APPENDIX
Assessing Software Reliability using Inter Failures Time Data

Dr. R Satya Prasad
Associate Prof.,
Dept. of Computer Science & Eng.,
Acharya Nagarjuna University

Bandla Srinivasa Rao
Associate Prof.,
Dept. of Computer Science
VRS & YRN College

Dr. R.R. L Kantham
Professor,
Dept. of Statics
Acharyan Nagarjuna University

ABSTRACT
For critical business applications continuous availability is a requirement. Software reliability is an important component of continuous application availability. A single software defect can cause a system failure. To avoid these failures, reliable software is required. Due to schedule pressure, resource limitations, and unrealistic requirements in software development process, developing reliable software is difficult. To monitor software process variations and to improve reliability, the statistical Process Control (SPC) can be applied to software development process. SPC is a methodology that aims to provide process control in statistical terms. Control charts are the most common tools for determining whether a software process is under statistically control or not. In this paper we proposed a control mechanism, based on time between failures observations using exponential distribution, which is based on Non Homogeneous Poisson Process (NHPP).

General Terms
Control Chart, SQC, Probability limits, pdf, cdf, Time between failures

Keywords
Software reliability, Statistical Process Control (SPC), NHPP, MLE, Probability limits, Exponential Distribution

1. INTRODUCTION
Software reliability is the probability of failure free operation of a software in a specified environment during specified duration [Musa, 1998]. Statistical Process Control (SPC) is known to be a powerful tool to improve process, to enhance quality and productivity [Florac, 1999]. One of the possible measures for software reliability is the use of mean time between failures (MTBF) data. As a preliminary study for applying SPC, we tried on time between failures data [Xie, 2002] to predict software reliability using some control chart mechanism [Florac, 1999].

2. BACK GROUND THEORY
This section presents the theory that underlines Goel-Okumoto NHPP exponential model, and Maximum Likelihood Estimation to time domain, for ungrouped 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 can be written as:

\[
m(t) = aF(t)
\]

Where \( F(t) \) is a cumulative distribution function. The failure intensity function is given by [Swapna et al., 1998]:

\[
\lambda(t) = aF'(t)
\]

2.1. NHPP Model
The Non Homogeneous Poisson Process (NHPP) based software reliability growth models are proved to be quite successful in practical software reliability engineering [Musa et al., 1987]. 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. The parameters can be estimated by using Maximum Likelihood Estimate (MLE) based on various assumptions. Like 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. (Ohba, 1984, Pham, 1993). The present paper deals with Goel-Okumoto model applied on Inter failure times data [Xie et al., 2002] which is of time domain.

Let \( \{N(t), t \geq 0 \} \) be the cumulative number of software failures by time \( t \). \( m(t) \) is the mean value function, representing the expected number of software failures by time \( t \). \( \lambda(t) \) is failure intensity function, which is proportional to the residual fault content [Goel-Okumoto, 1979]. Thus

\[
m(t) = a(1 - e^{-bt}) \quad \text{a>0, b>0, t>0} \\
\lambda(t) = \frac{dm(t)}{dt} = b(a - m(t))
\]

Here \( a \) denotes the initial fault contained in a program and \( b \) represents the fault detection rate. In software reliability, the initial number of faults and faults detection rate are always unknown.

3. PARAMETER ESTIMATION USING MAXIMUM LIKELIHOOD ESTIMATION
Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for \( m(t) \) is known for a given model, the parameter in the solution needs to be determined. Parameter estimation is achieved by applying a technique of Maximum Likelihood Estimate (MLE) using Goel-Okumoto model. The MLE is consistent and asymptotically normally distributed as the sample size increases (Zhao, 1996). To estimate \( a \) and \( b \), for a sample of \( n \) units, first obtain the likelihood function (L): 

\[
L = e^{-m(s_n)} \prod_{k=1}^{n} m'(s_k)
\]

(3.1)
To solve equation (3.1), we take the logarithm of both sides

$$\log L = -m(s_n) + \sum_{k=1}^{n} \log m(s_k) \quad (3.2)$$

In order to estimate the parameters ‘a’ and ‘b’, we can take the derivative of the above equation (3.2) with respect to ‘a’ and ‘b’, and equating these derivatives to zero and solving the resulting equations for ‘a’ and ‘b’, we find the estimates as follows.

$$i.e. \frac{\partial \log L}{\partial a_n} = 0, \frac{\partial \log L}{\partial b} = 0 \quad (3.3)$$

$$g(b) = \frac{n}{b^2} - \frac{n}{b^2} + \frac{n}{b^2} + \frac{b^2}{(1-e^{-bn})} \quad (3.4)$$

$$g'(b) = \frac{n}{b^2} - \frac{n}{b^2} + \frac{n}{b^2} + \frac{b^2}{(1-e^{-bn})^2} \quad (3.5)$$

The value of ‘b’ in the above equation can be obtained using Newton Raphson method.

### 4. ESTIMATION BASED ON TIME BETWEEN FAILURES DATA

Based on the time between failures data given in Table-1, we compute the software failure process through mean value control chart. We use cumulative time between failures data for software reliability monitoring through SPC. The parameters observed for Goel-Okumoto model applied on the given time domain data are as follows:

<table>
<thead>
<tr>
<th>Failure No.</th>
<th>Time between failures</th>
<th>Failure No.</th>
<th>Time between failures</th>
<th>Failure No.</th>
<th>Time between failures</th>
<th>Failure No.</th>
<th>Time between failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>7</td>
<td>5.15</td>
<td>13</td>
<td>3.39</td>
<td>19</td>
<td>1.92</td>
</tr>
<tr>
<td>2</td>
<td>1.44</td>
<td>8</td>
<td>3.83</td>
<td>14</td>
<td>9.11</td>
<td>20</td>
<td>4.13</td>
</tr>
<tr>
<td>3</td>
<td>22.47</td>
<td>9</td>
<td>21</td>
<td>15</td>
<td>2.18</td>
<td>21</td>
<td>7.40</td>
</tr>
<tr>
<td>4</td>
<td>1.36</td>
<td>10</td>
<td>12.97</td>
<td>16</td>
<td>15.53</td>
<td>22</td>
<td>17.07</td>
</tr>
<tr>
<td>5</td>
<td>3.43</td>
<td>11</td>
<td>0.47</td>
<td>17</td>
<td>25.72</td>
<td>23</td>
<td>3.99</td>
</tr>
<tr>
<td>6</td>
<td>13.2</td>
<td>12</td>
<td>6.23</td>
<td>18</td>
<td>2.79</td>
<td>24</td>
<td>176.06</td>
</tr>
</tbody>
</table>

| Time between failures | 81.07 | 2.27 | 15.63 | 120.78 | 30.81 | 34.19 |

Table-1: Time between failures data (Xie et al., 2002)

### 5. CONTROL CHART

Control charts are sophisticated statistical data analysis tools, which include upper and lower limits to detect any outliers.

They are frequently used in SPC analysis [Koutras et al., 2007]. We used control chart mechanism to identify the

$$a = 31.698171, b = 0.003962$$

‘$\hat{a}$’ and ‘$\hat{b}$’ are Maximum Likelihood Estimates (MLEs) of parameters and the values can be computed using numerical iterative method for the given time between failures data shown in Table 1. Using values of ‘a’ and ‘b’ we can compute $m(t)$. Now equate the pdf of $m(t)$ to 0.00135, 0.99865, and 0.5 and the respective control limits are given by

$$T_a = (1 - e^{-a \cdot t}) = 0.99865$$

$$T_c = (1 - e^{-b \cdot t}) = 0.5$$

$$T_b = (1 - e^{-b \cdot t}) = 0.00135$$

These limits are convert at $m(t_a)$, $m(t_c)$ and $m(t_b)$ are given by

$$m(t_a) = 31.6767602, m(t_c) = 21.321140, m(t_b) = 0.08546968$$

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. A point below the control limit $m(t_a)$ indicates an alarming signal. A point above the control limit $m(t_b)$ indicates better quality. If the points are falling within the control limits it indicates the software process is in stable (MacGregor and Kourti). The values of control limits are as shown in Table-2.

### Table-2: Successive Difference of mean value function

<table>
<thead>
<tr>
<th>Failure No.</th>
<th>Cumulative failures</th>
<th>$m(t)$</th>
<th>$m(t)$</th>
<th>Failure No.</th>
<th>Cumulative failures</th>
<th>$m(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>5.585775764</td>
<td>0.160109911</td>
<td>16</td>
<td>151.78</td>
<td>14.32535611</td>
</tr>
<tr>
<td>2</td>
<td>31.46</td>
<td>7.314687474</td>
<td>2.383586679</td>
<td>17</td>
<td>177.54</td>
<td>16.00847828</td>
</tr>
<tr>
<td>3</td>
<td>53.93</td>
<td>6.098274153</td>
<td>0.137569469</td>
<td>18</td>
<td>180.29</td>
<td>13.18095679</td>
</tr>
<tr>
<td>4</td>
<td>55.29</td>
<td>6.235843622</td>
<td>0.34368381</td>
<td>19</td>
<td>182.21</td>
<td>16.2985492</td>
</tr>
<tr>
<td>5</td>
<td>58.72</td>
<td>6.579527432</td>
<td>1.27994674</td>
<td>20</td>
<td>186.34</td>
<td>16.54848354</td>
</tr>
<tr>
<td>6</td>
<td>71.92</td>
<td>7.859432106</td>
<td>0.481483904</td>
<td>21</td>
<td>215.82</td>
<td>20.23914448</td>
</tr>
<tr>
<td>7</td>
<td>77.07</td>
<td>8.34091601</td>
<td>0.351758112</td>
<td>22</td>
<td>237.88</td>
<td>20.98850796</td>
</tr>
<tr>
<td>8</td>
<td>80.9</td>
<td>8.692674122</td>
<td>1.836637975</td>
<td>23</td>
<td>277.87</td>
<td>21.15647921</td>
</tr>
<tr>
<td>9</td>
<td>101.9</td>
<td>10.5293121</td>
<td>1.060330131</td>
<td>24</td>
<td>453.93</td>
<td>26.4504792</td>
</tr>
<tr>
<td>10</td>
<td>114.87</td>
<td>11.5896423</td>
<td>0.037410054</td>
<td>25</td>
<td>535</td>
<td>27.89213213</td>
</tr>
<tr>
<td>11</td>
<td>115.34</td>
<td>11.6270522</td>
<td>0.489356342</td>
<td>26</td>
<td>537.27</td>
<td>27.92620918</td>
</tr>
<tr>
<td>12</td>
<td>121.57</td>
<td>12.1164068</td>
<td>0.261247815</td>
<td>27</td>
<td>552.9</td>
<td>28.1527065</td>
</tr>
<tr>
<td>13</td>
<td>124.96</td>
<td>12.3776664</td>
<td>0.684916199</td>
<td>28</td>
<td>673.68</td>
<td>29.50109689</td>
</tr>
<tr>
<td>14</td>
<td>134.07</td>
<td>13.0625726</td>
<td>0.160265529</td>
<td>29</td>
<td>704.49</td>
<td>29.7534513</td>
</tr>
<tr>
<td>15</td>
<td>136.25</td>
<td>13.2228381</td>
<td>1.10251794</td>
<td>30</td>
<td>738.68</td>
<td>29.9990315</td>
</tr>
</tbody>
</table>

| $m(t)$ | 1.68312175 | 0.172478506 | 0.117592238 | 0.249934515 | 3.690660937 | 0.74936348 |

$T_a = 1.68312175$, $T_c = 0.172478506$, $T_b = 0.117592238$, $T_a = 0.249934515$, $T_a = 3.690660937$. 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. A point below the control limit $m(t_a)$ indicates an alarming signal. A point above the control limit $m(t_b)$ indicates better quality. If the points are falling within the control limits it indicates the software process is in stable (MacGregor and Kourti). The values of control limits are as shown in Table-2.
process variation by placing the successive difference of cumulative mean values shown in table 2 on y axis and failure number on x axis and the values of control limits at mean value function are placed on Mean Value Chart, we obtained Figure 1. The Mean Value Chart shows that the successive differences of \( m(t) \) at 10\(^{th} \) and 25\(^{th} \) failure data has fallen below \( \mu(t) \), which indicates the failure process is identified.

