# 27. Statistical Models for Predicting Reliability of Software Systems in Random Environments

After a brief overview of existing models in software reliability in Sect. 27.1, Sect. 27.2 discusses a generalized nonhomogeneous Poisson process model that can be used to derive most existing models in the software reliability literature. Section 27.3 describes a generalized random field environment (RFE) model incorporating both the testing phase and operating phase in the software development cycle for estimating the reliability of software systems in the field. In contrast to some existing models that assume the same software failure rate for the software testing and field operation environments, this generalized model considers the random environmental effects on software reliability. Based on the generalized RFE model, Sect. 27.4 describes two specific RFE reliability models, the $\gamma$-RFE and $\beta$-RFE models, for predicting software reliability in field environments. Section 27.4 illustrates the models using telecommunication software failure data. Some further research considerations based on the generalized software reliability model are also discussed.





Many software reliability models have been proposed to help software developers and managers understand and analyze the software development process, estimate the development cost and assess the level of software reliability. Among these software reliability models, models based on the nonhomogeneous Poisson process (NHPP) have been successfully applied to model the software failure processes that possess certain trends such as reliability growth or deterioration. NHPP models are very useful to predict software failures and software reliability in terms of time, and to determine when to stop testing and release the software [27.1].

Currently most existing NHPP software reliability models have been carried out through the fault intensity rate function and the mean-value functions $m(t)$ within a controlled operating environment [27.2–13]. Obviously, different models use different assumptions and therefore provide different mathematical forms for the mean-value function $m(t)$. Table 27.1 shows a summary of several existing models appearing in the software reliability engineering literature [27.14]. Generally, these models are applied to software testing data and then to make predictions of software failures and reliability in the field. The underlying assumption for this application is that the field environments are the same as, or close to, a testing environment; this is valid for some software systems that are only used in one environment throughout their entire lifetime. However, this assumption is not valid for many applications where a software program may be used in many different locations once it is released.

The operating environments for the software in the field are quite different. The randomness of the field environment will affect software failure and software reliability in an unpredictable way. *Yang* and *Xie* [27.15] mentioned that the operational reliability and testing reliability are often different from each other, but they assumed that the operational failure rate is still close to the testing failure rate, and hence that the difference between them is that the operational failure rate decreases with time, while the testing failure rate remains constant. *Zhang* et al. [27.16] proposed an NHPP software reliability calibration model

**Table 27.1** Summary of NHPP software reliability models [27.14]

| Model name | Model type | MVF [$m(t)$] | Comments |
|---|---|---|---|
| Goel–Okumoto (G–O) | Concave | $m(t) = a(1 - e^{-bt})$<br>$a(t) = a$<br>$b(t) = b$ | Also called exponential model |
| Delayed S-shaped | S-shaped | $m(t) = a[1 - (1 + bt)e^{-bt}]$<br>$a(t) = a$<br>$b(t) = \frac{b^2 t}{1+bt}$ | Modification of G–O model to obtain S-shape |
| Inflection S-shaped SRGM | Concave | $m(t) = \frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$<br>$a(t) = a$<br>$b(t) = \frac{b}{1+\beta e^{-bt}}$ | Solves a technical condition with the G–O model. Becomes the same as G–O if $\beta = 0$ |
| HD/G–O model | Concave | $m(t) = \log\left[(e^a - c)/(e^{a e^{-bt}} - c)\right]$ | Same as G–O when $c = 0$ |
| Yamada exponential | Concave | $m(t) = a\left(1 - e^{-r\alpha(1-e^{(-\beta t)})}\right)$<br>$a(t) = a$<br>$b(t) = r\alpha\beta e^{-\beta t}$ | Attempts to account for testing effort |
| Yamada Rayleigh | S-shaped | $m(t) = a\left(1 - e^{-r\alpha(1-e^{(-\beta t^2/2)})}\right)$<br>$a(t) = a$<br>$b(t) = r\alpha\beta t e^{-\beta t^2/2}$ | Attempts to account for testing effort |
| Yamada imperfect debugging model (1) | S-shaped | $m(t) = \frac{ab}{\alpha+b}(e^{\alpha t} - e^{-bt})$<br>$a(t) = a e^{\alpha t}$<br>$b(t) = b$ | Assumes exponential fault-content function and constant fault-detection rate |
| Yamada imperfect debugging model (2) | S-shaped | $m(t) = a(1 - e^{-bt})(1 - \frac{\alpha}{b}) + \alpha a t$<br>$a(t) = a(1 + \alpha t)$<br>$b(t) = b$ | Assumes constant fault-introduction rate $\alpha$ and constant fault-detection rate |
| PNZ model | S-shaped and concave | $m(t) = \frac{a}{1+\beta} e^{-bt}\left[(1 - e^{-bt})(1 - \frac{\alpha}{b}) + \alpha a t\right]$<br>$a(t) = a(1 + \alpha t)$<br>$b(t) = \frac{b}{1+\beta e^{-bt}}$ | Assumes introduction rate is a linear function of testing time, and the fault-detection rate function is nondecreasing and inflexion S-shaped |
| Pham–Zhang model | S-shaped and concave | $m(t) = \frac{1}{1+\beta e^{-bt}}\left[(c + a)(1 - e^{-bt}) - \frac{a}{b-\alpha}(e^{-\alpha t} - e^{-bt})\right]$<br>$a(t) = c + a(1 - e^{-\alpha t})$<br>$b(t) = \frac{b}{1+\beta e^{-bt}}$ | Assume constant introduction rate is exponential function of testing time, and the error-detection function is nondecreasing with an inflexion S-shaped model |

by introducing a calibration factor. This calibration factor, $K$, obtained from software failures in both the testing and field operation phases will be a multiplier to the software failure intensity. This calibrated software reliability model can be used to assess and adjust the predictions of software reliability in the operation phase.

Instead of relating the operating environment to the failure intensity $\lambda$, in this chapter we assume that the effect of the operating environment is to multiply the unit failure-detection rate $b(t)$ achieved in the testing environment using the concept of the proportional hazard approach suggested by *Cox* [27.17]. If the operating environment is more liable to software failure, then the unit fault-detection rate increases by some factor $\eta$ greater than 1. Similarly, if the operating environment is less liable to software failure, then the unit fault-detection rate decreases by some positive factor $\eta$ less than 1.

This chapter describes a model based on the NHPP model framework for predicting software failures and evaluating the software reliability in random field environments. Based on this model, developers and engineers can further develop specific software reliability models customized to various applications.

| Notations | |
|---|---|
| $R(t)$ | Software reliability function |
| $\eta$ | Random environmental factor |
| $G(\eta)$ | Cumulative distribution function of $\eta$ |
| $\gamma$ | Shape parameter of gamma distributions |
| $\theta$ | Scale parameter of gamma distributions |
| $\alpha, \beta$ | Parameters of beta distributions |
| $N(t)$ | Counting process which counts the number of software failures discovered by time $t$ |
| $m(t)$ | Expected number of software failures detected by time $t$, $m(t) = E[N(t)]$ |
| $a(t)$ | Expected number of initial software faults plus introduced faults by time $t$ |
| $m_1(t)$ | Expected number of software failures in testing by time $t$ |
| $m_2(t)$ | Expected number of software failures in the field by time $t$ |
| $a_1(t)$ | Expected number of initial software faults plus introduced faults discovered in the testing by time $t$ |
| $a$ | Number of initial software faults at the beginning of testing phase, is a *software parameter* that is directly related to the software itself |
| $T$ | Time to stop testing and release the software for field operations |
| $a_F$ | Number of initial software faults in the field (at time $T$) |
| $b(t)$ | Failure detection rate per fault at time $t$, is a *process parameter* that is directly related to testing and failure process |
| $p$ | Probability that a fault will be successfully removed from the software |
| $q$ | Error introduction rate at time $t$ in the testing phase |
| MLE | Maximum likelihood estimation |
| RFE-model | Software reliability model subject to a random field environment |
| $\gamma$-RFE | Software reliability model with a gamma distributed field environment |
| $\beta$-RFE | Software reliability model with a beta distributed field environment |
| NHPP | Non-homogeneous Poisson process |
| SRGM | software reliability growth model |
| HD | Hossain–Ram |
| PNZ | Pham–Nordman–Zhang |
| G–O | Goel–Okumoto |
| NHPP | nonhomgeneous Poisson process |
| MLE | maximum likelihood estimation |
| RFE | random field environment |



