CHAPTER-3

SOFTWARE RELIABILITY GROWTH MODELS AND DATA ANALYSIS WITH APARETO TEST-EFFORT

We study software reliability growth models with a Pareto-testing effort. In this testing, the error detection phenomenon is modeled by Non-Homogenous Poisson Process. It is assumed that the error detection rate to the amount of testing effort spent during the testing phase is proportional to the current error content. For the model, the software reliability measures and estimation methods of parameters are investigated. Here we show that Pareto-testing effort function can be expressed as a software test effort curve. Using the model, the method of data analysis for the software reliability measurement with actual software data is developed.

3.1 Introduction

Software reliability is the probability that a given software will be functioning without failure for a specified period of time in a specified environment. Hence, software reliability is a key factor in software development process and software quality. Yamada et al. (1986, 1933), Huang and Kuo (2002), Musa (1999) proposed a SRGMs which described the explicit relationship amount the calendar testing time, the amount of test effort and the number of software errors detected by testing. The test effort is measured by the number of CPU hours, the number of executed test cases and so on.

The size and complexity of computer systems have grown rapidly for the last several decades. Software costs as a percentage of total computer systems costs continue to increase; while associated hardware costs
continue to decrease. The quantitative assessment of software quality can be conducted through many approaches; however, it is sometimes difficult for the project managers to measure software quality and productivity. Nevertheless, reliability may be the most important quality attribute of commercial software since it quantifies software failures during the development process. Although we can test maintainability, usability, or efficiency, but the key issue for software testing is still reliability. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (Lyu (1996)). Its evaluation includes two types of activities; reliability estimation and reliability prediction. Since the early 1970s, many analytical software reliability growth models (SRGMs) have been proposed for estimation of reliability growth of products during software development processes. There are two main categories of reliability estimation models: SRGMs and statistical models. The models in the former class can estimate the software reliability using the failure history of the program. On the other hand, the latter models apply the success/failure information of a program from a random sample of test causes without making any corrections on the discovered errors (Xie (1991) and Musa et al. (1987)).

In recent years software systems such as operating systems, control programs, and application programs have become more complex and larger than ever. It is quite natural to produce reliable software systems efficiently since the breakdown of the computer systems, which is caused by software errors, results in a tremendous loss and damage for social life. Then, software reliability is one of the key issues in modern software product development. Many efforts have been devoted to study of measuring software reliability quantitatively in the area of software
There are several existing software reliability models, especially applicable to the software testing phase in the software reliability development process, which are of great use to estimate and predict software reliability. During the software testing phase, a software system is tested to detect software errors remaining in the system and correct them. If it is assumed that correction of errors does not introduce any new errors, the probability that no failure occurs for a fixed time interval, i.e., the reliability, increases with the progress of software testing. A software reliability model describing such an error detection phenomenon is called a software reliability growth model (SRGM) (Ramamoorthy & Bastani (1982)). Applying the SRGM’s to the observed software error data, the important software reliability measures, such as the number of errors remaining in the system and the software reliability function, can estimated. Then, using the software reliability data analyses based on the SRGM’s, we can evaluate software reliability Goel and Okumoto (1979), Littlewood (1980), Musa (1975), and Yamada et al. (1983).

Software reliability measurement and management in the software development process are essential to produce quality and reliable software efficiently and efficiently. Quantitative measurement and management are characterize the product reliability. In particular based on software-error data analyses, it is very important to evaluate software reliability during the software testing phase. Several software-reliability models have been developed to describe a software-error detection phenomenon during the testing phase and to measure software reliability. Models which are concerned with the relationship between the time-interval between software failures at the time span of the software testing, are called software-reliability growth models (Ramamoorthy & Bastani
These models enable us to estimate software reliability measures such as the mean initial error content, the mean time-interval between failures, the mean number of remaining errors at an arbitrary testing time point, and the software reliability function. Interesting work has been done by Chi, et al. (1989), Goel & Okumoto (1979), Kuo (1983), Musa et al. (1987), Musa & Okumoto (1984), Yamada (1991), and Yamada & Osaki (1985).

