Chapter

3

Software Reliability Modeling Survey

William Farr
Naval Surface Warfare Center

3.1 Introduction

With the ever-increasing role that software is playing in our systems, concern has steadily grown over the quality of the software component. Since the most important facet of quality is reliability, software reliability engineering (SRE) has generated quite a bit of interest and research in the software community. One particular aspect of SRE that has received the most attention is software reliability modeling. This chapter will present some of the more important models that have appeared in the recent literature, from both a historical and applications perspective. Not all of the models are considered because of space limitation (e.g., only time series models are considered). For additional models and further elaboration on the models considered in this chapter, you are referred to some other books on the subject of software reliability modeling [Musa87, Xie91a].

For each model that we do consider, we'll provide some motivation for it, present its assumptions and data required for implementation, show the model form and the resulting estimates, and conclude with some general comments about the model's implementation and provide an example in some cases of the estimation process. The derived estimates will be based upon the maximum likelihood procedure. In some cases, least-squares estimation will also be considered if the likelihood function is difficult to solve analytically. Other estimation procedures are also applicable (e.g., method-of-moments), but in this chapter we'll primarily concentrate on maximum likelihood estimates because of their many desirable properties, (e.g., asymptotic normality, asymptotic efficiency, and invariance).

Before considering the models we'll first provide a historical perspective of the development of this field and some needed theoretical results from reliability theory, which we'll use in each model development. We'll then go into the models. We'll first consider the exponential class of models, as that is the most important. Other distributional forms for the failure data, including Weibull and gamma, will then be considered. We'll also discuss some models based upon a Bayesian perspective and compare and contrast this approach with the more traditional one. The models that are presented were selected based upon a number of criteria. Important models in the historical development of this field were considered. In addition, the models that have been applied the most, based upon a literature review, were included. Another consideration was to select a model that was fairly typical of the class that it represents. Some important models may still have been left out in this review, but you are provided an extensive reference list for additional readings. Finally, we present some current research in generalizing the models and in extending them to the early phases of the life cycle.

This chapter will only consider models for reliability based on the time domain, i.e., models using either the elapsed time between software failures or the number of failures occurring over a specified time period.

3.2 Historical Perspective and Implementation

3.2.1 Historical background

Software reliability modeling has, surprisingly to many, been around since the early 1970s, with pioneering works by [Mora72, Mora75a, Shoo72, Shoo73, Shoo76, Shoo77a, Shoo77b, Cout73]. The basic approach is to model past failure data to predict future behavior. This approach employs either the observed number of failures discovered per time period or the observed time (actual wall clock or some measures of computer execution time) between failures of the software. The models therefore fall into two basic classes, depending upon the types of data the model uses:

1. Failures per time period
2. Time between failures

These classes are, however, not mutually disjoint. There are models that can handle either data type. Moreover, many of the models for one data type can still be applied even if the user has data of the other type, as explained by the two data transformation procedures in Chap. 1.

Either of these data transformation procedures requires that you test the applied model to determine the adequacy of the resulting fit. In the following sections the models will be introduced with the data class in which they first appeared in the literature.

These classes can themselves be considered part of the larger *time domain* approach to software reliability modeling, in contrast to the *error seeding and tagging* approach and the *data domain* approach. You are referred to [Farr83] and [Xie91a], among others, where these alternative approaches are described.

Since the development of these models was based upon concepts adapted from hardware reliability theory, you may want to review some reliability functions and concepts that show the relationships among these different functions. This is provided in App. B, Sec. B.2. We will make extensive use of these relationships in the development of the models throughout this chapter.

3.2.2 Model classification scheme

To aid in our development of the models in the ensuing sections, we'll need to discuss a model classification scheme that was proposed by Musa and Okumoto [Musa83]. It allows relationships to be established for models within the same classification groups and shows where model development has occurred. For this scheme Musa and Okumoto classified models in terms of five different attributes. They are:

1. *Time domain.* Wall clock versus execution time.
2. *Category.* The total number of failures that can be experienced in infinite time. This is either *finite* or *infinite,* the two subgroups.
3. *Type.* The distribution of the number of the failures experienced by time t. Two important types that we will consider are the Poisson and binomial.
4. *Class.* (Finite failure category only.) Functional form of the failure intensity expressed in terms of time.
5. *Family.* (Infinite failure category only.) Functional form of the failure intensity function expressed in terms of the expected number of failures experienced.

We will be especially concerned in the following sections with the category and type groupings. For the category group, suppose we let $M(t)$ be the random number of failures (faults) that are experienced by time t with mean value function $\mu(t)$, i.e., $\mu(t) = E[M(t)]$. If $\lim_{t \to \infty} \mu(t) < \infty$ (i.e., is finite), we have a finite failure model; otherwise we have a model of the infinite failure subgroup. In Secs. 3.3 and 3.4, we'll deal

with the former group (finite failure models), while in Sec. 3.5 we'll specifically look at the latter group. Note that in this chapter we do not distinguish *faults* and *failures* exclusively. We assume there is a one-to-one relationship between them.

For the type consideration we will now relate some important properties of the Poisson and binomial groups. We will make use of these relationships for specific classes of models in subsequent sections.

First for the Poisson type, we consider that we have a Poisson process over time. By this we mean that if we let $t_0 = 0, t_1, \ldots, t_{i-1}, t_i, \ldots, t_n = t$ be a partition of our time interval 0 to t and $\mu(t)$ is as defined as above, then we have a Poisson process if each f_i, $i = 1, \ldots, n$ (the number of faults detected in the ith interval, t_{i-1} to t_i), are independent Poisson random variables with means, $E[f_i] = \mu(t_i) - \mu(t_{i-1})$. Thus for each of the random variables f_i's, $i = 1, \ldots, n$, the probability density function is:

$$P(f_i = x) = e^{-(\mu(t_i) - \mu(t_{i-1}))}[(\mu(t_i) - \mu(t_{i-1}))]^x/x! \qquad \text{for } x = 0, 1, \ldots$$

Note: If $\mu(t)$ is a linear function of time, i.e., $\mu(t) = \alpha t$ for some constant $\alpha > 0$, we say the Poisson process, $M(t)$, is a homogeneous Poisson process (HPP). If, however, it is nonlinear we refer to the process as being a nonhomogeneous Poisson process (NHPP).

If we have a Poisson process model, we can show a relationship between the failure intensity function and the reliability function (hence the hazard rate and the probability density function using the relationships established in App. B, Sec. B.2). Suppose we denote $R(t + \Delta t \mid t)$ as the conditional reliability function that the software will still operate after $t + \Delta t$ given that it has not failed after time t. Let $\mu(t)$ be the mean value function for the cumulative number of failures and $\lambda(t)$ be the failure intensity function, then

$$\begin{aligned} R(t + \Delta t \mid t) &= P(f_t = 0 \mid t) \text{ where } f_t \text{ is a Poisson random variable over the interval } t \text{ to } t + \Delta t \\ &= \exp(-(\mu(t + \Delta t) - \mu(t))) \\ &= \exp(-\lambda(t')\Delta t), \; t' \text{ is a point in the interval } t \text{ to } t + \Delta t \text{ and using the definition of } \lambda(t) \\ &= \exp\left(-\int_t^{t+\Delta t} \lambda(x)\, dx\right) \text{ using the mean value theorem of integrals} \end{aligned}$$

The relationship between the failure intensity function and the hazard rate for a Poisson process can also be derived. It can be shown (see Prob. 3.1) that

$$z(\Delta t \mid t_{i-1}) = \lambda(t_{i-1} + \Delta t) \tag{3.1}$$

where t_{i-1} is the time of the $(i-1)$st failure and Δt is any point such that $t_{i-1} \leq t_{i-1} + \Delta t < t_i$. This shows that the conditional hazard rate and the failure intensity function are the same if the failure intensity function is evaluated at the current time $t_{i-1} + \Delta t$.

Another relationship that one can establish for the Poisson type of models is [Musa87]

$$\mu(t) = \alpha F_a(t) \tag{3.2}$$

where α is some constant and $F_a(t)$ is the cumulative distribution function of the time to failure of an individual fault a. From this, if we consider also distributions that belong to the finite failure category (i.e., $\lim_{t\to\infty} \mu(t) < \infty$), we have that $\lim_{t\to\infty} \mu(t) = \alpha$, since $\lim_{t\to\infty} F_a(t) = 1$. Thus α represents the eventual number of faults detected in the system if it could have been observed over an infinite amount of time. Using Eq. (3.2) and the relationship between the mean value function and the failure intensity function, we have also for the Poisson type of models

$$\lambda(t) = \mu'(t) = \alpha f_a(t) \tag{3.3}$$

where $f_a(t)$ is the probability density function of the time to failure of the individual fault a.

For the binomial type of models, we have the following assumptions:

1. There is a fixed number of faults (N) in the software at the beginning of the time in which the software is observed.
2. When a fault is detected it is removed immediately.
3. Using the notation of [Musa87], if T_a is the random variable denoting the time to failure of fault a, then the T_a's, $a = 1, \ldots, n$ are independently and identically distributed random variables as $F_a(t)$ for all remaining faults.

The cumulative distribution function, $F_a(t)$, density function, $f_a(t)$, and hazard rate function, $z_a(t)$, are the same for all faults for this class. Moreover, for this class no new faults are introduced into the software in the fault detection/correction process. [Musa87] shows for this class that the failure intensity function is obtained from the probability density function for a single fault as:

$$\lambda(t) = N f_a(t) \tag{3.4}$$

The mean value function is in turn related to the cumulative distribution function, $F_a(t)$, as

$$\mu(t) = N F_a(t) \tag{3.5}$$

Notice the similarity between Eqs. (3.4) and (3.3) as well as between (3.5) and (3.2). For the binomial we have a fixed number of faults at start, N, while for the Poisson type, α is the eventual number of faults that could be discovered over an infinite amount of time.

3.2.3 Model limitations and implementation issues

In fitting any model to a given data set, you are cautioned about some limitations for this type of analysis. First, you must be aware of a given model's assumptions. For example, if a selected model makes the assumption that the time intervals over which the software is observed or tested are all of the same magnitude (e.g., Schneidewind's model), don't attempt to use this model if this is not the case for your data. There are other assumptions that may not hold, but the model may be fairly robust with respect to violations. One such assumption is the distributional one about the number of failures per unit time or the time between failures. One can still do a credible job in fitting the data for a selected model even if its distributional assumption is violated. The only way to tell is to ask just how well the model is doing in tracking and predicting the data. The procedures discussed in Chap. 4 will help considerably in answering this question.

A second model limitation and implementation issue concerns future predictions. If the environment in which the software is being tested or observed changes considerably from the one in which the data have been collected, you can't expect to do well in predicting future behavior. If the software is being operated in a different manner (i.e., new capabilities are being exercised that were not used before, or a different testing methodology is employed), the failure history of the past will not reflect these changes, and poor predictions may result. Too many times model users tend to extrapolate either too far into the future or make reliability predictions for an environment in which little if any data have been gathered. Developing operational profiles is very important if one wants to predict future reliability in the user's environment (see Chap. 5).

For both violations of assumptions and considerations for predictions, one option that may be available to the practitioner is to use the most recent data if sufficient current data are available. Recent data may be more representative of the environment in which the software is employed than data collected in the distant past. This same reasoning applies to violations of assumptions. Current data may be more stable and reflective of the assumptions than past data. This is the basic idea behind Schneidewind's model 2 (see Sec. 3.3.3). It appears to us that there is nothing to preclude this approach on other models. Model validation should be the final word.

A final comment on implementation: this modeling approach is primarily applicable from integrated testing onward. The software must have matured to the point that *extensive changes are not being routinely made.* The models can't have a credible performance if the software is changing so fast that gathering data on one day is not the same as gathering data on another day. Different approaches and models need to be considered if that is the case. At the conclusion of this chapter some approaches for the earlier phases are presented.

3.3 Exponential Failure Time Class of Models

In the literature on software reliability, this class has the most articles written on it. Using Musa and Okumoto's classification scheme, this group consists of all finite failure models with the functional form of the failure intensity function being exponential. The binomial types in this class are all characterized by a per-fault constant hazard rate (i.e., $z(t) = \phi$); the hazard rate function before the ith fault that has been detected is a function of the remaining number of faults (i.e., $N - (i - 1)$); and the failure intensity function is exponential in form (i.e., $\lambda(t) = N\phi \exp(-\phi t)$). The Poisson types in this class are all characterized by a per-fault constant hazard rate (i.e., $z(t) = \phi$) and an exponential time to failure of an individual fault (i.e., $f_X(x) = \phi \exp(-\phi x)$). Since we have either a homogeneous or nonhomogeneous Poisson process, the number of faults that occur over any fixed period of time is a Poisson random variable. For the time-between-failures models, the distribution is exponential.

3.3.1 Jelinski-Moranda de-eutrophication model

3.3.1.1 Overview of the model. One of the earliest models proposed, which is still being applied today, is the de-eutrophication model developed by Jelinski and Moranda [Mora72], while working on some Navy projects for McDonnell Douglas. The elapsed time between failures is taken to follow an exponential distribution with a parameter that is proportional to the number of remaining faults in the software, i.e., the mean time between failures at time t is $1/\phi(N - (i - 1))$. Here t is any point in time between the occurrence of the $(i - 1)$st and the ith fault occurrence. The quantity ϕ is the proportionality constant and N is the total number of faults in the software from the initial point in time at which the software is observed. Figure 3.1 illustrates the impact that finding a fault has on the hazard rate. One can see as each fault is discovered that the hazard rate is reduced by the proportionality constant

ϕ. This indicates that the impact of each fault removal is the same. In Musa and Okumoto's classification scheme, this is a binomial type model.