## 27.1 A Generalized NHPP Software Reliability Model

A generalized NHPP model studied by *Zhang* et al.[27.7] can be formulated as follows:

$$m'(t) = \eta b(t)[a(t) - pm(t)] \,, \tag{27.1}$$

$$a'(t) = q \cdot m'(t) \,, \tag{27.2}$$

where $m(t)$ is the number of software failures expected to be detected by time $t$. If the marginal conditions are given as $m(0) = 0$ and $a(0) = a$, then for a specific environmental factor $\eta$, the solutions to (27.1) and (27.2) are, given in [27.7], as follows

$$m_\eta(t) = a \int_0^t \eta b(u) \mathrm{e}^{-\int_0^u \eta(p-q)b(\tau)\mathrm{d}\tau} \mathrm{d}u \,, \tag{27.3}$$

$$a_\eta(t) = a \left[ 1 + \int_0^t \eta q b(u) \mathrm{e}^{-\int_0^u \eta(p-q)\cdot b(\tau)\mathrm{d}\tau} \mathrm{d}u \right] . \tag{27.4}$$

This is the generalized form of the NHPP software reliability model. When $p = 1$, $\eta = 1$ and $q = 0$, then for any given function $a(t)$ and $b(t)$, all the functions listed in Table 27.1 can easily be obtained.

## 27.2 Generalized Random Field Environment (RFE) Model

The testing environment is often a controlled environment with much less variation compared to the field environments, which may be quite different for the field application software. Once a software program is released, it may be used in many different locations and various applications in industries. The operating environments for the software are quite different. Therefore, the randomness of the field environment will affect the cumulative software failure data in an unpredictable way.

Figure 27.1 shows the last two phases of the software life cycle: in-house testing and field operation [27.18]. If $T$ is the time to stop testing and release the software for field operations, then the time period $0 \le t \le T$ refers to the time period of *software testing*, while the time period $T \le t$ refers to the post-release period—*field operation*.

The environmental factor $\eta$ is used to capture the uncertainty about the environment and its effects on the software failure rate. In general, software testing is carried out in a controlled environment with very small variations, which can be used as a reference environment where $\eta$ is constant and equal to 1. For the field operating environment, the environmental factor $\eta$ is assumed to be a nonnegative random variable (RV) with probability density function (PDF) $f(\eta)$, i. e.

$$\eta = \begin{cases} 1 & t \le T \\ \text{RV with PDF } f(\eta) & t \ge T \end{cases} . \tag{27.5}$$

If the value of $\eta$ is less than 1, this indicates that the conditions are less favorable to fault detection than that of testing environment. Likewise, if the value of $\eta$ is greater than 1, it indicates that the conditions are more favorable to fault detection than that of the testing environment.

From (27.3) and (27.5), the mean-value function and the function $a_1(t)$ during testing can be obtained as

$$m_1(t) = a \int_0^t b(u) \mathrm{e}^{-\int_0^u (p-q)b(\tau)\mathrm{d}\tau} \mathrm{d}u \quad t \le T ,$$

$$a_1(t) = a\left[1 + \int_0^t qb(u) \times \mathrm{e}^{-\int_0^u (p-q)\cdot b(\tau)\mathrm{d}\tau} \mathrm{d}u\right] \quad t \le T . \tag{27.6}$$

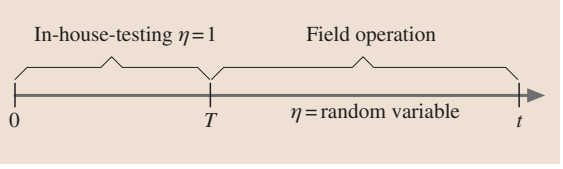


**Fig. 27.1** Testing versus field environment where $T$ is the time to stop testing and release the software

For the field operation where $t \ge T$ the mean-value function can be represented as

$$\begin{aligned} m_2(t) &= m_1(T) + \int_0^\infty m_\eta(t) f(\eta)\mathrm{d}\eta \\ &= m_1(T) + \int_0^\infty \left[a_\mathrm{F} \int_T^t \eta b(u) \times \mathrm{e}^{-\int_T^u \eta(p-q)b(\tau)\mathrm{d}\tau} \mathrm{d}u\right] f(\eta)\mathrm{d}\eta \quad t \ge T \\ &= m_1(T) + \int_T^t a_\mathrm{F} b(u) \left[\int_0^\infty \eta \times \mathrm{e}^{-\eta\int_T^u (p-q)b(\tau)\mathrm{d}\tau} f(\eta)\mathrm{d}\eta\right] \mathrm{d}u , \end{aligned} \tag{27.7}$$

where $a_\mathrm{F}$ is the number of faults in the software at time $T$. Using the Laplace transform formula, the mean-value function can be rewritten as

$$\begin{aligned} m_2(t) &= m_1(T) + \int_T^t a_\mathrm{F} b(u) \times \left(-\frac{\mathrm{d}F^*(s)}{\mathrm{d}s}\bigg|_{s=\int_0^u (p-q)b(\tau)\mathrm{d}\tau}\right) \mathrm{d}u , \quad t \ge T \\ &= m_1(T) + \frac{a_\mathrm{F}}{(p-q)} \times \int_T^t \left\{-\mathrm{d}F^*\left[(p-q)\int_T^u b(\tau)\mathrm{d}\tau\right]\right\} , \end{aligned}$$

where $F^*(s)$ is the Laplace transform of the PDF $f(x)$ and

$$\int_0^\infty x\,\mathrm{e}^{-x\cdot s} f(x)\,\mathrm{d}x = -\frac{\mathrm{d}F^*(s)}{\mathrm{d}s}$$

or, equivalently,

$$\begin{aligned} m_2(t) &= m_1(T) - \frac{a_\mathrm{F}}{(p-q)} \\ &\quad \times F^*\left[(p-q)\int_T^u b(\tau)\,\mathrm{d}\tau\right]\Bigg|_T^t, \quad t \ge T \\ &= m_1(T) + \frac{a_\mathrm{F}}{(p-q)} \\ &\quad \times \left\{F^*(0) - F^*\left[(p-q)\int_T^t b(\tau)\,\mathrm{d}\tau\right]\right\}. \end{aligned}$$

Notice that $F^*(0) = \int_0^\infty \mathrm{e}^{-0x} f(x)\,\mathrm{d}x = 1$, so

$$\begin{aligned} m_2(t) &= m_1(T) + \frac{a_\mathrm{F}}{(p-q)} \\ &\quad \times \left\{1 - F^*\left[(p-q)\int_T^t b(\tau)\,\mathrm{d}\tau\right]\right\} \quad t \ge T. \end{aligned}$$

The expected number of faults in the software at time $T$ is given by

$$\begin{aligned} a_\mathrm{F} &= a_1(T) - p m_1(T) \\ &= a\left[1 - \int_0^t (p-q) b(u)\,\mathrm{e}^{-\int_0^u (p-q)\cdot b(\tau)\,\mathrm{d}\tau}\,\mathrm{d}u\right] \\ &= a\,\mathrm{e}^{-\int_0^t (p-q) b(\tau)\,\mathrm{d}\tau}. \end{aligned}$$

The generalized RFE model can be obtained as

$$m(t) = \begin{cases} \frac{a}{(p-q)}\left(1 - \mathrm{e}^{-(p-q)\int_0^u b(\tau)\,\mathrm{d}\tau}\right) & t \le T \\ \frac{a}{(p-q)}\left\{1 - \mathrm{e}^{-(p-q)\int_0^T b(\tau)\,\mathrm{d}\tau}\right. \\ \left.\times F^*\left[(p-q)\int_T^t b(\tau)\,\mathrm{d}\tau\right]\right\} & t \ge T. \end{cases} \tag{27.8}$$

The model in (27.8) is a generalized software reliability model subject to random field environments. The next section presents specific RFE models for the gamma and beta distributions of the random field environmental factor $\eta$.

