CHAPTER 3
PREDICTION OF PARETO TYPE IV SOFTWARE RELIABILITY

3.1 Introduction

Software reliability is a key part in software quality. It is the probability that given software functions without failure in a given environmental condition during a specified time. That is, it is the probability of failure-free execution of the software for a specified time in a specified environment. Software reliability can be improved by increasing the testing effort and by correcting detected faults. Reliability tends to change continuously during testing due to the addition of problems in new code or to the removal of problems by debugging errors. There are two important parts to provide reliability: fault detection and fault isolation. The design has to consider both aspects. Since performance requirements influence the selection of data structures and algorithms, it is important to check performance factors at the design phase.

To estimate the performance of the design, the information on usage pattern, design structure, and installation characteristics are needed. The specifications describe the level and what security looks like while design considers its implementation. So, good engineering methods can largely improve software reliability. The study of software reliability can be categorized into three parts: modelling, measurement and improvement. Software reliability modelling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, one of the well-known and simplest model is our Pareto type IV model.

Most software models contain the following parts: assumptions, factors, and a mathematical function that relates the reliability with the factors. The mathematical function is usually higher order exponential or logarithmic. Software modelling techniques can be divided into two subcategories: prediction modelling and estimation modelling. Both kinds of modelling techniques are based on observing and accumulating failure data and analyzing with statistical inference.
The content of this chapter is published in the following journal.

*Dr R.Satya Prasad, G.Sridevi. “Pareto Type IV Software Reliability Growth Model”, Elixir Comp, Sci. & Engg. 64 (2013):19124-19129. ISSN: 2229-712X.*

### 3.2 Model Formulation

Software reliability models can be classified according to probabilistic assumptions. When a Markov process represents the failure process; the resultant model is called Markovian Model. Second one is fault counting model which describes the failure phenomenon by stochastic process like Homogeneous Poisson Process (HPP), Non Homogeneous Poisson Process (NHPP) and Compound Poisson Process etc. A majority of failure count models are based upon NHPP described in the following lines.

A software system is subjected to failures at random times caused by errors present in the system. Let $N(t), t > 0$ be a counting process representing the cumulative number of failures by time ‘t’. Since there are no failures at $t=0$ we have

$$N(0) = 0$$

It is to assume that the number of software failures during non-overlapping time intervals do not affect each other. In other words, for any finite collection of times $t_1 < t_2 < \ldots < t_n$. The ‘n’ random variables $(t_i), \{N(t_2) - N(t_1)\}, \ldots, \{N(t_n) - N(t_{n-1})\}$ are independent. This implies that the counting process $\{N(t), t > 0\}$ has independent increments.

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

$$m(t)\begin{cases} 0, & t = 0 \\ a, & t \rightarrow \infty \end{cases}$$
Where ‘a’ is the expected number of software errors to be eventually detected. Suppose \( N(t) \) is known to have a Poisson probability mass function with parameters \( m(t) \) i.e.,

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

then \( N(t) \) is called an NHPP. Thus the stochastic behavior of software failure phenomena can be described through the \( N(t) \) process. Various time domain models have appeared in the literature (Kantam and Subbarao, 2009) which describe the stochastic failure process by an NHPP which differ in the mean value function \( m(t) \).

The proposed mean value function \( m(t) \) of Pareto Type IV model is given by

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

(3.2.1)

Where \( [m(t)/a] \) is the cumulative distribution function of Pareto type IV distribution (Johnson et al., 1994) for the present choice.

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

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

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

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

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

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

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

\[
= a \left[ 1 + \left( \frac{t}{c} \right) \right]^{-b}
\]
Let $S_k$ be the time between $(k-1)^{th}$ and $k^{th}$ failure of the software product. Let $X_k$ be the time up to the $k^{th}$ failure. Let us find out the probability that time between $(k-1)^{th}$ and $k^{th}$ failures, i.e., $S_k$ exceeds a real number ‘$s$’ given that the total time up to the $(k-1)^{th}$ failure is equal to $x$.

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

\[ R \frac{S_k}{X_{k-1}}(s/x) = e^{-[m(x+s) - m(s)]} \]  
\[ (3.2.2) \]

This Expression is called Software Reliability.