A lot of development resources are consumed by software development projects. During the software testing phase, software reliability is highly related to the amount of development resources spent on detecting and correcting latent software errors. Musa et al. (1987) developed a scheme for classifying existing software reliability growth models and demonstrated that: (a) execution time (i.e., test-effort) is a better domain for software reliability modeling than calendar time, and (b) execution time can be transformed into calendar time based on resources available. Yamada et al. (1986) proposed a new and simple software-reliability growth model which describes the explicit relationship among the calendar testing time, the amount of test-effort, and the number of software errors detected by testing. The test-effort is measured by the man-power spent during testing phase, the number of CPU hours, the number of executed test cases, and so on. That is, they proposed software-reliability growth models incorporating the effect of test-effort on the software-reliability growth. They described the time-dependent behaviour of test-effort expenditures by using Rayleigh and exponential curves.

This describes the time-dependent behaviour of test effort by a Pareto curve. Assuming that the error detection rate in software testing is proportional to the current error content and the proportionality depends
on the current test-effort at an arbitrary testing time, a plausible software-reliability growth model based on a non-homogeneous Poisson process, (Ascher and Feingold (1984)) is developed and its application are presented.

3.2 Pareto Test Effort Function

We know that actual test effort data expressed various consumption patterns, sometimes the test effort consumption are difficult to describe only by exponential, Weibull or logistic curve. Therefore, we try to incorporate a Pareto-test effort function instead of above consumption function as the test effort during the software development process. A great deal of resources such as time, money and manpower is spent in developing a software system so that the dead line will be met. Commonly, a Rayleigh curve has been used to estimate and predict the time-dependent behaviour of resources consumed in the software development process (Basili et al. (1978) and Putnam (1978)). Yamada et al. (1986) proposed software reliability growth models incorporating the amount of test-effort spent on software testing by assuming that the test-effort during the testing can be described by the Rayleigh curve as well as the software development effort. However, in many software testing situations it is sometimes difficult to describe the test-effort expenditures by only exponential or Rayleigh curve since actual test-effort data show various expenditure patterns. Then, we offer Pareto curve as the test-effort function due to the flexibility in describing the test-effort expenditure patterns.

The test-effort function representing cumulative test resource expenditures at testing time \((0, t]\) (calendar time) is given by a Pareto curve as:

\[
W(t) = \alpha(1 - (1 + \beta t)^{-\theta}) \quad \alpha > 0, \beta > 0, \theta > 0
\]  

3.1
\[ w(t) = W'(t) = a^\beta \theta (1 + \beta t) - ^{-1} \]

### 3.2.1 Least Square Estimation of Parameter

**Estimation of parameters:** SRGM parameters can be estimated by the least square estimation (LSE) method (Musa et al. (1999)). The parameters are determined for then observed data pairs in the form \((t_k, W_k), k = 1, 2, \ldots, n\) where \(W_k\) is the cumulative test effort spent in \((0, t_k]\).

LSE is used here to fit the Pareto curve with the actual software failure data set. The parameters \(\alpha, \beta \text{ and } \theta\) in the Pareto test function (3.1) and (3.2) can be estimated by the method of least squares Draper & Smith (1981).

The least square estimators \(\hat{\alpha}, \hat{\beta} \text{ and } \hat{\theta}\) can be obtained by minimizing the following equations:

\[
\text{Minimize } S(\alpha, \beta, \theta) = \sum_{k=1}^{n} (w_k - w(t_k))^2 \\
= \sum_{k=1}^{n} \left[ \log w_k - \log \alpha - \log \beta - \log \theta + (\theta + 1) \log(1 + \beta t_k) \right]^2
\]

Where \(w_k\) is the current test effort spent at testing time \(t\). The normal equations are:

\[
\frac{\partial S}{\partial \alpha} = \sum_{k=1}^{n} 2 \left[ \log w_k - \log \alpha - \log \beta - \log \theta + (\theta + 1) \log(1 + \beta t_k) \right] \left( -\frac{1}{\alpha} \right) = 0
\]
\[
\Rightarrow \Sigma \log w_k - \log x - \log \beta - \log \theta + (\theta + 1)\sum_{k=1}^{n} \log(1 + \beta t_k) = 0 \quad 3.4
\]

\[
\frac{\partial s}{\partial \beta} = \sum_{k=1}^{n} 2[\log w_k - \log x - \log \beta - \log \theta + (\theta + 1)\log(1 + \beta t_k)] \left( \frac{-1}{\beta} + (\theta + 1)\frac{1}{1 + \beta t_k} \right) = 0
\]

