reliability and performance of software system, since it is the main attribute of software. To identify and eliminate human errors in software development process and also to improve software quality, 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 quality 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 the quality of the final product which 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.”

The control charts can be classified into several categories, as per several distinct criteria. Depending on the number of quality characteristics under investigation, charts can be divided into univariate control charts and multivariate control charts. Furthermore, the quality characteristic of interest may be a continuous random variable or alternatively a discrete attribute. Control charts should be capable to create an alarm when a shift in the level of one or more parameters of the underlying distribution or a non-random behavior occurs. Normally, such a situation will be reflected in the control chart by failure 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.

SPC is a powerful tool to optimize the amount of information needed for use in making management decisions. Statistical techniques provide an understanding of the business baselines, insights for process improvements, communication of value and results of processes, and active and visible involvement. SPC provides real time analysis to establish controllable process baselines; learn, set, and dynamically improves process capabilities; and focus business areas which need improvement. The early detection of software failures will improve the software reliability. 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]. Many factors influence the process, resulting in variability. The causes of process variability can be broadly classified into two categories, viz., assignable causes and chance causes.

The control limits can then be utilized to monitor the failure times of components. After each failure, the time can be plotted on the chart. If the plotted point falls between the calculated control limits, it indicates that the process is in the state of statistical control and no action is warranted. If the point falls above the UCL, it indicates that the process average, or the failure occurrence rate, may have decreased which results in an increase in the time between failures. This is an important indication of possible process improvement. If this happens, the management should look for possible causes for this improvement and if the causes are discovered then action should be taken to maintain them. If the plotted point falls below the LCL, it indicates that the process average, or the failure occurrence rate, may have increased which results in a decrease in the failure time. This means that process may have deteriorated and thus actions should be taken to identify and the causes may be removed. It can be noted here that the parameters a, b and c should normally be estimated with the data from the failure process.
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 [9].

**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].

**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.

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

Where ‘a’ is the expected number of software errors to be eventually detected.
Suppose \( N(t) \) is known ppgs \( m(t) \) i.e.,

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

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 \( m(t) \).

**Proposed Model Description**

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

The Cumulative distributive function (CDF) is given by

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

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

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

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.

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

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

This is also a Poisson model with mean ‘\( a \)’.

Let \( N(t) \) be the number of errors remaining in the system at time ‘\( t \)’.

\[
N(t) = N(\infty) - N(t)
\]

\[
E[N(t)] = E[N(\infty)] - E[N(t)]
\]

\[
= a - m(t)
\]

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

\[
= a(1 + t^c)^{-b}
\]
Illustrating the MLE Method

Parameter Estimation based on Time Domain Data

In this section we develop expressions to estimate the parameters of the Burr type XII model based on time 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 time domain data. In this paper parameters are estimated from the time domain data.

The mean value function of Burr type XII model is given by
\[ m(t) = a \left[ 1 - \left(1 + t^c \right)^{-b} \right], \quad t \geq 0 \] (3.1.1)

We conduct an experiment and obtain N independent observations \( t_1, t_2, ..., t_n \). The likelihood function for time domain data [5] is given by
\[ L = \prod_{i=1}^{N} a b c \left( \frac{t_i}{1 + t_i^c} \right)^{b-1} e^{-\frac{a b c}{1 - (1 + t_i^c)^b}} \]

\[ \log L = -a + a (1 + t_i^c)^{-b} + \sum_{i=1}^{n} \left[ \log a + \log b + \log c + (c-1) \log t_i - (b+1) \log (1 + t_i^c) \right] \] (3.1.2)

Taking the Partial derivative with respect to ‘a’ and equating to ‘0’.
\[ \frac{\partial \log L}{\partial a} = 0 \]
\[ \therefore a = \frac{n (1 + t_i^c)^b}{(1 + t_i^c)^b - 1} \] (3.1.3)

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

\[ \frac{\partial^2 \log L}{\partial b^2} = g'(b) = -n \left[ \log \left( \frac{1}{1 + t_i^c} \right) \left( \frac{(t_i^c)^b \log(t_i^c)}{(t_i^c)^b - 1} \right) + \frac{1}{b^2} \right] \] (3.1.4)