It is significantly early detection of failures using Mean Value Chart. The software quality is determined by detecting failures at an early stage. The remaining failure data shown in Figure-1 is stable. No failure data fall outside \( \mu(t) \). It does not indicate any alarm signal.

6. CONCLUSION

This mean value chart exemplifies that, the first out – of – control situation is noticed at the 10\(^{th} \) failure and the second at 25\(^{th} \) failure with the corresponding successive difference of mean values falling below the LCL. The assignable cause for this is to be investigated and promoted. In comparison, the time control chart for the same data given in Xie et al (2002) reveal that an out - of - control for the first time above the UCL occurred at 23\(^{rd} \) failure. Since the data of the time-control chart are inter-failure times, a point above UCL for time-control chart is also a preferable criterion for the product. The time control chart gives the first out - of - control signal in a positive way, but at the 23\(^{rd} \) failure. Hence, it is claimed that the Mean Value Chart proposed by us detects out - of - control in a positive way much earlier than the time-control chart. Therefore, earlier detections are possible in failures control chart.

7. ACKNOWLEDGMENTS

We gratefully acknowledge the support of Department of Computer Science and Engineering, Acharya Nagarjuna University for providing necessary facilities to carry out the research work.

8. REFERENCES


Monitoring Software Reliability using Statistical Process Control: An MMLE Approach

Dr. R Satya Prasad¹, Bandla Sreenivasa Rao² and Dr. R.R. L Kantham³
¹Department of Computer Science & Engineering, Acharya Nagarjuna University, Guntur, India
profrsp@gmail.com,

²Department of Computer Science, VRS & YRN College, Chirala, India
sreenibandla@yahoo.com

³Department of Statistics, Acharya Nagarjuna University, Guntur, India
kantam_rrl@rediffmail.com

ABSTRACT
This paper consider an MMLE (Modified Maximum Likelihood Estimation) based scheme to estimate software reliability using exponential distribution. The MMLE is one of the generalized frameworks of software reliability models of Non Homogeneous Poisson Processes (NHPPs). The MMLE gives analytical estimators rather than an iterative approximation to estimate the parameters. In this paper we proposed SPC (Statistical Process Control) Charts mechanism to determine the software quality using inter failure times data. The Control charts can be used to measure whether the software process is statistically under control or not.

KEYWORDS
Software Reliability, Statistical Process Control, Control Charts, NHPP, Exponential Distribution, MMLE, Inter Failure Times Data

1. INTRODUCTION
The software reliability is one of the most significant attributes for measuring software quality. The software reliability can be quantitatively defined as the probability of failure free operation of a software in a specified environment during specified duration.[1]. Thus, probabilistic models are applied to estimate software reliability with the field data. Various NHPP software reliability models are available to estimate the software reliability. The MMLE is one of such NHPP based software reliability model.[2]. The software reliability models can be used quantitative management of quality (3). This is achieved by employing SPC techniques to the quality control activities that determines whether a process is stable or not. The objective of SPC is to establish and maintain statistical control over a random process. To achieve this objective, it is necessary to detect assignable causes of variation that contaminate the random process. The SPC had proven useful for detecting assignable causes(4).

2. BACKGROUND
This section presents the theory that underlies exponential distribution 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: \( m(t) = aF(t) \). where, \( F(t) \) is a cumulative distribution function. The failure intensity function \( \lambda(t) \) in case of the finite failure NHPP models is given by: \( \lambda(t) = aF'(t) \)[5][6]

### 2.1 Exponential NHPP Model

When the data is in the form of inter failure times also called Time between failures, we will try to estimate the parameters of an NHPP model based on exponential distribution [6]. Let \( N(t) \) be an NHPP defined as

\[
p(N(t) = k) = \frac{\lambda^k}{k!} e^{-\lambda t}, \quad k = 0, 1, 2, \ldots, n
\]

Here \( m(t) \) is the mean value function of the process of an NHPP given by

\[
m(t) = a(1 - e^{-\lambda t}) \quad a > 0, \ b > 0, t \geq 0
\]

(2.1.1)

The intensity function of the process is given by

\[
\lambda(t) = \frac{d\ln(m(t))}{dt} = b(a - m(t))
\]

(2.1.2)

### 2.2 Modified Maximum Likelihood Estimation

The constants \( 'a', \ 'b' \) which appear in the mean value function and hence in NHPP, in intensity function (error detection rate) and various other expressions are called parameters of the model. In order to have an assessment of the software reliability \( 'a', \ 'b' \) are to be known or they are to be estimated from a software failure data. Suppose we have \( n \) time instants at which the first, second, third,..., \( n^{th} \) failures of a software are experienced. In other words if \( s_k \) is the total time to the \( k^{th} \) failure, \( s_k \) is an observation of random variable \( s_k \) and \( n \) such failures are successively recorded. The joint probability of such failure time realizations \( s_1, s_2, s_3, \ldots, s_n \) is

\[
L = e^{-m(s_k)} \prod_{k=1}^{n} \lambda(s_k)
\]

(2.2.1)

The function given in equation (2.1.3)(2.2.1) is called the likelihood function of the given failure data. Values of \( 'a', \ 'b' \) that would maximize \( L \) are called maximum likelihood estimators (MLEs) and the method is called maximum likelihood (ML) method of estimation. Accordingly \( 'a', \ 'b' \) would be solutions of the equations

\[
\frac{\partial \ln L}{\partial a} = 0, \quad \frac{\partial \ln L}{\partial b} = 0
\]

Substituting the expressions for \( m(t), \lambda(t) \) given by equations (2.1.1) and (2.1.2) in equation (2.2.1), taking logarithms, differentiating with respect to \( 'a', \ 'b' \) and equating to zero, after some joint simplification we get
MLE of ‘b’ is an iterative solution of equation (2.1.5) (2.2.3) which when substituted in equation (2.1.4) gives MLE of ‘a’. In order to get the asymptotic variances and co-variance of the MLEs of ‘a’, ‘b’ we needed the elements of the information matrix obtained through the following second order partial derivative.

\[
\frac{\partial^2 \log L}{\partial \alpha \partial \beta} = \frac{1}{n} - m \frac{\partial^2 \log L}{\partial \alpha^2} - \frac{2}{b} - n \frac{\partial^2 \log L}{\partial \alpha \partial \beta} + \frac{m e^{\alpha x}}{\partial \beta} e^{\alpha x}
\]  

(2.2.4)

Expected values of negatives of the above derivative would be the following information matrix

\[
E = \begin{bmatrix}
\frac{\partial^2 \log L}{\partial \alpha \partial \alpha} & \frac{\partial^2 \log L}{\partial \alpha \partial \beta} \\
\frac{\partial^2 \log L}{\partial \beta \partial \alpha} & \frac{\partial^2 \log L}{\partial \beta \partial \beta}
\end{bmatrix}
\]  

Inverse of the above matrix is the asymptotic variance covariance matrix of the MLEs of ‘a’, ‘b’. Generally the above partial derivatives evaluated at the MLEs of ‘a’, ‘b’ are used to get consistent estimator of the asymptotic variance covariance matrix.

However in order to overcome the numerical iterative way of solving the log likelihood equations and to get analytical estimators rather than iterative, some approximations in estimating the equations can be adopted from [2] [8] and the references there in. We use two such approximations here to get modified MLEs of ‘a’ and ‘b’.

Equation (2.2.3) can be written as

\[
\sum_{k=1}^{n} \frac{S_k \alpha^{k-1} x_k^2}{e^{\alpha x_k} + e^{\beta}} - \alpha m x_k e^{\alpha x_k} = 0
\]  

(2.2.5)

Let us approximate the following expressions in the L.H.S of equation (2.2.5) by linear functions in the neighborhoods of the corresponding variables.

\[
S_k \alpha^{k-1} x_k^2 = m S_k + c, \ n = 1, 2, \ldots, n.
\]  

(2.2.6)

where \( S_k \) is the slope and ‘c’ is the intercepts in equations (2.2.6) are to be suitably found. With such values equations (2.2.6) when used in equation (2.2.5) would give an approximate MLE for ‘b’ as

\[
\hat{b} = \frac{1}{e^{\beta}} + \frac{1}{m \sum_{k=1}^{n} y_k e^{\beta}}
\]  

(2.2.7)

where \( \hat{b} = \sum_{k=1}^{n} \frac{y_k}{e^{\beta}} \)

We suggest the following method to get the slopes and intercepts in the R.H.S of equations (2.2.6).

\[
F (x) = \frac{n}{\sum_{k=1}^{n} x_k}
\]  

(2.2.8)
Given a natural number ‘n’ we can get the values of \( \mathcal{Z} \) and \( \mathcal{Z}' \) by inverting the above equations through the function \( F(z) \) the L.H.S of equation (2.2.6) we get

\[
\begin{align*}
m &= \frac{2}{b} \log \left( \frac{\exp(\frac{1}{2})}{\exp(\frac{1}{2})} \right) \quad (2.2.11) \\
c &= \frac{2}{b} \log \left( \frac{\exp(\frac{1}{2})}{\exp(\frac{1}{2})} \right) - m \quad (2.2.12)
\end{align*}
\]

It can be seen that the evaluation of \( \mathcal{Z} \) and \( \mathcal{Z}' \) can be computed free from any data. Given the data observations and sample size using these values along with the sample data in equation (2.1.12) we get an approximate MLE of ‘b’. Equation (2.2.2) gives approximate MLE of ‘a’.

### 3. ESTIMATION BASED ON INTER FAILURE TIMES DATA

Based on the time between failures data given in Table-1, we compute the software failure process through mean value control chart. We use cumulative time between failures data for software reliability monitoring through SPC. The parameters obtained from Goel-Okumoto model applied on the given time domain data are as follows:

\[
\begin{align*}
a &= 33.396342, \\
b &= 0.003962
\end{align*}
\]

‘\( \hat{a} \)’ and ‘\( \hat{b} \)’ are Modified Maximum Likelihood Estimates (MMLEs) of parameters and the values can be computed using analytical method for the given time between failures data shown in Table 1. Using values of ‘a’ and ‘b’ we can compute \( m(t) \). Now equate the pdf of \( m(t) \) to 0.00135, 0.99865, and 0.5 and the respective control limits are given by

\[
\begin{align*}
T_0 &= (1 - e^{-2t}) = 0.99865 \\
T_1 &= (1 - e^{-2t}) = 0.5 \\
T_2 &= (1 - e^{-2t}) = 0.00135
\end{align*}
\]

These limits are convert at \( m(t_0) \), \( m(t_1) \) and \( m(t_2) \) are given by

\[
\begin{align*}
m(t_0) &= 33.39612359302906, \\
m(t_1) &= 16.6901710073401, \\
m(t_2) &= 0.00450056100100
\end{align*}
\]

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. A point below the control limit \( m(t_1) \) indicates an alarming signal. A point above the control limit \( m(t_2) \) indicates better quality. If the points are falling within the control limits it indicates the software process is in stable [9]. The values of control limits are as shown in Table-2.
Table-1: Time between failures data (Xie et al., 2002)