\[
\Rightarrow \sum_{k=1}^{n} [\log w_k - \log x - \log \beta - \log \theta + (\theta + 1)\log(1 + \beta t_k)] \left( \frac{\beta t_k - 1}{1 - \beta t_k} \right) = 0 \quad 3.5
\]

\[
\frac{\partial s}{\partial \theta} = \sum_{k=1}^{n} 2[\log w_k - \log x - \log \beta - \log \theta + (\theta + 1)\log(1 + \beta t_k)] \left[ \frac{-1}{\theta} + \log(1 + \beta t_k) \right] = 0
\]

\[
\Rightarrow \sum_{k=1}^{n} [\log w_k - \log x - \log \beta - \log \theta + (\theta + 1)\log(1 + \beta t_k)] [\theta \log(1 + \beta t_k) - 1] = 0 \quad 3.6
\]

These equations can be solved numerically to obtain $\alpha$, $\beta$ and $\theta$.

### 3.3 Software Reliability Growth Model

A number of SRGMs have been proposed on the subject of software reliability. Among these models, Goel and Okumoto used an NHPP as the stochastic process to describe the fault process (Lyu (1996), Huang et al. (2002) and Boehm et al. (2002)) modify the model and incorporate the concept of testing-effort in an NHPP model to get a better description of the software fault detection phenomenon. We also proposed a new SRGM with the Pareto testing-effort function to predict the behaviour of failure occurrences and the fault content of a software product. Based on our past experimental results, this approach is suitable for estimating the reliability of software application during the development process.

#### 3.3.1 Model Description

An implemented software system is tested in the software development process. During the testing phase software errors remaining in the system cause software failures, and the errors are detected and corrected by test personnel. A software failure is defined as an unacceptable departure of program operation caused by a software error remaining in the system. Following the usual assumptions in the area of software reliability growth
modeling (Goel (1985)), we assume that the number of detected errors to
the current test-effort expenditures is proportional to the current error
content. Let \( m(t) \) represent the expected number of errors detected by
testing time \( t \) (calendar time) which is assumed to be a bound non-
decreasing function of \( t \) with \( m(0)=0 \). Then, using the Pareto test-effort
function in eqn.(1), we have following differential equation:

\[ \frac{\partial m(t)}{\partial t} / w(t) = r[a-m(t)] \quad a > 0, \quad 0 < r < 1 \] 3.7

Where \( a \) is the initial error content in the system and \( r \) is the error
detection rate per error (per unit test-effort expenditures at testing time \( t \)).
Solving the differential equation (3.7), we have

\[ m(t) = a(1-e^{-rW(t)}) \] 3.8

Substituting (3.1) for \( W(t) \) in (3.8), we get,

\[ m(t) = a[1-e^{-\alpha[1-(\alpha r)^t]}] \] 3.9

From eqn. (3.8), we have the following important relationship between
\( m(t) \) and \( W(t) \):

\[ W(t) = \frac{1}{r} \log \left( \frac{a}{a-m(t)} \right) \] 3.10

For stochastic modeling of a software error detection phenomenon, let
\([N(t), t \geq 0]\) be a counting process representing the cumulative number of
errors detected by testing time \( t \). Defining the expected value of \( N(t) \) by
\( m(t) \) in eqn.(3.8), we can describe a software reliability growth model
incorporating the Pareto test-effort function by an NHPP as:

\[ \Pr[N(k) = n] = \frac{[m(t)]^n e^{-m(t)}}{n!}, \quad n=0,1,2, \ldots \] 3.11

\[ = \text{Pois} m(n; m(t)) \]

Where \( m(t) \) is called mean value function of the NHPP (Goel and
Okumoto (1979) and Yamada and Osaki (1985)) and \( \text{Pois} (n; m(t)) \) is a
Poisson pmf parameter with the intensity function of the NHPP is given by:

\[ \lambda(t) = \frac{\partial m(t)}{\partial t} = \alpha r w(t) e^{-\alpha t} \]

which means the instantaneous error detection rate. From eqn. (3.11), we can show that the limit distribution of \( N(t) \) is a Poisson distribution with the following mean:

\[ m(\infty) = \alpha(1 - e^{-\alpha t}) \]

The equation (3.13) implies that even if a software system is tested during an infinity long duration, all errors remaining in the system can not be detected (Yamada et al. (1986)). Thus, the mean number of undetected errors \( d(t) \) if a test is applied for an infinite amount of time is:

\[ a - m(\infty) = a - \alpha(1 - e^{-\alpha t}) \Rightarrow d(t) = ae^{-\alpha t} \]