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.
\[
g(c) = \frac{\partial \log L}{\partial c} = 0
\]
\[
\frac{\partial \log L}{\partial c} = g(c) = \frac{-n}{(1+r^c)} \log(t) + \frac{n}{c} \sum_{i=1}^{n} 2 \log(t) \frac{t^c_i}{(1+t^c_i)} + \sum_{i=1}^{n} \log t_i \quad (3.1.6)
\]
\[
g'(c) = \frac{\partial^2 \log L}{\partial c^2} = 0
\]
\[
\frac{\partial^2 \log L}{\partial c^2} = g'(c) = \frac{n t^c}{(1+r^c)^2} \log t - \frac{n}{c^2} - 2 \log t \sum_{i=1}^{n} t^c_i \log t_i \left\{ \frac{1}{(1+t^c_i)^2} \right\} \quad (3.1.7)
\]

**Distribution of Time between failures**

The set of software errors analyzed here is borrowed from software development project as published in Pham (2005) [8]. Based on the inter failure data given in Table 1, we compute the software failures process through Mean Value Control chart. We used cumulative time between failures data for software reliability monitoring using Burr Type XII distribution. The use of cumulative number of failures is a different and new approach, which is of particular advantage in reliability.

<table>
<thead>
<tr>
<th>Table 1: Naval Tactical Data System (NTDS) Software Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Failure Number (n)</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>21</td>
</tr>
<tr>
<td>22</td>
</tr>
</tbody>
</table>
Solving equations by Newton Raphson Method for the NTDS test data, the iterative solutions for MLEs of $a$, $b$ and $c$ are

\[
\begin{align*}
    a &= 26.105273 \\
    b &= 0.998899 \\
    c &= 0.998903 
\end{align*}
\]

Using ‘$a$’ and ‘$b$’ and ‘$c$’ values, we compute $m(t)$. Now the control limits are calculated by the following equations taking the standard values 0.00135, 0.99865 and 0.5.

**Table 2**: Successive differences of Cumulative mean values

<table>
<thead>
<tr>
<th>Failure number (n)</th>
<th>Cumulative failures ($S_n$)</th>
<th>Mean values $m(t)$</th>
<th>Successive differences of $m(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>23.48244280</td>
<td>1.428384531</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>24.91082711</td>
<td>0.397396116</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>25.30822323</td>
<td>0.085979850</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>25.39420308</td>
<td>0.112886763</td>
</tr>
<tr>
<td>5</td>
<td>43</td>
<td>25.50708984</td>
<td><strong>0.025949741</strong></td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>25.53303958</td>
<td>0.055979652</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>25.58901923</td>
<td>0.069852456</td>
</tr>
<tr>
<td>8</td>
<td>58</td>
<td>25.65887169</td>
<td>0.034799104</td>
</tr>
<tr>
<td>9</td>
<td>63</td>
<td>25.69367079</td>
<td>0.040493258</td>
</tr>
<tr>
<td>10</td>
<td>70</td>
<td>25.73416405</td>
<td>0.005142713</td>
</tr>
<tr>
<td>11</td>
<td>71</td>
<td>25.73930676</td>
<td>0.028090139</td>
</tr>
<tr>
<td>12</td>
<td>77</td>
<td>25.76739690</td>
<td>0.004267318</td>
</tr>
</tbody>
</table>
Assuming an acceptable probability of false alarm of 0.27% the control limits can be obtained as [10]:

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

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

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

These limits are converted to \( m(t_u), m(t_c) \) and \( m(t_l) \) form. They are used to find whether the software process is in control or not by placing the points in Mean value chart shown in figure 1.