## 27.3 RFE Software Reliability Models

Obviously, the environmental factor $\eta$ must be nonnegative. Any suitable nonnegative distribution may be used to describe the uncertainty about $\eta$. In this section we present two RFE models. The first model is a $\gamma$-RFE model, based on the gamma distribution, which can be used to evaluate and predict software reliability in field environments where the software failure-detection rate can be either greater or less than the failure detection rate in the testing environment. The second model is a $\beta$-RFE model, based on the beta distribution, which can be used to predict software reliability in field environments where the software failure detection rate can only be less than the failure detection rate in the testing environment.

### 27.3.1 $\gamma$-RFE Model

In this model, we use the gamma distribution to describe the random environmental factor $\eta$. This model is called the $\gamma$-RFE model.

Assume that $\eta$ follows a gamma distribution with a probability density function as follows

$$f_\gamma(\eta) = \frac{\theta^\gamma \eta^{\gamma-1}\,\mathrm{e}^{-\theta\cdot\eta}}{\Gamma(\gamma)}, \gamma, \theta > 0; \eta \ge 0. \tag{27.9}$$

The gamma distribution has sufficient flexibility and has desirable qualities with respect to computations [27.18]. Figure 27.2 shows an example of the gamma density

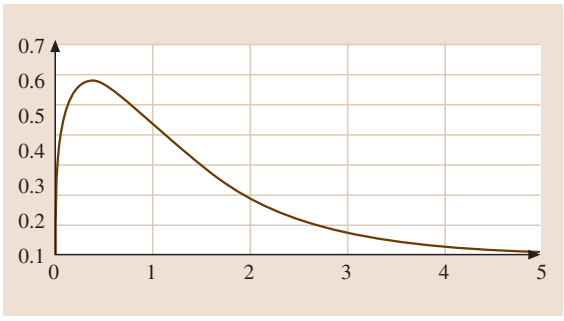


**Fig. 27.2** A gamma density function

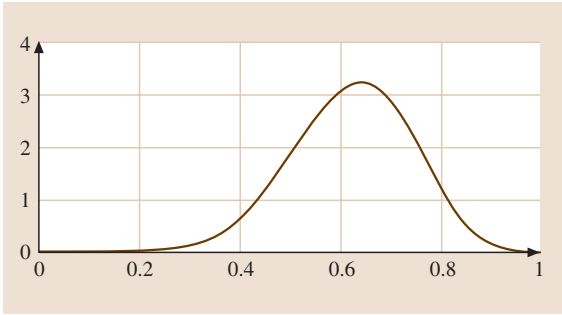


**Fig. 27.3** A PDF curve of the beta distribution



probability function. The gamma function seems to be reasonable to describe a software failure process in those field environments where the software failure-detection rate can be either greater (i. e., $\eta > 1$) or less than (i. e., $\eta < 1$) the failure-detection rate in the testing environment.

The Laplace transform of the probability density function in (27.9) is

$$F^*(s) = \left(\frac{\theta}{\theta + s}\right)^{\gamma} . \tag{27.10}$$

Assume that the error-detection rate function $b(t)$ is given by

$$b(t) = \frac{b}{1 + c\,\mathrm{e}^{-b\cdot t}} , \tag{27.11}$$

where $b$ is the asymptotic unit software-failure detection rate and $c$ is the parameter defining the shape of the learning curve, then from (27.8) the mean-value function of the $\gamma$-RFE model can be obtained as follows

$$m_{\gamma}(t) = \begin{cases} \frac{a}{(p-q)}\left[1 - \left(\frac{1+c}{\mathrm{e}^{bt}+c}\right)^{(p-q)}\right] & t \le T , \\ \frac{a}{(p-q)}\left[1 - \left(\frac{1+c}{\mathrm{e}^{bT}+c}\right)^{(p-q)} \right. \\ \left. \times \left(\frac{\theta}{\theta + (p-q)\ln\left(\frac{c+\mathrm{e}^{bt}}{c+\mathrm{e}^{bT}}\right)}\right)^{\gamma}\right] & t \ge T . \end{cases} \tag{27.12}$$

### 27.3.2 β-RFE Model

This section presents a model using the beta distribution that describes the random environmental factor $\eta$, called the $\beta$-RFE model.

The beta PDF is

$$f_{\beta}(\eta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\eta^{\alpha-1}(1-\eta)^{\beta-1} , \quad 0 \le \eta \le 1, \alpha > 0, \beta > 0 . \tag{27.13}$$

Figure 27.3 shows an example of the beta density function. It seems that the $\beta$-RFE model is a reasonable function to describe a software failure process in those field environments where the software failure-detection rate can only be less than the failure-detection rate in the testing environment. This is not uncommon in the software industry because, during software testing, the engineers generally test the software intensely and conduct an *accelerated* test on the software in order to detect most of the software faults as early as possible.

The Laplace transform of the PDF in (27.13) is

$$F^*_{\beta}(s) = \mathrm{e}^{-s} \cdot HG([\beta], [\alpha+\beta], s) , \tag{27.14}$$

where $HG([\beta], [\alpha+\beta], s)$ is a generalized hypergeometric function such that

$$HG([a_1, a_2, \ldots, a_m], [b_1, b_2, \ldots, b_n], s) = \sum_{k=0}^{\infty}\left(\frac{s^k \prod_{i=1}^{m}\frac{\Gamma(a_i+k)}{\Gamma(a_i)}}{\prod_{i=1}^{n}\frac{\Gamma(b_i+k)}{\Gamma(b_i)}k!}\right) ,$$

Therefore

$$\begin{aligned} F^*_{\beta}(s) &= \mathrm{e}^{-s}\sum_{k=0}^{\infty}\left(\frac{\Gamma(\alpha+\beta)\Gamma(\beta+k)s^k}{\Gamma(\beta)\Gamma(\alpha+\beta+k)k!}\right) \\ &= \sum_{k=0}^{\infty}\left(\frac{\Gamma(\alpha+\beta)\Gamma(\beta+k)}{\Gamma(\beta)\Gamma(\alpha+\beta+k)}\frac{s^k\,\mathrm{e}^{-s}}{k!}\right) \\ &= \sum_{k=0}^{\infty}\left(\frac{\Gamma(\alpha+\beta)\Gamma(\beta+k)}{\Gamma(\beta)\Gamma(\alpha+\beta+k)}\mathrm{Poisson}(k,s)\right) . \end{aligned}$$

where the Poisson probability density function is given by

$$\mathrm{Poisson}(k, s) = \frac{s^k\,\mathrm{e}^{-s}}{k!} .$$

Using the same error-detection rate function in (27.11) and replacing $F^*(s)$ by $F^*_{\beta}(s)$, the mean-value function of the $\beta$-RFE model is

$$m_{\beta}(t) = \begin{cases} \frac{a}{(p-q)}\left[1 - \left(\frac{1+c}{\mathrm{e}^{bt}+c}\right)^{(p-q)}\right] & t \le T , \\ \frac{a}{(p-q)}\left[1 - \left(\frac{1+c}{\mathrm{e}^{bT}+c}\right)^{(p-q)} \right. \\ \left. \times \sum_{k=0}^{\infty}\left(\frac{\Gamma(\alpha+\beta)\Gamma(\beta+k)\mathrm{Poisson}(k,s)}{\Gamma(\beta)\Gamma(\alpha+\beta+k)}\right)\right] & t \ge T . \end{cases} \tag{27.15}$$

where

$$s = (p-q)\left[\ln\left(\frac{c+\mathrm{e}^{bt}}{c+\mathrm{e}^{bT}}\right)\right] .$$