### 3.3 Maximum Likelihood Estimation

The parameters ‘$a$’, ‘$b$’ and ‘$c$’ are estimated by using Maximum Likelihood method and the values can be computed using iterative method for the given cumulative interval domain data. Using the estimators of ‘$a$’, ‘$b$’ and ‘$c$’ we can compute $m(t)$.

**Mathematical derivation for parameter estimation**

To estimate the values of ‘$a$’, ‘$b$’ and ‘$c$’ for a sample of $n$ units, first obtain the Likelihood function:

\[ L = \prod_{i=1}^{N} 2ab^2t_i e^{-(bt_i)^2} \]

Take the natural logarithm on both sides, The Log Likelihood function is given as:

\[ \log L = \sum_{i=1}^{k} (n_i - n_{i-1}) \cdot \log [m(t_i) - m(t_{i-1})] - m(t_k) \]  
\[ (3.3.1) \]

Take the mean value function of Pareto Type IV is of the form

\[ m(t) = a \left[ 1 - \left[ 1 + \left( \frac{t}{c} \right)^b \right] \right] \]  
\[ (3.3.2) \]
By substituting Equation (3.3.2) in Equation (3.3.1), we get

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

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

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

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

\[
\log L = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \log \left( \frac{c+t_i}{c} \right)^{b} - \log \left( \frac{c+t_{i-1}}{c} \right)^{b} \right] - a \left( \frac{c+t_k}{c} \right)^{b}
\]  \hspace{1cm} (3.3.3)

The parameter ‘a’ is estimated by taking the partial derivative w.r.t ‘a’ and equating to ‘0’. \( \text{i.e., } \frac{\partial \log L}{\partial a} = 0 \)

\[
\frac{\partial \log L}{\partial a} = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{a} + 0 \right] - 1 + \left( \frac{c+t_k}{c} \right)^{b}
\]

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

\[
\Rightarrow \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{a} \right] - 1 + \left( \frac{c+t_k}{c} \right)^{b} = 0
\]

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

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

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

(3.3.4)

By simplifying the Equation (3.3.3), we get

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

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

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

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

\[
LogL = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \log a + \log \left( \frac{(c+t_i)^b}{(c+t_{i-1})^b} \right) - b \log (c+t_{i-1}) - b \log (c+t_i) \right] - a + \left( \frac{c+t_k}{c} \right)^{-b}
\]
The parameter ‘b’ is estimated by using iterative Newton Raphson Method

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

which is substituted in finding ‘a’. Where \( g(b) \) & \( g'(b) \) are expressed as follows.

Taking the Partial derivative w.r.t ‘b’ and equating to ‘0’. (i.e \( g(b) = \frac{\partial \log L}{\partial b} = 0 \))

\[
\frac{\partial \log L}{\partial b} = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \begin{array}{c}
0 + \log c - \log(c + t_{i-1}) - \log(t_i + c) \\
+ \frac{1}{(c + t_i)^b - (c + t_{i-1})^b} \left\{ (c + t_i)^b \log(c + t_i) - (c + t_{i-1})^b \log(c + t_{i-1}) \right\}
\end{array} \right] - 0
\]

\[
+ a \left\{ \frac{c}{(c + t_k)^b} \log \left( \frac{c}{c + t_k} \right) \right\}
\]

Substitute ‘a’ value in the above equation, we get

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

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

Let c=1, we get

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

\[
+ \sum_{i=1}^{k} (n_i - n_{i-1}) \left\{ \frac{(1)^b}{(1 + t_k)^b - 1^b} \log \left( \frac{1}{1 + t_k} \right) \right\}
\]
\[ g(b) = \frac{\partial \log L}{\partial b} = 0 \]

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

(3.3.5)

Again taking the Partial derivative w.r.t ‘b’ and equating to ‘0’.