<table>
<thead>
<tr>
<th>Failure No.</th>
<th>Time between Failures</th>
<th>Failure No.</th>
<th>Time between Failures</th>
<th>Failure No.</th>
<th>Time between Failures</th>
<th>Failure No.</th>
<th>Time between Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>7</td>
<td>5.15</td>
<td>13</td>
<td>3.39</td>
<td>19</td>
<td>1.92</td>
</tr>
<tr>
<td>2</td>
<td>1.44</td>
<td>8</td>
<td>3.83</td>
<td>14</td>
<td>9.11</td>
<td>20</td>
<td>4.13</td>
</tr>
<tr>
<td>3</td>
<td>22.47</td>
<td>9</td>
<td>21</td>
<td>15</td>
<td>2.18</td>
<td>21</td>
<td>70.47</td>
</tr>
<tr>
<td>4</td>
<td>1.36</td>
<td>10</td>
<td>12.97</td>
<td>16</td>
<td>2.18</td>
<td>22</td>
<td>17.07</td>
</tr>
<tr>
<td>5</td>
<td>3.43</td>
<td>11</td>
<td>0.47</td>
<td>17</td>
<td>25.72</td>
<td>23</td>
<td>3.99</td>
</tr>
<tr>
<td>6</td>
<td>13.2</td>
<td>12</td>
<td>6.23</td>
<td>18</td>
<td>2.79</td>
<td>24</td>
<td>176.06</td>
</tr>
</tbody>
</table>

Table-2: Successive Difference of mean value function

<table>
<thead>
<tr>
<th>Failure No</th>
<th>Cumulative failures</th>
<th>m(t)</th>
<th>m(t) Successive Difference</th>
<th>Failure No</th>
<th>Cumulative failures</th>
<th>m(t)</th>
<th>m(t) Successive Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>3.745007495</td>
<td>0.168687503</td>
<td>16</td>
<td>151.78</td>
<td>15.092810</td>
<td>62</td>
</tr>
<tr>
<td>2</td>
<td>31.46</td>
<td>3.913694999</td>
<td>2.511282936</td>
<td>17</td>
<td>177.5</td>
<td>16.8661025</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>53.93</td>
<td>6.424977934</td>
<td>0.1449395</td>
<td>18</td>
<td>180.29</td>
<td>17.04782168</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>55.29</td>
<td>6.569917434</td>
<td>0.362096035</td>
<td>19</td>
<td>182.21</td>
<td>17.1717137</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>58.72</td>
<td>6.932013469</td>
<td>1.348473204</td>
<td>20</td>
<td>186.34</td>
<td>17.435038</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>71.92</td>
<td>8.280486673</td>
<td>0.507278516</td>
<td>21</td>
<td>256.81</td>
<td>21.32341928</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>77.07</td>
<td>8.787765189</td>
<td>0.370602904</td>
<td>22</td>
<td>273.88</td>
<td>22.11292853</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>80.9</td>
<td>9.158368093</td>
<td>1.935032465</td>
<td>23</td>
<td>277.87</td>
<td>22.28989853</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>101.9</td>
<td>11.09340056</td>
<td>1.11713536</td>
<td>24</td>
<td>453.93</td>
<td>27.8675148</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>114.87</td>
<td>12.21053592</td>
<td>0.039414228</td>
<td>25</td>
<td>535</td>
<td>29.38640162</td>
<td>0.03590267</td>
</tr>
<tr>
<td>11</td>
<td>115.34</td>
<td>12.24950515</td>
<td>0.515572704</td>
<td>26</td>
<td>537.27</td>
<td>29.42230429</td>
<td>0.238631489</td>
</tr>
<tr>
<td>12</td>
<td>121.57</td>
<td>12.76552285</td>
<td>0.275243684</td>
<td>27</td>
<td>552.9</td>
<td>29.66093578</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>124.96</td>
<td>13.04076653</td>
<td>0.72160932</td>
<td>28</td>
<td>673.68</td>
<td>31.08153524</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>134.07</td>
<td>13.76237588</td>
<td>0.168851459</td>
<td>29</td>
<td>704.49</td>
<td>31.34753639</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>136.25</td>
<td>13.93122731</td>
<td>1.161583304</td>
<td>30</td>
<td>738.68</td>
<td>31.60709258</td>
<td>2</td>
</tr>
</tbody>
</table>

4. CONTROL CHART

Control charts are sophisticated statistical data analysis tools, which include upper and lower limits to detect any outliers. They are frequently used in SPC analysis [10]. We used control chart mechanism to identify the process variation by placing the successive difference of cumulative mean values shown in table 2 on y axis and failure number on x axis and the values of control limits at mean value function are placed on Inter Failure Control chart, we obtained Figure 1. The Inter Failure Control chart shows that the successive differences of m(t) at 10th and 25th failure data has fallen below m(t-1), which indicates the failure process is identified. It is significantly early detection of failures using Inter Failure Control chart. The software quality is determined by detecting failures at an early stage. The remaining failure data shown in Figure-1 is stable. No failure data fall outside of control limits. It does not indicate any alarm signal.
5. CONCLUSION

This Mean value chart (Fig 1) exemplifies that, the first out – of – control and second our-of-control situation is noticed at the 10th failure and 25th failure with the corresponding successive difference of m(t) falling below the LCL. It results in an earlier and hence preferable out-of-control for the product. The assignable cause for this is to be investigated and promoted. The out of control signals in and the model suggested in Satya Prasad at el [2011] [13] are the same. We therefore conclude that adopting a modification to the likelihood method doesn’t alter the situation, but simplified the procedure of getting the estimates of the parameters, thus resulting in a preference of the present model to the one described in Satya Prasad et al [2011] [13].

6. ACKNOWLEDGMENT

We gratefully acknowledge the support of Department of Computer Science and Engineering, Acharya Nagarjuna University and PG Department of Computer Applications, VRS & YRN College, Chirala for providing necessary facilities to carry out the research work.

7. REFERENCES

3. Quantitative quality management through defect prediction and statistical process control (2nd World Quality Congress for Software, Japan, September 2000.).


AUTHORS PROFILE

Dr. R. Satya Prasad received Ph.D. degree in Computer Science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Andhra Pradesh. He received gold medal from Acharya Nagarjuna University for his outstanding performance in Masters Degree. He is currently working as Associate Professor and H.O.D, in the Department of Computer Science & Engineering, Acharya Nagarjuna University. His current research is focused on Software Engineering, Software reliability. He has published several papers in National & International Journals.

Bandla Srinivasa rao received the Master Degree in Computer Science and Engineering from Dr MGR Deemed University, Chennai, Tamil Nadu, India. He is Currently working as Associate Professor in PG Department of Computer Applications, VRS & YRN College, Chirala, Andhra Pradesh, India. His research interests include software reliability, Cryptography and Computer Networks. He has published several papers in National and International Journals.

R.R.L Kantam is a professor of statistics at Acharya Nagarjuan University, Guntur,India. He has 31 years of teaching experience in statistics at Under Graduate and Post Graduate Programs. As researcher in statistics, he has successfully guided 8 students for M.Phil in statistics and 5 students for Ph.D. in statistics. He has authored 54 research publications appeared various statistical journals published in India and other countries Like US, UK, Germany, Pakistan, Srilanka, and Bangladesh. He has been a referee for Journal of Applied Statistics (UK), METRON (Italy), Pakistan Journal of Statistics (Pakistan), IAPQR- Transactions (India), Assam Statistical Review (India) and Gujarat Statistical Review (India). He has been a special speaker in technical sessions of a number of seminars and Conferences, His area of research interest are Statistical Inference, Reliability Studies, Quality Control Methods and Actuarial Statistics. As a teacher his present teaching areas are Probability Theory, Reliability, and Actuarial Statistics. His earlier teaching topics include Statistical Inference, Mathematical analysis, Operations Research, Econometrics, Statistical Quality Control, Measure theory.

Bandla Srinivasa Rao  
Associate Professor.,  
Dept. of Computer Science  
VRS & YRN College

Dr. R Satya Prasad  
Associate Professor,  
Dept. of Computer Science & Eng.,  
Acharya Nagarjuna University

Dr. R.R. L Kantham  
Professor,  
Dept. of Statistics  
Acharyan Nagarjuna University

ABSTRACT
The nature and complexity of software have changed significantly in the last few decades. With the easy availability of computing power, deeper and broader applications are made. It has been extremely necessary to produce good quality software with high precision of reliability right in the first place. Olden day’s software errors and bugs were fixed at a later stage in the software development. Today to produce high quality reliable software and to keep a specific time schedule is a big challenge. To cope up the challenge many concepts, methodology and practices of software engineering have been evolved for developing reliable software. Better methods of controlling the process of software production are underway. One of such methods to assess the software reliability is using control charts. In this paper we proposed an NHPP based control mechanism by using order statistics with cumulative quantity between observations of failure data using mean value function of exponential distribution.

General Terms
Software reliability, Software quality, Six Sigma, Control Charts, PDF, CDF

Keywords: Ordered Statistics, Statistical Process Control (SPC), Exponential Distribution, Control Limits, software reliability, software quality

1. INTRODUCTION
As computer applications became more diverse and spread through almost every area of everyday life, reliability became a very important characteristic for software, since it is a matter of economy. To produce a software having reliability, it is necessary to measure and control its reliability. To do this, a number of models have been developed; new models try to make better predictions. Software reliability represents a user oriented view of software quality. It relates directly to operation rather than design of the program, and hence it is dynamic. For this reason software reliability is interested in failures occurrence and not faults in a program

1.1. Software reliability Modeling
The probability that a given program will work as intended by the user, i.e., without failures in a specified environment and for a specified duration can be termed as software reliability [1][2]. The aim of software engineer is to increase this probability and make it one if possible. To do this one must measure the reliability of the software. A commonly used approach for measuring software reliability is by using an analytical model whose parameters are generally estimated from available data on software failures. Reliability quantities have been defined with respect to time, although it is possible to define them with respect to other variables. We have taken inter failures time data of Musa(1975) which are random values. In reliability study there are two characteristics of a random process: 1) the probability distribution of the random variables, i.e., Poisson and 2) the variation of the process with time. A random process whose probability distribution varies with time is called non homogeneous. For the random process for time variation we can define two functions, the mean value function m(t), as the average cumulative failures associated with each time point and the failure intensity function λ(t) as the rate of change of mean value function. When there are changes in the software i.e. software corrections occur it is called non homogeneous process.

Let M(t) be the random process representing the number of failures experienced by time t, then the mean value function is defined by \( \mu(t) = E[M(t)] \), i.e. the expected number of failures at time t. the failure intensity function of the \( M(t) \) process is the instantaneous rate of change of the expected number of failures with respect to time or \( \lambda(t) = \frac{d\mu(t)}{dt} \).[3]

2. ORDERED STATISTICS
Let X denote a continuous random variable with Probability Density Function (PDF) f(x) and Cumulative Distribution Function (CDF) F(x), and let (X_1, X_1, ..., X_n) denote a random sample of size n drawn on X. The original sample observations may be unordered with respect to magnitude. A transformation is required to produce a corresponding ordered sample. Let (X_1, X_2, ..., X_n) denote the ordered random sample such that X_1 < X_2 < ... < X_n; then (X_1, X_2, ..., X_n) are collectively known as the order statistics derived from the parent X. The various distributional characteristics can be known from Balakrishnan and Cohen [4]. The inter-failure time data represent the time lapse between every two consecutive failures. On the other hand if a reasonable waiting time for failures is not a serious problem, we can group the inter-failure time data into non
overlapping successive sub groups of size 4 or 5 and add the failure times within each sub group. For instance if a data of 100 inter-failure times are available we can group them into 20 disjoint subgroups of size 5. The sum total in each subgroup would denote the time lapse between every 5th order statistics in a sample of size 5. In general for inter-failure data of size 'n', if r (any natural no) less than 'n' and preferably a factor n, we can conveniently divide the data into 'k' disjoint subgroups (k=n/r) and the cumulative total in each subgroup indicate the time between every rth failure. The probability distribution of such a time lapse would be that of the rth ordered statistics in a subgroup of size r, which would be equal to rth power of the distribution function of the original variable m(t).The whole process involves the mathematical model of the mean value function and knowledge about its parameters. If the parameters are known they can be taken as they are for the further analysis, if the parameters are not know they have to be estimated using a sample data by any admissible, efficient method of distribution. This is essential because the control limits depend on mean value function, which intern depends on the parameters. If software failures are quite frequent keeping track of inter-failure is tedious. If failures are more frequent order statistics are preferable. [5]