3.3.1.2 Assumptions and data requirements. The *basic assumptions* are:

1. The rate of fault detection is proportional to the current fault content of the software.
2. The fault detection rate remains constant over the intervals between fault occurrence.
3. A fault is corrected instantaneously without introducing new faults into the software.
4. The software is operated in a similar manner as that in which reliability predictions are to be made.
5. Every fault has the same chance of being encountered within a severity class as any other fault in that class.
6. The failures, when the faults are detected, are independent.

Note: The numbered assumptions 4 through 6 are fairly standard as we consider other models in this chapter. Assumption 4 is to ensure that the model estimates that are derived using data collected in one particular environment are applicable to the environment in which the reliability projections are to be made. The fifth assumption is to ensure that the various failures all have the same distributional properties. One severity class might have a different failure rate than the others, requiring a separate reliability analysis be done. The last assumption allows simplicity in deriving the maximum likelihood estimates. Since assumptions 4 through 6 will appear often in the models that follow, we'll refer to them as the *Standard Assumptions* for reliability modeling rather than repeat them in each model development.

The *data requirements* to implement this model are: the elapsed time between failures $x_1, x_2, \ldots, x_n$ or the actual times that the software failed $t_1, t_2, \ldots, t_n$, where $x_i = t_i - t_{i-1}, i = 1, \ldots, n$ with $t_0 = 0$.

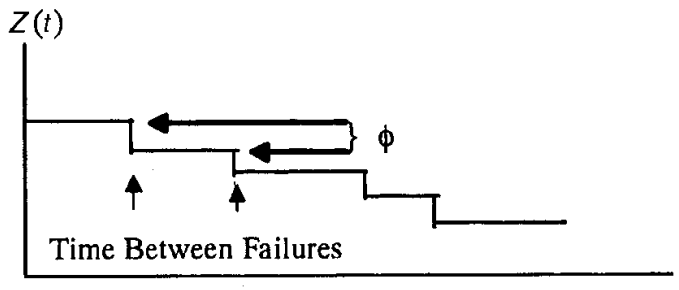

Figure 3.1 De-eutrophication process.

3.3.1.3 Model form. From the overview of the model and the assumptions from the previous section, we can determine that if the time-between-failure occurrences are $X_i = T_i - T_{i-1}$, $i = 1, \ldots, n$, then the X_i's are independent exponentially distributed random variables with mean = $1/\phi(N - (i-1)) = 1/z(X_i \mid T_{i-1})$. That is

$$f(X_i \mid T_{i-1}) = z(X_i \mid T_{i-1})\exp(-z(X_i \mid T_{i-1})X_i)$$

$$= \phi[N - (i-1)]\exp(-\phi[N - (i-1)]X_i)$$

Since this exponential model belongs to the binomial type, using Eqs. (3.2) and (3.3), we have specifically:

$$\mu(t) = N(1 - \exp(-\phi t)) \qquad \text{and} \qquad \lambda(t) = N\phi\exp(-\phi t)$$

for the mean value function and the failure intensity function. It is clearly a finite failures type model as $\lim_{t\to\infty} \mu(t) = \lim_{t\to\infty} (N(1 - \exp(-\phi t))) = N$.

3.3.1.4 Model estimation and reliability prediction. The maximum likelihood estimates, MLEs, calculated from the joint density of the X_i's, are the solutions to the following equations:

$$\hat{\phi} = \frac{n}{\hat{N}\left(\sum_{i=1}^{n} X_i\right) - \sum_{i=1}^{n}(i-1)X_i} \qquad \text{and}$$

$$\sum_{i=1}^{n} \frac{1}{\hat{N} - (i-1)} = \frac{n}{\hat{N} - \left(1/\sum_{i=1}^{n} X_I\right)\left(\sum_{i=1}^{n}(i-1)X_i\right)}$$

The second equation is solved by numerical techniques for the MLE of N, and then the solution is put into the first equation to find the MLE of ϕ. Using the MLEs, various reliability measures can then be derived by replacing the quantities N and ϕ in the reliability function of interest by the corresponding MLEs $\hat{N}$ and $\hat{\phi}$. An example is the estimated MTTF after n faults have been detected. The expression for this is MTTF for the $(n+1)$st fault = $1/z(x_{n+1} \mid t_n)$, so the MLE is $\hat{MTTF} = 1/\hat{z}(x_{n+1} \mid t_n) = 1/\hat{\phi}(\hat{N} - n)$.

Example 3.1 To illustrate the above results, suppose we have observed the following elapsed-time-between-failure occurrences as

$x_1 = 7, x_2 = 11, x_3 = 8, x_4 = 10, x_5 = 15, x_6 = 22, x_7 = 20, x_8 = 25, x_9 = 28$, and $x_{10} = 35$

Using the SMERFS reliability program (see [Farr93a, Farr93b] and App. A), the MLE estimates for N and ϕ are respectively $\hat{N} = 11.6$ and $\hat{\phi} = 0.0096$. Thus the estimated MTTF to the next failure is

$$MTTF = 1/\hat{\phi}(\hat{N} - n) = 1/0.0096(11.6 - 10) = 65.1.$$

3.3.1.5 Comments. Much has been written in the literature on this model and many variations of it have been proposed. Farr [Farr83] has in his survey a discussion of many of these variations as well as alternative ways of deriving estimates of the reliability measures, e.g., least-squares estimation. This model, however, has largely been replaced by some of the more recent models that will be discussed. Its importance is largely in setting the framework for future work in this modeling area.

3.3.2 Nonhomogeneous Poisson process (NHPP) model

3.3.2.1 Overview of the model. The nonhomogeneous Poisson process (NHPP) model is a Poisson type model that takes the number of faults per unit of time as independent Poisson random variables. The model was first proposed in 1979 by Amrit Goel and Kazu Okumoto [Goel79] and has formed the basis for the models using the observed number of faults per unit time group. A number of others are spin-offs from it, e.g., the S-shaped model of Yamada, [Yama83] which we'll consider later (see Sec. 3.4.2).

3.3.2.2 Assumptions and data requirements. Including the Standard Assumptions (see Sec. 3.3.1.2), the *basic assumptions* are:

1. The cumulative number of failures by time t, $M(t)$, follows a Poisson process with mean value function $\mu(t)$. The mean value function is such that the expected number of fault occurrences for any time t to $t + \Delta t$ is proportional to the expected number of undetected faults at time t. It is also assumed to be a bounded, nondecreasing function of time with $\lim_{t \to \infty} \mu(t) = N < \infty$, that is, it is a finite failure model.
2. The number of faults $(f_1, f_2, \ldots, f_n)$ detected in each of the respective intervals $[(t_0 = 0, t_1), (t_1, t_2), \ldots, (t_{i-1}, t_i), \ldots, (t_{n-1}, t_n)]$ is independent for any finite collection of times, $t_1 < t_2 < \cdots < t_n$.

The *data requirements* to implement this fault count model are:

1. The fault counts in each of the testing intervals, i.e., the f_i's.
2. The completion time of each period that the software is under observation, i.e., the t_i's.

3.3.2.3 Model form. From the assumptions it can be shown [Goel79] that the mean value function must be of the form

$$\mu(t) = N(1 - e^{-bt})$$

for some constants $b > 0$ and $N > 0$. N is the expected total number of faults to be eventually detected. (*Note:* N is not required to be an integer since it is the *expected number* of faults that will eventually be detected.) Since the failure intensity function is the derivative of $\mu(t)$ we have, therefore

$$\lambda(t) = Nbe^{-bt}$$

Notice that the failure intensity function is strictly decreasing for $t > 0$. Because it belongs to the exponential class, we have the distribution of a single individual fault, X:

$$f_X(x) = be^{-bx}$$

We thus have for the failure intensity function

$$\lambda(t) = Nbe^{-bt} = Nf_X(t)$$

which shows the relationship between the failure intensity function and probability density function for a single fault.

From the assumptions, we also have that each f_i, the fault count in the ith interval, is an independent Poisson random variable with mean $= \mu(t_i) - \mu(t_{i-1})$. Therefore the joint density of the f_i's, $i = 1, \ldots, n$, is

$$\prod_{i=1}^{n} \frac{[\mu(t_i) - \mu(t_{i-1})]^{f_i} \exp\{\mu(t_i) - \mu(t_{i-1})\}}{f_i!}$$

3.3.2.4 Model estimation and reliability prediction. Using the joint density given above, the maximum likelihood estimates (MLEs) of N and b can be obtained as the solutions for the following pair of equations:

$$\hat{N} = \frac{\sum_{i=1}^{n} f_i}{(1 - e^{-\hat{b}t_n})} \quad \text{and} \quad \frac{t_n e^{-\hat{b}t_n} \sum_{i=1}^{n} f_i}{(1 - e^{-\hat{b}t_n})} = \sum_{i=1}^{n} \frac{f_i(t_i e^{-\hat{b}t_i} - t_{i-1} e^{-\hat{b}t_{i-1}})}{e^{-\hat{b}t_{i-1}} - e^{-\hat{b}t_i}}$$

The second equation is solved for $\hat{b}$ by numerical methods, and the solution is then substituted into the first equation to find $\hat{N}$. MLEs are then obtained for other reliability measures by substituting the MLEs for N

and b in the expressions of the measures of interest. For example, the MLEs of the mean value function and the failure intensity function are

$$\hat{\mu}(t) = \hat{N}(1 - e^{-\hat{b}t}) \qquad \text{and} \qquad \hat{\lambda}(t) = \hat{N}\hat{b}e^{-\hat{b}t}$$

The MLE of the expected number of faults to be detected in the $(n + 1)$st observation period is similarly determined as

$$\text{Estimated expected number of faults in } (n+1)\text{st} = \hat{N}(e^{-\hat{b}t_n} - e^{-\hat{b}t_{n+1}})$$

3.3.2.5 Comments. Goel and Okumoto [Goel79] have also adapted this model to use the time of fault occurrences instead of the fault counts. Within this framework, [Okum80] have also determined an optimal release time for a software system. If the desired reliability is R for a specified operational time of O, then to achieve the desired result, the required amount of time that the software must be observed is

$$\text{Required time} = \frac{1}{b}\left[\ln(a(1 - e^{-bO})) - \left(\ln\left(\ln\left(\frac{1}{R}\right)\right)\right)\right]$$

In the paper [Okum80], they also determine the optimal release time based upon cost (cost of testing and of finding and fixing a fault in the testing environment versus the operational). You are referred to that article for the details or to [Farr83].

This model is equivalent to the model considered in Sec. 3.3.3 (type 1 model) if each of the periods that the software is observed are all of the same length, that is, $t_i = iL$, $i = 1, \ldots, n$ for some constant $L > 0$ and $N = \alpha/\beta$ where α and β are the parameters of Schneidewind's model.

3.3.3 Schneidewind's model

3.3.3.1 Overview of the model. The idea behind Schneidewind's model [Schn75] is that the current fault rate might be a better predicator of the future behavior than the observed rates in the distant past. The failure rate process may be changing over time so the current data may better model the present reliability. To reflect this idea, Schneidewind has three forms of the model that reflect the analyst's view of the importance of the data as functions of time. The data used are the number of faults per unit of time where all the time periods are of the same length. Suppose there are n units of time, all of some fixed length; then the three forms of the model are:

Model 1 Utilize all of the fault counts from the n periods. This reflects the view that all of the data points are of equal importance.

Model 2 Ignore the fault counts completely from the first through the $s - 1$ time periods, i.e., only use the data from periods s through n. This reflects the view that the early time periods contribute little if anything in predicting future behavior. For example, one can eliminate a learning curve effect by ignoring the first few time periods.

Model 3 Use the cumulative fault counts from the intervals 1 to $s - 1$ as the first data point and the individual fault counts for periods s through n as the additional data points. This is an approach, intermediate between the other two, that reflects the belief that a combination of the first $s - 1$ period is indicative of the failure rate process during the later stages.

Schneidewind [Schn93a, Schn93b, Schn93c, Schn93d] has recently developed criteria for the optimal selection of the s value. (We note that if $s = 1$, then models 2 and 3 become model 1.) This will be discussed further in Sec. 3.3.3.4.

3.3.3.2 Assumptions and data requirements. Including the Standard Assumptions (see Sec. 3.3.1.2), the *basic assumptions* are:

1. The cumulative number of failures by time t, $M(t)$, follows a Poisson process with mean value function $\mu(t)$. The mean value function is such that the expected number of fault occurrences for any time period is proportional to the expected number of undetected faults at that time. It is also assumed to be a bounded, nondecreasing function of time with $\lim_{t \to \infty} \mu(t) = \alpha/\beta < \infty$; for some constants $\alpha, \beta > 0$ (i.e., it is a finite failure model).
2. The failure intensity function is assumed to be an exponentially decreasing function of time. The failure intensity function $\lambda(t)$ is taken to be of the form $\lambda(t) = \alpha \exp(-\beta t)$. Therefore, large β implies a small failure rate, small β implies a large one. Moreover, we see α is the initial failure rate at time $t = 0$.
3. The number of faults (f_i) detected in each of the respective intervals are independent.
4. The fault correction rate is proportional to the number of faults to be corrected.
5. The intervals over which the software is observed are all taken to be of the same length, that is, $t_i = il$, for $i = 1, \ldots, n$ and l being some positive constant. (Note, without loss of generality, we can take $l = 1$ so that $t_i = i$.)