## 27.4 Parameter Estimation

### 27.4.1 Maximum Likelihood Estimation (MLE)

We use the MLE method to estimate the parameters in these two RFE models. Let $y_i$ be the cumulative number of software faults detected up to time $t_i$, $i = 1, 2, \ldots, n$. Based on the NHPP, the likelihood function is given by

$$L = \prod_{i=1}^{n} \frac{\left[m(t_i) - m(t_{i-1})\right]^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} \mathrm{e}^{-\left[m(t_i) - m(t_{i-1})\right]} . \tag{27.16}$$

The logarithmic form of the above likelihood function is

$$\ln L = \sum_{i=1}^{n} \left\{ (y_i - y_{i-1}) \ln \left[m(t_i) - m(t_{i-1})\right] - \left[m(t_i) - m(t_{i-1})\right] - \ln \left[(y_i - y_{i-1})!\right] \right\} . \tag{27.17}$$

In this analysis, the error-removal efficiency $p$ is given. Each model has five unknown parameters. For example, in the $\gamma$-RFE model, we need to estimate the following five unknown parameters: $a$, $b$, $q$, $\gamma$ and $\theta$. For the $\beta$-RFE model, we need to estimate: $a$, $b$, $q$, $\alpha$ and $\beta$. By taking derivatives of (27.18) with respect to each parameter and setting the results equal to zero, we can obtain five equations for each RFE model. After solving all those equations, we obtain the maximum likelihood estimates (MLEs) of all parameters for each RFE model.

**Table 27.2** Normalized cumulative failures and times during software testing

| Time | Failures | Time | Failures | Time | Failures |
|---|---|---|---|---|---|
| 0.0001 | 0.0249 | 0.0038 | 0.3483 | 0.0121 | 0.6766 |
| 0.0002 | 0.0299 | 0.0044 | 0.3532 | 0.0128 | 0.7015 |
| 0.0002 | 0.0647 | 0.0048 | 0.3682 | 0.0135 | 0.7363 |
| 0.0003 | 0.0647 | 0.0053 | 0.3881 | 0.0142 | 0.7761 |
| 0.0005 | 0.1095 | 0.0058 | 0.4478 | 0.0147 | 0.7761 |
| 0.0006 | 0.1194 | 0.0064 | 0.4876 | 0.0155 | 0.8159 |
| 0.0008 | 0.1443 | 0.0070 | 0.5224 | 0.0164 | 0.8259 |
| 0.0012 | 0.1692 | 0.0077 | 0.5473 | 0.0172 | 0.8408 |
| 0.0016 | 0.1990 | 0.0086 | 0.5821 | 0.0176 | 0.8458 |
| 0.0023 | 0.2289 | 0.0095 | 0.6119 | 0.0180 | 0.8756 |
| 0.0028 | 0.2637 | 0.0105 | 0.6368 | 0.0184 | 0.8955 |
| 0.0033 | 0.3134 | 0.0114 | 0.6468 | 0.0184 | 0.9005 |

Table 27.2 shows a set of failure data from a telecommunication software application during software testing [27.16]. The column "Time" shows the normalized cumulative time spent in software testing for this telecommunication application, and the column "Failures" shows the normalized cumulative number of failures occurring in the testing period up to the given time.

The time to stop testing is $T = 0.0184$. After the time $T$, the software is released for field operations. Table 27.3 shows the field data for this software release. Similarly, the column "Time" shows the normalized cumulative time spent in the field for this software application, and the time in Table 27.3 is continued from the time to stop testing $T$. The column "Failures" shows the normalized cumulative number of failures found after releasing the software for field operations up to the given time. The cumulative number of failures is the total number of software failures since the beginning of software testing.

To obtain a better understanding of the software development process, we show the actual results of the MLE solutions instead of the normalized results. In this study, let us assume that testing engineers have a number of years of experience of this particular product and software development skills and therefore conducted perfect debugging during the test. In other word, $p = 1$. The maximum likelihood estimates of all the parameters in the $\gamma$-RFE model are obtained as shown in Table 27.4.

**Table 27.3** Normalized cumulative failures and their times in operation

| Time | Failures | Time | Failures | Time | Failures |
|---|---|---|---|---|---|
| 0.0431 | 0.9055 | 0.3157 | 0.9751 | 0.7519 | 0.9900 |
| 0.0616 | 0.9104 | 0.3407 | 0.9751 | 0.7585 | 0.9900 |
| 0.0801 | 0.9204 | 0.3469 | 0.9751 | 0.7718 | 0.9900 |
| 0.0863 | 0.9254 | 0.3967 | 0.9751 | 0.7983 | 0.9900 |
| 0.1357 | 0.9303 | 0.4030 | 0.9801 | 0.8251 | 0.9900 |
| 0.1419 | 0.9353 | 0.4291 | 0.9851 | 0.8453 | 0.9900 |
| 0.1666 | 0.9453 | 0.4357 | 0.9851 | 0.8520 | 0.9900 |
| 0.2098 | 0.9453 | 0.4749 | 0.9851 | 0.9058 | 0.9900 |
| 0.2223 | 0.9502 | 0.5011 | 0.9851 | 0.9126 | 0.9900 |
| 0.2534 | 0.9502 | 0.5338 | 0.9851 | 0.9193 | 0.9900 |
| 0.2597 | 0.9502 | 0.5731 | 0.9851 | 0.9395 | 0.9950 |
| 0.2659 | 0.9502 | 0.6258 | 0.9900 | 0.9462 | 0.9950 |
| 0.2721 | 0.9552 | 0.6656 | 0.9900 | 0.9529 | 1.0000 |
| 0.2971 | 0.9602 | 0.6789 | 0.9900 | 0.9865 | 1.0000 |
| 0.3033 | 0.9701 | 0.7253 | 0.9900 | 1.0000 | 1.0000 |

**Table 27.4** MLE solutions for the $\gamma$-RFE model

| $\hat{a}$ | $\hat{b}$ | $\hat{q}$ | $\hat{c}$ | $\hat{\gamma}$ | $\hat{\theta}$ |
|---|---|---|---|---|---|
| 236.58 | 0.001443 | 0 | 0 | 0.2137 | 10.713 |

**Table 27.5** MLE solutions for the $\beta$-RFE model

| $\hat{a}$ | $\hat{b}$ | $\hat{q}$ | $\hat{c}$ | $\hat{\alpha}$ | $\hat{\beta}$ |
|---|---|---|---|---|---|
| 236.07 | 0.001449 | 0 | 0 | 0.1862 | 8.6922 |

Similarly, set $p = 1$, the MLE of all the parameters in the $\beta$-RFE model are obtained as shown in Table 27.5.

For both RFE models, the MLE results can be used to obtain more insightful information about the software development process. In this example, at the time to stop testing the software $T = 0.0184$, the estimated number of remaining faults in the system is $a_F = a - (p-q)m(T) = 55$.

## 27.4.2 Mean-Value Function Fits

After we obtain the MLEs for all the parameters, we can plot the mean-value function curve fits for both the $\gamma$-RFE and $\beta$-RFE models based on the MLE parameters against the actual software application failures.

Table 27.6 shows the mean-value function curve fits for both the models where the column $m_\gamma(t)$ and $m_\beta(t)$ show the mean-value function for the $\gamma$-RFE model and the $\beta$-RFE model, respectively.