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

(3.3.6)

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

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

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

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

\[ g(c) = \frac{\partial \log L}{\partial c} = 0 \]
\[ \log L = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \log a + b \log c - b \log (t_i + c) - b \log (t_{i-1} + c) \right] + \log \left[ \left( t_i + c \right)^b - \left( t_{i-1} + c \right)^b \right] - a + \frac{ac^b}{(t_k + c)^b} \]

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

Substitute ‘a’ value in the above equation, we get

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

Again substitute b=1, we get

\[ \frac{\partial \log L}{\partial c} = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{c} \frac{1}{(t_i + c)} - \frac{1}{(t_i + c)} + \frac{(t_i + c)^0 - (t_{i-1} + c)^0}{(t_i + c)^1 - (t_{i-1} + c)^1} \right] + \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{(t_k + c)} \right] \]

\[ \frac{\partial \log L}{\partial c} = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{c} \frac{1}{(t_i + c)} - \frac{1}{(t_i + c)} + \frac{1-1}{(t_i + c)^1 - (t_{i-1} + c)^1} \right] + \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{(t_k + c)} \right] \]
\[ g(c) = \frac{\partial \text{Log} L}{\partial c} = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{c} - \frac{1}{(t_{i-1} + c)} - \frac{1}{(t_i + c)} \right] + \sum_{i=1}^{k} (n_i - n_{i-1}) \left\{ \frac{1}{(t_i + c)} \right\} \] (3.3.7)

Taking the partial derivative again w.r.t ‘c’ and equating to ‘0’.

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

\[ \therefore g'(c) = \sum_{i=1}^{k} (n_i - n_{i-1}) \left[ \frac{1}{(t_{i-1} + c)^2} + \frac{1}{(t_i + c)^2} - \frac{1}{c^2} \right] - \sum_{i=1}^{k} (n_i - n_{i-1}) \left\{ \frac{1}{(t_i + c)^2} \right\} \] (3.3.8)

### 3.4 Data Analysis

#### Datasets Phase 1 and Phase 2 from Pham (2005)

A set of failure data taken from Pham (2005) given in Table 3.4.1 and 3.4.2.

#### Datasets Release #1, #2, #3 and #4 from Alan Wood Tandem Computers (1996)

A set of failure data taken from Wood (1996) given in Table 3.4.3 to 3.4.6 consists of the observation time(week), CPU Hours and the number of failures detected per week : defects found.
Table 3.4.1. Phase 1 System Test data Pham (2005)

<table>
<thead>
<tr>
<th>Week Index</th>
<th>Exposure Time (cum. System test hours)</th>
<th>Fault</th>
<th>Cumulative Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>356</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>712</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1068</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1424</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1780</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>2136</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>2492</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>2848</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>3204</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>3560</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>3916</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>4272</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>4628</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>4984</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>5340</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>16</td>
<td>5696</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>17</td>
<td>6052</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>18</td>
<td>6408</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>19</td>
<td>6764</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>7120</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>21</td>
<td>7476</td>
<td>2</td>
<td>26</td>
</tr>
</tbody>
</table>
Table 3.4.2. Phase 2 System Test data Pham (2005)

<table>
<thead>
<tr>
<th>Week Index</th>
<th>Exposure Time (cum. System test hours)</th>
<th>Fault</th>
<th>Cumulative Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>416</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>832</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1248</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1664</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>2080</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>2496</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>2912</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>3328</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>3744</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>4160</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>4576</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>4992</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>5408</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>14</td>
<td>5824</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>6240</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>16</td>
<td>6656</td>
<td>1</td>
<td>37</td>
</tr>
<tr>
<td>17</td>
<td>7072</td>
<td>2</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>7488</td>
<td>0</td>
<td>39</td>
</tr>
<tr>
<td>19</td>
<td>7904</td>
<td>0</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>8320</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>8736</td>
<td>1</td>
<td>43</td>
</tr>
</tbody>
</table>
### Table 3.4.3. Data Set Release #1 (Alan Wood Tandem Computers -1996)