The *data requirements* to implement this fault count model are: the fault counts in each of the testing intervals, i.e., the f_i for $i = 1, \ldots, n$.

3.3.3.3 Model form. From the assumptions, the cumulative mean number of faults by the ith time period is

$$D_i = \mu(t_i) = \frac{\alpha}{\beta}[1 - \exp(-\beta i)].$$

Thus the expected number of faults in the ith period is

$$m_i = D_i - D_{i-1} = \mu(t_i) - \mu(t_{i-1}) = \frac{\alpha}{\beta}[\exp(-\beta(i-1)) - \exp(-\beta i)]$$

Using the assumptions again pertaining to the f_i's being independent nonhomogeneous Poisson random variables and incorporating the concept of the different model types, we have the joint density

$$\frac{M_{s-1}^{F_{s-1}} \exp(-M_{s-1})}{F_{s-1}} \prod_{i=s}^{n} \frac{m_i^{f_i} \exp(-m_i)}{f_i!}$$

where s is some integer value chosen in the range 1 to n, M_{s-1} is the cumulative mean number of faults in the intervals up to $s-1$, and F_{s-1} is the cumulative number of faults detected up through interval $s-1$.

3.3.3.4 Model estimation and reliability prediction. [Schn75] derived the MLEs for α and β. They can also be found in [Farr83], [Geph78], and the AIAA *Recommended Practice for Software Reliability* [AIAA93]. Three different sets of equations are derived for each of the three models.

Model 1 estimates

$$\hat{\alpha} = \frac{\hat{\beta} F_n}{1 - \exp(-\hat{\beta} n)} \quad \text{and} \quad \frac{1}{\exp(\hat{\beta}) - 1} - \frac{n}{\exp(\hat{\beta} n) - 1} = \sum_{k=0}^{n-1} k \frac{f_{k+1}}{F_n}$$

where $F_n = \sum_{i=1}^{n} f_i$ and the f_i's are the fault counts in intervals 1 to n.

Model 2 estimates

$$\hat{\alpha} = \frac{\hat{\beta} F_{s,n}}{1 - \exp(-\hat{\beta}(n - s + 1))} \quad \text{and}$$

$$\frac{1}{\exp(\hat{\beta}) - 1} - \frac{n - s + 1}{\exp(\hat{\beta}(n - s + 1)) - 1} = \sum_{k=0}^{n-s} k \frac{f_{s+k}}{F_{s,n}}$$

where $F_{s,n} = \sum_{k=s}^{n} f_k$. Notice if we let $s = 1$, model 2 estimates become equivalent to model 1.

Model 3 estimates

$$\hat{\alpha} = \frac{\beta F_n}{1 - \exp(-\hat{\beta} n)} \quad \text{and}$$

$$\frac{(s-1)F_{s-1}}{\exp(\hat{\beta}(s-1)) - 1} + \frac{F_{s,n}}{\exp(\hat{\beta}) - 1} - \frac{nF_n}{\exp(\hat{\beta} n) - 1} = \sum_{k=0}^{n-s} (s + k - 1) f_{s+k}$$

where $F_{s-1} = \sum_{k=1}^{s-1} f_k$. We note again that if $s = 1$ is substituted into the above equations we obtain the equivalent estimates for model 1.

Recent work by [Schn93a, Schn93b, Schn93c] has been identifying the optimal s in model types 2 and 3. Three criteria have been developed, (1) the weighted least-squares criterion, (2) the mean square criterion for time to next failure(s), and (3) the mean square error criterion for cumulative failures. Each criterion is handled the same for the selection of s. The analyst seeks the value of s that minimizes the respective criterion. The procedure is as follows. For a given value of s and a selected model type (2 or 3) the corresponding MLEs of α and β are derived, and then the criterion is evaluated. This is done over a range of s values. The optimal s is the one that globally minimizes the chosen criterion. Schneidewind has proposed that if the global minimum of the criteria cannot be determined because of the computational complexity involved, the analyst can use the first value of s, starting from $s = 1$, in which the selection criteria have achieved a local minimum. This is illustrated in the example that follows. The formulas for these criteria are, respectively:

Weighted least-squares criterion

$$\text{WLS} = \frac{\sum_{k=s}^{n} \exp(\beta(k - s + 1))[\alpha/\beta(\exp(-\beta(k - s + 1)))(\exp(\beta) - 1) - f_i]^2}{(n - s + 1)}$$

Mean square criterion for time to next failure(s)

$$\text{MSE}_T(s) = \frac{\sum_{k=s}^{J-1} [[\log[\alpha/(\alpha - \beta(F_{s,k} + f_{j|k}))]/\beta - (k - s + 1)] - (j - k)]^2}{(J - s)}$$

$$\text{for } \frac{\alpha}{\beta} > (F_{s,k} + f_{j|k})$$

where $f_{j|k}$ = number of faults detected during time interval j since k; k is the index variable and j is the next interval index beyond k (that is, $j > k$) for which $f_j > 0$ and J is the maximum $j \leq n$ such that $f_{j|k} > 0$.

Mean square error criterion for cumulative failures

$$\mathrm{MSE_F}(s) = \frac{\sum_{k=s}^{n} [\alpha/(\beta(1 - \exp(-\beta(k - s + 1)))) - F_{s,k}]^2}{n - s + 1}$$

The $\mathrm{MSE_T}$ and the $\mathrm{MSE_F}$ are the preferred criteria. $\mathrm{MSE_T}$ looks at the squared error difference between the predicted number of periods required to generate a specified number of fault detections and the actual number it took. $\mathrm{MSE_F}$ compares the squared error difference between the predicted model cumulative fault counts and the actual values observed. The later is preferred if a failure count prediction is to be made, the former if a time to next failure prediction is of interest.

Example 3.2 To illustrate this model, suppose we have the following number of faults detected per unit of time (day, week, etc.): $f_1 = 20$, $f_2 = 18$, $f_3 = 25$, $f_4 = 30$, $f_5 = 35$, $f_6 = 36$, $f_7 = 31$ and $f_8 = 32$, $f_9 = 29$, $f_{10} = 26$, $f_{11} = 24$, $f_{12} = 21$, $f_{13} = 18$, $f_{14} = 20$. From the first few data points it appears that a learning curve effect may be present, so a model 2 might be appropriate with candidate s values of 4, 5, or 6. Using SMERFS, Table 3.1 was generated for model 2 to determine the optimal s.

The WLS and the $\mathrm{MSE_F}$ indicate an $s = 6$ may be appropriate. The $\mathrm{MSE_T}$ indicates the optimal s has not yet been reached. Since two of the three criteria indicate an s of 6 is appropriate, this was the model tried. Using the data points from 6 on, the MLEs are obtained as:

$$\hat{\beta} = 0.08305 \quad \text{and} \quad \hat{\alpha} = \frac{\hat{\beta} F_{6,14}}{1 - \exp(-\hat{\beta}(14 - 6 + 1))} = \frac{0.08305 \times 237}{1 - \exp(-0.08305 \times 9)} = 37.37$$

From this, various estimates of reliability measures can be calculated, such as

$$\text{Estimated total number of faults} = \frac{\hat{\alpha}}{\hat{\beta}} = 450$$

3.3.3.5 Comments. As was previously stated in the NHPP model, Schneidewind's model 1 is a special case of the NHPP model if all the

TABLE 3.1 SMERFS Output for the Optimal "s"

S	BETA	ALPHA	WLS	MSE-F	MSE-T
1	0.12061E–01	0.28334E+02	0.36602E+02	0.15082E+03	0.22780E+00
2	0.23640E–01	0.30825E+02	0.33908E+02	0.10539E+03	0.14059E+00
3	0.43698E–01	0.35016E+02	0.19540E+02	0.42996E+02	0.51981E–01
4	0.61054E–01	0.37698E+02	0.96338E+01	0.13589E+02	0.14931E–01
5	0.76042E–01	0.38840E+02	0.35426E+01	0.24855E+01	0.34200E–02
6	0.83051E–01	0.37390E+02	0.27012E+01	0.10740E+01	0.20634E–02
7	0.82575E–01	0.34331E+02	0.27868E+01	0.12602E+01	0.18962E–02

observation periods are the same length and we let $\alpha/\beta = N$ and $\beta = b$ be the correspondence between the parameters of the two models.

This model has been used extensively on IBM's Flight Control software for the Space Shuttle [Schn92b] with very good success, especially employing the procedure for determining the optimal s to obtain better fits to the data. This model is also one of the four selected models to start an initial attempt at model fitting as proposed in the AIAA's Recommended Practice for Software Reliability [AIAA93].

3.3.4 Musa's basic execution time model

3.3.4.1 Overview of the model. This model has had the widest distribution among the software reliability models and was developed by John Musa of AT&T Bell Laboratories [Musa75, Musa78, Musa79a, Musa79b, Musa80, Musa87]. Musa has been a leading contributor in this field and has been a major proponent of using models to aid in determining the reliability of software. As such, it is natural that his models (the basic execution and the logarithmic Poisson, see Sec. 3.5.3) have been applied in many diverse fields.

This model was one of the first to use the actual execution time of the software component on a computer for the modeling process. The times between failures are expressed in terms of computational processing units (CPU) rather than elapsed wall-clock time. Musa feels that execution time is more reflective of the actual stress induced on the software system than the amount of calendar time that has elapsed. The model does, however, have a feature to convert the execution time results to calendar time. This is accomplished by a second component of the model that functionally relates human and computer resources utilization with the execution time.

3.3.4.2 Assumptions and data requirements. Including the Standard Assumptions (see Sec. 3.3.1.2), the *basic assumptions* are:

1. The cumulative number of failures by time t, $M(t)$, follows a Poisson process with mean value function $\mu(t) = \beta_0[1 - \exp(-\beta_1 t)]$, where β_0, $\beta_1 > 0$. The mean value function is such that the expected number of failure occurrences for any time period is proportional to the expected number of undetected faults at that time. Since $\lim_{t \to \infty} \mu(t) = \lim_{t \to \infty}(\beta_0[1 - \exp(-\beta_1 t)]) = \beta_0$, it is a finite failure model. The parameter β_0 is the total number of faults that would be detected in the limit.
2. The execution times between the failures are piecewise exponentially distributed, i.e., the hazard rate for a single fault is constant. This is why this model belongs to the exponential class.

3. The quantities of the resources (number of fault-identification, -correction personnel and computer times) that are available are constant over a segment for which the software is observed.
4. Resource expenditures for the kth resource, $\Delta\chi_k$, associated with a change in MTTF from T_1 to T_2 can be approximated by $\Delta\chi_k \approx \theta_k \Delta t + \mu_\kappa \Delta m$, where Δt is the increment of execution time, Δm is the increment of failures experienced, θ_k is an execution time coefficient of resource expenditure, and μ_κ is a failure coefficient of resource expenditure.
5. Fault-identification personnel can be fully utilized and computer utilization is constant.
6. Fault-correction personnel utilization is established by the limitation of fault queue length for any fault-correction person. Fault queue is determined by assuming that fault correction is a Poisson process and that servers are randomly assigned in time.

Assumptions 3 through 6 are needed only if the second component of the basic execution model linking execution time and calendar time is desired.

The *data requirements* to implement this fault count model are:

- *For the basic execution time component.* Either the actual times that the software failed, $t_1, t_2, \ldots, t_n$ or the elapsed time between failures $x_1, x_2, \ldots, x_n$, where $x_i = t_i - t_{i-1}$.
- *For the basic calendar time component*
  1. The available resources for both identification and correction personnel and the number of computer shifts. We'll denote them as P_I, P_F, and P_C, respectively.
  2. The utilization factor for each resource, that is, ρ_I (=1), ρ_F, and ρ_C.
  3. The execution time coefficient of resource expenditure for each resource, that is, θ_I, θ_F (=0 usually), and θ_C.
  4. The failure coefficient of resource expenditure for each resource, that is, μ_1, μ_F, and μ_C.
  5. The maximum fault queue length Q for a fault correction personnel.
  6. The probability P that the fault queue length is no larger than Q.

3.3.4.3 Model form. Since $\mu(t) = \beta_0(1 - \exp(-\beta_1 t))$, the failure intensity function for this model is

$$\lambda(t) = \mu'(t) = \beta_0\beta_1\exp(-\beta_1 t)$$

We notice that for large β_1 the failure intensity function will decrease rapidly, while for a small one it will decrease slowly. In either case, the function decreases exponentially to 0. This is illustrated in Fig. 3.2.

By making the correspondence that $\beta_1 = B\phi$ and $\beta_0 = \nu_0$, where B is defined as the fault reduction factor (the proportionality constant relating the fault correction rate to the hazard rate) and ϕ is the constant hazard rate per individual fault, the preceding formulation can be put into the framework in which Musa originally introduced this model. (See [Musa87], p. 285, for this correspondence.)

Using the result of Sec. 3.2.2 and the above expressions for the mean value and failure intensity function, one can show (see Prob. 3.2) that the reliability function after $(i - 1)$ failures have occurred is $R(\Delta t \mid t_{i-1}) = \exp(-[\beta_0 \exp(-\beta_1 t_{i-1})][1 - \exp(-\beta_1 \Delta t)])$ for $0 \leq \Delta t$, and the conditional hazard rate is $z(\Delta t \mid t_{i-1}) = \beta_0 \beta_1 \exp(-\beta_1 t_{i-1}) \exp(-\beta_1 \Delta t)$ for $0 \leq \Delta t$.