2.1. Model Description
Considering failure detection as a non homogenous Poisson process with an exponentially decaying rate function, the expected number of failures observed by time t is given by
\[ m(t) = \sigma(1 - e^{-bt}) \] and the failure rate by \( \lambda(t) = m'(t) \). To calculate the parameter values and control limits using Order Statistics approach, we considered exponential distribution [8]. The mean value function of exponential distribution is
\[ m(t) = \sigma(1 - e^{-bt}) \]
In order to group the inter-failure time data into non overlapping successive sub groups of size r the mean value function can be written as
\[ m(t) = \sigma(1 - e^{-bt})' \]
\[ m(z_k) = [\sigma(1 - e^{-b_z})]' \]
\[ m'(z_k) = a^r r (1 - e^{-bx})^r-1 b e^{-bx} \]

The likelihood function L can be written as
\[ L = e^{-m(z_o)} \prod_{k=1}^{n} m'(z_k) \]
Substituting eq.2.1.1 in eq.2.2.2 we can write
\[ L = e^{-m(z_o)} \prod_{k=1}^{n} a^r r (1 - e^{-bx})^r-1 b e^{-bx} \]
\[ \log L = -m(z_o) + \sum_{k=1}^{n} [\log a^r + \log b + \log r + \log e^{-bx} + \log (1 - e^{-bx})^r-1] \]
\[ m(z_n) = [\sigma(1 - e^{-b_z})]' \]
Substitute equation 2.2.4 in 2.2.3 we get
\[ \log L = -[a(1 - e^{-b_z})]^r + \sum_{k=1}^{n} [\log a^r + \log b + \log r + \log e^{-bx} + \log (1 - e^{-bx})^r-1] \]

2.2. Parameter estimation and Control limits
Parameter estimation is a statistical method trying to estimate parameters based on inter failures time data which is based on ordered statistics. For the given observations using equations 2.2.8 and 2.2.9 the parameters ‘a’ and ‘b’ are computed by using the popular Newton Rapson method A program written in C was used for this purpose. [3]

Based on the time between failures data given in Table-1, we compute the software failure process through mean value control chart. We use cumulative time between failures data for software reliability monitoring through SPC. The parameters obtained from Goel-Okumoto model applied on the given time domain data are as follows:

<table>
<thead>
<tr>
<th>Data Set of Table 2</th>
<th>Order</th>
<th>( \hat{a} )</th>
<th>( \hat{b} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4th</td>
<td>2.415117</td>
<td>0.000099</td>
</tr>
<tr>
<td></td>
<td>5th</td>
<td>1.933309</td>
<td>0.000114</td>
</tr>
</tbody>
</table>

‘\( \hat{a} \)’ and ‘\( \hat{b} \)’ are ordered statistics of parameters and the values can be computed using analytical method for the given time between failures data shown in Table 1. Using values of ‘a’ and ‘b’ we can compute \( m(t) \). Now equate the pdf of m(t) to 0.00135, 0.99865, and 0.5 and the respective control limits are given by
\[ T_C = (1 - e^{-bt}) = 0.99865 \]
\[ T_c = (1 - e^{-\frac{m}{c}}) = 0.5 \]
\[ T_L = (1 - e^{-\frac{m}{c}}) = 0.00135 \]

These limits are convert at \( m(t_0), m(t_c) \) and \( m(t_L) \) are given by

\[ m(t_0) = 33.35125569382986 \]
\[ m(t_c) = 16.6981710073481 \]
\[ m(t_L) = 0.04508506100108 \]

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 and figure-2. A point below the control limit \( m(t_c) \) indicates an alarming signal. A point above the control limit \( m(t_0) \) indicates better quality. If the points are falling within the control limits it indicates the software process is in stable. [6]

STATISTICAL PROCESS CONTROL

Statistical process control is the application of statistical methods to provide the information necessary to continuously control or improve processes throughout the entire lifecycle of a product [7]. SPC techniques help to locate trends, cycles, and irregularities within the development process and provide clues about how well the process meets specifications or requirements. They are tools for measuring and understanding process variation and distinguishing between random inherent variations and significant deviations so that correct decisions can be made about whether to make changes to the process or product. One of such primary statistical technique used to assess process variation is the control chart. [8]

2.3. Control Chart

The control chart displays sequential process measurements relative to the overall process average and control limits. The upper and lower control limits establish the boundaries of normal variation for the process being measured. Variation within control limits is attributable to random or chance causes, while variation beyond control limits indicates a process change due to causes other than chance, a condition that may require investigation. [7] The upper control limit (UCL) and lower control limit (LCL) give the boundaries within which observed fluctuations are typical and acceptable. There are many different types of control charts, such as, [8, 9, 10].

2.4. Developing Control Chart

Given the n inter-failure data the values of \( m(t) \) at \( T_c, T_0, T_L \) and at the given n inter-failure times are calculated. Then successive differences of \( m(t) \)’s are taken, which leads to n-1 values. The graph with the said inter-failure times 1 to n-1 on X-axis, the n-1 values of successive differences \( m(t) \)’s on Y-axis, and the 3 control lines parallel to X-axis at \( m(T_c), m(T_0), m(T_L) \) respectively constitutes mean value chart to assess the software failure phenomena on the basis of the given inter-failures time data.

2.5. Illustration

The procedure of a mean value chart for failure software process will be illustrated with an example here. Table 1 show the time between failures of software product reported by Musa [1975] [11].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>18</td>
<td>120</td>
<td>35</td>
<td>227</td>
<td>52</td>
<td>21</td>
<td>69</td>
<td>529</td>
<td>86</td>
<td>860</td>
<td>103</td>
<td>108</td>
<td>120</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>19</td>
<td>26</td>
<td>36</td>
<td>65</td>
<td>53</td>
<td>233</td>
<td>70</td>
<td>379</td>
<td>87</td>
<td>983</td>
<td>104</td>
<td>0</td>
<td>121</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>113</td>
<td>20</td>
<td>114</td>
<td>37</td>
<td>176</td>
<td>54</td>
<td>134</td>
<td>71</td>
<td>44</td>
<td>88</td>
<td>707</td>
<td>105</td>
<td>3110</td>
<td>122</td>
<td>482</td>
</tr>
<tr>
<td>4</td>
<td>81</td>
<td>21</td>
<td>325</td>
<td>38</td>
<td>58</td>
<td>55</td>
<td>357</td>
<td>72</td>
<td>129</td>
<td>89</td>
<td>33</td>
<td>106</td>
<td>1247</td>
<td>123</td>
<td>5509</td>
</tr>
<tr>
<td>5</td>
<td>115</td>
<td>22</td>
<td>55</td>
<td>39</td>
<td>457</td>
<td>56</td>
<td>193</td>
<td>73</td>
<td>810</td>
<td>90</td>
<td>868</td>
<td>107</td>
<td>943</td>
<td>124</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>23</td>
<td>242</td>
<td>40</td>
<td>300</td>
<td>57</td>
<td>236</td>
<td>74</td>
<td>290</td>
<td>91</td>
<td>724</td>
<td>108</td>
<td>700</td>
<td>125</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>24</td>
<td>68</td>
<td>41</td>
<td>97</td>
<td>58</td>
<td>31</td>
<td>75</td>
<td>300</td>
<td>92</td>
<td>2323</td>
<td>109</td>
<td>875</td>
<td>126</td>
<td>1071</td>
</tr>
<tr>
<td>8</td>
<td>91</td>
<td>25</td>
<td>422</td>
<td>42</td>
<td>263</td>
<td>59</td>
<td>369</td>
<td>76</td>
<td>529</td>
<td>93</td>
<td>2930</td>
<td>110</td>
<td>245</td>
<td>127</td>
<td>371</td>
</tr>
<tr>
<td>9</td>
<td>112</td>
<td>26</td>
<td>180</td>
<td>43</td>
<td>452</td>
<td>60</td>
<td>748</td>
<td>77</td>
<td>281</td>
<td>94</td>
<td>1461</td>
<td>111</td>
<td>729</td>
<td>128</td>
<td>790</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>27</td>
<td>10</td>
<td>44</td>
<td>255</td>
<td>61</td>
<td>0</td>
<td>78</td>
<td>160</td>
<td>95</td>
<td>843</td>
<td>112</td>
<td>1897</td>
<td>129</td>
<td>6150</td>
</tr>
<tr>
<td>11</td>
<td>138</td>
<td>28</td>
<td>1145</td>
<td>46</td>
<td>197</td>
<td>62</td>
<td>232</td>
<td>79</td>
<td>828</td>
<td>96</td>
<td>12</td>
<td>113</td>
<td>447</td>
<td>130</td>
<td>3321</td>
</tr>
<tr>
<td>12</td>
<td>50</td>
<td>29</td>
<td>600</td>
<td>46</td>
<td>193</td>
<td>63</td>
<td>330</td>
<td>80</td>
<td>1011</td>
<td>97</td>
<td>261</td>
<td>114</td>
<td>386</td>
<td>131</td>
<td>1045</td>
</tr>
<tr>
<td>13</td>
<td>77</td>
<td>30</td>
<td>15</td>
<td>47</td>
<td>6</td>
<td>64</td>
<td>365</td>
<td>81</td>
<td>445</td>
<td>98</td>
<td>1800</td>
<td>115</td>
<td>446</td>
<td>132</td>
<td>648</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>31</td>
<td>36</td>
<td>48</td>
<td>79</td>
<td>65</td>
<td>1222</td>
<td>82</td>
<td>296</td>
<td>99</td>
<td>865</td>
<td>116</td>
<td>122</td>
<td>133</td>
<td>5485</td>
</tr>
<tr>
<td>15</td>
<td>108</td>
<td>32</td>
<td>4</td>
<td>49</td>
<td>816</td>
<td>66</td>
<td>543</td>
<td>83</td>
<td>1755</td>
<td>100</td>
<td>1435</td>
<td>117</td>
<td>990</td>
<td>134</td>
<td>1160</td>
</tr>
<tr>
<td>16</td>
<td>88</td>
<td>33</td>
<td>0</td>
<td>50</td>
<td>1351</td>
<td>67</td>
<td>10</td>
<td>84</td>
<td>1064</td>
<td>101</td>
<td>30</td>
<td>118</td>
<td>948</td>
<td>135</td>
<td>1864</td>
</tr>
<tr>
<td>17</td>
<td>670</td>
<td>34</td>
<td>8</td>
<td>51</td>
<td>148</td>
<td>68</td>
<td>16</td>
<td>85</td>
<td>1783</td>
<td>102</td>
<td>143</td>
<td>119</td>
<td>1082</td>
<td>136</td>
<td>4116</td>
</tr>
</tbody>
</table>
Table 3: 4th order cumulative faults and their m(t) successive difference.