![Mean Value Control Chart](image_url)
A point below the control limit $m(t_i)$ indicates an alarming signal. A point above the control limit $m(t_i)$ indicates better quality. If the points are falling within the control limits it indicates the software process is in stable condition [11].

$$m(t_i) = 26.0697749$$
$$m(t_i) = 0.03549806$$
$$m(t_i) = 13.0526256$$

By placing the failure cumulative data shown in table 2 on y axis and time between failures on x axis and the values of control limits are placed on Mean Value chart, we obtained Figure 1. The Mean Value Chart shows that the 5th failure data has fallen below $m(T_i)$ which indicates the failure process is identified. It is significantly early detection of failures using Mean Value chart.

**Conclusion**

In this paper, Burr type XII software reliability growth model with SPC is proposed. By observing the Mean Value control chart we have identified that the failure situation is detected at 5th point of Table 2. i.e., failure data has fallen below LCL. Hence our proposed Mean Value Chart detects out of control situation. This is a simple method for model validation and is very convenient for practitioners of software reliability. The methodology adopted in this paper is better than the methodology adopted by Xie et al. [2002] control chart [10] . Therefore; we may conclude that this model is the best choice for an early detection of software failures.

**Acknowledgements**

Our thanks to Department of Computer Science and Engineering, Nimra Women’s College of Engineering, Vijayawada, for providing necessary facilities to carry out the research work.

**References**


sridevi gutta:

We have reached a decision regarding your submission to Indian Journal of Science and Technology, "Detection of Burr Type XII Reliable Software using SPRT".

Our decision is to:
"Accept"

Dr. Natarajan Gajendran
Indian Journal Science and Technology
Phone +91-9940444695
indjst@gmail.com

Editor
Fee (Payment can be made by NEFT)
INR 7500 + 14% (Service Tax) (i.e., 8550 INR)

Name (Payable) to: Indian Society for Education and Environment,
A/C No: 16350100005111
A/C Type: Saving account,
Name of the Bank: Indian Overseas Bank,
IFSC Code: IOBA0001635,
Branch code: 001635
MICR code: 600020120
Branch address: 58 4th MAIN ROAD,GANDHI NAGAR, ADYAR,CHENNAI – 600020,
India
City: Chennai
Country: India
Bank phone: +91-044-28519488

PI inform us the UTR number; date of payment and article number.
The fee paid details PI be emailed to: indjst@gmail.com

With best regards
NG

Indian Journal of Science and Technology
http://www.indjst.org
Detection of Burr Type XII Reliable Software using Sequential Process Ratio Test

K.V.Murali Mohan¹, Dr.R.Satya Prasad ², Dr.G.Sridevi³*

¹ECE, H.M. Institute of Tech. & Science, Hyderabad, India; kvmuralimohan22@gmail.com
² CSE, Acharya Nagarjuna University, Andhra Pradesh, India; profrsp@gmail.com
³ CSE, KL University, Guntur, India; sridevi.gutta2012@gmail.com

Abstract

Background/Objectives: There is a need for the people to have the tools/mechanism to detect whether the software is reliable or not. Several methods came into existence to assess the software reliability. It takes more time to come to a decision in Classical Hypothesis testing because the conclusions can be drawn only after collecting the large amounts of data.

Methods/Statistical analysis: By adopting Sequential Analysis of Statistical science, it can be decided very quickly whether the software which is developed is reliable or unreliable. In this paper, we proposed a new type of statistical science procedure, Sequential Probability Ratio Test (SPRT) applied for Burr Type XII model based on Time domain data.

Findings: For the proposed Burr type XII model, we applied the SPRT methodology on five real time software failure datasets that were borrowed from different software services. The result exemplifies that the model has given a decision of rejection for all the datasets. Therefore our findings state that all used datasets are unreliable.

Application/Improvements: Applying SPRT procedure on datasets, we can come to an early conclusion of reliable/unreliable software.

Keywords: Burr type XII model, Sequential Probability Ratio Test, ML Estimation, NHPP, Time Domain Data.