<table>
<thead>
<tr>
<th>Test Week</th>
<th>CPU Hours</th>
<th>Percent CPU Hours</th>
<th>Defects Found</th>
<th>Predicted Total Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>519</td>
<td>-</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>968</td>
<td>-</td>
<td>24</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1,430</td>
<td>-</td>
<td>27</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1,893</td>
<td>-</td>
<td>33</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2,490</td>
<td>-</td>
<td>41</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>3,058</td>
<td>-</td>
<td>49</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>3,625</td>
<td>-</td>
<td>54</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>4,422</td>
<td>-</td>
<td>58</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>5,218</td>
<td>-</td>
<td>69</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>5,823</td>
<td>58</td>
<td>75</td>
<td>98</td>
</tr>
<tr>
<td>11</td>
<td>6,539</td>
<td>65</td>
<td>81</td>
<td>107</td>
</tr>
<tr>
<td>12</td>
<td>7,083</td>
<td>71</td>
<td>86</td>
<td>116</td>
</tr>
<tr>
<td>13</td>
<td>7,487</td>
<td>75</td>
<td>90</td>
<td>123</td>
</tr>
<tr>
<td>14</td>
<td>7,846</td>
<td>78</td>
<td>93</td>
<td>129</td>
</tr>
<tr>
<td>15</td>
<td>8,205</td>
<td>82</td>
<td>96</td>
<td>129</td>
</tr>
<tr>
<td>16</td>
<td>8,564</td>
<td>86</td>
<td>98</td>
<td>134</td>
</tr>
<tr>
<td>17</td>
<td>8,923</td>
<td>89</td>
<td>99</td>
<td>139</td>
</tr>
<tr>
<td>18</td>
<td>9,282</td>
<td>93</td>
<td>100</td>
<td>138</td>
</tr>
<tr>
<td>19</td>
<td>9,641</td>
<td>96</td>
<td>100</td>
<td>135</td>
</tr>
<tr>
<td>20</td>
<td>10,000</td>
<td>100</td>
<td>100</td>
<td>133</td>
</tr>
</tbody>
</table>
Table 3.4.4. Dataset Release #2 (Alan Wood Tandem Computers -1996)

<table>
<thead>
<tr>
<th>Test Week</th>
<th>CPU Hours</th>
<th>Percent CPU Hours</th>
<th>Defects Found</th>
<th>Predicted Total Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>384</td>
<td>-</td>
<td>13</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1,186</td>
<td>-</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1,471</td>
<td>-</td>
<td>26</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2,236</td>
<td>-</td>
<td>34</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2,772</td>
<td>-</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2,967</td>
<td>-</td>
<td>48</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>3,812</td>
<td>-</td>
<td>61</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>4,880</td>
<td>-</td>
<td>75</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>6,104</td>
<td>-</td>
<td>84</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>6,634</td>
<td>65</td>
<td>89</td>
<td>203</td>
</tr>
<tr>
<td>11</td>
<td>7,229</td>
<td>70</td>
<td>95</td>
<td>192</td>
</tr>
<tr>
<td>12</td>
<td>8,072</td>
<td>79</td>
<td>100</td>
<td>179</td>
</tr>
<tr>
<td>13</td>
<td>8,484</td>
<td>83</td>
<td>104</td>
<td>178</td>
</tr>
<tr>
<td>14</td>
<td>8,847</td>
<td>86</td>
<td>110</td>
<td>184</td>
</tr>
<tr>
<td>15</td>
<td>9,253</td>
<td>90</td>
<td>112</td>
<td>184</td>
</tr>
<tr>
<td>16</td>
<td>9,712</td>
<td>95</td>
<td>114</td>
<td>183</td>
</tr>
<tr>
<td>17</td>
<td>10,083</td>
<td>98</td>
<td>117</td>
<td>182</td>
</tr>
<tr>
<td>18</td>
<td>10,174</td>
<td>99</td>
<td>118</td>
<td>183</td>
</tr>
<tr>
<td>19</td>
<td>10,272</td>
<td>100</td>
<td>120</td>
<td>184</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 3.4.5. Dataset Release #3 (Alan Wood Tandem Computers - 1996)