<table>
<thead>
<tr>
<th>Fault</th>
<th>4-order Cumulative Faults</th>
<th>m(t)</th>
<th>Successive Difference's of m(t)</th>
<th>Fault</th>
<th>4-order Cumulative Faults</th>
<th>m(t)</th>
<th>Successive Difference's of m(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>227</td>
<td>0.053669607</td>
<td>0.050189929</td>
<td>18</td>
<td>16358</td>
<td>1.936901718</td>
<td>0.083134596</td>
</tr>
<tr>
<td>2</td>
<td>444</td>
<td>0.103859536</td>
<td>0.070964302</td>
<td>19</td>
<td>18287</td>
<td>2.020036314</td>
<td>0.079829413</td>
</tr>
<tr>
<td>3</td>
<td>759</td>
<td>0.174823838</td>
<td>0.064912354</td>
<td>20</td>
<td>20567</td>
<td>2.099865727</td>
<td>0.09363885</td>
</tr>
<tr>
<td>4</td>
<td>1056</td>
<td>0.239736192</td>
<td>0.191343658</td>
<td>21</td>
<td>24127</td>
<td>2.193504577</td>
<td>0.077302531</td>
</tr>
<tr>
<td>5</td>
<td>1986</td>
<td>0.431079851</td>
<td>0.131004192</td>
<td>22</td>
<td>28460</td>
<td>2.270807108</td>
<td>0.046687097</td>
</tr>
<tr>
<td>6</td>
<td>2676</td>
<td>0.562084043</td>
<td>0.296000509</td>
<td>23</td>
<td>32408</td>
<td>2.317494206</td>
<td>0.039546532</td>
</tr>
<tr>
<td>7</td>
<td>4434</td>
<td>0.858084551</td>
<td>0.097761832</td>
<td>24</td>
<td>37654</td>
<td>2.357040738</td>
<td>0.020362766</td>
</tr>
<tr>
<td>8</td>
<td>5089</td>
<td>0.955846384</td>
<td>0.042703058</td>
<td>25</td>
<td>42015</td>
<td>2.377403504</td>
<td>0.001034693</td>
</tr>
<tr>
<td>9</td>
<td>5389</td>
<td>0.998549441</td>
<td>0.132378119</td>
<td>26</td>
<td>42296</td>
<td>2.378438197</td>
<td>0.016427908</td>
</tr>
<tr>
<td>1</td>
<td>6380</td>
<td>1.13092756</td>
<td>0.12873383</td>
<td>27</td>
<td>48296</td>
<td>2.394866105</td>
<td>0.006274847</td>
</tr>
<tr>
<td>11</td>
<td>7447</td>
<td>1.259661391</td>
<td>0.053077534</td>
<td>28</td>
<td>52042</td>
<td>2.401140952</td>
<td>0.001810038</td>
</tr>
<tr>
<td>12</td>
<td>7922</td>
<td>1.312738924</td>
<td>0.22760806</td>
<td>29</td>
<td>53443</td>
<td>2.402950999</td>
<td>0.003163639</td>
</tr>
<tr>
<td>13</td>
<td>10258</td>
<td>1.540346985</td>
<td>0.075916166</td>
<td>30</td>
<td>56485</td>
<td>2.406114629</td>
<td>0.004113047</td>
</tr>
<tr>
<td>14</td>
<td>11175</td>
<td>1.616263151</td>
<td>0.102288223</td>
<td>31</td>
<td>62651</td>
<td>2.410227676</td>
<td>0.000973225</td>
</tr>
<tr>
<td>15</td>
<td>12559</td>
<td>1.718551373</td>
<td>0.061080293</td>
<td>32</td>
<td>64893</td>
<td>2.411200901</td>
<td>0.00261935</td>
</tr>
<tr>
<td>16</td>
<td>13486</td>
<td>1.779631666</td>
<td>0.103253053</td>
<td>33</td>
<td>76057</td>
<td>2.413820251</td>
<td>0.000925177</td>
</tr>
<tr>
<td>17</td>
<td>15277</td>
<td>1.882884719</td>
<td>0.054016999</td>
<td>34</td>
<td>88682</td>
<td>2.414745429</td>
<td></td>
</tr>
</tbody>
</table>
Fig-1: Mean Value Chart of 4th order data set

Table: 5th order cumulative faults and their m(t) successive difference

<table>
<thead>
<tr>
<th>Fault</th>
<th>5-order Cumulative</th>
<th>m(t)</th>
<th>Successive Difference’s Of m(t)’s</th>
<th>Fault</th>
<th>5-order Cumulative</th>
<th>m(t)</th>
<th>Successive Difference’s Of m(t)’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>342</td>
<td>0.073925386</td>
<td>0.04791294</td>
<td>15</td>
<td>17758</td>
<td>1.62782472</td>
<td>0.050149302</td>
</tr>
<tr>
<td>2</td>
<td>571</td>
<td>0.121838326</td>
<td>0.080156008</td>
<td>16</td>
<td>20567</td>
<td>1.677974022</td>
<td>0.069965628</td>
</tr>
<tr>
<td>3</td>
<td>968</td>
<td>0.201994334</td>
<td>0.189702018</td>
<td>17</td>
<td>25910</td>
<td>1.74793965</td>
<td>0.084558265</td>
</tr>
<tr>
<td>4</td>
<td>1986</td>
<td>0.391696352</td>
<td>0.183547444</td>
<td>18</td>
<td>29361</td>
<td>1.832497915</td>
<td>0.032788872</td>
</tr>
<tr>
<td>5</td>
<td>3098</td>
<td>0.575243796</td>
<td>0.270820129</td>
<td>19</td>
<td>37642</td>
<td>1.865286786</td>
<td>0.041557817</td>
</tr>
<tr>
<td>6</td>
<td>5049</td>
<td>0.846063925</td>
<td>0.03556388</td>
<td>20</td>
<td>42015</td>
<td>1.906844603</td>
<td>0.010389187</td>
</tr>
<tr>
<td>7</td>
<td>5324</td>
<td>0.879620314</td>
<td>0.11950945</td>
<td>21</td>
<td>45406</td>
<td>1.91723379</td>
<td>0.005154027</td>
</tr>
<tr>
<td>8</td>
<td>6380</td>
<td>0.999129764</td>
<td>0.125362531</td>
<td>22</td>
<td>49416</td>
<td>1.922387817</td>
<td>0.004007073</td>
</tr>
<tr>
<td>9</td>
<td>7644</td>
<td>1.124492295</td>
<td>0.196749357</td>
<td>23</td>
<td>53321</td>
<td>1.92639489</td>
<td>0.002484134</td>
</tr>
<tr>
<td>10</td>
<td>10089</td>
<td>1.321241652</td>
<td>0.059242997</td>
<td>24</td>
<td>56485</td>
<td>1.928879023</td>
<td>0.001341437</td>
</tr>
<tr>
<td>11</td>
<td>10982</td>
<td>1.380484649</td>
<td>0.090964117</td>
<td>25</td>
<td>62661</td>
<td>1.930220461</td>
<td>0.001561037</td>
</tr>
<tr>
<td>12</td>
<td>12559</td>
<td>1.471448766</td>
<td>0.100355072</td>
<td>26</td>
<td>74364</td>
<td>1.931781497</td>
<td>0.001125183</td>
</tr>
<tr>
<td>13</td>
<td>14708</td>
<td>1.571803838</td>
<td>0.056020882</td>
<td>27</td>
<td>84566</td>
<td>1.93290668</td>
<td>0.00027658</td>
</tr>
<tr>
<td>14</td>
<td>16185</td>
<td>0.073925386</td>
<td>0.04791294</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. CONCLUSION

The Mean value charts of Fig 1 and 2 have shown out of control signals i.e. below LCL. By observing Mean value charts, we identified that failures situation is detected at an early stages. The early detection of software failure will improve the software reliability. When the control signals are below LCL, it is likely that there are assignable causes leading to significant process deterioration and it should be investigated. Hence, we conclude that our control mechanism proposed in this chapter giving a positive recommendation for its use to estimate whether the process is in control or out of control.

4. REFERENCES

[12] Hong Pharm; System Reliability; Springer;2005;Page No 281

Fig 2: Mean Value Chart of 5th order data set
A Comparative Study For Assessing Software Reliability Using SPC

K. Ramchand H Rao
Department of Computer Science
A.S.N. Degree College
Tenali, India
ramkolasani@gmail.com

B. Srinivasa Rao
Department of Computer Science
VRS & YRN College
Cherala, India
sreenibandla@yahoo.com

Dr. R. Satya Prasad
Department of Computer Science and Engineering
Acharya Nagarjuna University
Guntur, India
profrsp@gmail.com

Abstract—Software reliability is defined as the probability that a software system operates without failure occurring for a specified time on specified operating conditions. Assessing software reliability and thereby maintaining software quality during software development and software usage is most important, as software are being used in all domains along with safety critical systems and non-safety critical systems. Statistical process control (SPC) can be applied to forecast software failure and thereby contribute significantly to the improvement of software reliability. Control charts are an important tool of SPC to determine software process for reliability. In this paper, two distinct control mechanisms developed by us to assess software reliability, based on time between failure observations using Half Logistic and Exponential distributions based NHPP are compared.

Keywords—Statistical Process Control, Software reliability, Control Charts, Probability limits, Half Logistic Distribution, Exponential Distribution.

I. INTRODUCTION

In the past decades, various statistical tools have been applied to solve the problem of software reliability. Software reliability is defined as the probability that a software system operates with no failure occurring for a specified time on specified operating conditions [1]. By estimating, predicting and managing the reliability, cost and performance of software based systems, the quality of software product is aided to get improved and the satisfaction of customers are gained. A failure is an observed departure of the external result of software operation from software requirements or user expectations. When a system in operation does not deliver its intended said to fail. A metric that is commonly used to describe software reliability is failure intensity, defined as the number of failures experienced per unit time period. Failure intensity is a good measure for reflecting the user perspective of software quality. When reliability growth is being experienced, failure intensity decreases over time.

As Computer applications become more diverse and spread through almost every area of everyday life, reliability become a very important characteristic of computer systems. From the first time computers were used until presently, people have been interested in reliable systems [2]. Since it is a matter of economy to produce a system having reliability above a specific level, it is necessary to measure and control its reliability or unreliability. To do this, a number of models have been and are being developed. New models try to make better predictions and to alleviate the problems resulting from unreasonable assumptions made by earlier ones. The most important software product characteristics are level of quality, time of delivery, and cost. These attributes are user-oriented rather than developer-oriented. Also time of delivery and cost are quantitative, whereas quality obviously cannot be defined quantitatively [3]. Reliability is one, and probably the most important, aspect of software quality.

Software reliability represents a user-oriented view of software quality. It relates directly to operation rather than design of the program, and hence it is dynamic rather that static. For this reason software reliability is interested in failures occurring and not faults in a program. Reliability measures are much more useful than fault measures.

In this paper we compared two control mechanisms developed by us to assess software reliability using SPC. The two control mechanisms are based on time between failure observations using Half Logistic Distribution (HLD) [4][5] and Exponential Distribution [6].
II. MODELS DESCRIPTION

A. Half Logistic Distribution

Let \( \{N(t), t \geq 0\}, m(t), \lambda(t) \) be the counting process, mean value function and intensity function of a software failure phenomenon.

The mean value function \( m(t) \) is finite valued, non decreasing, non negative and bounded with the boundary conditions

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

Here ‘a’ represents the expected number of software failures eventually detected. If \( \lambda(t) \) is the corresponding intensity function, \( \lambda(t) \) is a decreasing function of \( m(t) \) as a result of a decreasing trend for \( \lambda(t) \) with increase in \( m(t) \).

From the fact that \( \lambda(t) \) is the derivative of \( m(t) \) we get the following differential equation

\[
\frac{dm(t)}{dt} = \frac{b}{2a}\left[a^2 - m^2(t)\right]
\]

Whose solution is

\[
m(t) = \frac{a \left[1 - e^{-bt}\right]}{1 + e^{-bt}} \quad (1)
\]

An NHPP with its mean value function given in equation (1). Its intensity function is

\[
\lambda(t) = \frac{2aban}{\left(1 + e^{-bt}\right)^2} \quad (2)
\]

B. Exponential Distribution

Let \( \{N(t), t \geq 0\} \) be the cumulative number of software failures by time ‘t’. \( m(t) \) is the mean value function, representing the expected number of software failures by time ‘t’. \( \lambda(t) \) is failure intensity function, which is proportional to the residual fault content [7].

Thus

\[
m(t)=a(1-e^{-bt}) \quad a>0, b>0, t>0
\]

\[
\lambda(t) = aF(t) \quad (4)
\]

Here ‘a’ denotes the initial fault contained in a program and ‘b’ represents the fault detection rate. In software reliability, the initial number of faults and faults detection rate are always unknown.

III. PARAMETER ESTIMATION

Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for \( m(t) \) is known for a given model, the parameter in the solution needs to be determined. Parameter estimation is achieved by applying a technique of Maximum Likelihood Estimate (MLE) using Goel-Okumoto model. The MLE is consistent and asymptotically normally distributed as the sample size increases [8].