1. Introduction

Wald's procedure is particularly relevant if the data is collected sequentially. Sequential Analysis is different from Classical Hypothesis. With classical hypothesis, first the entire data need to be collected and then the analysis is done to attain the conclusions. But where as in Sequential Analysis, each and every test case is analyzed soon after the data has been collected, and also the results are compared with some threshold value incorporating the new information obtained with the current test case. This permits one to come up with the conclusions during the data collection itself, so that the final decision may be made at much earlier stage. Wald’s procedure is well suited if the data is collected sequentially. The main advantage of Sequential Analysis is that the decisions can be taken at an earlier time saving the human time and also in terms of money.

Sequential Probability Ratio Test (SPRT) is usually applied at the circumstances where we need to take decision between two simple hypotheses or a single decision point. The SPRT procedure can be used to distinguish the software under test into one of the two categories like reliable/unreliable, pass/fail and certified/uncertified. The software failure data
analysis can be done either by considering the time between failures or failure count in a specific time interval. Also it is assumed that the average number of recorded failures in a given time interval is directly proportional to the length of the interval. The random number of failure occurrences in a given interval can be explained by the Poisson process.

\[
P[N(t) = n] = \frac{e^{-\lambda t}(\lambda t)^n}{n!} \quad (1.1)
\]

As per the observations made by the\(^5\), the reliability predictions are misleading by applying software reliability growth models in classical hypotheses testing. According to the observations made by him, the statistical methods can be successfully applied to the failure data. His observations are demonstrated by applying the well-known sequential probability ratio test (SPRT) of\(^4\) for a software failure data to detect unreliable software components and compare the reliability of different software versions. In this paper, we consider one of the popular software reliability growth model Burr Type XII and adopt the principle of\(^5\) in detecting whether the software is reliable or unreliable in order to accept or reject the developed software. The theory proposed by\(^4\) is described in section 2 Implementation of SPRT for the proposed Burr type XII Software Reliability Growth Model is illustrated in section 3. Maximum Likelihood estimation method is used to estimate the parameters is presented in Section 4. Application of the decision rule to detect the unreliable software with reference to the Software Reliability Growth Model Burr Type XII is depicted in section 5.

2. Wald’s Sequential Test for a Poisson Process

The Sequential Probability Ratio Test (SPRT) was developed by Abraham Wald at Columbia University in 1943\(^4\). The SPRT procedure is used for quality control studies during the manufacturing of software products. The tests can be performed with fewer observations as compared to fixed sample size sets. Testing is performed on large volumes of data which consumes large amount of time in classical hypothesis and it can be reduced to a large extent by implementing Sequential Probability Ratio Tests. The SPRT methodology for Homogeneous Poisson Process given by Stieber is described below\(^5\).

Let \(\{N(t), t \geq 0\}\) be a homogeneous Poisson process with rate \(\lambda\). In this case, \(N(t)\) = number of failures up to time ‘t’ and \(\lambda\) is the failure rate (failures per unit time). If the system is put on test and that if 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 the data suggest that the failure rate is larger than \(\lambda_1\) and accept it with a high probability, if it is smaller than \(\lambda_0\). Here we have to specify two (small) numbers ‘\(\alpha\)’ and ‘\(\beta\)’, where ‘\(\alpha\)’ is the probability of falsely accepting the system. That is accepting the system even if \(\lambda \geq \lambda_1\). This is the “producer’s” risk. ‘\(\beta\)’ is the probability of falsely accepting the system. That is accepting the system even if \(\lambda \geq \lambda_1\). This is the “consumer’s” risk. Wald’s classical SPRT is very sensitive to the choice of relative risk required in the specification of the alternative hypothesis. With the classical SPRT, tests are performed continuously at every time point \(t > 0\) as additional data are collected. 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 \(\lambda_0, \lambda_1\) as the failure rates are respectively given by

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

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

The ratio \(\frac{P_1}{P_0}\) at any time ‘\(t\)’ 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

\[ \frac{p_1}{p_0} = \exp(\lambda_0 - \lambda_1) t + \left( \frac{\lambda_1}{\lambda_0} \right)^{N(t)} \]

The decision rule of SPRT is to decide in favour of \( \lambda_1 \) or \( \lambda_0 \) or to continue by observing the number of failures at a later time than \( t \) 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 the test process with one more observation in failure data, according to

\[ \frac{p_1}{p_0} \geq A \quad (2.3) \]

\[ \frac{p_1}{p_0} \leq B \quad (2.4) \]

\[ B < \frac{p_1}{p_0} < A \quad (2.5) \]