**Assumptions**

1. The software system is composed of \( N \) independent modules that are tested individually. The number of software faults remaining in each module can be estimated by an SRGM with Pareto testing-effort function.
2. For each module, the failure data have been collected and the parameters of each module can be estimated.
3. The total amount of testing resources expenditures available for the module testing process is fixed and denoted by \( W \).
4. If any of the software modules fails upon execution, the whole software system is in failure.
5. The system manager has to allocate the total testing resources \( W \) to each software module and minimize the number of faults remaining in the system during the testing period. Besides, the desired software reliability after the testing phase should achieve the reliability objective \( R_o \).
3.3.2 Software Reliability Measures

Based on the NHPP model with \( m(t) \), given by equation (3.8), two quantitative measures for software reliability assessments are derived (Goel & Okumoto (1979) and Yamada (1991)). Let \( \bar{N}(t) \) represent the number of errors remaining in the system at testing time \( t \). The expectation of \( \bar{N}(t) \), \( E[\bar{N}(t)] \), and its variance are given by:

\[
r(t) = E[\bar{N}(t)] = E[(\infty - \bar{N}(t))] = E[N(\infty)] - E[N(t)]
\]

\[
= m(\infty) - m(t) = a[e^{-\beta(t)} - e^{-\beta(\infty)}]
\]

\[
= \text{Var}[\bar{N}(t)].
\]

The software reliability representing the probability that a software failure does not occur in the time interval \((t, t+x)\) is given by:

\[
R = R(x/t) = e^{-\int m(t+x)-m(t)} = e^{-\beta[t+(t+x)-t]}
\]

The instantaneous mean time between failures (MTBF) at arbitrary testing time can be defined as a reciprocal of the instantaneous error detection rate in eqn. (3.12) (Yamada (1985)). Then, the instantaneous MTBF is given by:

\[
MTBF(t) = \frac{1}{\lambda(t)} = \frac{1 + \beta \int \theta \alpha e^{-\alpha(t+\beta)\psi}}{\alpha \beta \theta}
\]

3.4 Maximum Likelihood Estimation of Model Parameters

The estimated test-effort parameters \( \alpha, \beta \), and \( \theta \) in the Pareto test-effort function in eqn. (3.1) or (3.2) have been obtained by the methods of least-squares method. The estimators for \( a \) and \( r \) are determined for the \( n \) observed data pairs in the form \( [(t_i, y_i), (k=1,2,...,n; 0<t_i<t_{i+1}<...t_k)\] where \( y_i \) is the cumulative number of errors detected up to time \( t_i \) or \( 0, t_i \), then the likelihood unknown parameters, \( a \) and \( r \) in the NHPP model with \( m(t) \), is:

\[
L' = (a,r) \equiv P[N(t_i) = y_i, i = 1,2,...,n}
\]
\[ k = m(t) - m(t-1) \]

Where \( t_0 = 0 \) and \( y_0 = 0 \).

Taking the log of eqn. (3.17)

\[
L = \log L' = \sum_{k=1}^{n} (y_k - y_{k-1}) \log[m(t_k) - m(t_{k-1})] - \sum_{k=1}^{n} [m(t_k) - m(t_{k-1})] - \sum_{k=1}^{n} \log[(y_k - y_{k-1})!]
\]

From equation (3.8), we get,

\[
m(t_k) - m(t_{k-1}) = a[1 - e^{-rW(t_k)}] - 1 + e^{-rW(t_{k-1})}
\]

Thus,

\[
\sum_{k=1}^{n} [m(t_k) - m(t_{k-1})] = am(t) = a[1 - e^{-rW(t)}] 
\]

Taking the summation, we get,

\[
L = \sum_{k=1}^{n} (y_k - y_{k-1}) \log a + \sum_{k=1}^{n} (y_k - y_{k-1}) \log[e^{-rW(t_k)}] - a[1 - e^{-rW(t)}] - \sum_{k=1}^{n} \log[(y_k - y_{k-1})!]
\]