The constants ‘a’, ‘b’ which appear in the mean value function for HLD (1) and hence in NHPP, in intensity function (2) (error detection rate) and various other expressions are called parameters of the model. In order to have an assessment of the software reliability ‘a’, ‘b’ are to be known or they are to be estimated from a software failure data.

Suppose we have ‘n’ time instants at which the first, second, third..., \( n^{th} \) failures of a software are experienced. In other words if \( S_k \) is the total time to the \( k^{th} \) failure, \( S_k \) is an observation of random variable \( S_k \) and ‘n’ such failures are successively recorded. The joint probability of such failure time realizations \( S_1, S_2, S_3, \ldots, S_n \) is

\[
L = e^{-m(s_n)} \prod_{k=1}^{n} \lambda(s_k) \quad (5)
\]

The function given in equation (5) is called the likelihood function of the given failure data. Values of ‘a’, ‘b’ that would maximize \( L \) are called maximum likelihood estimators (MLEs) and the method is called maximum likelihood (ML) method of estimation. Accordingly ‘a’, ‘b’ would be solutions of the equations

\[
\frac{\partial \log L}{\partial a} = 0, \quad \frac{\partial \log L}{\partial b} = 0, \quad \frac{\partial^2 \log L}{\partial b^2} = 0
\]

Substituting the expressions for \( m(t), \lambda(t) \) given by equations (5) and (6) in equation (7), taking logarithms, differentiating with respect to ‘a’, ‘b’ and equating to zero, after some joint simplification we get

\[
a = n \left[ \frac{1}{1 - e^{-bS_n}} \right] \quad (6)
\]

\[
g(b) = \sum_{k=1}^{n} \frac{s_k}{b} - 2 \sum_{k=1}^{n} \frac{s_k e^{-bS}}{1 + e^{-bS}} - 2 \frac{s_k e^{-bS}}{1 - e^{-bS}} \left[1 - \frac{n}{1 - e^{-bS}}\right] = 0 \quad (7)
\]

\[
g'(b) = \frac{n}{b^2} + 2 \sum_{k=1}^{n} \frac{s_k e^{-bS}}{1 + e^{-bS}} + 2 s_n^2 e^{-bS_n} \left[\frac{1}{1 - e^{-bS_n}}\right] - n + e^{-2bS_n} - 2e^{-bS_n} \quad (8)
\]
The value of ‘b’ can be obtained using Newton-Raphson method which when substituted in equation (6) gives value of ‘a’.

For Exponential Distribution, to estimate ‘a’ and ‘b’, for a sample of n units, first obtain the likelihood function (L):

\[ L = e^{-m(s_n)} \prod_{k=1}^{n} m(s_k) \] (9)

To solve equation (3.1), we take the logarithm of both sides

\[ \log L = -m(s_n) + \sum_{k=1}^{n} \log m(s_k) \] (10)

In order to estimate the parameters ‘a’ and ‘b’, we can take the derivative of the above equation (10) with respect to ‘a’ and ‘b’, and equating these derivatives to zero and solving the resulting equations for ‘a’ and ‘b’, we find the estimates as follows.

\[ a = \frac{n}{\sum_{k=1}^{n} s_k - \frac{n}{b} + n s_n \frac{e^{-bs_n}}{1-e^{-b}}} \] (11)

\[ g(b) = \sum_{k=1}^{n} s_k \frac{n}{b} - n s_n \frac{e^{-bs_n}}{1-e^{-b}} \] (12)

\[ g'(b) = \frac{n}{b^2} - n s_n \left[ \frac{1}{1-e^{-bs_n}} + \frac{e^{-bs_n}}{(1-e^{-bs_n})^2} \right] e^{-bs_n} \] (13)

IV. CONTROL CHARTS: TO ESTIMATE THE TIME BETWEEN FAILURES

The selection of proper SPC charts is essential to effective statistical process control implementation and use. There are many charts which use statistical techniques. It is important to use the best chart for the given data, situation and need [9]. There are advances charts that provide more effective statistical analysis. The basic types of advanced charts, depending on the type of data are the variable and attribute charts. Variable control charts are designed to control product or process parameters which are measured on a continuous measurement scale. X-bar, R charts are variable control charts.

Attributes are characteristics of a process which are stated in terms of good are bad, accept or reject, etc. Attribute charts are not sensitive to variation in the process as variables charts. However, when dealing with attributes and used properly, especially by incorporating a real time pareto analysis, they can be effective improvement tools. For attribute data there are : p-charts, c-charts, np-charts, and u-charts. We have named the control chart as Failures Control Chart for HLD and Mean Value Chart for exponential Distribution in this paper. The said control chart helps to assess the software failure phenomena on the basis of the given inter-failure time data[10].

For a software system during normal operation, failures are random events caused by, for example, problem in design or analysis and in some cases insufficient testing of software. In this paper we applied Half Logistic Distribution [4][5] and Exponential Distribution [6] to time between failures data. This distribution uses cumulative time between failure data for reliability monitoring.

Equating the pdf of HLD’s m(t) (1) to 0.99865, 0.00135, 0.5 and the respective control limits are given by.

\[ m(t) = a \frac{1 - e^{-bt}}{1 + e^{-bt}} = 0.99865 \]

It gives

\[ t = \frac{7.300122639}{b} = t_u \] (14)

Similarly

\[ t = \frac{0.002700002}{b} = t_l \] (15)

\[ t = \frac{1.098612289}{b} = t_c \] (16)

The control limits are such that the point above the m(t_u) (14)(UCL) is an alarm signal. A point below the m(t_l)(15) (LCL) is an indication of better quality of software. A point within the control limits indicates stable process.

Equating the pdf of Exponential’s m(t) (3) to 0.00135, 0.99865, and 0.5 and the respective control limits are given by

\[ T_u = (1 - e^{-bt}) = 0.99865 \] (17)

\[ T_c = (1 - e^{-bt}) = 0.5 \] (18)

\[ T_l = (1 - e^{-bt}) = 0.00135 \] (19)
The procedure of a failures control chart for failure software process will be illustrated with an example here. Table 1 shows the time between failures of a software product [10].

### Table -1: Time between failures of a component[8]

<table>
<thead>
<tr>
<th>Failure number</th>
<th>Time between Failure (hrs)</th>
<th>Failure number</th>
<th>Time between Failure (hrs)</th>
<th>Failure number</th>
<th>Time between Failure (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>11</td>
<td>0.47</td>
<td>21</td>
<td>70.47</td>
</tr>
<tr>
<td>2</td>
<td>1.44</td>
<td>12</td>
<td>6.23</td>
<td>22</td>
<td>17.07</td>
</tr>
<tr>
<td>3</td>
<td>22.47</td>
<td>13</td>
<td>3.39</td>
<td>23</td>
<td>3.99</td>
</tr>
<tr>
<td>4</td>
<td>1.36</td>
<td>14</td>
<td>9.11</td>
<td>24</td>
<td>176.06</td>
</tr>
<tr>
<td>5</td>
<td>3.43</td>
<td>15</td>
<td>2.18</td>
<td>25</td>
<td>81.07</td>
</tr>
<tr>
<td>6</td>
<td>13.2</td>
<td>16</td>
<td>15.53</td>
<td>26</td>
<td>2.27</td>
</tr>
<tr>
<td>7</td>
<td>5.15</td>
<td>17</td>
<td>25.72</td>
<td>27</td>
<td>15.63</td>
</tr>
<tr>
<td>8</td>
<td>3.83</td>
<td>18</td>
<td>2.79</td>
<td>28</td>
<td>120.78</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>19</td>
<td>1.92</td>
<td>29</td>
<td>30.81</td>
</tr>
<tr>
<td>10</td>
<td>12.97</td>
<td>20</td>
<td>4.13</td>
<td>30</td>
<td>34.19</td>
</tr>
</tbody>
</table>

Table 2 shows the time between failures (cumulative) in hours, corresponding \( m(t) \) and successive difference between \( m(t) \)'s for HLD.

### Table 2- Successive difference of mean value function (m(t)) for HLD

<table>
<thead>
<tr>
<th>Failure number</th>
<th>Time between Failure (hrs) (cumulative)</th>
<th>m(t)</th>
<th>Successive Difference of m(t)</th>
<th>Failure number</th>
<th>Time between Failure (hrs) (cumulative)</th>
<th>m(t)</th>
<th>Successive Difference of m(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>2.364302301</td>
<td>0.112954122</td>
<td>15</td>
<td>136.25</td>
<td>10.35263373</td>
<td>1.07893884</td>
</tr>
<tr>
<td>2</td>
<td>31.46</td>
<td>2.477256423</td>
<td>1.75247064</td>
<td>16</td>
<td>151.78</td>
<td>11.43156761</td>
<td>1.720058168</td>
</tr>
<tr>
<td>3</td>
<td>53.93</td>
<td>4.229727062</td>
<td>0.105331424</td>
<td>17</td>
<td>177.5</td>
<td>13.15162578</td>
<td>0.181300217</td>
</tr>
<tr>
<td>4</td>
<td>55.29</td>
<td>4.335058486</td>
<td>0.265209098</td>
<td>18</td>
<td>180.29</td>
<td>13.332926</td>
<td>0.12414598</td>
</tr>
<tr>
<td>5</td>
<td>58.72</td>
<td>4.600267585</td>
<td>1.014117756</td>
<td>19</td>
<td>182.21</td>
<td>13.45707198</td>
<td>0.26531761</td>
</tr>
<tr>
<td>6</td>
<td>71.92</td>
<td>5.614385341</td>
<td>0.392552448</td>
<td>20</td>
<td>186.34</td>
<td>13.72238914</td>
<td>4.14567564</td>
</tr>
<tr>
<td>7</td>
<td>77.07</td>
<td>6.006937788</td>
<td>0.290696243</td>
<td>21</td>
<td>256.81</td>
<td>17.8680649</td>
<td>0.892060255</td>
</tr>
<tr>
<td>8</td>
<td>80.9</td>
<td>6.297634032</td>
<td>1.572927375</td>
<td>22</td>
<td>273.88</td>
<td>18.76012516</td>
<td>0.202130338</td>
</tr>
<tr>
<td>9</td>
<td>101.9</td>
<td>7.870561407</td>
<td>0.951568845</td>
<td>23</td>
<td>277.87</td>
<td>18.9622555</td>
<td>6.671109936</td>
</tr>
<tr>
<td>10</td>
<td>114.87</td>
<td>8.822130252</td>
<td>0.304170022</td>
<td>24</td>
<td>453.93</td>
<td>25.6336543</td>
<td>1.838854653</td>
</tr>
<tr>
<td>11</td>
<td>115.34</td>
<td>8.856300274</td>
<td>0.450773778</td>
<td>25</td>
<td>535</td>
<td>27.47222009</td>
<td>0.04287584</td>
</tr>
<tr>
<td>12</td>
<td>121.57</td>
<td>9.307074052</td>
<td>0.244275895</td>
<td>26</td>
<td>537.27</td>
<td>27.51509593</td>
<td>0.283919499</td>
</tr>
<tr>
<td>13</td>
<td>124.97</td>
<td>9.551349947</td>
<td>0.647546483</td>
<td>27</td>
<td>552.9</td>
<td>27.79901542</td>
<td>1.634249439</td>
</tr>
<tr>
<td>14</td>
<td>134.07</td>
<td>10.19889643</td>
<td>0.1537373</td>
<td>28</td>
<td>673.68</td>
<td>29.43326486</td>
<td>0.290356701</td>
</tr>
<tr>
<td>15</td>
<td>136.25</td>
<td>10.35263373</td>
<td>1.078938844</td>
<td>29</td>
<td>704.49</td>
<td>29.72362156</td>
<td>0.276439943</td>
</tr>
</tbody>
</table>
Table 3 shows the time between failures (cumulative) in hours, corresponding $m(t)$ and successive difference between $m(t)$’s for Exponential Distribution.

