## 1. Introduction

In recent years, more and more computers are used in safety-critical applications such as medicine, transportation, and nuclear energy. For a lot of software embedded system, software reliability has been the dominant driver of today's system reliability. It leads to a great demand for high quality software products. However, the poor performance due to unreliable software is exhibited by many systems. To improve the software quality, software reliability engineering plays an important role in many aspects throughout the software life cycle. For example, before software is released for installation or distribution to customers, accurate reliability estimates are required to verify the quality of the software. To assess the reliability of software is also critical in determining the optimal release time of a software system.

As we know, there are many reasons for software to fail, but none involves wear or tear. Usually, software fails because of a design problem. Other failures occur when the code is written, or when changes are introduced to a working system [Fenton and Pfleeger, 1997]. For the past several decades, various statistical models have been proposed to assess the software reliability. The Nonhomogenous Poisson process (NHPP) based software reliability growth models (SRGMs) are proved 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. Model parameters can be estimated by using maximum likelihood or least square estimate. Once the mean value function is determined the software reliability and the related measurements can be easily derived.

Various NHPP SRGMs have been built upon various assumptions. Many of the SGRMs 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 [Ohba, 1984][Pham, 1993]. The other assumption of many NHPP SRGMs is that each failure occurs independently and randomly in time according to the same distribution during the fault detection process [Musa et al., 1987]. However, in practice the failure distribution can be affected by many factors, such as the running environment, testing strategy and resource allocation [Zhao, 1993]. Once these factors are changed during the software-testing phase, this could result in a software failure intensity function that increases or decreases non-monotonically. It is identified as a change-point problem.

In this paper, we incorporate both imperfect debugging and change-point problem into the NHPP SRGM. Both of these features reflect more closely to a general SRGM.

## 2. Imperfect-Software-Debugging Model with Change Point

To consider the NHPP SRGM that integrates imperfect debugging with change-point problem, the following assumptions are made.

(a) When detected faults are removed at time $t$, it is possible to introduce new faults with introduction rate $(t)$.

$$(t) = \begin{cases} {}_1 & \text{when } 0 \le t \le \hat{o} \\ {}_2 & \text{when } t > \hat{o} \end{cases} \quad (\hat{o} \text{is the change point}).$$

(b) The fault detection rate represented as following is a step function

$$b(t) = \begin{cases} b_1 & \text{when } 0 \le t \le \hat{o} \\ b_2 & \text{when } t > \hat{o} \end{cases}.$$

(c) A Nonhomogeneous Poisson process models the fault detection phenomenon in the software system.

In previous studies, the parameter $\hat{o}$ is considered unknown and is to be estimated from the collected failure data. [Zhao, 1993][Hinkley, 1970][Chang, 1997]. Because the testing strategy and resource allocation can be tracked all the time during the fault detection process, it may be more reasonable to reconsider that the change-point $\hat{o}$ is a given. Therefore, in this paper we suppose the parameter $\hat{o}$ is allocated in a certain time point and is known before the model is created. According to these assumptions, one can derive the new set of differential equations to obtain the new mean value function:

$$\frac{\partial m(t)}{\partial t} = b(t) \cdot (a(t) - m(t));$$
$$\frac{\partial a(t)}{\partial t} = \quad (t) \cdot \frac{\partial m(t)}{\partial t};$$
$$a(0) = a;$$
$$m(0) = 0.$$

Solving the differential equations under the assumptions (a) and (b) yields

$$m(t)=\begin{cases}\dfrac{a}{1-\ _1}[1-e^{-(1-\ _1)b_1 t}] & \text{when } 0\le t\le \tau \\ \dfrac{a}{1-\ _2}[1-e^{-(1-\ _1)b_1\tau-(1-\ _2)b_2(t-\tau)}]+\dfrac{m(\tau)(\ _1-\ _2)}{1-\ _2} & \text{when } t>\tau\end{cases},$$

$$\lambda(t)=\frac{\partial m(t)}{\partial t}=\begin{cases}ab_1 e^{-(1-\ _1)b_1 t} & \text{when } 0\le t\le \tau \\ ab_2 e^{-(1-\ _1)b_1\tau-(1-\ _2)b_2(t-\tau)} & \text{when } t>\tau\end{cases}.$$

