An Improved NHPP Model with Time-Varying Fault Removal Delay

Xue Yang, Nan Sang, and Hang Lei

Abstract—In this paper, an improved NHPP model is proposed by replacing constant fault removal time with time-varying fault removal delay in NHPP model, proposed by Daniel R Jeske. In our model, a time-dependent delay function is established to fit the fault removal process. By using two sets of practical data, the descriptive and predictive abilities of the improved NHPP model are compared with those of the NHPP model, G-O model, and delayed S-shape model. The results show that the improved model can fit and predict the data better.

Index Terms—Fault removal delay, fault removal efficiency, non-homogeneous Poisson process (NHPP), software reliability, software reliability growth model.

1. Introduction

With the rapid growth of computing technology and complexity of software systems, the assurance of software quality becomes a critical concern. Software reliability is widely cited by many users and developers as one of the most important features of software products, and the estimation of software reliability provides a prediction of the software failure rate before system tests begin or at any point throughout\[1\]. Research activities in software reliability engineering have been conducted and many software reliability growth models (SRGM) have been proposed to assess the reliability of software.

Software reliability models based on non-homogeneous Poisson process (NHPP) have been quite successful tools in practical software reliability engineering\[2\], they are frequently used in stochastic simulations to model non-stationary point processes\[3\]. There are many NHPP models, such as Goel-Okumoto model\[4\], delayed S-shaped model\[5\], PNZ-SRGM model\[6\], TOE-SRGM model\[7\], CE-SRGM model\[8\] and so on. These models have been built upon various assumptions, and one common assumption is that detected faults are immediately removed. In fact, this assumption may not be realistic in software development. Software testing and debugging are very complex and expensive processes. Once a fault is discovered, a trouble ticket is created and assigned to one or more developers for analysis and code modifications. It is not unusual for a software fault to occur multiple times in the field before it is finally removed. Therefore, the time delayed by the correction process should not be neglected.

Daniel R. Jeske\[9\] supposed that perfect debugging is an acceptable assumption (based on thorough regression testing), but instantaneous fault removal is not. He proposed an NHPP model with the combination of perfect debugging and non-instantaneous fault removal. In this model, the expected time to remove a fault is a constant and the fault removal efficiency is also a constant. While in practice, with the process of test, it would be more difficult to make sure where the fault locates, and more time is consumed to remove this fault, then, the fault removal efficiency would decrease. So, the time to remove a fault should be a function of the time when the fault occurs.

In this paper, we discuss the impractical assumption that fault removal time is a constant since fault removal time usually changes with the moment when the fault occurs. And then we establish a corresponding time-dependent delay function to fit the fault removal process, and improve the constant fault removal delay NHPP model proposed by Daniel R. Jeske. After that, we validate the improved model by using some practical data. The results show that the improved model fits and predicts the data better.

The rest of this paper is organized as follows. Section 2 presents an explicit solution to a general class of NHPP SRGM. Section 3 proposes an improved NHPP SRGM that incorporates time-varying fault removal delay. Section 4 evaluates this new model using two sets of software failure data. The results show that incorporating time-varying fault removal delay into Daniel R. Jeske’s NHPP SRGM can improve both descriptive and short-term predictive ability.

2. General NHPP Model with Fault Removal Efficiency

Firstly, we give the notations to be used in the rest part of this paper:

\[ N(t) \] Counting process for the total number of faults in \([0,t)\).
m(t) Expected number of software failures by time t,
\[ m(t) = E[N(t)]. \]

a Total number of faults in the software.

b Fault detection rate per fault.

p(t) Fault removal efficiency function, i.e., percentage of faults eliminated by reviews, inspections and tests at time t.

\( \lambda(t) \) Failure intensity function, the failure rate of the software.

\[ \lambda(t) = d[m(t)]/dt . \]

\( x(t) \) Total number of faults detected and successfully removed by time t.

In the family of software reliability models, NHPP software reliability models have been widely used in analyzing and estimating the reliability related metrics of software products in many applications\[10\]. These models consider the debugging process as a counting process, which follows Poisson distribution with a time-dependent intensity function. In this section, we present a general NHPP model framework with fault removal efficiency. There are three assumptions for this model.

**Assumption A**: The fault detection process follows the NHPP.

**Assumption B**: The software failure rate at any time is a function of fault detection rate and the number of remaining faults presented at that time.

**Assumption C**: All faults are independent and equally detectable. No new faults are introduced during the fault removal process.

Assumption A is a widely accepted assumption, it means the total number of detected faults \( N(t) \) follows a Poisson distribution with parameter \( m(t) \). It can be denoted as:

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

Assumption B can be interpreted as follows: software failure rate is the product of the number of residual faults (which incorporates the concept of fault removal efficiency) and the average failure rate of a fault. When the developer modifies the code, no new faults were introduced to the software. This is captured by Assumption C.