The maximum likelihood estimate (MLE) of reliability growth parameters, \( a \) and \( r \) obtained by solving the following equations:

\[
\frac{\partial L}{\partial a} = \sum_{k=1}^{n} \frac{(y_k - y_{k-1})}{a} - 1 + e^{-rW(t_k)} = 0
\]

\[
\Rightarrow y_k / a = 1 - e^{-rW(t_k)}
\]

\[
\Rightarrow a = \frac{y_k}{1 - e^{-rW(t_k)}} = \frac{y_k}{1 - \theta_k}
\]

\[
\frac{\partial L}{\partial r} = \frac{\sum_{k=1}^{n} (y_k - y_{k-1}) [e^{-rW(t_k)}W(t_k) - e^{-rW(t_{k-1})}W(t_{k-1})]}{e^{-rW(t_k)} - e^{-rW(t_{k-1})}} = 0
\]

\[
\Rightarrow aW(t_k)e^{-rW(t_k)} = \frac{\sum_{k=1}^{n} (y_k - y_{k-1}) [W(t_k)e^{-rW(t_k)} - W(t_{k-1})e^{-rW(t_{k-1})}]}{e^{-rW(t_k)} - e^{-rW(t_{k-1})}} = 0
\]

Let \( \phi_k = e^{-rW(t_k)} \), \( k = 1, 2, ..., n \)
\[ aW(t)W_{k} = \frac{\sum_{k=0}^{n} (y_{k} - y_{k-1}) [W(t_{k}) \phi_{k} - W(t_{k+1}) \phi_{k-1}]}{\phi_{k+1} - \phi_{k}} = 0 \quad 3.21 \]

Which can be solved by numerical method to get the value of \( \hat{a} \) and \( \hat{r} \). If the sample size \( n \) of \((t_{k},y_{k})\) is sufficiently large, then the maximum likelihood estimates (MLE) \( \hat{a} \) and \( \hat{r} \) asymptotically follow a bivariate s-normal (BVN) distribution (Nelson (1982) Okumoto & Goel (1980)).

\[
\begin{pmatrix}
\hat{a} \\
\hat{r}
\end{pmatrix} \sim \text{BVN}
\begin{pmatrix}
\hat{a} \\
\hat{r}
\end{pmatrix}
\begin{pmatrix}
\Sigma
\end{pmatrix}
\quad n \to \infty \quad 3.22
\]

The variance-covariance matrix \( \Sigma \) in the asymptotic properties of eqn. (3.22) is useful of the Fisher information matrix \( F \), i.e. \( \Sigma = F^{-1} \), given by the expectation of the negative of the second partial derivatives of \( L \) as:

\[
F = \begin{bmatrix}
E \left( \frac{\partial^2 \ln L}{\partial a^2} \right) & E \left( \frac{\partial^2 \ln L}{\partial a \partial r} \right) \\
E \left( \frac{\partial^2 \ln L}{\partial r \partial a} \right) & E \left( \frac{\partial^2 \ln L}{\partial r^2} \right)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\frac{f_{s}}{a} & g_{s} \\
g_{s} & \frac{\hat{a} \sum_{i=1}^{n} (g_{i} - g_{i-1})^2}{(f_{s} - f_{s-1})}
\end{bmatrix}
\]

Where,

\[
g_{s} = W(t_{s})e^{-\hat{r}W(t_{s})} \quad 3.23
\]

and

\[
f_{s} = 1 - e^{-\hat{r}W(t_{s})}
\]

Substitute the value of \( \hat{a} \) and \( \hat{r} \) in eqn. (3.23) and calculate \( F^{-1} \). The estimated asymptotic variance-covariance matrix is:

\[
\Sigma = F^{-1} = \begin{bmatrix}
\text{Var}(\hat{a}) & \text{Cov}(\hat{a},\hat{r}) \\
\text{Cov}(\hat{a},\hat{r}) & \text{Var}(\hat{r})
\end{bmatrix}
\]

\( \Sigma \) is useful in quantifying the variability of the estimated parameters.
3.5 Real Software Data Analysis & Comparison Criteria

3.5.1 Comparison Criteria

We choose two comparison criteria of estimation described as follow:

(1) The Accuracy of Estimation Musa et al. (1987), Hou et al. (1994),
Goel et al. (1979) and Kuo et al. (2001).

\[ (AE) = \left| \frac{M_a - m}{M_a} \right| \]  \hspace{1cm} 3.24

Where \( M_a \) is the actual cumulative number of detected errors during the
test and after the test, and \( m \) is the estimated parameter \( a \) in Eqn. (3.14).

