

## Chapter 2

# NHPP Model and AHP for OSS Reliability Analysis

In this chapter, we focus on the X desktop environment of which the product is the software system developed under the open source project. There are many X desktop environments operating on UNIX-like operating systems which are known as free softwares. Such an X desktop environment is known as the software which offer the interface, design, operability to users. The purpose of X desktop environment is to improve operability by providing the graphical user interfaces on UNIX-like operating systems. At present, the representative examples include GNOME and KDE. Especially, GNOME and KDE have the characteristics of flexible and easy-to-use GUI and a long-use history. On the other hand, these are pointing out as the hefty operability, a lot of library, complexities of dependence, and so on.

In order to consider the effect of each software component on the reliability of a system developed in a distributed environment, we apply the AHP which well established decision making methods. Moreover, we propose the method of reliability assessment based on an SRGM incorporating the interaction among each software component. We also show several numerical examples of software reliability assessment by using the actual data.

## 2.1 Component Reliability Analysis of OSS

### 2.1.1 Reliability Assessment Based on SRGM

Many SRGM's have been used as a conventional methods to assess the reliability, quality control, and testing-process control of software development. Among others, nonhomogeneous Poisson process (NHPP) models have been discussed by many researchers, since these NHPP models can be easily applied during software development. In this section, we discuss NHPP models to analyze software fault-detection count data.

According to the growth curve of the cumulative number of detected faults, we assume that the software reliability in each software component is assessed by applying the following SRGM's based on the NHPP [1]:

- Exponential SRGM.
- Inflection S-shaped SRGM.

The NHPP models have been discussed by many researchers, since the models can be easily applied in actual software projects. Moreover, we apply the method of maximum-likelihood to estimate the model parameters. Below are the expressions for various software reliability assessment measures from the NHPP models.

### 2.1.2 Exponential SRGM

The mean value function of the exponential SRGM is given as follows:

$$E_i(t) = a_i(1 - e^{-b_i t}) \quad (a_i > 0, b_i > 0), \quad (2.1)$$

where  $E_i(t)$  represents the expected cumulative number of faults detected up to the module testing time  $t$  ( $t \geq 0$ ) is mean value functions for the  $i$ th software component. In Eq. (2.1),  $a_i$  ( $i = 1, 2, \dots, n$ ) is the expected number of initial inherent faults for the  $i$ th software component, and  $b_i$  ( $i = 1, 2, \dots, n$ ) the software failure rate per inherent fault for the  $i$ th software component.

### 2.1.3 Inflection S-Shaped SRGM

The mean value function of the inflection S-shaped SRGM is given as follows:

$$D_i(t) = \frac{a_i(1 - e^{-b_i t})}{(1 + c_i \cdot e^{-b_i t})} \quad (a_i > 0, b_i > 0, c_i > 0), \quad (2.2)$$

where  $a_i$  ( $i = 1, 2, \dots, n$ ) is the expected number of initial inherent faults for the  $i$ th software component, and  $b_i$  ( $i = 1, 2, \dots, n$ ) the software failure rate per inherent fault for the  $i$ th software component. Moreover,  $c_i$  ( $i = 1, 2, \dots, n$ ) represents the inflection rate for the  $i$ th software component.

### 2.1.4 Goodness-of-Fit Evaluation Criteria for Applied Model

In this chapter, we compare the model goodness-of-fit of two conventional SRGM's for the observed data set. We use the following goodness-of-fit evaluation criteria,

i.e., the Akaike's information criterion (AIC) and the mean square error (MSE). Suppose that  $K$  data pairs  $(t_k, y_k)$  ( $k = 1, 2, \dots, K$ ) are observed during the system testing-phase, where  $y_k$  is the cumulative number of software failures observed in the time interval  $(0, t_k]$ .

#### (A) AIC

AIC helps us to select the optimal model among ones estimated by the method of maximum-likelihood. It is given by

$$\text{AIC} = -2 \cdot (\text{the logarithmic maximum-likelihood}) + 2 \cdot (\text{the number of free model parameters}). \quad (2.3)$$

Differences among AIC values are significant, not the value themselves. A model possessing the smallest AIC best a data set when the difference between two models is greater than or equal to 1. However, there are no significant difference among two models in the case where the differences of AIC's are less than 1.

#### (B) MSE

The mean square error can be obtained by dividing the sum of square errors between the observed value,  $y_k$ , and its estimate,  $\hat{y}_k$ , by the number of pairs of data,  $n$ . That is,

$$\text{MSE} = \frac{1}{n} \sum_{k=1}^n (y_k - \hat{y}_k)^2. \quad (2.4)$$

$\hat{y}_k$  in Eq. (2.4) is obtained from  $\hat{y}_k = \hat{H}(t_k)$  ( $k = 1, 2, \dots, n$ ). A small mean square error indicates that the selected model fits the observed well.

We compare the two conventional SRGM's by using the above-described goodness-of-fit evaluation criteria. Concretely speaking, AIC is the first goodness-of-fit evaluation criterion, and MSE is the secondary goodness-of-fit measure, i.e., we select the appropriate model when the difference of value in AIC are greater than or equal to 1, otherwise we select the appropriate model based on the value of MSE.