According to the assumptions above, the software failure rate at any time is a function of fault detection rate and the number of remaining faults presented at that time. Fault removal rate is determined by the fault removal efficiency and software failure rate. Then,

\[
\lambda(t) = d[m(t)]/dt = b(a - x(t)) \quad (2)
\]

\[
\frac{dx(t)}{dt} = p(t) \frac{dm(t)}{dt} \quad (3)
\]

where \( x(t) \) is the total number of faults detected and successfully removed by time \( t \), \( m(t) \) presents the expected number of faults by time \( t \), and \( p(t) \) denotes the fault removal efficiency.

The marginal conditions for the differential (2) and (3) are as follows:

\[ m(0) = 0 \quad (4) \]

\[ x(0) = 0 . \quad (5) \]

Hence, the total number of faults detected and successfully removed by time \( t \) is

\[
x(t) = a(1 - \exp(-b \int_0^t p(\tau)d\tau)) . \quad (6)
\]

Therefore, the explicit expression of the expected number of faults by time \( t \) can be obtained as follows:

\[
m(t) = ab \int_0^t \exp(-b \int_0^\tau p(\tau)d\tau)d\tau . \quad (7)
\]

Thus, the reliability metrics, i.e., the expected number of residual faults, software reliability can be estimated respectively.

### 3. NHPP Model with Time-Varying Fault Removal Delay

Fault removal is a complex process, when a fault is discovered, a series of procedures would be processed, such as fault report, fault location and fault correction, so it is not unusual for a software fault to occur multiple times in the field before it is finally removed. Daniel R. Jeske\[9\] supposed that the expected time to remove a fault is \( \mu \) time-units, and the value of \( \mu \) is often known through experience with previous releases and other software products. Once a fault is detected, the expected number of subsequent occurrences before it is removed is \( \mu b \). Therefore, the fault removal efficiency \( p \) can be denoted as:

\[
\frac{1}{p} = 1 + \mu b . \quad (8)
\]

But in practice, for these faults that are detected later, it would be more difficult to locate where these faults are, and more time are consumed to remove them, so fault removal efficiency should decrease with testing time. Here, we suppose fault removal time delayed is a function of the time when the fault is detected, and it can be denoted as \( \mu (1 + ct) \), and \( c \) is the parameter for measuring how fast the fault removal time changes, then the expected number of subsequent occurrences of this fault before it is removed is \( \mu b (1 + ct) \). Therefore, we have

\[
\frac{1}{p(t)} = 1 + \mu b(1 + ct) \quad (9)
\]

where \( p(t) \) is the fault removal efficiency which varies with the testing time \( t \), so

\[
p(t) = \frac{1}{1 + \mu b(1 + ct)} . \quad (10)
\]

Substituting (10) into (6) and (7), we obtain

\[
x(t) = a \left[ 1 - \left( \frac{1 + \mu b(1 + ct)}{1 + \mu b} \right)^{\frac{1}{\mu c}} \right] \quad (11)
\]
\[ m(t) = \frac{a(1 + \mu b)}{(\mu c - 1)} \left( \frac{1 + \mu b t}{1 + \mu b} \right)^{\frac{1}{\mu c - 1}} - 1. \]  

It should be noted that fault removal efficiency that changes with testing time is introduced into this improved NHPP model.

### 4. Model Comparison

In this section, we estimate the parameters in \( m(t) \) and compare the newly improved NHPP model with the NHPP model proposed by Daniel R. Jeske and another two classical NHPP models: G-O model and Delayed S-shape model by using two sets of practical data.

The results below indicate that the improved model has better descriptive and short-term predictive power.

#### 4.1 Parameter Estimation and Comparison Criteria

Once the analytical expression for \( m(t) \) is derived, the parameters in the \( m(t) \) need to be estimated. Here, we use the maximum likelihood estimate (MLE) method to estimate the parameters \( a, b \) and \( c \) in \( m(t) \).

In this paper, the performance of the newly improved NHPP model is evaluated by using the sum of squared error (SSE) and Akaike’s information criterion (AIC). Both the descriptive and predictive abilities of these models are considered.

The sum of squared error (SSE) is the criterion which sum up the squares of the residuals of the actual data and the mean value function (\( m(t) \)) of each model in terms of the number of actual faults at any time points. The SSE function can be expressed as follows:

\[ SSE = \sum_{k=1}^{n} [y_k - \hat{m}(t_k)]^2 \]  

(13)

where \( y_k \) is the total number of faults observed at time \( t_k \) according to the testing data, and \( \hat{m}(t_k) \) is the estimated cumulative number of faults at time \( t_k \) obtained from the fitted mean value function \( m(t) \), \( k = 1, 2, \ldots, n \).