<table>
<thead>
<tr>
<th>Failure No</th>
<th>Cumulative failures</th>
<th>$m(t)$</th>
<th>Successive Difference $m(t)$</th>
<th>Failure No</th>
<th>Cumulative failures</th>
<th>$m(t)$</th>
<th>Successive Difference $m(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>3.554</td>
<td>0.160109911</td>
<td>16</td>
<td>151.78</td>
<td>14.32</td>
<td>1.683122175</td>
</tr>
<tr>
<td>2</td>
<td>31.46</td>
<td>3.714</td>
<td>2.383586679</td>
<td>17</td>
<td>157.54</td>
<td>16.00</td>
<td>0.172478506</td>
</tr>
<tr>
<td>3</td>
<td>53.93</td>
<td>6.092</td>
<td>0.137569469</td>
<td>18</td>
<td>180.29</td>
<td>13.18</td>
<td>0.117592238</td>
</tr>
<tr>
<td>4</td>
<td>55.29</td>
<td>6.235</td>
<td>0.34368381</td>
<td>19</td>
<td>182.21</td>
<td>16.29</td>
<td>0.249934515</td>
</tr>
<tr>
<td>5</td>
<td>58.72</td>
<td>6.579</td>
<td>1.2799.4674</td>
<td>20</td>
<td>184.36</td>
<td>16.54</td>
<td>3.690669037</td>
</tr>
<tr>
<td>6</td>
<td>71.92</td>
<td>7.589</td>
<td>0.481483904</td>
<td>21</td>
<td>186.81</td>
<td>20.23</td>
<td>0.74936348</td>
</tr>
<tr>
<td>7</td>
<td>77.07</td>
<td>8.340</td>
<td>0.351758112</td>
<td>22</td>
<td>187.38</td>
<td>20.98</td>
<td>0.167971248</td>
</tr>
<tr>
<td>8</td>
<td>80.9</td>
<td>8.693</td>
<td>1.83637975</td>
<td>23</td>
<td>188.59</td>
<td>21.13</td>
<td>5.293999998</td>
</tr>
<tr>
<td>9</td>
<td>101.9</td>
<td>10.529</td>
<td>1.060330131</td>
<td>24</td>
<td>189.11</td>
<td>26.45</td>
<td>1.441652925</td>
</tr>
<tr>
<td>10</td>
<td>114.87</td>
<td>11.586</td>
<td>0.037140054</td>
<td>25</td>
<td>190.32</td>
<td>27.89</td>
<td>0.34077054</td>
</tr>
<tr>
<td>11</td>
<td>115.34</td>
<td>11.627</td>
<td>0.489356342</td>
<td>26</td>
<td>190.92</td>
<td>27.92</td>
<td>0.226497314</td>
</tr>
<tr>
<td>12</td>
<td>121.57</td>
<td>12.114</td>
<td>2.66247815</td>
<td>27</td>
<td>193.12</td>
<td>28.15</td>
<td>1.34863376</td>
</tr>
<tr>
<td>13</td>
<td>124.96</td>
<td>12.377</td>
<td>0.684916199</td>
<td>28</td>
<td>193.89</td>
<td>29.50</td>
<td>0.252475261</td>
</tr>
<tr>
<td>14</td>
<td>134.07</td>
<td>13.062</td>
<td>0.160265529</td>
<td>29</td>
<td>194.97</td>
<td>29.75</td>
<td>0.246358013</td>
</tr>
<tr>
<td>15</td>
<td>136.25</td>
<td>13.223</td>
<td>1.10251794</td>
<td>30</td>
<td>196.21</td>
<td>29.99</td>
<td>0.29990315</td>
</tr>
</tbody>
</table>

The values of ‘a’ and ‘b’ are computed by using the well know iterative Newton-Rapson method. These values are used to compute, $T_u$, $T_l$, $T_c$ i.e. UCL, LCL, CL

For Half Logistic Distribution, the values of a and b are 31.524466 and 0.005006 and

$m(T_u)/UCL = 31.6767602$
$m(T_l)/LCL = 0.008546968$
$m(T_c)/CL = 21.1321140$

The values of $m(t)$ at $T_u$, $T_l$, $T_c$ and at the given 30 inter-failure times are calculated. Then the $m(t)$’s are taken, which leads to 29 values. The graph with the said inter-failure times 1 to 30 on X-axis, the 29 values of $m(t)$’s on Y-axis, and the 3 control lines parallel to X-axis at $m(T_l)$, $m(T_u)$, $m(T_c)$ respectively constitutes Failure Control Chart and Mean Value Chart to assess the software failure phenomena on the basis of the given inter-failures time data.
VI. CONCLUSION

The failures control chart (Figure 1) and the mean value chart (Figure 2) exemplifies that, the first out-of-control situation is noticed at the 10\textsuperscript{th} failure with the corresponding successive difference of \(m(t)\) falling below the LCL. It results in an earlier and hence preferable out-of-control for the product. The assignable cause for this is to be investigated and promoted. In comparison, the time control chart for the same data given in Xie et al [10] reveals an out-of-control for the first time above the UCL at 23\textsuperscript{rd} failure. Since the data of the time-control chart are inter-failure times, a point above UCL for time-control chart is also a preferable criterion for the product. The time control chart gives the first out-of-control signal in a positive way, but at the 23\textsuperscript{rd} failure. Hence it is claimed that the failures control chart and mean value chart detects out-of-control in a positive way much earlier than the time-control chart. Therefore, earlier detections are possible in failures control chart and mean value chart. Since both control mechanisms are making the detection at the same point, either mechanism based on Half Logistic Distribution [5] or Exponential Distribution [6] is preferable.

References

[2] Fedon Kadiferli; A Reliability Model for a large-scale software system; Thesis; Bogazici University, 1987


K Ramchand H Rao, received Master’s degree in Technology with Computer Science from Dr. M.G.R University, Chennai, Tamilnadu, India. He is currently working as Associate Professor and Head of the Department, in the Department of Computer Science, A.S.N. Degree College, Tenali, which is affiliated to Acharya Nagarjuna University. He has 18 years teaching experience and 2 years of Industry experience at Morgan Stanly, USA as Software Analyst. He is currently pursuing Ph.D., at Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. His research area is software Engineering. He has published several papers in National & International Journals.

Bandla Srinivasa Rao received the Master Degree in Computer Science and Engineering from Dr. MGR Deemed University, Chennai, Tamil Nadu, India. He is currently working as Associate Professor in PG Department of Computer Applications, VRS & YRN College, Chirala, Andhra Pradesh, India. His research interests include software reliability, Cryptography and Computer Networks. He has published several papers in National and International Journals.

Dr. R. Satya Prasad received Ph.D. degree in Computer Science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Andhra Pradesh. He received gold medal from Acharya Nagarjuna University for his outstanding performance in Masters Degree. He is currently working as Associate Professor and H.O.D, in the Department of Computer Science & Engineering, Acharya Nagarjuna University. His current research is focused on Software Engineering, Software reliability. He has published several papers in National & International Journals.
A Comparative Study of Software Reliability Models Using SPC on Ungrouped Data

G. Krishna Mohan  
Reader,  
Dept. of Computer Science  
P.B. Siddhartha college  
Vijayawada.  
km_mm_2000@yahoo.com

B. Srinivasa Rao  
Associate Professor  
Dept. of Computer Science  
VRS & YRN College  
Chirala  
sreenibandla@yahoo.com

Dr. R. Satya Prasad  
Associate Professor  
Dept. of CS&E.  
Acharya Nagarjuna University  
Nagarjuna Nagar.  
profrsp@gmail.com

Abstract—Control charts are widely used for process monitoring. Software reliability process can be monitored efficiently by using Statistical Process Control (SPC). It assists the software development team to identify failures and actions to be taken during software failure process and hence, assures better software reliability. If not many, few researchers proposed SPC based software reliability monitoring techniques to improve Software Reliability Process. In this paper we propose a control mechanism based on the cumulative quantity between observations of time domain failure data using mean value function of Weibull and Goel-Okumoto distribution, which are based on Non Homogenous Poisson Process (NHPP). The Maximum Likelihood Estimation (MLE) method is used to derive the point estimators of the distributions.

Keywords—Statistical Process Control, Software reliability, Weibull Distribution, Goel-Okumoto distribution, Mean Value function, Probability limits, Control Charts.

I. 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 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 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 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 nonrandom 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 parameter a, b should normally be estimated with the data from the failure process. Since a, b are the parameters in the proposed distributions, any traditional estimator can be used.

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 should in fact depend on the actual product or process [9].

II. LITERATURE SURVEY

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: \( f(t; \theta_1, \theta_2, \ldots, \theta_k) \). 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: \( m(t) = aF(t) \), where, F(t) is a cumulative distribution 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

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proved to be quite successful in practical software reliability engineering [4]. 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 [5,6].

Let \( N(t), t \geq 0 \) be the cumulative number of software failures by time ‘t’. \( m(t) \) is the mean value function, representing the expected number of software failures by time ‘t’. \( \lambda(t) \) is the failure intensity function, which is proportional to the residual fault content. Thus \( m(t) = a(1 - e^{-dt}) \) and \( \lambda(t) = \frac{dm(t)}{dt} = b(a - m(t)) \), where ‘a’ denotes the initial number of faults contained in a program and ‘b’ represents the fault detection rate. In software reliability, the initial number of faults and the fault detection rate are always unknown. The maximum likelihood technique can be used to evaluate the unknown parameters. In NHPP SRGM \( \lambda(t) \) can be expressed in a more general way as \( \lambda(t) = \frac{dm(t)}{dt} = b(t)[a(t) - m(t)] \), where \( a(t) \) is the time-dependent fault content function which includes the initial and introduced faults in the program and \( b(t) \) is the time-dependent fault detection rate. A constant \( a(t) \) implies the perfect debugging assumption, i.e no new faults are introduced during the debugging process. A constant \( b(t) \) implies the imperfect debugging assumption, i.e when the faults are removed, then there is a possibility to introduce new faults.

B. Goel-Okumoto distribution

The Goel-Okumoto model is a simple NonHomogenous Poisson Process (NHPP) model with the mean value function \( m(t) = a(1 - e^{-at}) \) [12]. Where the parameter ‘a’ is the number of initial faults in the software and the parameter ‘b’ is the fault detection rate. The corresponding failure intensity
function is given by \( \lambda(t) = abe^{-bt} \). The probability density function of a Goel-Okumoto model has the form: \( f(t) = be^{-bt} \). The corresponding cumulative distribution function is: \( F(t) = 1 - e^{-bt} \).

C. Weibull distribution

The Weibull distribution is a generalization of exponential distribution, which is recovered for \( \beta = 1 \). Although the exponential distribution has been widely used for times-between-event, Weibull distribution is more suitable as it is more flexible and is able to deal with different types of aging phenomenon in reliability. Hence in reliability monitoring of equipment failures, the Weibull distribution is a good alternative. The probability density function of a two-parameter Weibull model has the form:
\[
 f(t) = b\beta t^{\beta-1} e^{-bt^\beta} \quad \text{where } b > 0 \text{ is a scale parameter and } \beta > 0 \text{ is a shape parameter.}
\]
The corresponding cumulative distribution function is:
\[
 F(t) = 1 - e^{-bt^\beta} \quad .
\]

The mean value function \( m(t) = ab \frac{1-e^{-(bt^\beta)}}{b\beta} \) . The failure intensity function is given as:
\[
 \lambda(t) = b\beta t^{\beta-1} e^{-bt^\beta} \quad .
\]

D. MLE (Maximum Likelihood) Parameter Estimation

The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. The method of maximum likelihood is considered to be more robust (with some exceptions) and yields estimators with good statistical properties. In other words, MLE methods are versatile and apply to many models and to different types of data. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense. Using today's computer power, however, mathematical complexity is not a big obstacle. If we conduct an experiment and obtain \( N \) independent observations, \( t_1, t_2, \ldots, t_N \). The likelihood function [7] may be given by the following product:
\[
 L(t_1, t_2, \ldots, t_N | \theta_1, \theta_2, \ldots, \theta_N) = \prod_{i=1}^{N} f(t_i; \theta_1, \theta_2, \ldots, \theta_N) 
\]
Likely hood function by using \( \lambda(t) \) is:
\[
 L = \prod_{i=1}^{n} \lambda(t_i)
\]
The logarithmic likelihood function is given by:
\[
 \log L = \log \left( \prod_{i=1}^{n} \lambda(t_i) \right) = \sum_{i=1}^{n} \log \left[ \lambda(t_i) \right] - m(t_n)
\]
The maximum likelihood estimators (MLE) of \( \theta_1, \theta_2, \ldots, \theta_k \) are obtained by maximizing \( L \) or \( \Lambda \), where \( \Lambda \) is \( \ln L \). By maximizing \( \Lambda \), which is much easier to work with than \( L \), the maximum likelihood estimators (MLE) of \( \theta_1, \theta_2, \ldots, \theta_k \) are the simultaneous solutions of \( k \) equations such as:
\[
 \frac{\partial \log L}{\partial \theta_j} = 0, \quad j = 1, 2, \ldots, k
\]

The parameters ‘a’ and ‘b’ are estimated as follows. The parameter ‘b’ is estimated by iterative Newton Raphson Method using \( b_{n+1} = b_n - \frac{g(b_n)}{g'(b_n)} \), which is substituted in finding ‘a’.