For the development of the calendar time component of this model you are referred to [Musa87].

3.3.4.4 Model estimation and reliability prediction. Suppose we have observed n failures of the software system at times $t_1, t_2, \ldots, t_n$, and from the last failure time t_n an additional time of x $(x \geq 0)$ has elapsed without failure (that is, $t_n + x$ is therefore the total time the software component has been observed since the start). Using the model assumptions, the likelihood function for this class is obtained as

$$L(\beta_0, \beta_1) = \beta_0^n \beta_1^n \left[\prod_{i=1}^{n} \exp(-\beta_1 t_i) \right] \exp(-\beta_0 [1 - \exp(-\beta_1 (t_n + x))])$$

so the MLEs of β_0 and β_1 are obtained as the solutions to the following pair of equations:

$$\hat{\beta}_0 = \frac{n}{1 - \exp(-\hat{\beta}_1 (t_n + x))} \quad \text{and} \quad \frac{n}{\hat{\beta}_1} - \frac{n(t_n + x)}{\exp(\hat{\beta}_1 (t_n + x)) - 1} - \sum_{i=1}^{n} t_i = 0$$

Once the estimates of β_0 and β_1 are obtained, we can use the invariance property of the MLEs to estimate other reliability measures. These

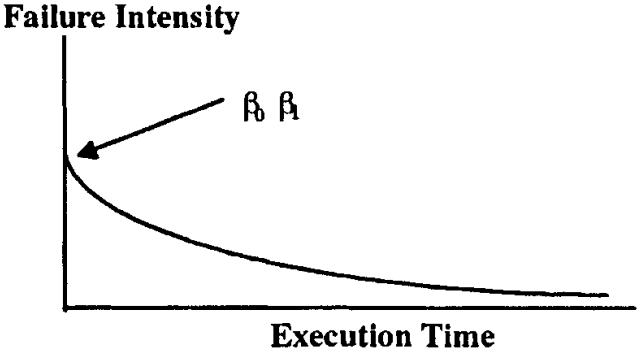

Figure 3.2 Failure intensity function for Musa's basic execution model.

include the reliability function, hazard rate, failure intensity function, etc. An example is the estimate of the failure intensity function. Since the function is $\lambda(t; \beta_0, \beta_1) = \beta_0\beta_1\exp(-\beta_1 t)$, the MLE of this function is $\hat{\lambda}(t; \hat{\beta}_0, \hat{\beta}_1) = \hat{\beta}_0\hat{\beta}_1\exp(-\hat{\beta}_1 t)$.

Example 3.3 Suppose the observed times of failures are $t_1 = 10$, $t_2 = 18$, $t_3 = 32$, $t_4 = 49$, $t_5 = 64$, $t_6 = 86$, $t_7 = 105$, $t_8 = 132$, $t_9 = 167$, $t_{10} = 207$, with an additional 15 CPU hours of no failure after the last one. All the units are taken to be in hours and are all measured in execution time. Using the above equations, the MLEs of β_0 and β_1 are the solution of the following equations:

$$\hat{\beta}_0 = \frac{10}{1 - \exp(-222\hat{\beta}_1)} \quad \text{and} \quad \frac{10}{\hat{\beta}_1} - \frac{2220}{\exp(222\hat{\beta}_1) - 1} - 870 = 0$$

The solutions to these equations are found to be: $\hat{\beta}_0 = 13.6$ and $\hat{\beta}_1 = 0.006$. Estimates of other reliability measures can then be calculated.

3.3.4.5 Comments. [Musa87] recommends this model in contrast to his logarithmic Poisson (see Sec. 3.5.3) if you wish to predict early reliability before program execution is initiated and failure data observed; if the program is substantially changing over time as the failure data are observed; and if you are interested in seeing the impact of a new software engineering technology on the development process.

3.3.5 Hyperexponential model

3.3.5.1 Overview of the model. This model is an extension of the classical exponential models considered by Musa and Goel (see Secs. 3.3.4 and 3.3.2). The hyperexponential model was first considered by Ohba [Ohba84] and has been addressed in variations by others (e.g., [Yama85] and [Lapr91]). The basic idea is that the different sections (or classes) of the software experience an exponential failure rate; however, the rates vary over these sections to reflect their different natures. This could be due to different programming groups doing the different parts, old versus new code, sections written in different languages, etc. The basic idea is that different failure behaviors are represented in the different sections. We thus reflect the sum of these different exponential growth curves, not by another exponential, but by a hyperexponential growth curve. If in observing a software system, you notice that different clusters of that software appeared to behave differently in their failure rates, the hyperexponential model may be more appropriate than the classical exponential model that assumes a similar failure rate.

3.3.5.2 Assumptions and data requirements. The *basic assumptions* are as follows. Suppose there are K sections (classes of the software) such that *within each class:*

1. The rate of fault detection is proportional to the current fault content within that section of the software.
2. The fault detection rate remains constant over the intervals between fault occurrence.
3. A fault is corrected instantaneously without introducing new faults into the software.

And for the software system as a whole:

4. The cumulative number of failures by time t, $M(t)$, follows a Poisson process with mean value function $\mu(t) = N \sum_{i=1}^{K} p_i[1 - \exp(-\beta_i t)]$ where $0 < \beta_i < 1$, $\sum_{i=1}^{K} p_i = 1$, $0 < p_i < 1$ and N is finite. (Notice it is a finite failure model.)

The Standard Assumptions of Sec. 3.3.1.2 are again assumed to hold.

The *data requirements* to implement this fault count model are:

1. The fault counts in each of the testing intervals, i.e., the f_i's.
2. The completion time of each period that the software is under observation, i.e., the t_i's.

3.3.5.3 Model form. Notice that if $K = 1$ we have the NHPP model of Sec. 3.3.2. Also, $\lim_{t \to \infty} \mu(t) = N$; so, as before, N represents the expected total number of faults to be eventually detected. (*Note:* N is not required to be an integer since it is the *expected number* of faults that will eventually be detected.) For the ith class, we also note that Np_i is the expected number of faults within that class. Since the failure intensity function is the derivative of $\mu(t)$, we therefore have

$$\lambda(t) = N \sum_{i=1}^{K} p_i \beta_i \exp(-\beta_i t)$$

Notice that the failure intensity function is strictly decreasing for $t > 0$.

3.3.5.4 Model estimation and reliability prediction. By letting $N_i^* = Np_i$, that is, N_i^* is the number of faults in the ith class, one can obtain the MLE estimates for each class as the MLE estimates given in the NHPP model (see Sec. 3.3.2.4). The MLE estimate of N is then found as the sum of the MLEs over the classes.

3.3.5.5 Comments. If there are only two classes (e.g., new versus old code; easy versus difficult to detect faults), this model is called the *modified exponential software reliability growth model* [Yama85].

Laprie et al. [Lapr91] considered a variation of this model for the situation where $K = 2$. They considered a hyperexponential model with failure rate function

$$\lambda(t) = \frac{p_1\zeta_1\exp(-\zeta_1 t) + p_2\zeta_2\exp(-\zeta_2 t)}{p_1\exp(-\zeta_1 t) + p_2\exp(-\zeta_2 t)}$$

From this they derived an expression for the unavailability of a system including both hardware and software components and then showed how the techniques could be extended. As discussed in Chap. 2, this is a significant step in helping bridge the gap between hardware and software models.

3.3.6 Others

We'll briefly describe some additional exponential models and variations of some of the models considered in this section. The first is the inflection S-shaped software reliability growth model proposed by Ohba [Ohba84], which has a mean value function of the form:

$$\mu(t) = N\left(\frac{1 - \exp(-\beta t)}{1 + \psi(r)\exp(-\beta t)}\right) \qquad \text{where } \psi(r) = \frac{1-r}{r},\ r > 0$$

The parameter r is the inflection rate that indicates the ratio of detectable faults to the total number of faults in the software. This model basically assumes that the error discovery rate increases throughout a test period. If r equals 1 we have the basic exponential growth curve considered in this section.

Everett [Ever92] proposed an extension of Musa's basic execution model, which he referred to as the *extended execution time* model. The mean value function for this model is of the form:

$$\mu(t) = \beta_0\left[1 - \int_0^1 \exp(-((a+1)\beta_1 t)x^a)\,dx\right] \qquad \text{where } a \geq 0$$

While the basic execution model was a function of two parameters, this one has an additional one, a. This parameter reflects the nonuniformity of instruction execution. The larger it gets the more nonuniform the execution is. If $a = 0$, this becomes the basic execution model, while as a gets large, this model tracks the logarithmic Poisson considered in Sec. 3.5.

Brooks and Motley [Broo80] and [AIAA93] formulated models of the Poisson and binomial type that are finite failures models. Each use the fault count per unit interval for model parameter estimation. These

two models are especially mentioned because they are among the few that treat the situation where not all of the code is being tested equally, and/or not all of the software is complete at the time of testing. Some of the modules are under test while others are still to be written. These models specifically factor in those components that are under test at both the system or module level. You are referred to the references for details.

3.4 Weibull and Gamma Failure Time Class of Models

For Weibull and gamma failure time classes, we take the per-fault failure distribution to be the traditional Weibull and gamma distributions, respectively, rather than the exponential distribution considered in the previous section. These are important distributions because of the great flexibility given for failure modeling because of the shape and scale parameters that define them. Many hardware failure processes are modeled using these distributions. As such they were naturally one of the first groups to be applied in the software arena.

3.4.1 Weibull model

3.4.1.1 Overview of the model. One of the most widely used models for hardware reliability modeling is the Weibull distribution. It can accommodate increasing, decreasing, or constant failure rates because of the great flexibility expressed through the model's parameters. This model belongs to the finite failures category and is of the binomial type using the [Musa83] classification scheme of Sec. 3.2.2.

3.4.1.2 Assumptions and data requirements. Including the Standard Assumptions of Sec. 3.3.1.2, the *basic assumptions* are:

1. There is a fixed number of faults (N) in the software at the beginning of the time in which the software is observed.
2. The time to failure of fault a, denoted as T_a, is distributed as a Weibull distribution with parameters α and β (that is, the density function of T_a is $f_a(t) = \alpha\beta t^{\alpha-1}\exp(-\beta t^{\alpha})$), with $\alpha, \beta > 0$ and $t \geq 0$. (Since the per-fault distribution is $f_a(t)$, the per-fault hazard rate is $z_a(t) = \alpha\beta t^{\alpha-1}$.)
3. The number of faults $(f_1, f_2, \ldots, f_n)$ detected in each of the respective intervals $[(t_0 = 0, t_1), (t_1, t_2), \ldots, (t_{i-1}, t_i), \ldots, (t_{n-1}, t_n)]$ are independent for any finite collection of times.

The *data requirements* to implement this fault count model are:

1. The fault counts in each of the testing intervals, i.e., the f_i's.
2. The completion time of each period that the software is under observation, i.e., the t_i's.

3.4.1.3 Model form. Since this model belongs to the binomial type, from Eqs. (3.2), (3.3), and the cumulative distribution function for a Weibull, we have for the failure intensity function and the mean value function:

$$\lambda(t) = Nf_a(t) = N\alpha\beta t^{\alpha-1}\exp(-\beta t^{\alpha}) \qquad \text{and}$$

$$\mu(t) = NF_a(t) = N(1 - \exp(-\beta t^{\alpha}))$$

Notice that $\lim_{t\to\infty} \mu(t) = N$, the total number of faults in the system at the start. Also, from the assumptions we have that if $\alpha = 1$, the distribution f_a becomes the exponential, and if it equals 2 we have the Rayleigh distribution, another important failure model in hardware reliability theory. We also note for the case $\alpha = 2$ that this becomes the early model considered by Schick-Wolverton [Schi73]. You can also see that if $0 < \alpha < 1$, the per-fault hazard rate is decreasing with respect to time; if α equals 1 (exponential) it is constant; and if $\alpha > 1$, it increases.

The form of the conditional hazard rate is shown to be (see Prob. 3.3):

$$z(t \mid t_{i-1}) = (N - i + 1)\alpha\beta(t + t_{i-1})^{\alpha-1} \qquad \text{for } t_{i-1} \le t + t_{i-1} < t_i$$

This function is plotted in Fig. 3.3 for $0 < \alpha < 1$ to contrast its behavior with the exponential class illustrated in Fig. 3.1. In that figure, the change occurred at each fault detection and it was a constant change. For the Weibull distribution, the change occurs at a fault detection, but the change is not constant. The effect on the hazard rate decreases with time because of the power function component.

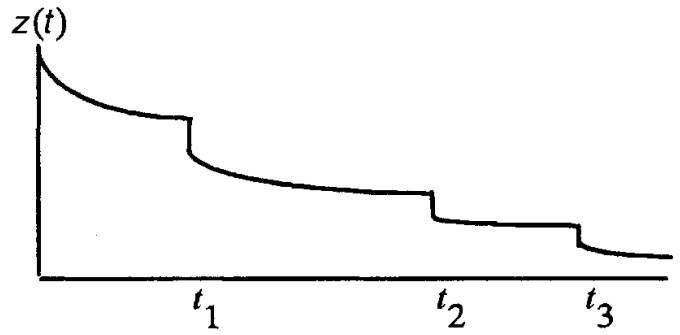

Figure 3.3 Program hazard rate function for Weibull.