The $\gamma$-RFE and $\beta$-RFE models yield very close fits and predictions on software failures. Figure 27.4 shows the mean-value function curve fits for both the $\gamma$-RFE model and $\beta$-RFE model. Both models appear to be a good fit for the given data set. Since we are particularly interested in the fits and the predictions for software failure data during field operation, we also plot the detailed mean-value curve fits for both the $\gamma$-RFE model and the $\beta$-RFE model in Fig. 27.5.

For the overall fitting of the mean-value function against the actual software failures, the mean squared error (MSE) is 23.63 for the $\gamma$-RFE model fit, and is 23.69 for the $\beta$-RFE model. We can also obtain the fits and predictions for software failures by applying some existing NHPP software reliability models to the same set of failure data. Since all these existing models assumes a constant failure-detection rate throughout both the software testing and operation periods, we only apply the software testing data to the software models and then predict the software failures in the field environments.

Figure 27.6 shows the comparisons of the mean-value function curve fits between the two RFE models and some existing NHPP software reliability models. It appears that the two models that

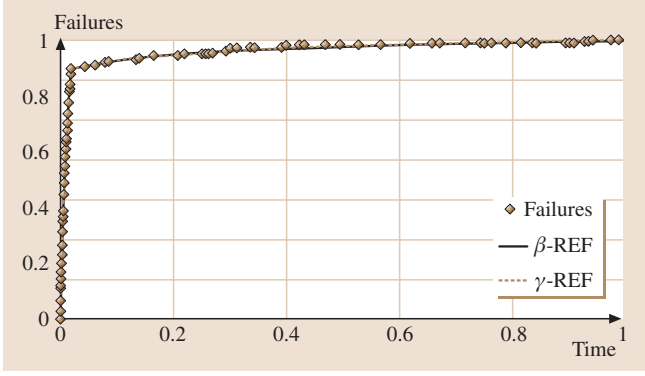


**Fig. 27.4** Mean-value function curve fits for both RFE models

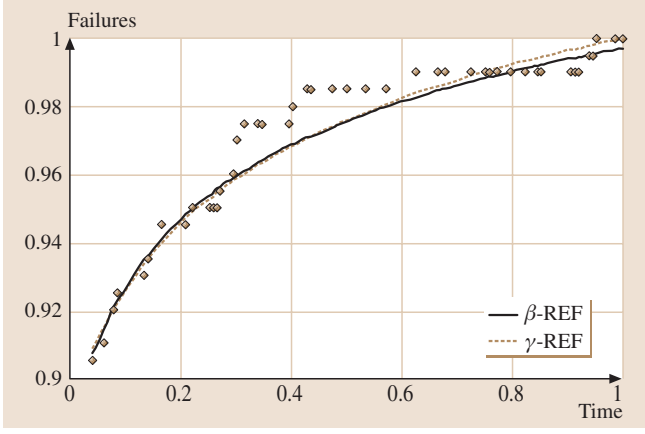


**Fig. 27.5** Mean-value function fitting comparisons

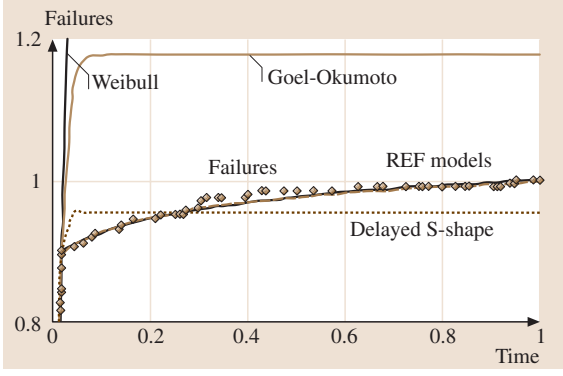


**Fig. 27.6** Model comparisons

**Table 27.6** The mean-value functions for both RFEs models

| Time | Failures | $m_\gamma(t)$ | $m_\beta(t)$ | Time | Failures | $m_\gamma(t)$ | $m_\beta(t)$ |
|---|---|---|---|---|---|---|---|
| 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.1357 | 0.9303 | 0.9340 | 0.9341 |
| 0.0001 | 0.0249 | 0.0085 | 0.0085 | 0.1419 | 0.9353 | 0.9352 | 0.9354 |
| 0.0002 | 0.0299 | 0.0152 | 0.0152 | 0.1666 | 0.9453 | 0.9398 | 0.9399 |
| 0.0002 | 0.0647 | 0.0219 | 0.0219 | 0.2098 | 0.9453 | 0.9469 | 0.9467 |
| 0.0003 | 0.0647 | 0.0302 | 0.0302 | 0.2223 | 0.9502 | 0.9487 | 0.9485 |
| 0.0005 | 0.1095 | 0.0466 | 0.0467 | 0.2534 | 0.9502 | 0.9530 | 0.9525 |
| 0.0006 | 0.1194 | 0.0547 | 0.0548 | 0.2597 | 0.9502 | 0.9538 | 0.9533 |
| 0.0008 | 0.1443 | 0.0708 | 0.0709 | 0.2659 | 0.9502 | 0.9545 | 0.9540 |
| 0.0012 | 0.1692 | 0.1023 | 0.1025 | 0.2721 | 0.9552 | 0.9553 | 0.9547 |
| 0.0016 | 0.1990 | 0.1404 | 0.1406 | 0.2971 | 0.9602 | 0.9582 | 0.9575 |
| 0.0023 | 0.2289 | 0.1915 | 0.1917 | 0.3033 | 0.9701 | 0.9589 | 0.9582 |
| 0.0028 | 0.2637 | 0.2332 | 0.2335 | 0.3157 | 0.9751 | 0.9603 | 0.9594 |
| 0.0033 | 0.3134 | 0.2667 | 0.2670 | 0.3407 | 0.9751 | 0.9628 | 0.9618 |
| 0.0038 | 0.3483 | 0.3053 | 0.3056 | 0.3469 | 0.9751 | 0.9635 | 0.9624 |
| 0.0044 | 0.3532 | 0.3422 | 0.3426 | 0.3967 | 0.9751 | 0.9681 | 0.9667 |
| 0.0048 | 0.3682 | 0.3718 | 0.3721 | 0.4030 | 0.9801 | 0.9686 | 0.9672 |
| 0.0053 | 0.3881 | 0.4003 | 0.4007 | 0.4291 | 0.9851 | 0.9708 | 0.9692 |
| 0.0058 | 0.4478 | 0.4332 | 0.4336 | 0.4357 | 0.9851 | 0.9713 | 0.9697 |
| 0.0064 | 0.4876 | 0.4648 | 0.4651 | 0.4749 | 0.9851 | 0.9743 | 0.9725 |
| 0.0070 | 0.5224 | 0.4998 | 0.5002 | 0.5011 | 0.9851 | 0.9761 | 0.9742 |
| 0.0077 | 0.5473 | 0.5332 | 0.5335 | 0.5338 | 0.9851 | 0.9783 | 0.9762 |
| 0.0086 | 0.5821 | 0.5772 | 0.5775 | 0.5731 | 0.9851 | 0.9808 | 0.9785 |
| 0.0095 | 0.6119 | 0.6205 | 0.6208 | 0.6258 | 0.9900 | 0.9839 | 0.9813 |
| 0.0105 | 0.6368 | 0.6600 | 0.6602 | 0.6656 | 0.9900 | 0.9860 | 0.9833 |
| 0.0114 | 0.6468 | 0.6953 | 0.6955 | 0.6789 | 0.9900 | 0.9867 | 0.9839 |
| 0.0121 | 0.6766 | 0.7210 | 0.7211 | 0.7253 | 0.9900 | 0.9890 | 0.9860 |
| 0.0128 | 0.7015 | 0.7479 | 0.7479 | 0.7519 | 0.9900 | 0.9902 | 0.9871 |
| 0.0135 | 0.7363 | 0.7684 | 0.7684 | 0.7585 | 0.9900 | 0.9905 | 0.9874 |
| 0.0142 | 0.7761 | 0.7924 | 0.7924 | 0.7718 | 0.9900 | 0.9911 | 0.9879 |
| 0.0147 | 0.7761 | 0.8050 | 0.8049 | 0.7983 | 0.9900 | 0.9923 | 0.9890 |
| 0.0155 | 0.8159 | 0.8294 | 0.8292 | 0.8251 | 0.9900 | 0.9934 | 0.9900 |
| 0.0164 | 0.8259 | 0.8522 | 0.8520 | 0.8453 | 0.9900 | 0.9943 | 0.9908 |
| 0.0172 | 0.8408 | 0.8713 | 0.8710 | 0.8520 | 0.9900 | 0.9945 | 0.9910 |
| 0.0176 | 0.8458 | 0.8804 | 0.8801 | 0.9058 | 0.9900 | 0.9966 | 0.9929 |
| 0.0180 | 0.8756 | 0.8897 | 0.8893 | 0.9126 | 0.9900 | 0.9969 | 0.9932 |
| 0.0184 | 0.8955 | 0.8987 | 0.8983 | 0.9193 | 0.9900 | 0.9971 | 0.9934 |
| 0.0184 | 0.9005 | 0.8995 | 0.8991 | 0.9395 | 0.9950 | 0.9979 | 0.9941 |
| 0.0431 | 0.9055 | 0.9092 | 0.9092 | 0.9462 | 0.9950 | 0.9981 | 0.9943 |
| 0.0616 | 0.9104 | 0.9153 | 0.9155 | 0.9529 | 1.0000 | 0.9983 | 0.9945 |
| 0.0801 | 0.9204 | 0.9208 | 0.9210 | 0.9865 | 1.0000 | 0.9995 | 0.9956 |
| 0.0863 | 0.9254 | 0.9224 | 0.9227 | 1.0000 | 1.0000 | 1.0000 | 0.9960 |