Another criterion we use is AIC, which can be expressed as follows:

\[ AIC = -2 \log ( \text{likelihood fun. at its max. value}) + 2N \]  

(14)

where \( N \) represents the number of parameters in the model. The AIC measures the ability of a model to maximize the likelihood function that is directly related to the degrees of freedom during fitting, increasing the number of parameters will usually result in a better fit. AIC criterion takes the degree of freedom into consideration by assigning a model with more parameters a larger penalty.

Therefore, the lower the SSE and AIC values, the better the model performs.

#### 4.2 Data from a Real Time Control System

In this section, we use the data set that is documented in [10] to examine the goodness-of-fit and predictive power of the improved model.

There are totally 136 faults reported, and the TBF is extremely long from the 122nd fault to the 123rd fault, and the TBFs after the 123rd fault increases tremendously. It implies that the reliability grows, and the system becomes stabilized. Here, we use the first 122 data points for the goodness-of-fit evaluation and the remaining data points for the predictive power test.

According to the experience, we use 580 seconds as the value of \( \mu \). The SSE and AIC values for goodness-of-fit and prediction are listed in Table 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>SSE(fit)</th>
<th>SSE(predict)</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-O</td>
<td>7615.1</td>
<td>704.8</td>
<td>426.1</td>
</tr>
<tr>
<td>Delayed S-shape</td>
<td>51729.2</td>
<td>257.7</td>
<td>546.0</td>
</tr>
<tr>
<td>Daniel R’s model</td>
<td>5212.7</td>
<td>125.6</td>
<td>410.2</td>
</tr>
<tr>
<td>Improved model</td>
<td>4315.4</td>
<td>107.5</td>
<td>397.0</td>
</tr>
</tbody>
</table>

Table 1 shows that for the improved model, the SSE value for the goodness-of fit is 4315.4, which is smaller than that of Daniel R’s Model. The SSE value for prediction and the AIC value are also the lowest among all models.

#### 4.3 AT&T Data

This section examines models using the data collected from the testing of a network management system developed by the AT&T[11], which is known as System T. It receives telemetry events (such as alarms), facility-performance information, and diagnostic messages, and forwards this data to operators for further action.

There are totally 22 faults detected. We use the first 19 data points to fit the models and estimate parameters in the models, and use the remaining data to compare the prediction power of the models. Here, we take 85 time units as the value of \( \mu \). The AIC values and SSE values for both the goodness-of-fit and prediction power are listed in Table 2.

<table>
<thead>
<tr>
<th>Model</th>
<th>SSE(fit)</th>
<th>SSE(predict)</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-O</td>
<td>26.70</td>
<td>2.27</td>
<td>78.48</td>
</tr>
<tr>
<td>Delayed S-shape</td>
<td>21.95</td>
<td>9.55</td>
<td>86.70</td>
</tr>
<tr>
<td>Daniel R’s model</td>
<td>22.13</td>
<td>1.35</td>
<td>73.86</td>
</tr>
<tr>
<td>Improved model</td>
<td>20.35</td>
<td>1.07</td>
<td>72.41</td>
</tr>
</tbody>
</table>

Table 2 shows that the SSE values for the goodness-of-fit are 20.35 and 1.07 respectively, and the AIC value is 72.41. They are all smaller than the values of other three models. Overall, the improved model is the best descriptive and predictive model.
5. Summary

In this paper, we discuss the impractical assumption in Daniel R Jeske’s model that fault removal time is a constant, and research how fault removal time varies with fault detection time. In practice, fault removal time usually changes with the moment when the fault occurs. For this reason, an improved NHPP model is established by replacing constant fault removal time with time-varying fault removal delay in Daniel R Jeske’s model. By using two sets of practical data, the ability of our improved NHPP model is compared with that of classical NHPP models. The results show that inclusion of various fault removal delay can improve both the descriptive and the predictive ability of a model.

References


Xue Yang was born in Chongqing, China, in 1982. She received her B.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2005. She is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, UESTC. Her research interests include software reliability testing, software reliability modeling.

Nan Sang was born in Sichuan, China, in 1964. He received the M.S. degree from Sichuan University (SCU), Chengdu, in 1987. He is currently a professor with UESTC. His research interests include embedded real-time and high dependability technology, real-time software engineering.

Hang Lei was born in Sichuan, China, in 1960. He received the Ph.D. degree from UESTC, Chengdu, in 1997. He is currently a professor with UESTC. His research interests include software reliability testing and evaluation, hardware-software codesign of embedded systems.