<table>
<thead>
<tr>
<th>Test Week</th>
<th>CPU Hours</th>
<th>Percent CPU Hours</th>
<th>Defects Found</th>
<th>Predicted Total Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>162</td>
<td>-</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>499</td>
<td>-</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>715</td>
<td>-</td>
<td>13</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1,137</td>
<td>-</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1,799</td>
<td>-</td>
<td>28</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2,438</td>
<td>-</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>2,818</td>
<td>-</td>
<td>48</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>3,574</td>
<td>71</td>
<td>54</td>
<td>163</td>
</tr>
<tr>
<td>9</td>
<td>4,234</td>
<td>84</td>
<td>57</td>
<td>107</td>
</tr>
<tr>
<td>10</td>
<td>4,680</td>
<td>93</td>
<td>59</td>
<td>93</td>
</tr>
<tr>
<td>11</td>
<td>4,955</td>
<td>98</td>
<td>60</td>
<td>87</td>
</tr>
<tr>
<td>12</td>
<td>5,053</td>
<td>100</td>
<td>61</td>
<td>84</td>
</tr>
</tbody>
</table>

### Table 3.4.6. Dataset Release #4 (Alan Wood Tandem Computers - 1996)

<table>
<thead>
<tr>
<th>Test Week</th>
<th>CPU Hours</th>
<th>Percent CPU Hours</th>
<th>Defects Found</th>
<th>Predicted Total Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>254</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>788</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1054</td>
<td>-</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1393</td>
<td>-</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2216</td>
<td>-</td>
<td>11</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>2880</td>
<td>-</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>3593</td>
<td>-</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>4281</td>
<td>-</td>
<td>25</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>5180</td>
<td>-</td>
<td>27</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>6003</td>
<td>53</td>
<td>29</td>
<td>84</td>
</tr>
<tr>
<td>11</td>
<td>7621</td>
<td>67</td>
<td>32</td>
<td>53</td>
</tr>
<tr>
<td>12</td>
<td>8783</td>
<td>78</td>
<td>32</td>
<td>44</td>
</tr>
<tr>
<td>13</td>
<td>9604</td>
<td>85</td>
<td>36</td>
<td>45</td>
</tr>
<tr>
<td>14</td>
<td>10064</td>
<td>89</td>
<td>38</td>
<td>46</td>
</tr>
<tr>
<td>15</td>
<td>10560</td>
<td>93</td>
<td>39</td>
<td>48</td>
</tr>
<tr>
<td>16</td>
<td>11008</td>
<td>97</td>
<td>39</td>
<td>48</td>
</tr>
<tr>
<td>17</td>
<td>11237</td>
<td>99</td>
<td>41</td>
<td>50</td>
</tr>
<tr>
<td>18</td>
<td>11243</td>
<td>99</td>
<td>42</td>
<td>51</td>
</tr>
<tr>
<td>19</td>
<td>11305</td>
<td>100</td>
<td>42</td>
<td>52</td>
</tr>
</tbody>
</table>
3.5 Calculations and Results

Solving equations in Section 3.3 by Newton Raphson Method (N-R) method for all the data sets, the iterative solutions for MLEs of a, b, c and the reliabilities of given software failure datasets are shown in Table 3.5.1.

The estimator of the Reliability function from the equation (3.2.2) at any time x is given by

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

(i) The Reliability of the Phase 1 System Test data is given by

\[ R_{S_1/X_1}(7476/4272) = e^{-[m(4272+7476)-m(7476)]} \]

\[ = e^{-[m(11748)-m(7476)]} \]

\[ = e^{-[36.73759-36.71833]} \]

\[ = e^{-[0.019263]} \]

\[ = 0.980922 \]

(ii) The Reliability of the Phase 2 System Test data is given by

\[ R_{S_2/X_2}(8736/2080) = e^{-[m(2080+8736)-m(8736)]} \]

\[ = e^{-[m(10816)-m(8736)]} \]

\[ = e^{-[58.77548-58.76152]} \]

\[ = e^{-[0.013965]} \]

\[ = 0.986133 \]
(iii)  The Reliability of the Release #1 System Test data is given by

\[ R_{S_{12}/X_{11}}(10000/7083) = e^{-[m(7083+10000) - m(10000)]} \]

\[ = e^{-[m(17083) - m(7083)]} \]

\[ = e^{-[123.7665 - 123.6601]} \]

\[ = e^{-[0.106425]} \]

\[ = 0.899043 \]

