2.1 Survey

This chapter presents a brief survey of the development of research work in Software Reliability Growth Models (SRGMs) in chronological order. Twenty eight landmark research contributions have been selected for giving a thorough insight into the SRGMs problems and solution, which covered almost entire spectrum of the growth models. These historical information help me looking at a new problem in the field evaluate proposed solutions for their degree of promise. They are Hudson (1967); Jelinski and Moranda (1972) and Shooman (1972); Schneidewind (1972); Schick and Wolverton (1973); Crow (1974); Schneidewind (1975); Morand (1975); Musa (1975); Shooman and Natarajan (1976); Schick and Wolverton (1978); Goel and Okumoto (1978); Goel and Okumoto (1979b); Littlewood (1981); Keiller et al. (1983); Yamada et al. (1983); Musa and Okumoto (1983); Yamada and Osaki (1985); Yamada (1986); Trachtenberg (1990); Yamada et al. (1993); Zeephongsekul et al. (1994); Kuo et al. (2001); Bokhari et al. (2002); Malaiya et al. (2002); Teng and Pham (2002); Huang and Kuo (2002) and Huang et al. (2004).

Amongst them two main research work have been widely introduced main results with innovation. They are Yamada and Osaki (1985) and Yamada et al. (1993).

The first study of software reliability appears to have conducted by Hudson (1967). He viewed software development as a birth and death process (a type
of Markov process). Fault generation (through design changes, faults created in fixing other faults etc. was a birth, and fault correction was a death. The number of faults exiting at any time defined the state of the process. The transition probabilities related to the birth and death functions. He generally confined his work to pure death process, for reasons of mathematical tractability. He assumed that the rate of detection of fault was proportional to the number of fault remaining and a positive power of the time. In other words, the rate of fault detection was assumed to increase with time. It was shown that the number of faults detected follows a binomial distribution whose mean value as a function of time has the form of Weibull function. Data from the system test phase of one program were presented reasonable agreement between model and data is obtained if the system test phase is split into three overlapping sub phases and separate fits made for each.

The next major steps were made by Jelinski and Moranda (1972) and Shooman (1972). Both assumed a hazard rate for failures that mass piecewise constant and proportional to the number of faults remaining. The hazard changes at each fault correction by a constant amount, but is constant between corrections. Jelinski and Moranda applied maximum likelihood estimation to determine the total number of faults in the software and the constant of proportionality between number of faults remaining and hazard rate. Shooman postulated that the hazard rate was proportional to the fault density per instruction, the number of unique instructions executed per unit time, and cause failures. The faulty density is the difference between the inherent or original fault density and the faults is the difference between the inherent or original fault density and the faults corrected per instruction. The profile of the latter quality as a function of time was assumed to be related to
the project personnel profile in time. Several different fault correction profiles were proposed. The choice would depend on the particular project one was working with.

Schneidewind (1972) initially approached software reliability modeling from an empirical viewpoint. He recommended the investigation of different reliability functions and selection of the distribution that best fit the particular project in question. Suggested candidates were the exponential, normal, gamma and Weibull distributions. In looking at data, Schneidewind found that the best distribution varied from project to project. He indicated the importance of determining confidence intervals for the parameters estimated rather than just relying on point estimates.

Another early model was proposed by Schick and Wolverton (1973). The hazard rate assumed was proportional to the product of the number of faults remaining and the time. Hence the size of the changes in hazard rate (at fault correction) increases with time. Wolverton (1973) suggested a model in which the hazard rate was proportional to the number of faults remaining and a power of the time. This power could be varied to fit the data.

Crow (1974) proposed a model for the reliability estimation of hardware systems during development testing. It is non-homogenous Poisson process with a failure intensity function that is power function in time. It can be applied to software with certain ranges of parameter values.

In later Schneidewind (1975) viewed fault detections per time intervals as a non-homogenous Poisson process with an experiment mean value function. He applied either least square or maximum likelihood estimation to the determination of the parameters of the process. Schneidewind also suggested
that the time lag between failure detection and correction be determined from actual data and used to correct the time scale in forecasts.