The results have integrated imperfect debugging and change-point problem into the NHPP SRGMs. The maximum likelihood theory can be used to evaluate the unknown parameters of the proposed model. Taking a snap view, it can be figured out that the mean value function is the same as Pham's imperfect debugging model, while $0<t\le\tau$. However, if the change-point is exhibited (when $t>\tau$), the developed function will become more complexity than the other models. Let the fault introduction rate is a constant ( $_1=\ _2=\ $ ) during the fault detection process, the mean value function for time $t>\tau$ can be simplified as:

$$m(t)=\frac{a}{1-}[1-e^{-(1-\ )(b_1\tau+b_2(t-\tau))}].$$

The previous equation also appears that the intensity function $\lambda(t)$ is not a continuous function of time except when $b_1=b_2$. Following the same definition of Goel and Okumoto [1979], the conditional reliability function of this developed model can be obtained by

$$R(x\mid t)=\exp\{-(m(t+x)-m(t))\}$$

$$=\begin{cases}\exp\{-\dfrac{a}{1-\ _1}(e^{-(1-\ _1)b_1 t}-e^{-(1-\ _1)b_1(t+x)})\} & \text{when } 0<t\le\tau \\ \exp\{-\dfrac{a}{1-\ _2}(e^{-(1-\ _1)b_1\tau-(1-\ _2)b_2(t-\tau)}(1-e^{-(1-\ _2)b_2 x}))\} & \text{when } t>\tau\end{cases}$$

The above model can be used to conduct the problem with single type of fault. However, based on the severity that assesses the impact of the fault on the user, software faults can be classified into various types. Severity examines whether the fault can actually be evidenced as a failure, and the degree to which that failure would affect the user [Fenton and Pfleeger, 1997]. To conduct multiple fault types problem, the above mean value function can be extended and rewritten as:

$$m(t) = \sum_{i \in \text{all types of fault}} m_i(t),$$

$$m_i(t) = \begin{cases} \dfrac{ap_i}{1-\ _{i,1}}[1 - e^{-(1-\ _{i,1})bi,_1 t}] & \text{when } 0 \le t \le \text{²} \\ \dfrac{ap_i}{1-\ _{i,2}}[1 - e^{-(1-\ _{i,1})b_{i,1}\text{²} - (1-\ _{i,2})b_{i,2}(t-\text{²})}] + \dfrac{m(\text{²})(\ _{i,1} - \ _{i,2})}{1-\ _{i,2}} & \text{when } t > \text{²} \end{cases},$$

where $p_i$ is the content proportion of type $i$ fault,

$_{i,1}$ and $_{i,2}$ are the introduction rates for type $i$ fault.

The further modified model can be applied to conduct the software reliability estimation problem not only for the imperfect debugging and change-point case but also the multiple fault types problem. The only difficulty is that more parameters need to be estimated at the same time.

## 3. Cost Models

Assume

$C_1$: Cost of fixing an error during the testing phase,
$C_2$: Cost of fixing an error during the operation phase,
$C_3$: Cost of testing per unit time,
$C_4$: The cost of the system failure for each time,
$T$: Software release time,
$f(t)$: probability density function of the life cycle length .

Then the total software system cost can be defined as

$$EC(T) = C_1 m(T) + \int_T^\infty C_2 (m(t) - m(T)) g(t) dt + \int_T^\infty C_4 (m(t) - m(T)) g(t) dt + C_3 T .$$

To determine the optimal software release time that minimizes the expected software system cost, we let

$$\frac{dEC(T)}{dT} = C_1 m'(T) - C_2 m'(T) F_c(T) - C_4 m'(T) F_c(T) + C_3 = 0$$

where $F_c(T) = \int_T^\infty f(t) dt$.

Set $h(T) = C_2 m'(T) F_c(T) + C_4 m'(T) F_c(T) - C_1 m'(T)$

If $h(0) < C_3$ then $T^* = 0$

If $h(0) \geq C_3$ then $T^* = h^{-1}(C_3)$

## 4. Conclusions

The developed NHPP SRGM is unique in that it allows for the analysis of software failure data with change point, imperfect debugging, and various fault types. The modeling frameworks presented in this paper aim at extension of the change-point problems in the imperfect NHPP SRGM. The results from this investigation to develop such a model for software testing data are very promising. In this study, the change position of the fault detection rate is given at a certain point since we believe the testing process is well tracked. The assumption may not be realistic in some case. However, it is easy to modify this assumption or drop it by assuming the change-point is an additional variable.

Changes in the fault-detection rate are common during the software testing process. When the fault-detection rate is not a smooth function, the new model provides a high possibility to improve both the descriptive and predictive properties of the imperfect NHPP SRGM. In order to further verify the modeling approach in this paper, more applications in a general industrial setting are required.