(2) The mean of square fitting Errors

\[ (MSE) = \frac{\sum_{i=1}^{k} (m(t_i) - m_i)^2}{k} \]  \hspace{1cm} 3.25

The lower MSE indicates less fitting errors and better performance Kapur
et al. (1996).

3.5.2 Actual Software Data Analysis

The set of real data in table-3.1 is from the study by Obha (1984). The
system is PL/1 data base application software, consisting of
approximately 1,317,000 lines of code. During the nineteen weeks
experiments, 47.65 CPU times were consumed and about 328 software
errors were removed. The original data report gives that the total
cumulative number of detected faults after a long period of testing is 358
faults Hou et al (1994) & Obha (1984). In order to estimate the
parameters \( \alpha, \beta \) and \( \theta \) of the Pareto-testing function; we fit the actual
testing-effort data into equations (3.1) and (3.2) and solve it by using the
method of least squares. That is, we will minimize the sum of squares.
Hence, we obtain the estimates through numerical procedures. These
estimated parameters are:

\[ \hat{\alpha} = 3874.124972, \]
\[ \hat{\beta} = 0.000014157, \]
and \[ \hat{\theta} = 45.038014 \]

The estimated Pareto test-effort functions are

\[ \hat{W}(t) = 3874.12497 \times 0.0000141 \times 45.03801 \left( 1 + 0.0000141 . t \right)^{-45.038014 - 1} \]
\[ \hat{W}(t) = 3874.12497 \left( 1 - (1 + 0.0000141 . t)^{-45.038014 - 1} \right) \]

Figure 3.1 and figure 3.2 shows the fitting of the estimated testing-effort by using Equation (3.27) and (3.28). Here, the fitted curves are shown as a dotted line and actual software data shown by solid line. Using the estimated parameters \( \alpha \), \( \beta \) and \( \theta \), the other parameters \( a, r \) in eqn. (3.8) can be solved by maximum likelihood estimation (MLE) method.

For these estimates, the optimality was numerically. These estimated parameters are:

\[ a = 772.73317 \]
\[ r = 0.0128612 \]

Table 3.2 also summarizes the experimental results of estimated parameters with their standard errors and 95% confidence bound. The estimated mean value function is

\[ \hat{m}(t) = 772.73317 \left[ 1 - e^{-0.01286\times3874.12497 \times (1 + 0.0000141 . t)^{-45.038014 - 1}} \right] \]

Where \[ \hat{W}(t) = 3874.12497 \left( 1 - (1 + 0.0000141 . t)^{-45.038014 - 1} \right) \]

Similarly, we plotted a fitted curve of the estimated mean value function with the actual data in figure 3.3. Intensity function is shown by figure 3.4. Also a regression analysis depends on test-effort and number of failure respectively in Table 3.3 and 3.4. From figures (3.1), (3.2), (3.3) and (3.4) the comparison criteria shows that our SRGM is better fit than the other models for PL/1 application program.
Table-3.1 Software Failure Data