The approximate values of the constants A and B are taken as

\[ A \approx \frac{1 - \beta}{\alpha}, \quad B \approx \frac{\beta}{1 - \alpha} \]

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

\[ N_U(t) = at + b_2 \quad (2.6) \]

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

\[ N_L(t) = at - b_1 \quad (2.7) \]

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 equations (2.6) and (2.7) where

\[ a = \frac{\lambda_1 - \lambda_0}{\log \left( \frac{\lambda_1}{\lambda_0} \right)} \]

\[ b_1 = \frac{\log \left( \frac{1 - \alpha}{\beta} \right)}{\log \left( \frac{\lambda_1}{\lambda_0} \right)} \]

\[ b_2 = \frac{\log \left( \frac{1 - \beta}{\alpha} \right)}{\log \left( \frac{\lambda_1}{\lambda_0} \right)} \]

The parameters \( \alpha, \beta, \lambda_0 \) and \( \lambda_1 \) can be chosen in several ways. One way suggested by \( \lambda_0 = \frac{\lambda \log(q)}{q - 1} \)

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

If \( \lambda_0 \) and \( \lambda_1 \) are chosen in this way, the slope of \( N_U(t) \) and \( N_L(t) \) 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 or components.

3. Sequential Probability Ratio Test for Burr Type XII SRGM

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

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

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

\[ m(t) = a \left[ 1 - (1 + t^b)^{-b} \right], \quad t \geq 0 \]
It can also be written as

\[
P_0 = \frac{e^{-m_0 t} [m_0(t)]^{N(t)}}{N(t)!}
\]

\[
P_1 = \frac{e^{-m_1 t} [m_1(t)]^{N(t)}}{N(t)!}
\]

Where \(m_1(t), m_0(t)\) represents the mean value function at stated parameters indicating reliable software and unreliable software respectively. The mean value function \(m(t)\) comprises the parameters ‘a’, ‘b’ and ‘c’. The two specifications of NHPP for \(b\) are considered as \(b_0, b_1\) where \((b_0 < b_1)\) and two specifications of \(c\) say \(c_0, c_1\) where \((c_0 < c_1)\). For our proposed model, \(m(t)\) at \(b_1\) is said to be greater than \(b_0\) and \(m(t)\) at \(c_1\) is said to be greater than \(c_0\). The same can be denoted symbolically as \(m_0(t) < m_1(t)\). The implementation of SPRT procedure is illustrated below.

System is said to be reliable and can be accepted if

\[
\frac{P_1}{P_0} \leq B
\]

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

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

System is said to be unreliable and rejected if

\[
\frac{P_1}{P_0} \geq A
\]

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

\[i.e., \quad N(t) \geq \frac{\log \left( \frac{1-\beta}{\alpha} \right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} \tag{3.2}\]
Continue the test procedure as long as

$$\frac{\log \left( \frac{\beta}{1-\alpha} \right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)} < N(t) < \frac{\log \left( \frac{1-\beta}{\alpha} \right) + m_1(t) - m_0(t)}{\log m_1(t) - \log m_0(t)}$$

(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:

$$N(t) \leq \frac{\log \left( \frac{\beta}{1-\alpha} \right) + a \left[ (1+t^n)^{-\beta_h} - (1+t^n)^{-\beta_h} \right]}{\log a \left[ (1+t^n)^{-\beta_h} \right]}$$

(3.4)

Rejection Region:

$$N(t) \geq \frac{\log \left( \frac{1-\beta}{\alpha} \right) + a \left[ (1+t^n)^{-\beta_h} - (1+t^n)^{-\beta_h} \right]}{\log a \left[ (1+t^n)^{-\beta_h} \right]}$$

(3.5)