Moranda (1975) has also proposed two variants of the Jelinski-Moranda model. In the “geometric de-entrophication process” the hazard rate decreases in steps that from a geometric progression (rather than being of constant amount). The second, called the “geometric poisson” model, has a hazard rate which also decreases in a geometric progression. However, the decrements occur at fixed intervals rather that of each failure correction.

Musa (1975) presented an execution time model of software reliability (referred to as the “basic execution time model”). This theory built on earlier contribution, but also broke new ground in several ways. He postulated that execution time, the actual processor time utilized in executing the program, was the best practical measure of the failure inducing stress that was being placed on the program. Hence, he concluded that software reliability theory should be based on execution time rather than calendar time. Calendar time does not account for varying usage of the program in either test or operation. The removal of this confounding factor greatly simplifies modeling. An execution time model is superior in ability to model the failure process simply, in conceptual insight, and in predication validity.

Schick and Wolverton (1978) also proposed a modified model. It can be shown that the hazard rate for this model is a parabolic function instead of a linear function in time. Although this hazard rate is close in form to that of the Weibull distribution, it is clearly different.

Goel and Okumoto (1978) developed a modification of the Jelinski-Moranda model for the case of imperfect debugging. It is based on a view of debugging as a Markov process, with appropriate transition probabilities
between states. Several useful quantities can be derived analytical with the mathematics remaining tractable. Keiller (1983) developed this idea further, including the possibility of introducing a new fault due to the repair activity. Goel and Okumoto (1979b), reasoning from assumptions similar to those of Jelinski and Moranda, described failure detection as a non-homogeneous poisson process (NHPP) with an exponentially decaying rate function. The cumulative number of failures detected and the distribution of the number of remaining failures are both found to be Poisson. Maximum likelihood estimation methods were developed for both the cases when failure data are given in terms of failure intervals and failures per intervals. A simple modification of the NHPP model was investigated by Yamada et al. (1983). Where the cumulative number of failures detected is described as an S-shaped curve.

The different fault model proposed by Littlewood (1981) may be viewed as a variant of the general Littlewood-Verral model. It is similar is viewing the hazard rate as a random variable and in using Bayesian inference. Reliability growth is modeled through two mechanisms. One is the number of faults remaining. The second is the variation of the per-fault hazard rate with time. The second source follows from the hypotheses that failures occur with different frequencies due to the variation in frequency with which different input states at the program are executed. The most frequently occurring faults are detected and corrected first Littlewood considers that uncertainties in the relative frequencies of execution of different input states than uncertainties in fault correction.

Killer et al. (1983) investigated a model similar to the Littlewood-Verral general model. It characterizes the randomness of the hazard rate with the
same distribution. However, it uses a different parameter of that distribution to express reliability change.

Yamada et al. (1983) investigates a stochastic model for a software error detection process in which the growth curves of the number of detected software errors for the observed data is S-shaped. The software error detection model is a non-homogeneous poisson process where the mean value function has an S-shaped growth curve. The model is applied an actual software error data. Statistical inference on the unknown parameters is discussed. The model fits the observed data better than other models.

Musa and Okumoto (1983), examination of the basic concepts under lying software reliability modeling and development of a classification scheme have helped to clarify and organize comparisons and to suggest possible new models.

Musa and Okumoto (1984b), the work of Musa and Okumoto (1983) led to the development pf the Musa Okumoto logarithmic poisson execution time model, which combines with high predictive validity.

Yamada and Osaki (1985) summarizes existing software reliability growth models (SRGMs) described by non-homogeneous poisson process. The SRGM’s are classified in terms of the software reliability growth index of the error detection rate per error. The maximum likelihood estimations based on the reliability evaluation. Using actual software error data observed by software testing, application examples of the existing SRGM’s are illustrated.

Yamada et al. (1986) develops realistic software reliability growth models incorporating the effect of testing effort. The software error detection
phenomenon in software testing is indicated by non-homogeneous poisson process. The software reliability assessment measures and the estimation methods of parameters are investigated. Testing effort expenditures are described by exponential and Rayleigh curves. Least squares estimators and maximum likelihood estimators are used for the reliability growth parameters. The software reliability data analyses use actual data. The software reliability growth models with testing effort can consider the relationship between the software reliability growth and the effect of testing effort. Thus, the proposed methods will enable us to evaluate software reliability more realistically.