### 2.1.5 Weight Parameter for Each Component Based on AHP

The AHP developed in the 1970s is utilized widely in Europe and the United States for management issues, energy problems, decision-making, urban planning. The AHP is considered to be one of the most effective methods for decision-making support [2, 3].

When considering the effect of debugging process on an entire system in the development of a software reliability assessment method for distributed development environment, it is necessary to grasp the deeply-intertwined factors, such as programming path, size of a component, skill of fault reporter, and so on.

Also, it is rare that collected data sets entirely contain all the information needed to assess software reliability, although these data sets for deeply-intertwined factors completely collected from the bug tracking system. Therefore, it is difficult to estimate the effect of each component on the entire system by using the collected data sets only.

In this chapter, we propose a reliability assessment method based on the AHP to estimate the effect of each component on the entire system in a complicated environment. Specifically, we can assess the importance level of faults detected for each component, the size of component, the skill of fault reporter and so on, as evaluation criteria for the AHP.

Let  $w_i (i = 1, 2, \dots, n)$  be the weight parameters for evaluation criteria of the AHP. Then, the pair comparison matrix is given as follows:

$$A = \begin{bmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \dots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \dots & \frac{w_2}{w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \dots & \frac{w_n}{w_n} \end{bmatrix}. \quad (2.5)$$

We can obtain the weight parameter  $\alpha_i$  for each evaluation criterion from the above pair comparison matrix by using the following geometric average:

$$\alpha_i = \sqrt[n]{\prod_{j=1}^n x_{ij}}, \quad (2.6)$$

$$x_{ij} = \frac{w_i}{w_j}.$$

Therefore, the total weight parameter for each evaluation criterion is given by the following equation:

$$\beta_i = \frac{\alpha_i}{\sum_{i=1}^n \alpha_i}. \quad (2.7)$$

By using the weight parameter  $\beta_i$  in Eq.(2.7), we can obtain the total weight parameter  $p_i$  which represents the level of importance for each component.

## 2.2 Reliability Analysis for Entire OSS System

### 2.2.1 Logarithmic Execution Time Model

The operating environment of OSS possesses characteristics of the susceptible to various application softwares. Therefore, it is different from a conventional software system developed by an identical organization. Then, the expected number of detected faults continues to increase from the effect of the interaction among application softwares, i.e., the number of detected faults can not converge to a fixed value.

As mentioned above, we apply the logarithmic Poisson execution time model based on the assumption that the number of detected faults is infinity. Thus, we consider the following structure of the mean value function  $\mu(t)$  for the entire system because an NHPP model is characterized by its mean value function:

$$\mu(t) = \frac{1}{\theta - P} \ln[\lambda_0(\theta - P)t + 1] \quad (0 < \theta, 0 < \lambda_0, 0 < P < 1), \quad (2.8)$$

where  $\lambda_0$  is the intensity of initial inherent failure,  $\theta$  the reduction rate of the failure intensity rate per inherent fault. Moreover, we assume that the parameter  $P$  in Eq. (2.8) represents the following weighted average in terms of the weight parameter  $p_i$  estimated by the AHP and the software failure rate per inherent fault  $b_i$  for the  $i$ th software component in Eqs. (2.1) or (2.2):

$$P = \sum_{i=1}^n p_i \cdot b_i, \quad (2.9)$$

where  $n$  represents the number of software components and  $p_i (i = 1, 2, \dots, n)$  the weight parameter for each component.

### 2.2.2 Reliability Assessment Measures

We can give the following expressions as software reliability assessment measures derived from the NHPP model given by Eq. (2.8):

- *Software reliability*

The software reliability can be defined as the probability that a software failure does not occur during the time-interval  $(t, t+x]$  ( $t \geq 0, x \geq 0$ ) after the testing-time  $t$ . The software reliability is given by

$$R(x|t) = \exp[\mu(t) - \mu(t+x)], \quad (t \geq 0, x \geq 0). \quad (2.10)$$

- *Instantaneous mean time between software failures*

The instantaneous mean time between software failures ( $MTBF_I$ ) measures the frequency of software failure-occurrence, and is given by

$$MTBF_I(t) = \frac{1}{\frac{d\mu(t)}{dt}}. \quad (2.11)$$

- *Cumulative mean time between software failures*

The cumulative mean time between software failures ( $MTBF_C$ ) is given as follows:

$$MTBF_C(t) = \frac{t}{\mu(t)}. \quad (2.12)$$

## References

1. S. Yamada, *Software Reliability Modeling: Fundamentals and Applications* (Springer, Tokyo, 2014)
2. T. Satty, *The Analytic Hierarchy Process* (McGraw-Hill, New York, 1980)
3. E. Kinoshita, *Introductory AHP (in Japanese)* (JUSE Press, Tokyo, 2000)



<http://www.springer.com/978-3-319-31817-2>

OSS Reliability Measurement and Assessment

Yamada, S.; Tamura, Y.

2016, X, 185 p. 83 illus., Hardcover

ISBN: 978-3-319-31817-2