The reliability function is obtained from the cumulative distribution function as $R(t) = 1 - F(t) = \exp(-\beta t^{\alpha})$ and, hence, from Eq. B.28 the MTTF is

$$\text{MTTF} = \int_0^{\infty} R(t)\,dt = \Gamma\left(\frac{1}{\alpha} + 1\right) \Big/ \beta^{\frac{1}{\alpha}}$$

where $\Gamma(\bullet)$ is the gamma function.

3.4.1.4 Model estimation and reliability prediction. [Cout73] shows that the parameters α and β can be estimated using method of moments, least squares, MLE, or even graphical procedures. For the case of the least squares estimates, suppose we let $b = \ln(\beta)$, $Y_i = \ln\,[\ln[1/(1 - F(i))]]$, with $F(i) = \sum_{j=1}^{i} f_j / \sum_{j=1}^{n} f_j$, the normalized cumulative number of faults found up through the ith time period, and $X_i = \ln(t_i)$. Starting with the cumulative distribution function, you can obtain the equation for a straight line of the form $Y = \alpha X + b$. Using the points (X_i, Y_i) calculated from the data, the usual least squares estimates (LSEs) for the slope and intercept are easily obtained. The estimates of α and β are then obtained as the slope estimate and $\hat{\beta} = \exp(\hat{b})$, respectively. Corresponding estimates of the reliability measures are obtained by substituting the associated LSEs for α and β in the measures expressions. [Wago73] also addresses the issues of parameter estimation for this distribution class.

3.4.2 S-shaped reliability growth model

3.4.2.1 Overview of the model. The S-shaped (or delayed S-shaped) reliability growth model ([Yama83], [Ohba84]) will be illustrative of the gamma distribution class. Here the per-fault failure distribution is gamma. The number of failures per time period, however, is a Poisson type model using the classification scheme of Musa and Okumoto [Musa83] rather than the binomial considered in the previous section. It, like the Weibull, is a finite failures model, i.e., $\lim_{t \to \infty} \mu(t) < \infty$. It is patterned after the Goel-Okumoto model considered in Sec. 3.3.2. Yamada, Ohba, and Osaki felt that the mean value function $\mu(t)$ is often a characteristic S-shaped curve rather than the exponential growth of the Goel-Okumoto model. The software error detection process can be described as an S-shaped growth curve to reflect the initial learning curve at the beginning, as test team members become familiar with the software, followed by growth and then leveling off as the residual faults become more difficult to uncover.

3.4.2.2 Assumptions and data requirements. Including the Standard Assumptions of Sec. 3.3.1.2, the *basic assumptions* are:

1. The cumulative number of failures by time t, $M(t)$, follows a Poisson process with mean value function $\mu(t)$. The mean value function is of the form $\mu(t) = \alpha[1 - (1 + \beta t)e^{-\beta t}]$ for $\alpha, \beta > 0$. This is a bounded, non-decreasing function of time with $\lim_{t \to \infty} \mu(t) = \alpha < \infty$, that is, it is a finite failure model.
2. The time between failures of the $(i - 1)$st and the ith depends on the time to failure of the $(i - 1)$st.
3. When a failure occurs, the fault which caused it is immediately removed and no other faults are introduced.

The *data requirements* to implement this model are:

1. The failure times, t_i's, of the software system, or
2. The number of faults detected, f_i, in each period of observation of the software along with the associated lengths l_i of those periods, $i = 1, \ldots, n$.

If data of type 1 are available, the data of the second type can be constructed by first forming a partition of the time period over which the software is observed and then counting up the number of faults that fall in each respective period of the partition. As a consequence, this model can be used for either the time-between-failures data or the number of faults per time period.

3.4.2.3 Model form. Suppose we have a partition of the time interval over which the software is observed. This partition could represent the testing intervals of the software. Let T^* denote this partition, that is, $t_0^* = 0 < t_1^* < \cdots < t_n^*$. Suppose $f_1, f_2, \ldots, f_n$ are the number of software faults detected in each interval of the partition, that is, f_i is the number of faults occurring in the interval of length $l_i = t_i^* - t_{i-1}^*$. From the assumptions we have, each f_i is an independent Poisson random variable with mean

$$\mu(t_i^*) - \mu(t_{i-1}^*) = \alpha[1 - (1 + \beta t_i^*)e^{-\beta t_i^*}] - \alpha[1 - (1 + \beta t_{i-1}^*)e^{-\beta t_{i-1}^*}]$$

$$= \alpha[(1 + \beta t_{i-1}^*)e^{-\beta t_{i-1}^*} - (1 + \beta t_i^*)e^{-\beta t_i^*}]$$

Also, from the mean value function $\mu(t) = \alpha[1 - (1 + \beta t)e^{-\beta t}]$, we have the failure intensity function $\lambda(t) = \mu'(t) = \alpha\beta^2 t e^{-\beta t}$. The model gets its S-shaped form because of the mean value function. Moreover, you can see that $\lim_{t \to \infty} \mu(t) = \alpha < \infty$, so we indeed have a finite failures model with α being the total number of faults in the system. If you were to plot the failure intensity function, you would see that it increases up to time $t = 1/\beta$ and then begins to decrease asymptotically approaching the time

axis. Since we have a Poisson type as well as a finite failures model, using Eq. (3.3), the per-fault time distribution between failures is $f_a(t) = \beta^2 t e^{-\beta t}$, as $\lambda(t) = \alpha f_a(t)$. This is the gamma distribution.

Using the above relationships one can also establish (See Prob. 3.4) the following reliability measures for this model:

The reliability of the function at time $t_i + \Delta t$ given a failure at time

$$t_i = \exp(-\alpha[(1 + \beta t_i)e^{-\beta t_i} - (1 + \beta(t_i + \Delta t))e^{-\beta(t_i + \Delta t)}])$$

The hazard rate function at time $t_i + \Delta t$ given a failure at time

$$t_i = \alpha\beta^2(t_i + \Delta t)\exp(-\beta(t_i + \Delta t))$$

The expected number of faults in the ith period of length

$$l = \alpha\left[\left(1 + \beta\sum_{j=i}^{i-1} l_j\right)e^{-\beta\sum_{j=i}^{i-1} l_j} - \left(1 + \beta\left(l + \sum_{j=i}^{i-1} l_j\right)\right)e^{-\beta\left(l + \sum_{j=i}^{i-1} l_j\right)}\right]$$

3.4.2.4 Model estimation and reliability prediction. The joint density of the fault counts over the given partition is

$$\prod_{i=1}^{n} \frac{[\mu(t_i^*) - \mu(t_{i-1}^*)]^{f_i}}{f_i!} \exp(-(\mu(t_i^*) - \mu(t_{i-1}^*)))$$

using the assumptions from the previous section. The MLEs of α and β are then shown to be (see Prob. 3.5) the solutions of the following pair of equations:

$$\sum_{i=1}^{n} f_i = \hat{\alpha}(1 - (1 + \hat{\beta}t_n^*)e^{-\hat{\beta}t_n^*}) \qquad \text{and}$$

$$\hat{\alpha}(t_n^*)^2 e^{-\hat{\beta}t_n^*} = \sum_{i=1}^{n}\left(\frac{\left(\sum_{j=1}^{i} f_j - \sum_{j=1}^{i-1} f_j\right)((t_i^*)^2 e^{-\hat{\beta}t_i^*} - (t_{i-1}^*)^2 e^{-\hat{\beta}t_{i-1}^*})}{((1 + \hat{\beta}t_{i-1})e^{-\hat{\beta}t_{i-1}} - (1 + \hat{\beta}t_i)e^{-\hat{\beta}t_i})}\right)$$

For implementation purposes, if you have the observed failure times (t_i's), you could let the $\sum_{j=1}^{i} f_i$'s be the cumulative number of failures up to time t_i and let the partition correspond to the failure time points (i.e., let $t_i^* = t_i$); on the other hand, if you have the number of faults detected per period (the f_i's), with the associated period lengths (the l_i's), you would let $t_i^* = \sum_{j=1}^{i} l_j$ in the above equations. These equations are solved through standard numerical analysis techniques. Such an implementation is incorporated into the SMERFS [Farr93a, Farr93b] software package.

MLEs of the associated reliability metrics are derived, as in past sections, by replacing the respective parameters α and β according to their corresponding MLE estimates.

3.4.2.5 Comments. For many applications at our facility, this model has been successfully applied. Many times this model was able to successfully fit a given data set when others couldn't. Excellent fits and predictions can be obtained when the S-shaped behavior is present in your data set. Trend analysis (see Chap. 10) can help to detect this.

3.5 Infinite Failure Category Models

Using Musa and Okumoto's classification scheme [Musa83] for this category of models, the $\lim_{t \to \infty} \mu(t) = \infty$ for the mean value function of the process. This means that the software will never be completely fault free. This could be caused by additional faults being introduced in the software through the error correction process.

3.5.1 Duane's model

3.5.1.1 Overview of the model. Originally proposed for hardware reliability, one of the earliest models was Duane's model [Duan64]. While at General Electric, Duane noticed that if the cumulative failure rate versus the cumulative testing time was plotted on *ln-ln* paper, it tended to follow a straight line. Crow [Crow74] observed that this behavior could be represented as a Weibull process. This process is a nonhomogeneous Poisson process in which the failure intensity function has the same form as the hazard rate for a Weibull distribution. This same behavior has been observed for software systems and has been used to develop various reliability estimates based upon this result. This model is sometimes referred to as the *power model* since the mean value function for the cumulative number of failures by time t is taken as a power of t, that is, $\mu(t) = \alpha t^{\beta}$ for some $\beta > 0$ and $\alpha > 0$. (For the case where $\beta = 1$, we have the homogeneous Poisson process model.) This model is an infinite failures model since $\lim_{t \to \infty} \mu(t) = \infty$.

3.5.1.2 Assumptions and data requirements. Including the Standard Assumptions of Sec. 3.3.1.2, the *basic assumption* is:

1. The cumulative number of failures by time t, $M(t)$, follows a Poisson process with mean value function $\mu(t) = \alpha t^{\beta}$ for some $\beta > 0$ and $\alpha > 0$.

The *data requirement* to implement this model is: either the actual times that the software failed, $t_1, t_2, \ldots, t_n$, or the elapsed time between failures $x_1, x_2, \ldots, x_n$, where $x_i = t_i - t_{i-1}$ and $t_0 = 0$.

3.5.1.3 Model form. From assumption 1 we have a Poisson process with a mean value function of $\mu(t) = \alpha t^{\beta}$. If T is the total time the software is observed, then we have

$$\frac{\mu(T)}{T} = \frac{\alpha T^{\beta}}{T} = \frac{\text{expected number of failures by time } T}{\text{total testing time}}$$

so that if we take the natural log of both sides of the equations we have

$$Y = \ln\left(\frac{(\mu(t))}{T}\right) = \ln\left(\frac{\alpha T^{\beta}}{T}\right) = \ln(\alpha) + (\beta - 1)\ln(T)$$

We can thus see if the first equation is plotted on *ln-ln* paper versus observed time T, or the second equation is plotted on regular paper versus $\ln(T)$ we will obtain a straight line. It is this form that is fitted to a given data set.

The failure intensity function is obtained by taking the derivative of the mean value function, that is, $\lambda(t) = d\mu(t)/dt = \alpha\beta t^{\beta - 1}$. From this function we see that the failure intensity function is strictly increasing for $\beta > 1$, a constant for the case of a homogeneous Poisson process ($\beta = 1$), and strictly decreasing for $1 > \beta > 0$ only. For $\beta > 1$, there can be no reliability growth!

3.5.1.4 Model estimation and reliability prediction. Crow [Crow74] derived the maximum likelihood estimates as:

$$\hat{\alpha} = \frac{n}{t_n^{\hat{\beta}}} \qquad \text{and} \qquad \hat{\beta} = \frac{n}{\sum_{i=1}^{n-1} \ln(t_n/t_i)}$$

Maximum likelihood estimates are then derived for the mean value and failure intensity function, $\mu(t)$ and $\lambda(t)$, by replacing the parameters α and β according to their maximum likelihood estimates (MLEs). In his paper Crow also shows that the MLE for the mean time to failure (MTTF) is $\hat{MTTF} = t_n/n\hat{\beta}$ for the $(n + 1)$st failure, and he provides tables that can be used to construct confidence intervals for this reliability measure.

3.5.2 Geometric model

3.5.2.1 Overview of the model. The geometric model was proposed by Moranda [Mora75b, Mora79] and is a variation of the Jelinski-Moranda model considered in Sec. 3.3.1. The time between failures is taken to be an exponential distribution whose mean decreases in a geometric fash-

ion. The discovery of the earlier faults is taken to have a larger impact on reducing the hazard rate than the later ones. As failures occur the hazard rate decreases in a geometric progression. Figure 3.4 illustrates this behavior. The function is initially a constant, D, but it decreases geometrically ($0 < \phi < 1$) as each failure occurs. The change in the reduction of the function is seen to get smaller as more failures occur, reflecting the smaller impact of the later-occurring faults.