include consideration of the field environment on the software failure-detection rate perform better in terms of the predictions for software failures in the field.

### 27.4.3 Software Reliability

Once the MLEs of all the parameters in (27.12) and (27.14) are obtained, the software reliability within

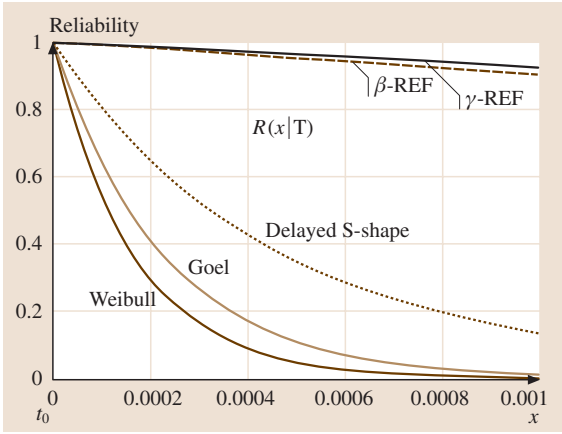


**Fig. 27.7** Reliability prediction comparisons

$(t, t+x)$ can be determined as

$$R(x|t) = \mathrm{e}^{-[m(t+x)-m(t)]} \,. \tag{27.18}$$

Let $T = 0.0184$, and change $x$ from 0 to 0.004, then we can compare the reliability predictions between the two RFE models and some other NHPP models that assume a constant failure-detection rate for both software testing and operation. The reliability prediction curves are shown in Fig. 27.7. From Fig. 27.7, we can see that the NHPP models without consideration of the environmental factor yield much lower predictions for software reliability in the field than the two proposed RFE software reliability models.

### 27.4.4 Confidence Interval

#### γ-RFE Model

To see how good the reliability predictions given by the two RFE models are, in this section we describe how to construct confidence intervals for the prediction of software reliability in the random field environments. From Tables 27.4 and 27.5, the MLEs of $c$ and $q$ are equal to zero and, if $p$ is set to 1, then the model in (27.12) becomes

$$m(t) = \begin{cases} a\left(1 - \mathrm{e}^{-b\cdot t}\right) & t \le T \,, \\ a\left[1 - \mathrm{e}^{-b\cdot T}\left(\frac{\theta}{\theta + b(t-T)}\right)^{\gamma}\right] & t \ge T \,. \end{cases} \tag{27.19}$$

This model leads to the same MLE results for the parameters $a$, $b$, $\gamma$ and $\theta$, and also yields exactly the same mean-value function fits and predictions as the model in (27.12). To obtain the confidence interval for the reliability predictions for the $\gamma$-RFE model, we derive the variance–covariance matrix for all the maximum likelihood estimates as follows.

If we use $x_i$, $i = 1, 2, 3$, and 4, to denote all the parameters in the model, or

$$x_1 \to a \quad x_2 \to b \quad x_3 \to \theta \quad x_4 \to \gamma \,.$$

The Fisher information matrix $H$ can be obtained as

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix} \,, \tag{27.20}$$

where

$$h_{ij} = E\left(-\frac{\partial^2 L}{\partial x_i \partial x_j}\right) i, j = 1, \cdots, 6 \,, \tag{27.21}$$

where $L$ is the log-likelihood function in (27.18).

If we denote $z(t_k) = m(t_k) - m(t_{k-1})$ and $\Delta y_k = y_k - y_{k-1}$, $k = 1, 2, \ldots, n$, then we have

$$\frac{\partial^2 L}{\partial x_i \partial x_j} = \sum_{k=1}^{n}\left[-\frac{\Delta y_k}{z(t_k)^2}\frac{\partial z(t_k)}{\partial x_i}\cdot\frac{\partial z(t_k)}{\partial x_j} + \left(\frac{\Delta y_k - z(t_k)}{z(t_k)}\cdot\frac{\partial^2 z(t_k)}{\partial x_i \partial x_j}\right)\right] \,. \tag{27.22}$$

Then we can obtain each element in the Fisher information matrix $H$. For example,

$$\begin{aligned} h_{11} &= E\left(-\frac{\partial^2 L}{\partial x_1^2}\right) \\ &= \sum_{k=1}^{n}\left\{\sum_{\Delta y_k=0}^{\infty}\left[\frac{\Delta y_k}{z(t_k)^2}\left(\frac{\partial z(t_k)}{\partial a}\right)^2\right] \times \frac{[z(t_k)]^{\Delta y_k}\,\mathrm{e}^{-z(t_k)}}{(\Delta y_k)!}\right\} \\ &= \sum_{k=1}^{n}\left\{\sum_{\Delta y_k=0}^{\infty}\left[\frac{\Delta y_k}{z(t_k)^2}\left(\frac{z(t_k)}{a}\right)^2\right] \times \frac{[z(t_k)]^{\Delta y_k}\,\mathrm{e}^{-z(t_k)}}{(\Delta y_k)!}\right\} \\ &= \sum_{k=1}^{n}\left(\frac{1}{a^2}\sum_{\Delta y_k=0}^{\infty}\Delta y_k \frac{[z(t_k)]^{\Delta y_k}\,\mathrm{e}^{-z(t_k)}}{(\Delta y_k)!}\right) \\ &= \sum_{k=1}^{n}\left[\frac{1}{a^2}\cdot z(t_k)\right] \\ &= \frac{1}{a^2}m(t_n) \,. \end{aligned} \tag{27.23}$$

The variance matrix, $V$, can also be obtained

$$V = (H)^{-1} = \begin{pmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ v_{41} & v_{42} & v_{43} & v_{44} \end{pmatrix} . \tag{27.24}$$

The variances of all the estimate parameters are given by

$$\begin{aligned} \mathrm{Var}(\hat{a}) &= \mathrm{Var}(x_1) = v_{11} , \\ \mathrm{Var}(\hat{b}) &= \mathrm{Var}(x_2) = v_{22} , \\ \mathrm{Var}(\hat{\gamma}) &= \mathrm{Var}(x_3) = v_{33} , \\ \mathrm{Var}(\hat{\theta}) &= \mathrm{Var}(x_4) = v_{44} . \end{aligned} \tag{27.25}$$