Continuation Region:

$$\frac{\log \left( \frac{\beta}{1-\alpha} \right) + a \left[ (1+t^n)^{-\beta_h} - (1+t^n)^{-\beta_h} \right]}{\log a \left[ (1+t^n)^{-\beta_h} \right]} < N(t) < \frac{\log \left( \frac{1-\beta}{\alpha} \right) + a \left[ (1+t^n)^{-\beta_h} - (1+t^n)^{-\beta_h} \right]}{\log a \left[ (1+t^n)^{-\beta_h} \right]}$$

(3.6)

For the specified model, it may be observed that the decision rules are exclusively based on the strength of the sequential procedure $(\alpha, \beta)$ and the value of the mean value functions namely $m_0(t)$ and $m_1(t)$. As described by $^5$, these decision rules become decision lines if the mean value function is linear in '$t$' passing through origin, that is $m(t) = \lambda t$. The equations (3.1), (3.2) and (3.3) are considered as generalizations for the decision procedure of $^6$. SPRT procedure is applied on live software failure data sets and the results that were analyzed are illustrated in Section 5.
4. Parameter Estimation

In this section we develop expressions to estimate the parameters of the Burr type XII model based on time domain data. Parameter estimation is very significant in software reliability prediction. Once the analytical solution form is known for a given model, parameter estimation is achieved by applying a well-known estimation, Maximum Likelihood Estimation (MLE) \(^2\).

The main idea behind Maximum Likelihood parameter assessment is to decide the parameters that maximize the probability (likelihood) of the specimen data. In the other words, MLE methods are versatile and applicable to most models and for different types of data. In this paper parameters are estimated from the time domain data.

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

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

(4.1)

The parameters a, b, c are estimated with Maximum Likelihood (ML) estimation.

The likelihood function for time domain data is given by

\[ L = e^{-m(t)} \prod_{i=1}^{n} m(t_i) \]  

(4.2)

Substituting Eq. (4.1) in eq. (4.2) we get,

\[ L = e^{-m(t)} \prod_{i=1}^{n} \frac{abct_i^{-1}}{1 + t_i^{-b+1}} \]

LogL = \(-a + a(1 + t_i^c)^{-b} + \sum_{i=1}^{n} \left[ \log a + \log b + \log c + (c - 1) \log t_i - (b + 1) \log(1 + t_i^c) \right] \)

(4.3)

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

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

\[
\therefore a = \frac{n(1 + t_i^c)^b}{(1 + t_i^c)^b - 1} \quad (4.4)
\]

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

\[ b_{n+1} = b_n - \frac{g'(b)}{g(b)}, \text{ Where } g(b) \text{ and } g'(b) \text{ are expressed as follows.} \]

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

\[ \frac{\partial \log L}{\partial b} = g(b) = \frac{n \log \left( \frac{1}{t+1} \right)}{(t+1)^b - 1} + \frac{n}{b} - \sum_{i=1}^{n} \log(t_i + 1) \]  

(4.5)

\[ g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0 \]
\[ \frac{\partial^2 \text{Log} L}{\partial b^2} = g'(b) = -n \left[ \log \left( \frac{1}{t+1} \right) \left( t+1 \right)^b \log(t+1) \right] + \frac{1}{b^2} \]  

(4.6)

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

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

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

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

\[ \frac{\partial \text{Log} L}{\partial c} = g(c) = \frac{-n}{1+t^c} \log(t) + \frac{n}{c} - \sum_{i=1}^{n} 2 \log(t) \frac{t_i^c}{(1+t_i^c)} + \sum_{i=1}^{n} \log t_i \]

(4.7)

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

\[ \frac{\partial^2 \text{Log} L}{\partial c^2} = g'(c) = \frac{m^c \log t}{(1+t^c)^2} - \frac{n}{c^2} - 2 \log t \sum_{i=1}^{n} t_i^c \log t_i \left( \frac{1}{(1+t_i^c)^2} \right) \]

(4.8)

## 5. SPRT Analysis of Live Datasets

In this section, the SPRT methodology is applied on five different data sets that are borrowed from 2.7 and SONATA software services. The decisions are evaluated based on the considered mean value function. Based on the estimates of the parameter ‘b’ in each mean value function, we have chosen the specifications of \( b_0 = b - \delta \), \( b_1 = b + \delta \) and \( c_0 = c - \delta, c_1 = c + \delta \) and apply SPRT such that \( b_0 < b < b_1 \) and \( c_0 < c < c_1 \). Assuming the \( \delta \) value is 0.5, the choices are given in the following Table 1.