3.5.2.2 Assumptions and data requirements. Including ths Standard Assumptions of Sec. 3.3.1.2, the *basic assumptions* are:

1. The fault detection rate forms a geometric progression and is constant between fault detections, that is, $z(t) = D\phi^{i-1}$, where $0 < \phi < 1$ and $t_{i-1} \le t < t_i$, with t_{i-1} being the time of the $(i-1)$st failure.
2. There is an infinite number of total faults in the system, i.e., $\lim_{t \to \infty} \mu(t) = \infty$, where $\mu(t)$ is the mean value function of the process.
3. The time between fault detection follows an exponential distribution.

The *data requirement* to implement this model is: either the actual times that the software failed, $t_1, t_2, \ldots, t_n$, or the elapsed time between failures $x_1, x_2, \ldots, x_n$, where $x_i = t_i - t_{i-1}$ and $t_0 = 0$.

3.5.2.3 Model form. Using assumptions 1 and 3, we have the density for the time between failures of the ith and $(i-1)$st is exponential of the form: $f(X_i) = D\phi^{i-1}\exp(-D\phi^{i-1}X_i) = z(t_{i-1})\exp(-z(t_{i-1})X_i)$. Thus the expected time between failures is

$$E(X_i) = \frac{1}{z(t_{i-1})} = \frac{1}{D\phi^{i-1}} \qquad \text{for } i = 1, \ldots, n$$

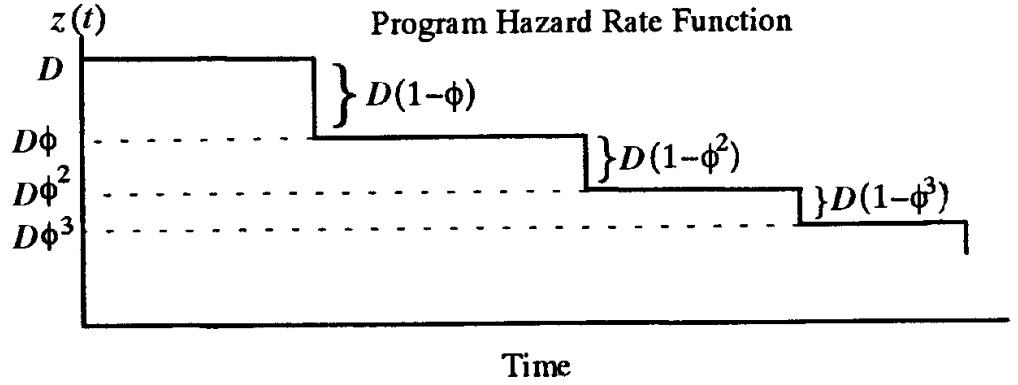

Figure 3.4 Geometric model hazard rate function.

Using the fact that $E(X_i) \approx 1/\lambda(t)$ and $i \approx \mu(t)$ (see [Musa87]), it follows (see Prob. 3.6) that

$$\mu(t) = \frac{1}{\beta} \ln([D\beta \exp(\beta)]t + 1) \qquad \text{and}$$

$$\lambda(t) = \frac{D \exp(\beta)}{[D\beta \exp(\beta)]t + 1} \qquad \text{where } \beta = -\ln(\phi) \text{ for } 0 < \phi < 1$$

Clearly, $\lim_{t \to \infty} \mu(t) = \infty$, so we indeed have an infinite failures category model.

3.5.2.4 Model estimation and reliability prediction. From the previous section and the assumptions, the joint density function for the X_i's is

$$\prod_{i=1}^{n} f(X_i) = D^n \prod_{i=1}^{n} \phi^{i-1} \exp\left(-D \sum_{i=1}^{n} \phi^{i-1} X_i\right)$$

Taking the natural log of this function and taking the partials with respect to ϕ and D, the maximum likelihood estimates are the solutions of the following pair of equations:

$$\hat{D} = \frac{\hat{\phi} n}{\sum_{i=1}^{n} \hat{\phi}^i X_i} \qquad \text{and} \qquad \frac{\sum_{i=1}^{n} i\hat{\phi}^i X_i}{\sum_{i=1}^{n} \hat{\phi}^i X_i} = \frac{n+1}{2}$$

Again, the MLE of the various reliability measures can be obtained using the invariance property of the MLE estimates. For example, the MLE of the failure intensity function is $\hat{\lambda}(t) = \hat{D} \exp(\hat{\beta})/\{[\hat{D}\hat{\beta} \exp(\hat{\beta})]t + 1\}$ where $\hat{\beta} = -\ln(\hat{\phi})$.

It is left as an exercise to you (Prob. 3.7) to show that the least squares estimates based upon the X_i's are the solutions to the following pair of equations:

$$\hat{D}_{\text{LS}} = \frac{\sum_{i=1}^{n} \frac{1}{\hat{\phi}_{\text{LS}}^{2(i-1)}}}{\sum_{i=1}^{n} \frac{x_i}{\hat{\phi}_{\text{LS}}^{i-1}}} \qquad \text{and}$$

$$\left(\sum_{i=1}^{n} \frac{1}{\hat{\phi}_{\text{LS}}^{2(i-1)}}\right)\left(\sum_{i=1}^{n} \frac{X_i(i-1)}{\hat{\phi}_{\text{LS}}^{i-1}}\right) = \left(\sum_{i=1}^{n} \frac{X_i}{\hat{\phi}_{\text{LS}}^{i-1}}\right)\left(\sum_{i=1}^{n} \frac{(i-1)}{\hat{\phi}_{\text{LS}}^{2(i-1)}}\right)$$

3.5.2.5 Comments. An extension of this model was proposed by Lipow and is discussed in [Suke76]. Lipow relaxed the assumption of an infinite number of faults being present in the code. For his model, the hazard rate is taken to be of the form $z(t) = D\phi^{n_{i-1}}$ for $t_{i-1} \leq t < t_i$. The term n_{i-1} is the cumulative number of faults found up to the ith interval that the software is being observed. D and ϕ are as defined in the geometric model. Another variation proposed by Moranda [Mora75b] was the geometric Poisson. This takes the number of failures during specified time intervals as being independent Poisson random variables with the mean for the ith interval to be $D\phi^{i-1}$. Again we see the geometric progression of the mean starting as we progress from an initial value of D.

3.5.3 Musa-Okumoto logarithmic Poisson

3.5.3.1 Overview of the model. The logarithmic Poisson proposed by Musa and Okumoto [Musa84] is another model that has been extensively applied. It is also a nonhomogeneous Poisson process with an intensity function that decreases exponentially as failures occur. The exponential rate of decrease reflects the view that the earlier discovered failures have a greater impact on reducing the failure intensity function than those encountered later. It is called *logarithmic* because the expected number of failures over time is a logarithmic function.

3.5.3.2 Assumptions and data requirements. Including the Standard Assumptions of Sec. 3.3.1.2, the *basic assumptions* are:

1. The failure intensity decreases exponentially with the expected number of failures experienced, that is, $\lambda(t) = \lambda_0 \exp(-\theta\mu(t))$, where $\mu(t)$ is the mean value function, $\theta > 0$ is the failure rate decay parameter, and $\lambda_0 > 0$ is the initial failure rate.
2. The cumulative number of failures by time t, $M(t)$, follows a Poisson process.

Because of assumption 1, it follows that $\mu(t) = \ln(\lambda_0\theta t + 1)/\theta$ (Prob. 3.8), and therefore $\lambda(t) = \lambda_0/(\lambda_0\theta t + 1)$. Clearly this is an infinite failure model.

The *data requirements* are: either the actual times that the software failed, $t_1, t_2, \ldots, t_n$, or the elapsed time between failures $x_1, x_2, \ldots, x_n$, where $x_i = t_i - t_{i-1}$.

3.5.3.3 Model form. The plot of the failure intensity function versus time is illustrated in Fig. 3.5. This illustrates the exponential type decay and the fact that the earlier encountered failures have a more dramatic impact than the later ones. The parameter θ controls the shape of the curve.

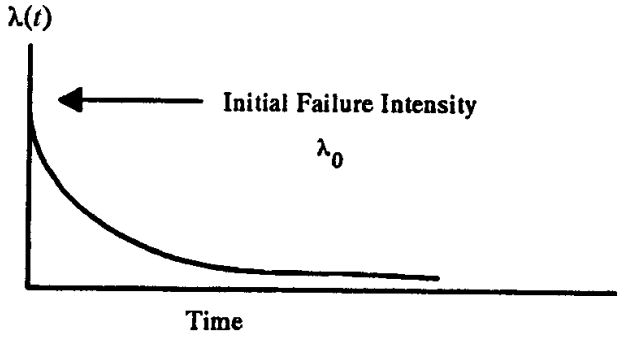

Figure 3.5 Failure intensity function for logarithmic poisson model.

A second expression of the logarithmic Poisson model to aid in obtaining the maximum likelihood estimates is through a reparameterization of the model. We let $\beta_0 = \theta^{-1}$ and $\beta_1 = \lambda_0\theta$. The intensity and mean value functions become in this case: $\lambda(t) = \beta_0\beta_1/(\beta_1 t + 1)$ and $\mu(t) = \beta_0 \ln(\beta_1 t + 1)$.

[Musa87] derives the program reliability and the hazard rate functions after the $(i-1)$st failure, respectively, as:

$$R(\Delta t \mid t_{i-1}) = \left[\frac{\beta_1 t_{i-1} + 1}{\beta_1(t_{i-1} + \Delta t) + 1}\right]^{\beta_0} \qquad \text{for } \Delta t \geq 0$$

and

$$z(\Delta t \mid t_{i-1}) = \beta_0\beta_1/(\beta_1(t_{i-1} + \Delta t) + 1) \qquad \text{for } \Delta t \geq 0$$

3.5.3.4 Model estimation and reliability prediction. Using the reparameterized model, the maximum likelihood estimates of β_0 and β_1 are shown in [Musa87] to be the solutions of the following pair of equations:

$$\hat{\beta}_0 = \frac{n}{\ln(1 + \hat{\beta}_1 t_n)} \qquad \text{and} \qquad \frac{1}{\hat{\beta}_1}\sum_{i=1}^{n}\frac{1}{1 + \hat{\beta}_1 t_i} = \frac{n t_n}{(1 + \hat{\beta}_1 t_n)\ln(1 + \hat{\beta}_1 t_n)}$$

As in the previous sections, estimates of various reliability measures are then obtained by using the invariance property of the MLE, i.e., substituting in an expression the MLEs $\hat{\beta}_0$ and $\hat{\beta}_1$ for the corresponding parameters β_0 and β_1. For example, from the above the MLE of the failure rate function is $\hat{\lambda}(t) = \hat{\beta}_0\hat{\beta}_1/(\hat{\beta}_1 t + 1)$.

Example 3.4 Suppose we use the same data as in Example 3.3, i.e.,

$t_1 = 10, t_2 = 18, t_3 = 32, t_4 = 49, t_5 = 64, t_6 = 86, t_7 = 105, t_8 = 132, t_9 = 167, t_{10} = 207$

and there is an additional 15 CPU hours of no failure after the last one. All the units are taken to be in hours and are measured in execution time. Using the SMERFS software package, the solutions to the MLE equations of the previous section are found to be: $\hat{\beta}_0 = 7.93$ and $\hat{\beta}_1 = 0.01139$. Various reliability estimates can then be obtained using the invariance property of the MLE. For example, the hazard rate function is

$$\hat{\lambda}(t;\, \hat{\beta}_0, \hat{\beta}_1) = \hat{\lambda}(t;\, 7.933, 0.01139) = \hat{\beta}_0\hat{\beta}_1/(\hat{\beta}_1 t + 1) = 0.0903/(0.011385t + 1)$$

The estimate of the initial hazard rate is therefore $\hat{\lambda}(0;\, \hat{\beta}_0, \hat{\beta}_1) = 0.0903$, and the estimate of the hazard rate from the time of the last failure is $\hat{\lambda}(207;\, \hat{\beta}_0, \hat{\beta}_1) = 0.0269$, that is, a little less than three failures per 100 hours of CPU operation.

3.5.3.5 Comments. This model is especially applicable when it is anticipated that the program's operational use will be decidedly nonuniform. The nonuniformity tends to make the failures encountered earlier have more of an impact than later ones. It is this impact that this model is especially good at capturing. This model was also one of the selected models in the AIAA Recommended Practice Standard [AIAA93] as a good candidate to try initially for fitting your reliability data set.

3.6 Bayesian Models

This group of models views reliability growth and prediction in a Bayesian framework rather than the traditional one considered in the previous sections. The previous models allow change in the reliability only when an error occurs. Most of them also look at the impact of each fault as being of the same magnitude. A Bayesian model takes a subjective viewpoint in that if no failures occur while the software is observed then the reliability should increase, reflecting the growing confidence in the software by the user. The reliability is therefore a reflection of both the number of faults that have been detected and the amount of failure-free operation. This reflection is expressed in terms of a prior distribution representing the view from past data and a posterior distribution that incorporates past and current data.

The Bayesian models also reflect the belief that different faults have different impacts on the reliability of the program. The number of faults is not as important as their impacts. If we have a program that has a number of faults in seldomly used code, is that program less reliable than one that has only one fault in the part of the code that is used often? The Bayesian would say no! The Bayesian modeler says that it is more important to look at the behavior of the software than to estimate the number of faults in it. The mean time to failure would therefore be a very important statistic in this framework.

The prior distribution reflecting the view of the model parameters from past data is an essential part of this methodology. It reflects the viewpoint that one should incorporate past information, say projects of similar nature, etc., in estimating reliability statistics for the present and future. This distribution is simultaneously one of the Bayesian's framework strengths and weaknesses. One should incorporate the past, but *how* is the question.