The actual numerical results for the $\gamma$-RFE model variance matrix are

$$V_\gamma = \begin{pmatrix} 703.8472 & -0.005387 & -88.6906 & -2.6861 \\ -0.005387 & 7.3655\times10^{-8} & 1.11\times10^{-3} & 3.097\times10^{-5} \\ -88.6906 & 1.11\times10^{-3} & 92.4287 & 1.1843 \\ -2.6861 & 3.097\times10^{-5} & 1.1843 & 0.0238 \end{pmatrix} . \tag{27.26}$$

### β-RFE Model

The model in (27.14) can also be simplified given that the estimates of both $q$ and $c$ are equal to zero and $p$ is set to 1. The mean-value function becomes

$$m_\beta(t) = \begin{cases} a(1-\mathrm{e}^{-bt}) & t \le T , \\ a\left[1-\mathrm{e}^{-bT} \right. & \\ \left. \times \sum_{k=0}^{\infty} \left( \frac{\Gamma(\alpha+\beta)\Gamma(\beta+k)\mathrm{Poisson}[k,b(t-T)]}{\Gamma(\beta)\Gamma(\alpha+\beta+k)} \right) \right] & \\ & t \ge T . \end{cases} \tag{27.27}$$

This model leads to the same MLE results for the parameters $a$, $b$, $\alpha$ and $\beta$, and also yields exactly the same mean-value function fits and predictions. To obtain the confidence interval for the reliability predictions for the $\beta$-RFE model, we need to obtain the variance–covariance matrix for all the maximum likelihood estimates.

If we use $x_i$, $i = 1, 2, 3$, and 4, to denote all the parameters in the model, or

$$x_1 \to a \quad x_2 \to b \quad x_3 \to \alpha \quad x_4 \to \beta ,$$

and go through similar steps as for the $\gamma$-RFE model, the actual numerical results for the $\beta$-RFE model variance matrix can be obtained as

$$V_\beta = \begin{pmatrix} 691.2 & -0.00536 & -2.728 & -66.2172 \\ -0.00536 & 7.4485\times10^{-8} & 2.671\times10^{-5} & 0.00085 \\ -2.7652 & 2.671\times10^{-5} & 0.01820 & 0.8295 \\ -66.2172 & 0.00085 & 0.8295 & 60.5985 \end{pmatrix} . \tag{27.28}$$

### Confidence Interval of the Reliability Predictions

If we define a partial derivative vector for the reliability $R(x|t)$ in (27.18) as

$$vR(x|t) = \left( \frac{\partial R(x|t)}{\partial x_1}, \frac{\partial R(x|t)}{\partial x_2}, \frac{\partial Rb(x|t)}{\partial x_3}, \frac{\partial R(x|t)}{\partial x_4} \right) \tag{27.29}$$

then the variance of $R(x|t)$ in (27.18) can be obtained as

$$\mathrm{Var}\left[R(x|t)\right] = vR(x|t)V\left[vR(x|t)\right]^T . \tag{27.30}$$

Assume that the reliability estimation follows a normal distribution, then the 95% confidence interval for the reliability prediction $R(x|t)$ is

$$\begin{aligned} &\left[ R(x|t) - 1.96\times\sqrt{\mathrm{Var}\left[R(x|t)\right]} \right. , \\ &\left. R(x|t) + 1.96\times\sqrt{\mathrm{Var}\left[R(x|t)\right]} \right] . \end{aligned} \tag{27.31}$$

Figures 27.8 and 27.9 show the 95% confidence interval of the reliability predicted by the $\gamma$-RFE and $\beta$-RFE models, respectively.

We plot the reliability predictions and their 95% confidence interval for both the $\gamma$-RFE model and the

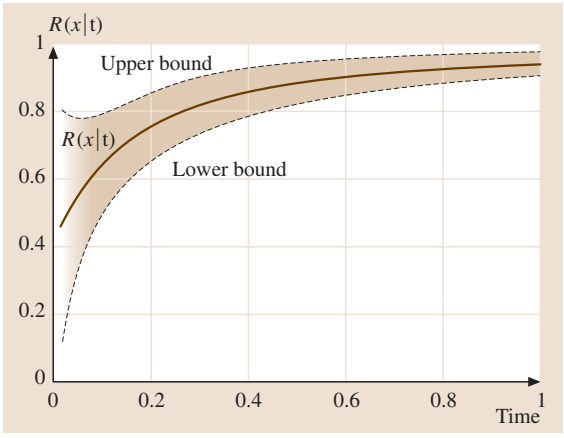


**Fig. 27.8** $\gamma$-RFE model reliability growth curve and its 95% confidence interval

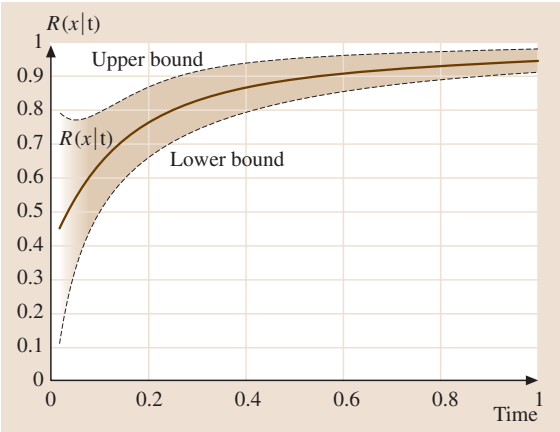


**Fig. 27.9** $\beta$-RFE model reliability growth prediction and its 95% confidence interval

$\beta$-RFE model in Fig. 27.10. For this given application data set, the reliability predictions for the $\gamma$-RFE model and the $\beta$-RFE model are very close to each other, as are their confidence intervals. Therefore, it would not matter too much which one of the two RFE models were used to evaluate the software reliability for this application. However, will these two RFE models always yield similar reliability predictions for all software applications? or, which model should one choose for applications if they are not always that close to each other? We will try to answer these two questions in the next section. Figure 27.11 shows the 95% confidence interval for the mean-value function fits and predictions from the $\gamma$-RFE model.

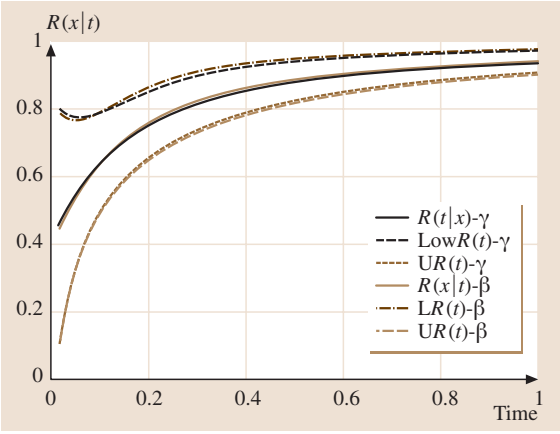


**Fig. 27.10** Reliability growth prediction curves and their 95% confidence intervals for the $\gamma$-RFE model and the $\beta$-RFE model

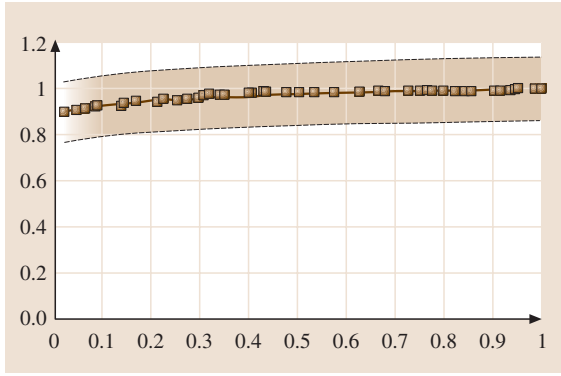

**Fig. 27.11** Mean-value function curve fit and its 95% confidence intervals for the $\gamma$-RFE model

## 27.4.5 Concluding and Remarks

Table 27.7 shows all the maximum likelihood estimates of all the parameters and other fitness measures. The maximum likelihood estimates (MLEs) on common parameters, such as $a$—the initial number of faults in the software, and $b$—the unit software failure-detection rate during testing, are consistent for both models. Both models provide very close predictions for software reliability and also give similar results for the mean and variance of the random environment factor $\eta$.