### Table 1. Estimates of \( a, b, c \) & specifications of \( b_0, b_1, c_0, c_1 \)

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Estimate of ‘a’</th>
<th>Estimate of ‘b’</th>
<th>( b_0 )</th>
<th>( b_1 )</th>
<th>Estimate of ‘c’</th>
<th>( c_0 )</th>
<th>( c_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xie</td>
<td>30.040800</td>
<td>0.999825</td>
<td>0.499825</td>
<td>1.499825</td>
<td>0.999619</td>
<td>0.499619</td>
<td>1.499619</td>
</tr>
<tr>
<td>AT &amp; T</td>
<td>22.032465</td>
<td>0.999859</td>
<td>0.499859</td>
<td>1.499859</td>
<td>0.999611</td>
<td>0.499611</td>
<td>1.499611</td>
</tr>
<tr>
<td>IBM</td>
<td>15.051045</td>
<td>0.999530</td>
<td>0.499530</td>
<td>1.499530</td>
<td>0.999196</td>
<td>0.499196</td>
<td>1.499196</td>
</tr>
<tr>
<td>NTDS</td>
<td>26.105273</td>
<td>0.998899</td>
<td>0.498899</td>
<td>1.498899</td>
<td>0.998903</td>
<td>0.498903</td>
<td>1.498903</td>
</tr>
<tr>
<td>SONATA</td>
<td>30.016391</td>
<td>0.999958</td>
<td>0.499958</td>
<td>1.499958</td>
<td>0.999920</td>
<td>0.499920</td>
<td>1.499920</td>
</tr>
</tbody>
</table>
Using the specification \( b_0, b_1 \) and \( c_0, c_1 \) the mean value functions \( m_0(t) \) and \( m_1(t) \) are computed for each ‘t’. Later the decisions are made based on the decision rules specified by the equations (3.4), (3.5) for the data sets. At each ‘t’ of the data set, the strengths \((\alpha, \beta)\) are considered as \((0.05, 0.2)\). SPRT procedure is applied on five different data sets and the necessary calculations are given in the Table 2.

**Table 2. SPRT Analysis for 5 data sets**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>T</th>
<th>N(t)</th>
<th>R.H.S. of equation (3.4) Acceptance region ((\leq))</th>
<th>R.H.S. of equation (3.5) Rejection region ((\geq))</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xie</td>
<td>30.02</td>
<td>1</td>
<td>0.003606</td>
<td>0.005130</td>
<td>Rejection</td>
</tr>
<tr>
<td>AT &amp; T</td>
<td>5.50</td>
<td>1</td>
<td>0.115010</td>
<td>0.164486</td>
<td>Rejection</td>
</tr>
<tr>
<td>IBM</td>
<td>10</td>
<td>1</td>
<td>0.023281</td>
<td>0.040821</td>
<td>Rejection</td>
</tr>
<tr>
<td>NTDS</td>
<td>9</td>
<td>1</td>
<td>0.047184</td>
<td>0.065189</td>
<td>Rejection</td>
</tr>
<tr>
<td>SONATA</td>
<td>52.50</td>
<td>1</td>
<td>0.0010008</td>
<td>0.001499</td>
<td>Rejection</td>
</tr>
</tbody>
</table>

**6. Conclusion**

The SPRT methodology for the proposed software reliability growth model Burr type XII is applied for the software failure data sets. Hence, it is observed that we are able to come to an conclusion in less time regarding the reliability or unreliability of a software product. The results exemplifies that the model has given a decision of rejection for all the data sets at various time instant of the data. Therefore, we may conclude that, applying SPRT on data sets we can come to an early conclusion of reliable / unreliable of software.

**7. References**