The basic idea on the mathematics behind this theory is as follows. Suppose we have a distribution for our reliability data that depends upon some unknown parameters, ξ, that is, $f_T(\mathbf{t} \mid \xi)$ and a prior $g(\xi; \phi)$ that reflects our views on those parameters, ξ, from historical data. Once additional data have been gathered through the vector $\mathbf{t}$ (note that boldfacing of a component denotes a possible vector of subcomponents to allow for multidimensionality, that is, $\xi = (\xi_1, \xi_2, \ldots, \xi_K)$), our view of the parameter ξ changes. That change is reflected in the posterior distribution which is calculated as

$$h(\xi \mid \mathbf{t}; \phi) = \frac{f_T(\mathbf{t} \mid \xi) g(\xi; \phi)}{\int \cdots \int f_T(\mathbf{t} \mid \xi) g(\xi; \phi)\, d\xi} = \frac{f_T(\mathbf{t} \mid \xi) g(\xi; \phi)}{f_T(\mathbf{t}; \phi)}$$

Using the posterior distribution, various estimates of ξ can then be obtained leading to reliability estimates involving ξ. A common Bayesian procedure is to define a loss function, $l(\hat{\xi}(\mathbf{t}), \xi)$, where $\hat{\xi}(\mathbf{t})$ is an estimate of ξ, and then choose the estimate of ξ that minimizes the expected loss using the posterior distribution. For a squared-error function or quadratic loss function, that is, $l(\hat{\xi}(\mathbf{t}), \xi) = (\hat{\xi}(\mathbf{t}) - \xi)^2$, the estimate is the mean of the posterior distribution, that is, $E\{\xi \mid \mathbf{t}\}$. You are referred to any mathematical statistics book for further details, e.g., [Mood74].

3.6.1 Littlewood-Verrall reliability growth model

3.6.1.1 Overview of the model. The Littlewood-Verrall model [Litt73, Litt78, Litt80a] is probably the best example of this class of models. It is also the one recommended in the AIAA Recommended Practice Standard [AIAA93] if you are looking for an initial candidate Bayesian model for fitting your data. The model tries to account for fault generation in the fault correction process by allowing for the probability that the software program could become less reliable than before. With each fault correction, a sequence of software programs is generated. Each is obtained from its predecessor by attempting to fix the fault. Because of uncertainty, the new version could be better or worse than its predecessor; thus another source of variation is introduced. This is reflected in the parameters that define the failure time distributions, which are taken to be random. The distribution of failure times is, as in the earlier models, assumed to be exponential with a certain failure rate, but it is that rate that is assumed to be random rather than constant as before. The distribution of this rate, as reflected by the prior, is assumed to be a gamma distribution.

3.6.1.2 Assumptions and data requirements. The *basic assumptions* are:

1. Successive execution times between failures, that is, X_i's, are assumed to be independent exponential random variables with parameter ξ_i, $i = 1, \ldots, n$.
2. The ξ_i's form a sequence of independent random variables, each with a gamma distribution of parameters α and $\psi(i)$. The function $\psi(i)$ is taken to be an increasing function of i that describes the quality of the programmer and the difficulty of the task. A good programmer would have a more rapidly increasing function than a poorer one.
3. The software is operated in a manner similar to the anticipated operational usage.

By requiring the function ψ to be increasing, the condition $P\{\xi_j < L\} \geq P\{\xi_{j-1} < L\}$ for all j is satisfied. This reflects the intention to make the program better after a fault is discovered and corrected, but it cannot be assured that the goal is achieved.

The *data requirements* are: the time-between-failure occurrences, i.e., the x_i's.

3.6.1.3 Model form. To calculate the posterior distribution we first need the marginal distribution of the x_i's. The prior distribution is of the form:

$$g(\xi_i; \psi(i), \alpha) = \frac{[\psi(i)]^{\alpha} \xi_i^{\alpha-1} \exp(-\psi(i)\xi_i)}{\Gamma(\alpha)}, \quad \xi_i > 0$$

Using this prior and the following conditional exponential distribution for the x_i's: $f_{X_i}(x_i \mid \xi_i) = \xi_i \exp(-\xi_i x_i)$ for $x_i > 0$, the marginal distribution of the x_i (Prob. 3.9) can be shown to be:

$$f(x_i \mid \alpha, \psi(i)) = \frac{\alpha[\psi(i)]^{\alpha}}{[x_i + \psi(i)]^{\alpha+1}} \qquad \text{for } x_i > 0$$

that is, a Pareto distribution, so that the joint density is

$$f(x_1, x_2, \ldots, x_n) = \frac{\alpha^n \prod_{i=1}^{n} [\psi(i)]^{\alpha}}{\prod_{i=1}^{n} [x_i + \psi(i)]^{\alpha+1}} \qquad \text{for } x_i > 0, i = 1, \ldots, n$$

The posterior distribution for the ξ_i's is therefore obtained as (see Prob. 3.10):

$$h(\xi_1, \xi_2, \ldots, \xi_n) = \frac{\prod_{i=1}^{n} \xi_i^{\alpha} \exp\left(-\sum_{i=1}^{n} \xi_i(x_i + \psi(i))\right)}{[\Gamma(\alpha+1)]^n \prod_{i=1}^{n} (x_i + \psi(i))^{\alpha+1}} \quad \text{for } \xi_i > 0, i = 1, \ldots, n$$

Each ξ_i is an independent gamma distribution with parameters $\alpha + 1$ and $1/(x_i + \psi(i))$. Therefore, if you use a quadratic loss function, the Bayesian estimate of ξ_i is the mean; namely, $(\alpha + 1)/(x_i + \psi(i))$.

Littlewood and Verrall suggest a linear and quadratic form for the $\psi(i)$ function, that is, $\psi(i) = \beta_0 + \beta_1 i$ (the linear form) and $\psi(i) = \beta_0 + \beta_1 i^2$ (the quadratic form).

The failure intensity functions for the linear and quadratic forms can be shown (see [Musa87]) to be

$$\lambda_{\text{linear}}(t) = \frac{\alpha - 1}{\sqrt{\beta_0^2 + 2\beta_1 t(\alpha - 1)}} \quad \text{and}$$

$$\lambda_{\text{quadratic}}(t) = \frac{\upsilon_1}{\sqrt{t^2 + \upsilon_2}}((t + (t^2 + \upsilon_2)^{1/2})^{1/3} - (t - (t^2 + \upsilon_2)^{1/2})^{1/3})$$

where $\upsilon_1 = (\alpha - 1)^{1/3}/(18\beta_1)^{1/3}$ and $\upsilon_2 = 4\beta_0^3/(9(\alpha - 1)^2\beta_1)$.

3.6.1.4 Model estimation and reliability prediction. Using the marginal distribution function for the x_i's, the maximum likelihood estimates of α, β_0, and β_1 can be found as the solutions to the following system of equations:

$$\frac{n}{\hat{\alpha}} + \sum_{i=1}^{n} \ln(\hat{\psi}(i)) - \sum_{i=1}^{n} \ln(x_i + \hat{\psi}(i)) = 0$$

$$\hat{\alpha} \sum_{i=1}^{n} \frac{1}{\hat{\psi}(i)} - (\hat{\alpha} + 1) \sum_{i=1}^{n} \frac{1}{x_i + \hat{\psi}(i)} = 0$$

$$\hat{\alpha} \sum_{i=1}^{n} \frac{i'}{\hat{\psi}(i)} - (\hat{\alpha} + 1) \sum_{i=1}^{n} \frac{i'}{x_i + \hat{\psi}(i)} = 0$$

where $\psi(i) = \beta_0 + \beta_1 i'$ and i' is either i or i^2. Using a uniform prior for α, Littlewood and Verrall [Litt73] obtain the marginal distribution of the x_i's as a function of β_0 and β_1 only. The MLEs of these two parameters are then obtained. You are referred to that reference for further discussions.

A third procedure is to derive the least squares estimates. Using the fact that

$$E\{X_i\} = \int_0^\infty \frac{\alpha x_i [\psi(i)]^\alpha}{[x_i + \psi(i)]^{\alpha+1}}\, dx_i = \frac{[\psi(i)]}{\alpha - 1}$$

the least squares estimates are those parameters that minimize:

$$S(\alpha, \beta_0, \beta_1) = \sum_{i=1}^{n} \left(x_i - \frac{[\psi(i)]}{\alpha - 1} \right)^2$$

You are referred to [Farr83] for further details.

Estimates of various reliability measures are then obtained accordingly, using any of the estimates derived from above. For example, the estimate of the mean time to failure for the ith failure is:

$$\text{MTTF} = \frac{[\hat{\psi}(i')]}{\hat{\alpha} - 1}$$

where i' is the linear or quadratic term for i.

Example 3.5 Suppose we use the data set that we have considered before, i.e., the following elapsed time between failures where the units of time are CPU hours:

$x_1 = 10, x_2 = 8, x_3 = 14, x_4 = 17, x_5 = 15, x_6 = 22, x_7 = 19, x_8 = 27, x_9 = 35, x_{10} = 40$

Using the SMERFS program and the linear form, the MLE estimates for α, β_0, and β_1 are obtained as $\hat{\alpha} = 10415$, $\hat{\beta}_0 = 57269$, and $\hat{\beta}_1 = 28162$. The estimate of the failure intensity function is thus

$$\text{Failure intensity function (linear) at time } t = \hat{\lambda}(t; \hat{\alpha}, \hat{\beta}_0, \hat{\beta}_1) = \frac{(\hat{\alpha} - 1)}{\sqrt{\hat{\beta}_0^2 + 2\hat{\beta}_1 t(\hat{\alpha} - 1)}}$$

so the estimated current failure intensity is with $t = 207$, $\hat{\lambda}(207) = 0.02949$. An estimate of the mean time to failure is obtained as MTTF $= [\hat{\beta}_0 + \hat{\beta}_1(11)]/(\hat{\alpha} - 1) =$ 35.25 CPU hours.

3.6.1.5 Comments. A paper by Mazzuchi and Soyer [Mazz88] considers a variation of this model by assuming all of the parameters α, β_0, and β_1 are random variables with appropriate priors. Employing some approximations because of computational difficulties, they then obtain some corresponding results.

Musa [Musa84a] considered the use of a rational function for $\psi(i)$. He felt that this parameter should be inversely related to the number of failures remaining. The form of this function was expressed as:

$$\psi(i) = \frac{N(\alpha + 1)}{\lambda_0(N - i)}$$

Here N is the expected number of faults within the software as time becomes infinite, λ_0 is the initial failure intensity function, and α is the parameter of the gamma distribution considered earlier. The index i is the failure index. You can see that as the number of remaining failures decreases, the scale parameter, $\psi(i)$, increases.

Another variation of this model is the one considered by Keiller et al. [Keil83]. Again, successive failures follow an exponential distribution with an associated gamma prior. However for this case, the reliability growth is induced by the shape parameter α rather than the scale parameter $\psi(i)$. See [Keil83] for additional details.

3.6.2 Other Bayesian models

A Bayesian formulation of the Jelinski-Moranda model was considered by [Lang88] and [Litt80c]. The Jelinski-Moranda model [Lang88] assumes that the time between failures is an exponential distribution. In addition, the parameters that define the distribution are themselves random variables. [Litt80c] uses a similar approach to derive the additional testing time required to achieve a specified reliability for a given operational time. Others who have taken this same Bayesian approach with this model are: Jewell [Jewe85], who allows the parameter, λ, of the Poisson distribution for N to itself be a random variable; Littlewood and Sofer [Litt87], who consider a reparameterized version; and Csenki [Csen90], who uses this Bayesian approach to derive the distribution for the time to next failure.

Other Bayesian approaches include: Kyparisi and Singpurwalla's [Kypa84] Bayesian nonhomogeneous Poisson process model; Liu's [Liu87] Bayesian geometric model; Becker and Camarinopoulos' [Beck90] Bayesian approach for a program that may be error free; and Thompson and Chelson's [Thom80] Bayesian model. [Xie91a] provides additional details on these models as well as other pertinent references.

3.7 Model Relationships

This section will provide an overall structure for software reliability modeling.

3.7.1 Generalized exponential model class

The generalized exponential model first appeared in the AIAA's *Recommended Practice for Software Reliability* [AIAA93] standard as one of the recommended models because of its generality. It reflects work

by Shooman and Musa to simplify the modeling process by having a single set of equations to represent a number of important models having the exponential hazard rate function considered in Sec. 3.3. The overall idea is that the failure occurrence rate is proportional to the number of faults remaining, and the failure rate remains constant between failures while it is reduced by the same amount when a fault is removed. Besides the Standard Assumptions of Sec. 3.3.1.2, the other assumptions of the model are:

1. The failure rate is proportional to the current fault content of the software.
2. The faults that caused a failure are corrected instantaneously without additional faults being introduced by the correction process.

The data required are the usual time between failures, x_i's, or the time of the failures, the t_i's.

The model form is expressed as:

$$z(t) = K[E_0 - E_c(t)]$$

where $z(\bullet)$ is the software hazard rate function; t is a time or resource variable for measuring the progress of the project; K is a constant of proportionality denoting the failures per unit of t; E_0 is the initial number of faults in the software; and E_c is the number of faults in the software which have been found and corrected after t units have been expended. Table 3.2 reflects how this model is related to some of the models considered in Sec. 3.3.