The underlying rationale for this phenomenon is the similarity between the gamma and beta distributions when the random variable $\eta$ is close to zero. In this application, the field environments are much less liable to software failure than the testing environment. The random field environmental factor, $\eta$, is mostly much less than 1 with mean $(\eta) \approx 0.02$.

Figure 27.12 shows the probability density function curves of the environmental factor $\eta$ based on the MLEs of all the parameters for both the $\gamma$-RFE model and

**Table 27.7** MLEs and fitness comparisons

| Parameter | $\gamma$-RFE | $\beta$-RFE |
|---|---|---|
| $\hat{a}$ | 236.5793016 | 236.0745369 |
| $\hat{b}$ | 0.001443362 | 0.001448854 |
| $\hat{\theta}$ | 10.7160153 | |
| $\hat{\gamma}$ | 0.213762945 | |
| $\hat{\alpha}$ | | 0.186224489 |
| $\hat{\beta}$ | | 8.692191792 |
| Mean | 0.019948 | 0.020975 |
| Variance | 0.0018615 | 0.002079 |
| MSE | 23.63 | 23.69 |
| Likelihood | **−136.1039497** | **−129.7811199** |

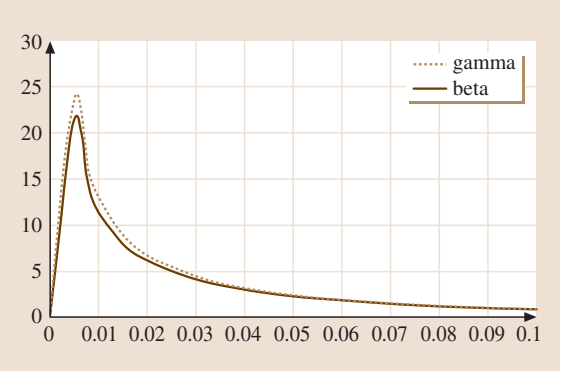


**Fig. 27.12** Probability density function curves comparison for the environmental factor $\eta$

the $\beta$-RFE model. We observe that the PDF curves for the beta and gamma distributions are also very close to each other. The two RFEs models give similar results because this software application is much less likely to fail in the field environment, with mean $(\eta) = 0.02$. If the mean $(\eta)$ is not so close to 0, then we would expect to have different prediction results from the $\gamma$-RFE model and the $\beta$-RFE model.

We suggest the following criteria as ways to select between the two models discussed in this chapter for predicting the software reliability in the random field environments:

1. Software less liable to failure in the field than in testing, i. e., $\eta \leq 1$

   In the $\gamma$-RFE model, the random field environmental factor, $\eta$ following a gamma distribution, ranges from 0 to $+\infty$. For the $\beta$-RFE model, the random field environmental factor, $\eta$ following a beta distribution, ranging from 0 to 1. Therefore, the $\beta$-RFE model will be more appropriate for describing field environments in which the software application is likely to fail than in the controlled testing environment.

   For this given application, we notice that, when the field environmental factor $\eta$ is much less than 1 [mean$(\eta) = 0.02$], the $\gamma$-RFE model yields similar results to the $\beta$-RFE model. However, we also observe that the $\gamma$-RFE model does not always yield similar results to the $\beta$-RFE model when $\eta$ is not close to 0. In this case, if we keep using the $\gamma$-RFE model instead of the $\beta$-RFE model, we would expect to see a large variance in the maximum likelihood estimates for all the unknown parameters, and hence a wider confidence interval for the reliability prediction.
2. Smaller variance of the RFE factor $\eta$

   A smaller variance of the random environmental factor $\eta$ will generally lead to a smaller confidence interval for the software reliability prediction. It therefore represents a better prediction in the random field environments.
3. Smaller variances for the common parameters $a$ and $b$

   The software parameter $a$ and the process parameter $b$ are directly related to the accuracy of reliability prediction. They can also be used to investigate the software development process. Smaller variances of $a$ and $b$ would lead, in general, to smaller confidence intervals for the mean-value function predictions and reliability predictions.
4. Smaller mean squared error (MSE) of the mean-value function fits

   A smaller MSE for the mean-value function fits means a better fit of the model to the real system failures. This smaller MSE will usually lead to a better prediction of software failures in random field environments.

The above criteria can be used with care to determine which RFE model should be chosen in practice. They may sometime provide contradictory results. In the case of contradictions, practitioners can often consider selecting the model with the smaller confidence interval for the reliability prediction.



## References


27.1 H. Pham, X. Zhang: A software cost model with warranty and risk costs, IEEE Trans. Comput. **48**, 71–75 (1999)

27.2 H. Pham, L. Normann, X. Zhang: A general imperfect debugging NHPP model with S-shaped fault detection rate, IEEE Trans. Reliab. **48**, 169–175 (1999)

27.3 A. L. Goel, K. Okumoto: Time-dependent error-detection rate model for software and other performance measures, IEEE Trans. Reliab. **28**, 206–211 (1979)

27.4 M. Ohba: Software reliability analysis models, IBM J. Res. Dev. **28**, 428–443 (1984)

27.5 H. Pham: *Software Reliability* (Springer, London 2000)

27.6 S. Yamada, M. Ohba, S. Osaki: S-shaped reliability growth modeling for software error detection, IEEE Trans. Reliab. **33**, 475–484 (1984)

27.7 X. Zhang, X. Teng, H. Pham: Considering fault removal efficiency in software reliability assessment, IEEE Trans. Syst. Man Cybern. A **33**, 114–120 (2003)
27.8 H. Pham, X. Zhang: NHPP software reliability and cost models with testing coverage, Eur. J. Oper. Res. **145**, 443–454 (2003)
27.9 X. Zhang, H. Pham: Predicting operational software availability and its Applications to telecommunication systems, Int. J. Syst. Sci. **33**(11), 923–930 (2002)
27.10 H. Pham, H. Wang: A quasi renewal process for software reliability and testing costs, IEEE Trans. Syst. Man Cybern. A **31**, 623–631 (2001)
27.11 X. Zhang, Mi-Young Shin: Exploratory analysis of environmental factors for enhancing the software reliability assessment, J. Syst. Softw. **57**, 73–78 (2001)
27.12 L. Pham, H. Pham: A Bayesian predictive software reliability model with pseudo-failures, IEEE Trans. Syst. Man Cybern. A **31**(3), 233–238 (2001)
27.13 X. Zhang, H. Pham: Comparisons of nonhomogeneous Poisson process software reliability models and its applications, Int. J. Syst. Sci. **31**(9), 1115–1123 (2000)
27.14 H. Pham: Software reliability and cost models: perspectives, comparison and practice, Eur. J. Oper. Res. **149**, 475–489 (2003)
27.15 B. Yang, M. Xie: A study of operational, testing reliability in software reliability analysis, Reliab. Eng. Syst. Safety **70**, 323–329 (2000)
27.16 X. Zhang, D. Jeske, H. Pham: Calibrating software reliability models when the test environment does not match the user environment, Appl. Stochastic Models Bus. Ind. **18**, 87–99 (2002)
27.17 D. R. Cox: Regression models and life tables (with discussion), J. R. Stat. Soc. Ser. B **34**, 133–144 (1972)
27.18 X. Teng, H. Pham: A software cost model for quantifying the gain with considerations of random field environment, IEEE Trans. Comput. **53**, 3 (2004)