<table>
<thead>
<tr>
<th>Times of Observation (in week)</th>
<th>Current execution time (in CPU hr)</th>
<th>Cumulative execution time</th>
<th>Number of failure</th>
<th>Cumulative number of failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.45</td>
<td>2.45</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>2.45</td>
<td>4.9</td>
<td>29</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>1.96</td>
<td>6.86</td>
<td>22</td>
<td>66</td>
</tr>
<tr>
<td>4</td>
<td>0.98</td>
<td>7.84</td>
<td>37</td>
<td>103</td>
</tr>
<tr>
<td>5</td>
<td>1.68</td>
<td>9.52</td>
<td>2</td>
<td>105</td>
</tr>
<tr>
<td>6</td>
<td>3.37</td>
<td>12.89</td>
<td>5</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>4.21</td>
<td>17.1</td>
<td>36</td>
<td>146</td>
</tr>
<tr>
<td>8</td>
<td>3.37</td>
<td>20.47</td>
<td>29</td>
<td>175</td>
</tr>
<tr>
<td>9</td>
<td>0.96</td>
<td>21.45</td>
<td>4</td>
<td>179</td>
</tr>
<tr>
<td>10</td>
<td>1.92</td>
<td>23.45</td>
<td>27</td>
<td>206</td>
</tr>
<tr>
<td>11</td>
<td>2.88</td>
<td>26.23</td>
<td>27</td>
<td>233</td>
</tr>
<tr>
<td>12</td>
<td>1.44</td>
<td>27.67</td>
<td>22</td>
<td>255</td>
</tr>
<tr>
<td>13</td>
<td>3.26</td>
<td>30.93</td>
<td>21</td>
<td>276</td>
</tr>
<tr>
<td>14</td>
<td>3.84</td>
<td>34.77</td>
<td>22</td>
<td>298</td>
</tr>
<tr>
<td>15</td>
<td>3.84</td>
<td>38.61</td>
<td>6</td>
<td>304</td>
</tr>
<tr>
<td>16</td>
<td>2.3</td>
<td>40.91</td>
<td>7</td>
<td>311</td>
</tr>
<tr>
<td>17</td>
<td>1.76</td>
<td>42.67</td>
<td>9</td>
<td>320</td>
</tr>
<tr>
<td>18</td>
<td>1.99</td>
<td>44.66</td>
<td>5</td>
<td>325</td>
</tr>
<tr>
<td>19</td>
<td>2.99</td>
<td>47.65</td>
<td>3</td>
<td>328</td>
</tr>
</tbody>
</table>
Table-3.2 Experiment result of different parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Estimate</th>
<th>Standard error</th>
<th>95% confidence interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>3874.12497</td>
<td>47215.39951</td>
<td>-96228.05080 - 103966.30074</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.000014</td>
<td>0.0035238</td>
<td>-0.0074560 - 0.004843</td>
</tr>
<tr>
<td>$\theta$</td>
<td>45.03801</td>
<td>11561.79945</td>
<td>-24464.88193 - 24554.95795</td>
</tr>
<tr>
<td>a</td>
<td>772.73317</td>
<td>145.88334</td>
<td>464.94620 - 1080.52013</td>
</tr>
<tr>
<td>r</td>
<td>0.01286</td>
<td>0.003051</td>
<td>0.0064242 - 0.01929839</td>
</tr>
</tbody>
</table>

Table-3.3 Regression analysis depends on test-effort

<table>
<thead>
<tr>
<th>Source</th>
<th>Degree of freedom</th>
<th>Sum of squares</th>
<th>Mean squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>3</td>
<td>14954.25588</td>
<td>4984.75196</td>
</tr>
<tr>
<td>Residual</td>
<td>16</td>
<td>31.3382</td>
<td>1.95863</td>
</tr>
<tr>
<td>Uncorrected total</td>
<td>19</td>
<td>14985.594</td>
<td>-</td>
</tr>
<tr>
<td>(corrected total)</td>
<td>18</td>
<td>3800.27821</td>
<td>-</td>
</tr>
</tbody>
</table>

$R^2=1$-Residual SS/ corrected SS= 0.99175

Table-3.4 Regression analysis depends on Number of failures

<table>
<thead>
<tr>
<th>Source</th>
<th>Degree of freedom</th>
<th>Sum of squares</th>
<th>Mean squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>2</td>
<td>953049.03471</td>
<td>476524.51736</td>
</tr>
<tr>
<td>Residual</td>
<td>17</td>
<td>2659.96529</td>
<td>156.4855</td>
</tr>
<tr>
<td>Uncorrected total</td>
<td>19</td>
<td>955709.000</td>
<td>-</td>
</tr>
<tr>
<td>(corrected total)</td>
<td>18</td>
<td>196108.94737</td>
<td>-</td>
</tr>
</tbody>
</table>

$R^2=1$-Residual SS/ corrected SS= 0.98644
Figure 3.1. Observed/Estimated cumulative Testing Effort vs Time

Figure 3.2. Observed/Estimated Current Testing Effort Function vs Time
Figure 3.3. Observed/Estimated Cumulative Number of failures vs Time

Figure 3.4. Observed/Estimated Current Testing Effort Function vs Time
3.6 Conclusion

We have discussed a software reliability growth model based on NHPP, which incorporates Pareto testing-effort expenditure. We conclude that the Pareto-testing function can be used to represent a software reliability growth process for a wide range of testing effort curve and give a reasonable predictive capability for the real failure data. Regression analysis is also discussed. We observed that $R^2$ is very close to 1. This slow that Pareto test effort function mean value function for error gives better fit in this case. Figures also represent the same result for fitting. Hence we conclude that Pareto Test consumption curve can be used in software reliability analysis.