TABLE 3.2 Generalized Exponential Model Relationships

Model	Original hazard rate function	Parameter equivalences
Generalized form	$K[E_0 - E_c(t)]$	
Shooman model	$K'[E_0/I_T - \varepsilon_c(t)]$	$\varepsilon_c = E_c/I_T$ where I_T is the number of instructions $K' = KI_T$
Jelinski-Moranda	$\phi(N - (i-1))$	$\phi = K, N = E_0, (i-1) = E_c(t)$
Basic execution model	$\beta_1\beta_0[1 - \mu(t)/\beta_0]$ where $\mu(t) = \beta_0[1 - \exp(-\beta_1 t)]$	$\beta_0 = E_0, \beta_1 = K, \mu(t) = E_c(t)$
Logarithmic Poisson	$\beta_1\beta_0\exp(-\mu(t)/\beta_0)$ where $\mu(t) = \beta_0\ln(\beta_1 t + 1)$	$\beta_1\beta_0 = KE_0$, $E_0 - E_c(t) = E_0\exp(-\mu(t)/\beta_0)$

3.7.2 Exponential order statistic model class

For this model the failure times of a software reliability growth process are modeled as order statistics of independent and nonidentically distributed exponential random variables. Miller [Mill86] developed this generalization and showed that the Jelinski-Moranda, Goel's NHPP, the logarithmic Poisson, and the power law models (e.g., Duane's model), among others, are special cases of this class. Miller lets a set of random variables, X_i, $i = 1, \ldots, n$, be independent random variables with respective rates λ_i. He then defines a stochastic process, $N(t)$, that is the number of failures that have occurred by time t and an associated process $\{T_i, i = 1, \ldots .\}$ that is the times of the failures, that is, $T_i = \min\{t: N(t) \geq i\}$. The $N(t)$ and the T_i are the counting and the occurrence-time processes of an exponential order statistic model with parameter set $\underset{\sim}{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. The exponential order statistic model is characterized by this parameter set. The only restriction on it is that $\lambda_i \geq 0$, $i = 1, 2, \ldots$, and $\sum_{i=1}^{\infty} \lambda_i < \infty$. Miller shows some special cases of the class by letting the λ_i's be various functions (e.g., constant, geometric, logarithmic, etc.). He then goes on to prove various results concerning this general class, including showing how a wide variety of known models are obtained as special cases. Estimation of the parameters is not considered in this paper. You are referred to his paper for additional details.

3.8 Software Reliability Prediction in Early Phases of the Life Cycle

All of the approaches considered so far in this chapter attempt to predict the reliability of the software in the later stages of the life cycle (integrated test and beyond). What is lacking are models that are applicable in the earlier phases, when less costly changes can be made. This section will briefly describe some approaches in this area. More research is, however, greatly needed.

3.8.1 Phase-based model

Gaffney and Davis [Gaff88, Gaff90] of the Software Productivity Consortium developed the phase-based model. It makes use of fault statistics obtained during the technical review of requirements, design, and the implementation to predict the reliability during test and operation.

The assumptions for this model are:

1. The development effort's current staffing level is directly related to the number of faults discovered during the development phase.

2. The fault discovery curve is monomodal.
3. Code size estimates are available during the early phases of a development effort. The model expects that fault densities will be expressed in terms of the number of faults per thousand lines of source code, which means that faults found during the requirements analysis and software design will have to be normalized by the code size estimates.

Their model is then expressed as:

$$\Delta V_t = \text{number of discovered faults per KLOC from time } t-1 \text{ to } t$$

$$= E[\exp(-B(t-1)^2 - \exp(-Bt^2)]$$

where E = total lifetime fault rate expressed in faults per thousand source lines of code (KLOC)

t = fault discovery index, with $t = 1$ means *requirements analysis*, $t = 2$ means *software design*, $t = 3$ means *implementation*, $t = 4$ means *unit test*, $t = 5$ means *software integration* $t = 6$ means *system test*, $t = 7$ means *acceptance test* (note t is not treated in the traditional sense that we have used it previously)

and

$$B = \frac{1}{2\tau_p^2}$$

where τ_p is the defect discovery phase constant, the peak of a continuous curve fit to the failure data. This is the point at which 39 percent of the faults have been discovered.

The cumulative form of the model is $V_t = E[1 - \exp(-Bt^2)]$, where V_t is the number of faults per KLOC that have been discovered through phase t. As data become available B and E can be estimated. This quantity can also be used to estimate the number of remaining faults at stage t by multiplying $E\exp(-Bt^2)$ by the number of source line statements at that point.

3.8.2 Predicting software defects from Ada designs

This section, unlike the others that consider prediction in the early phases, addresses one particular language, Ada. Agresti and Evanco [Agre92] attempted to develop models for predicting defect density based on product and process characteristics for Ada development efforts. The model they considered was:

$$\log(DD) = a_0 + \sum_{i=1}^{m} a_i * \log(X_i)$$

that is, a multivariate linear regression with the dependent variable being the log of the defect density (*DD*), and the independent variables being the log of various product characteristics of the Ada design and process characteristics. Among the variables considered were:

- *Content coupling measures* of the external or architectural complexity of the design
  1. Number of exported declarations per library unit
  2. The ratio of the number of import declarations to export declarations
  3. Number of content couples per library unit
- *Volatility*
  1. Number of modifications per library unit
  2. The number of nondefect modifications per unit library
- *Reuse*
  1. Fraction of the total compilation units that were new or extensively modified
- *Complexity*
  1. The ratio of imports from within a subsystem to the total imports

Using data from various projects, the multivariate regression analyses were conducted with resulting models explaining 63 to 74 percent of the variation in the dependent variable. Content coupling emerged as a consistently significant variable in these models. We plan to apply this approach on additional data sets. The results appear encouraging for this phase of the life cycle and this particular language.

3.8.3 Rome Laboratory work

One of the earliest and most well known efforts to predict software reliability in the earlier phases of the life cycle was the work initiated by the Air Force's Rome Laboratory [RL92] (see updates in this reference). For their model, they developed predictions of fault density which they could then transform into other reliability measures such as failure rates. To do this the researchers selected a number of factors that they felt could be related to fault density at the earlier phases. Included in the list were:

A *Application type* (e.g., real-time control systems, scientific, information management).

D *Development environment* (characterized by development methodology and available tools). The types of development environments considered are organic, semidetached, and embedded modes.

Requirements and design representation metrics

SA Anomaly management
ST Traceability
SQ Incorporation of quality review results into the software

Software implementation metrics

SL Language type (assembly, high-order, etc.)
SS Program size
SM Modularity
SU Extent of reuse
SX Complexity
SR Incorporation of standards review results into the software

The initial fault density prediction is then:

$$\delta_0 = A * D * (SA * ST * SQ) * (SL * SS * SM * SU * SX * SR)$$

Once the initial fault density has been found, a prediction of the initial failure rate is made as [Musa87]

$$\lambda_0 = F * K * (\delta_0 * \text{number of lines of source code}) = F * K * W_0$$

The number of inherent faults = W_0 = (δ_0 * number of lines of source code); F is the linear execution frequency of the program; and K is the fault expose ratio ($1.4 * 10^{-7} \leq K \leq 10.6 * 10^{-7}$). By letting $F = R/I$, where R is the average instruction rate and I is the number of object instructions in the program, and then further rewriting I as $I_s * Q_x$, where I_s is the number of source instructions and Q_x is the code expansion ratio (the ratio of machine instructions to source instructions—an average value of 4 is indicated), the initial failure rate can be expressed as

$$\lambda_0 = \left(R * \frac{K}{Q_X}\right) * \left(\frac{W_0}{I_s}\right)$$

3.9 Summary

This chapter has presented an overview of the classes of models that have appeared most often in the literature on software reliability modeling. We clearly have not covered the myriad of models that have appeared to date—to do so would require a large book unto itself. We have tried to give an overview of the important classes of these models

along with important examples within those classes. You should not be dismayed by the number of classes and the associated models. This should drive home two important points. First, that the field has matured to the point that it can be applied in practical situations and give meaningful results and, second, that there is no one model that is best in all situations. We firmly believe that to successfully apply software reliability modeling, you need to select the model that is most appropriate for the data set and the environment in which the data were collected.

We also must warn you to exercise care in applying the results of this chapter. As can be seen from the results presented, the mathematics can be quite complex. This cautionary statement is not made to discourage the application of these results, but to warn you that the results cannot be blindly applied. You don't need a detailed understanding of the mathematical derivations to apply it. You must, however, have an understanding of what software reliability models can and cannot do. A little time spent in learning this can be the difference between a successful program and one that isn't. Data collection, quality, and monitoring, as well as what has been emphasized for model selection and validation are all essential if these models are to be successfully applied.

This field will continue to evolve as newer models are developed. As this occurs, new ideas will no doubt emerge to improve the prediction and estimation process. This is especially true for software reliability prediction in the earlier phases of the life cycle (e.g., the requirements and the design phase).

Many challenges yet await the researcher and practitioner: distributed and parallel-based software systems, high-integrity software systems, and fault tolerance are just a few. The field of software engineering is continuously evolving. The software development process is changing, which, in turn, will present new problems for the software reliability engineer. We have made and will continue to make great strides in providing quantitative answers to the question, "Just how good is the software?"

Problems

3.1 Show for a Poisson process that the conditional hazard rate function, $z(t \mid t_i)$, with t_i being the time of the ith failure, and the failure intensity function, $\lambda(t)$ are the same if the failure intensity function is evaluated at the current time $t_i + t$ with $t \geq 0$, that is,

$$z(t \mid t_i) = \lambda(t_i + t) \qquad \text{for } t \geq 0$$

For this case, what can we then conclude about the relationship between the mean value function, $\mu(t)$, and the hazard rate?

3.2 Using the expressions for the mean value and failure intensity functions for the Musa basic execution model of Sec. 3.3, show that the reliability function and the hazard rate function after $(i-1)$ failures have occurred are

$$R(\Delta t \mid t_{i-1}) = \exp(-[\beta_0 \exp(-\beta_1 t_{i-1})][1 - \exp(-\beta_1 \Delta t)]) \qquad \text{for } \Delta t \geq 0,$$

and $$z(\Delta t \mid t_{i-1}) = \beta_0 \beta_1 \exp(-\beta_1 t_{i-1}) \exp(-\beta_1 \Delta t) \qquad \text{for } \Delta t \geq 0$$

3.3 Show for the Weibull model considered in Sec. 3.4 that the conditional hazard rate after $(i-1)$ failures have occurred is:

$$z(\Delta t \mid t_{i-1}) = (N - i + 1)\alpha\beta(t_{i-1} + \Delta t)^{\alpha - 1} \qquad \text{for } \Delta t \geq 0 \text{ and } 0 < \alpha < 1$$

What happens to this function as α approaches 1?

3.4 Derive the conditional reliability function and the hazard rate function after a failure has occurred at time t_i for the S-shaped model in Sec. 3.4. Derive also the expected number of faults to be detected in the ith interval of testing assuming it is of length l.

3.5 Derive the maximum likelihood estimates of α and β of the S-shaped model considered in Sec. 3.4. What are the resulting estimates for the data set considered in Example 3.2? Using the results from Prob. 3.4, what is the expected number of faults in the next testing period where all periods have been normalized to length 1?

3.6 Show that the mean value and failure rate function for the geometric model of Sec. 3.5 are

$$\mu(t) = \frac{1}{\beta}\ln([D\beta \exp(\beta)]t + 1) \qquad \text{and} \qquad \lambda(t) = \frac{D\exp(\beta)}{[D\beta\exp(\beta)] + 1}$$

where $\beta = -\ln(\phi)$ for $0 < \phi < 1$. (*Hint:* See Sec. 3.5.2.3.)

3.7 Using the formula for the expected time between failures and the actual times $(X_1, X_2, \ldots, X_n)$, derive the least squares estimates for D and β of the geometric model considered in Sec. 3.5. Using the data from Example 3.3, calculate the least squares estimates for D and β, and then the corresponding estimates for $\mu(t)$ and $\lambda(t)$ using the results from Prob. 3.6.

3.8 Show that the mean value and failure intensity function for the logarithmic Poisson in Sec. 3.5.3 are $\mu(t) = \ln(\lambda_0\theta t + 1)/\theta$ and $\lambda(t) = \lambda_0/(\lambda_0\theta t + 1)$, respectively.

3.9 For the Littlewood-Verrall model in Sec. 3.6, taking the conditional distribution of the time between failures, (x_i) to be exponential with parameter ξ_i, and the prior distribution to be a gamma distribution with parameters $\psi(i)$ and α, show that the marginal distribution of x_i is

$$f(x_i \mid \alpha, \psi(i)) = \frac{\alpha[\psi(i)]^{\alpha}}{[x_i + \psi(i)]^{\alpha+1}} \qquad \text{for } x_i > 0$$

3.10 Using the results from Prob. 3.9 and Sec. 3.6.1.3, show the resulting posterior distribution of the ξ_i's for the Littlewood-Verrall model is

$$h(\xi_1, \xi_2, \ldots, \xi_n) = \frac{\prod_{i=1}^{n} \xi_i^{\alpha} \exp\left(-\sum_{i=1}^{n} \xi_i(x_i + \psi(i))\right)}{[\Gamma(\alpha+1)]^n \prod_{i=1}^{n} (x_i + \psi(i))^{\alpha+1}} \qquad \text{for } \xi_i > 0, i = 1, \ldots, n$$