(iv)  The Reliability of the Release #2 System Test data is given by

\[ R_{S_{14}/X_{13}}(10272/8847) = e^{-[m(8847+10272) - m(10272)]} \]

\[ = e^{-[m(19119) - m(10272)]} \]

\[ = e^{-[158.067642 - 157.9959567]} \]

\[ = e^{-[0.071685327]} \]

\[ = 0.930823755. \]

(v)  The Reliability of the Release #3 System Test data is given by

\[ R_{S_{0}/X_{8}}(5053/4234) = e^{-[m(4234+5053) - m(5053)]} \]

\[ = e^{-[m(9287) - m(5053)]} \]

\[ = e^{-[83.65400348 - 83.60058558]} \]

\[ = e^{-[0.053417907]} \]

\[ = 0.947983761. \]
(vi) The Reliability of the Release #4 System Test data is given by

\[
R_{S_0/X_8}(11305/5180) = e^{-\left[ m(5180 + 11305) - m(5180) \right]}
\]

\[
= e^{-\left[ m(16485) - m(5180) \right]}
\]

\[
= e^{-\left[ 60.56286 - 60.48309 \right]}
\]

\[
= e^{-0.07977}
\]

\[
= 0.923329
\]

Table 3.5.1. Parameter Estimations and Reliabilities of the Software Failure data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Estimate of ‘a’</th>
<th>Estimate of ‘b’</th>
<th>Estimate of ‘c’</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pham (2005)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase 1 Data</td>
<td>36.772248</td>
<td>0.978993</td>
<td>9.541455</td>
<td>0.980922</td>
</tr>
<tr>
<td>Pham (2005)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase 2 Data</td>
<td>58.835596</td>
<td>0.978993</td>
<td>9.541455</td>
<td>0.986133</td>
</tr>
<tr>
<td>Wood (1996)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release #1 Dataset</td>
<td>123.844535</td>
<td>0.978352</td>
<td>9.144224</td>
<td>0.899043</td>
</tr>
<tr>
<td>Wood (1996)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release #2 Dataset</td>
<td>158.153536</td>
<td>0.977674</td>
<td>8.745810</td>
<td>0.930824</td>
</tr>
<tr>
<td>Wood (1996)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release #3 Dataset</td>
<td>83.720313</td>
<td>0.971698</td>
<td>5.978218</td>
<td>0.947983</td>
</tr>
<tr>
<td>Wood (1996)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release #4 Dataset</td>
<td>60.600888</td>
<td>0.977674</td>
<td>8.745810</td>
<td>0.923329</td>
</tr>
</tbody>
</table>
3.6 Conclusion

Software reliability is an important quality measure that quantifies the operational profile of computer systems. This model is primarily useful in estimating and monitoring software reliability, viewed as a measure of software quality. Equations to obtain the maximum likelihood estimates of the parameters based on interval domain data are developed. This analysis shows that phase 2 dataset is high reliable when compared to all the remaining datasets. This is a simple method for model validation and is very convenient for practitioners of software reliability.
Program to find unknown parameters a, b and c of Pareto Type IV using Newton Raphson Method for Interval domain data

#include<stdio.h>
#include<conio.h>
#include<math.h>
#define N 28

double g(double b,int s[],int n[],int sn);
double gdash(double b,int s[],int n[],int sn);
double gc(double c,int s[],int n[],int sn);
double gcdash(double c,int s[],int n[],int sn);

main()
{
    int i,j,k,sk;
    int s[N]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28};
    double savg,g1,g2,g3,g4,a;
    double b[25],c[25];
    clrscr();
    sk=0;
    printf("********Newton Raphson Method********");
    c[0]=b[0]=1.0;
    i=-1;
    do
    {
        //printf("B iteraton");