III. ILLUSTRATING THE MLE METHOD

A. Goel-Okumoto parameter estimation

The likelihood function is given as, \( L = \prod_{i=1}^{N} abe^{-(bt_i)} \).
Taking the natural logarithm on both sides, The Log Likelihood function is given as:
\[
 \log L = \sum_{i=1}^{N} \log(abe^{-(bt_i)}) - d[1 - e^{-(bt_i)}] 
\]
Taking the Partial derivative with respect to ‘a’ and equating to ‘0’, (i.e \( \frac{\partial \log L}{\partial a} = 0 \)).
\[
 a = \left[ \frac{n}{1 - e^{-(bt_i)}} \right]
\]
Taking the Partial derivative with respect to ‘b’ and equating to ‘0’, (i.e \( g(b) = \frac{\partial \log L}{\partial b} = 0 \)).
\[
 g(b) = \sum_{i=1}^{n} t_i - \frac{n}{b} + n\lambda \cdot \left[ \frac{e^{-bt_i}}{1 - e^{-bt_i}} \right] = 0
\]
Taking the partial derivative again with respect to ‘b’ and equating to ‘0’, (i.e \( g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0 \)).
\[
 g'(b) = \frac{n}{b^2} - \frac{n\lambda}{b} \cdot \left[ \frac{1}{1 - e^{-bt_i}} + \frac{e^{-bt_i}}{(1 - e^{-bt_i})^2} \right] \cdot e^{-bt_i} = 0
\]

B. Weibull parameter estimation

The likelihood function, assuming \( \beta = 2 \) (Rayleigh) is given as, \( L = \prod_{i=1}^{N} 2ab^2t_i e^{-(bt_i)^2} \).

Taking the natural logarithm on both sides, The Log Likelihood function is given as:
\[
 \log L = \sum_{i=1}^{N} \log(2ab^2t_i e^{-(bt_i)^2}) - d[1 - e^{-(bt_i)^2}] 
\]
Taking the Partial derivative with respect to ‘a’ and equating to ‘0’, (i.e \( \frac{\partial \log L}{\partial a} = 0 \)).
\[
 a = \left[ \frac{n}{1 - e^{-(bt_i)^2}} \right]
\]
Taking the Partial derivative with respect to ‘b’ and equating to ‘0’, (i.e \( g(b) = \frac{\partial \log L}{\partial b} = 0 \)).
\[
 g(b) = \frac{2n}{b} - 2b \sum_{i=1}^{n} t_i^2 - \frac{2n \lambda b^2}{(1 - e^{-(bt_i)^2})} = 0
\]
Taking the partial derivative again with respect to ‘b’ and equating to ‘0’. (i.e $g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0$).

$g'(b) = 2n\left(\frac{-1}{b^2}\right) - 2\sum_{i=1}^{n} t_i - 2m_n \left\{ \left(1 - e^{-(b_n)^t}\right) - \frac{2b_n t_i e^{-(b_n)^t}}{1 - e^{-(b_n)^t}} \right\}$

C. Distribution of Time between failures

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 Goel-Okumoto and Weibull distributions. The use of cumulative quality is a different and new approach, which is of particular advantage in reliability.

‘$\hat{a}$’ and ‘$\hat{b}$’ are Maximum Likely hood Estimates of parameters and the values can be computed using iterative method for the given cumulative time between failures data [10] shown in table 1. Using ‘a’ and ‘b’ values we can compute $m(t)$.

### Table 1. Time between failures of a software

<table>
<thead>
<tr>
<th>Failure Number</th>
<th>Time between failure(h)</th>
<th>Failure Number</th>
<th>Time between failure(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.02</td>
<td>16</td>
<td>15.53</td>
</tr>
<tr>
<td>2</td>
<td>1.44</td>
<td>17</td>
<td>25.72</td>
</tr>
<tr>
<td>3</td>
<td>22.47</td>
<td>18</td>
<td>2.79</td>
</tr>
<tr>
<td>4</td>
<td>1.36</td>
<td>19</td>
<td>1.92</td>
</tr>
<tr>
<td>5</td>
<td>3.43</td>
<td>20</td>
<td>4.13</td>
</tr>
<tr>
<td>6</td>
<td>13.2</td>
<td>21</td>
<td>70.47</td>
</tr>
<tr>
<td>7</td>
<td>3.15</td>
<td>22</td>
<td>17.07</td>
</tr>
<tr>
<td>8</td>
<td>3.83</td>
<td>23</td>
<td>3.99</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>24</td>
<td>176.06</td>
</tr>
<tr>
<td>10</td>
<td>12.97</td>
<td>25</td>
<td>81.07</td>
</tr>
<tr>
<td>11</td>
<td>0.47</td>
<td>26</td>
<td>2.27</td>
</tr>
<tr>
<td>12</td>
<td>6.23</td>
<td>27</td>
<td>15.63</td>
</tr>
<tr>
<td>13</td>
<td>3.39</td>
<td>28</td>
<td>120.78</td>
</tr>
<tr>
<td>14</td>
<td>9.11</td>
<td>29</td>
<td>30.81</td>
</tr>
<tr>
<td>15</td>
<td>2.18</td>
<td>30</td>
<td>34.19</td>
</tr>
</tbody>
</table>

Assuming an acceptable probability of false alarm of 0.27%, the control limits can be obtained as [10]:

$T_U = 1 - e^{-(b_n)^t} = 0.99865$

$T_C = 1 - e^{-(b_n)^t} = 0.5$

$T_L = 1 - e^{-(b_n)^t} = 0.00135$

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 and figure 2. A point below the control limit $m(t_L)$ indicates an alarming signal. A point above the control limit $m(t_U)$ indicates better quality. If the points are falling within the control limits, it indicates the software process is in stable condition [11]. The values of parameter estimates and the control limits are given in table 2 and 3 respectively.

### Table 2. Parameter estimates

<table>
<thead>
<tr>
<th>model</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO</td>
<td>31.698171</td>
<td>0.003962</td>
</tr>
<tr>
<td>Weibull</td>
<td>30.051592</td>
<td>0.003416</td>
</tr>
</tbody>
</table>

### Table 3. Control limits.

<table>
<thead>
<tr>
<th>model</th>
<th>$m(t_U)$</th>
<th>$m(t_C)$</th>
<th>$m(t_L)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO</td>
<td>31.676760</td>
<td>21.132114</td>
<td>0.085469</td>
</tr>
<tr>
<td>Weibull</td>
<td>30.011170</td>
<td>15.025870</td>
<td>0.040570</td>
</tr>
</tbody>
</table>

### Table 4. Mean successive differences of GO

<table>
<thead>
<tr>
<th>FN</th>
<th>m(t)</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.554578</td>
<td>0.160101</td>
</tr>
<tr>
<td>2</td>
<td>3.714687</td>
<td>2.383587</td>
</tr>
<tr>
<td>3</td>
<td>6.098274</td>
<td>0.137569</td>
</tr>
<tr>
<td>4</td>
<td>6.235844</td>
<td>0.343684</td>
</tr>
<tr>
<td>5</td>
<td>6.579527</td>
<td>1.279946</td>
</tr>
<tr>
<td>6</td>
<td>7.859432</td>
<td>0.481484</td>
</tr>
<tr>
<td>7</td>
<td>8.346016</td>
<td>0.351758</td>
</tr>
<tr>
<td>8</td>
<td>8.692674</td>
<td>1.836638</td>
</tr>
<tr>
<td>9</td>
<td>10.529312</td>
<td>1.060330</td>
</tr>
<tr>
<td>10</td>
<td>11.589642</td>
<td>0.037410</td>
</tr>
<tr>
<td>11</td>
<td>11.627052</td>
<td>0.489356</td>
</tr>
<tr>
<td>12</td>
<td>12.116408</td>
<td>0.261248</td>
</tr>
<tr>
<td>13</td>
<td>12.377656</td>
<td>0.684916</td>
</tr>
<tr>
<td>14</td>
<td>13.062573</td>
<td>0.160266</td>
</tr>
<tr>
<td>15</td>
<td>13.222838</td>
<td>1.105258</td>
</tr>
<tr>
<td>16</td>
<td>14.325356</td>
<td>1.683122</td>
</tr>
<tr>
<td>17</td>
<td>16.008478</td>
<td>0.172479</td>
</tr>
<tr>
<td>18</td>
<td>13.180956</td>
<td>0.117592</td>
</tr>
<tr>
<td>19</td>
<td>16.298549</td>
<td>0.249935</td>
</tr>
<tr>
<td>20</td>
<td>16.548483</td>
<td>3.690661</td>
</tr>
<tr>
<td>21</td>
<td>20.239144</td>
<td>0.749363</td>
</tr>
<tr>
<td>22</td>
<td>20.988508</td>
<td>0.167971</td>
</tr>
<tr>
<td>23</td>
<td>21.156479</td>
<td>5.293999</td>
</tr>
<tr>
<td>24</td>
<td>26.450479</td>
<td>1.441653</td>
</tr>
<tr>
<td>25</td>
<td>27.892132</td>
<td>0.034077</td>
</tr>
<tr>
<td>26</td>
<td>27.926209</td>
<td>0.226497</td>
</tr>
<tr>
<td>27</td>
<td>28.132706</td>
<td>1.348363</td>
</tr>
<tr>
<td>28</td>
<td>29.501069</td>
<td>0.252475</td>
</tr>
<tr>
<td>29</td>
<td>29.735345</td>
<td>0.246358</td>
</tr>
<tr>
<td>30</td>
<td>29.999903</td>
<td></td>
</tr>
</tbody>
</table>
The Mean Value chart of weibull shows that the 1st, 10th, and 25th failure data has fallen below \( m(t_f) \). The Mean Value Chart of weibull shows that the 1st, 10th and 25th failure data has fallen below \( m(t_f) \). The successive differences of mean values below \( m(t_f) \) indicates the failure process. In the present scenario, it is significantly early detection of failure through weibull using Mean Value Chart. The software quality is determined by detecting failures at an early stage for the corresponding \( m(t_f) \), which is below \( m(t_f) \). It indicates that the failure process is detected at an early stage compared with Xie et al. (2002) control chart [10], which detects the failure at 23rd point for the inter failure data above the UCL. Hence our proposed Mean Value Chart detects out of control failure at an earlier stage than the situation in the time control chart. The early detection of software failure will improve the software reliability. When the time between failures is less than LCL, it is likely that there are assignable causes leading to significant process deterioration and it should be investigated. On the other hand, when the time between failures has exceeded the UCL, there are probably reasons that have lead to significant improvement.

**REFERENCES**

© 2012, IJARCSSE All Rights Reserved