        i=i+1;
        g1=g(b[i],s,n,s[N-1]);
        g2=gdash(b[i],s,n,s[N-1]);
        b[i+1]=b[i]-(g1/g2);
printf("n\n\nb[%d]=%f b[%d]=%f",i,b[i],i+1,b[i+1]);
printf("n\n\nt b[%d]-b[%d]=%f",i+1,i,fabs(b[i+1]-b[i]));
}
while(fabs(b[i+1]-b[i])>=0.1);
//printf("n Final i value=%d",i);
j=-1;
do {
//printf("n C Iteration");
j=j+1;
g3=gc(c[j],s,n,s[N-1]);
g4=gcdash(c[j],s,n,s[N-1]);
c[j+1]=c[j]*(g3/g4);
printf("n\n c[%d]=%f",j+1,c[j+1]);
printf("n\n\tc[%d]-c[%d]=%f",j+1,j,fabs(c[j+1]-c[j]));
}
while(fabs(c[j+1]-c[j])>=0.1);
for(k=1;k<N;k++) {
  {f1=f1+(n[k]-n[k-1]);}
}f2=f2*(s[N-1]+c[j+1],b[i+1]);f3=pow((s[N-1]+c[j+1],b[i+1])-pow(c[j+1],b[i+1]));a=f2/f3;
printf("n\n\nb[%d]=%f is the MLE of b=%f",i+1,b[i+1],b[i+1]);
printf("n\n\nc[%d]=%f is the MLE of c=%f\n\ta=%f",j+1,c[j+1],c[j+1],a);
printf("n\n***************");
getch();
}
/*Function for calculating g(b)*/

double g(double b,int s[N],int n[N],int sn)
{
    int i;
    double d1,d2,e=0.0,gval,d;
    double c1,c2,c3,c4,c5=0.0,c6,c7,e1,e2;
    for(i=1;i<N;i++)
    {
        e=e+(n[i]-n[i-1]);
    }
    for(i=1;i<N;i++)
    {
        c1=(double)(s[i]+1);
        c2=(double)(s[i-1]+1);
        c3=pow(c1,b);
        c4=pow(c2,b);
        c5=c5+((log(c2)-log(c1))+(c3*log(c1)-c4*log(c2))/(c3-c4));
    }
    d1=(double)sn+1;
    d2=pow(d1,b);
    c6=(log(1/d1)*(1/(d2-1)));
    gval=(e*c5)+(e*c6);
    printf("gval=%.2f",gval);
    return gval;
}

/*function for calculating g’ (b)*/

double gdash(double b,int s[N],int n[N],int sn)
{
    int i;
    double gdval,c1,c2,c3,c4,c5=0.0;
double d1,d2,d3,d4,e;
for(i=1;i<N;i++)
{
    e=e+(n[i]-n[i-1]);
}
for(i=1;i<N;i++)
{
    c1=(double)(s[i]+1);
    c2=(double)(s[i-1]+1);
    c3=pow(c1,b);
    c4=pow(c2,b);
    c5=c5+(2*((c4*c3*log(c2/c1)*log(c1)))/(pow((c3-c4),2)));
}
d1=(double)(sn+1);
d2=pow(d1,b);
d3=(d2*log(d1))/(pow((d2-g1),2));
d4=e*log(d1)*d3;
gdval=(e*c5)+d4;
printf("gdval=%f",gdval);
return gdval;

/*function for calculating g*(c)*/

double gc(double c,int s[N],int n[N],int sn)
{
    int i;
    double gcval,c1=0.0,c2=0.0,c3;
    for(i=1;i<N;i++)
    {
        c1=c1+(n[i]-n[i-1]);
    }
    for(i=1;i<N;i++)
    {
    }
\{ 
c2=c2+((1/c)-(1/(s[i-1]+c)))-(1/(s[i]+c))); 
\}
c3=1/(sn+c);
geval=c1*(c2+c3);
printf("nc=%f",geval);
return geval;
\} 
/*function for calculating g'(c)*/

double gcdash(double c,int s[N],int n[N],int sn)
\{
int i;
double gcdval,c1=0.0,c2=0.0,c3,c4,c5,c6,c7;
for(i=1;i<N;i++)
\{
    c1=c1+(n[i]-n[i-1]);
\}
for(i=1;i<N;i++)
\{
    c2=c2+((-1/(c*c))+(1/((s[i-1]+c)*(s[i-1]+c)))+(1/((s[i]+c)*(s[i]+c))));
\}
c7=1/((sn+c)*(sn+c));
geval=c1*(c2-c7);
printf("ngc=%f",gcdval);
return gcdval;
\}