Trachtenberg (1990) proposes that software failures rates are the product of the software average error “size” apparent error density, and workload. Models of these factors are developed that are consistent with the assumptions of classical software reliability models. The linear (Jelinski-Moranda, Shooman, Musa), geometric (Moranda, Ramamoorthy-Bastani) and Rayleigh (Schick-Wolverton) models are special cases of the general theory. Linear reliability models (Jelinski-Moranda, Shooman, Musa) result from assumptions that the average of remaining error and workload are constant, and its apparent error density equals its real error density. Geometric reliability models (Ramamoorthy-Bastani, Moranda) differ from linear models in assuming that the average error size decreases geometrically as errors are corrected, while the Rayleigh model (Schick-Wolverton) differs in assuming that the average size of remaining errors increases linearly with time. The theory shows that the abstract proportionality constants of classical models are composed of more fundamental and more intuitively meaningful factor, viz., the initial values
of averages of remaining errors, real error density, workload and error content. A general software reliability models permits reliability engineers to model diverse reliability factors found in software. This section shows how the assumed behavior of the reliability primitives (average-error size, error density, and workload) is modeled to accommodate this diversity.

Yamada et al. (1993) developed a software reliability growth model incorporating the amount of test effort expended during the software testing phase. The time dependent behavior of test-effort expenditures is described by a Weibull curve. Assuming that the error detection rate to the amount of test effort spent during the testing phase is proportional to the current error content, the model is formulated by testing by a non-homogeneous poisson process. Using the model, the method of data analysis for software reliability measurement is developed. This model is applied to the prediction of additional test effort expenditures to achieve the objectives number of errors detected by software testing and the determination of the optimum time to stop software testing for release.

Zeephongsekul et al. (1994) presents a software reliability growth model which incorporates the possibility of introducing new faults into a software system due to the imperfect debugging of the original faults in the system. The original faults manifest themselves as primary failures and are assumed to be distributed as a non-homogeneous poisson process. Imperfect debugging of each primary failure induces a secondary failure which is assumed to occur in a delayed sense from the occurrence time of the primary failure. The mean total number of failures, comprising the primary & secondary failures, is obtained. We also developed a cost model and
consider some optimal policies based on the model. Parameters are estimated using maximum likelihood.

Kuo et al. (2001) proposes a new software scheme for constructing software reliability growth models (SRGM) based on a non-homogeneous poisson process. The main focus is to provide an efficient parametric decomposition method for software reliability modeling, which considers both testing efforts and fault detection rates (FDR). In general, the software fault detection/removal mechanism depend on previously detected/removed faults and on how testing effort are used. From practical field studies, it is likely that we can estimate the testing efforts consumption pattern and predicts the trends of a FDR. A set of time-variable testing effort based FDR models were developed that have the inherent flexibility of capturing a wide range at possible fault detection trends: increasing, decreasing and constant. This scheme has a flexible structure and can model a wide spectrum of software development environments, considering various testing efforts. The paper describes the FDR, which can be obtained from historical records of previous releases or other similar software projects, and incorporates the related testing activities into this new modeling approach. The applicability of our model and the related parametric decomposition methods are demonstrated through general real data sets from various software projects. The evaluation results show that the proposed framework to incorporate testing efforts and FDR for SRGM has a fairly accurate prediction capability and it depicts the real-life situation more faithfully. This technique can be applied to wide range of software systems.

Malaiya et al. (2002) discuss software test-coverage measure quantity and the degree of thoroughness of testing. Tools are now available that measure
test-coverage is terms of blocks, branches, computation-use, predicate-uses, etc. that are covered, and reliability. An log-arithmetic exponential (LE) model is presented that relates testing effort to test coverage (block, branch, computation use, or predicate-use). The model is based on the hypothesis that the enumerable elements (like branches or blocks) for the coverage measure have various probabilities of being exercised; just like defects have various probabilities of being encountered. This model allows relating a test-coverage measure directly with defect-coverage. The model is fitted to 4 data-sets for programs with real defects. In the model, defect coverage can predict the time to next failure.

The LE model can eliminate variables like test-application strategy from consideration. It is suitable for a high reliability application where automatic (or manual) test generation is used to cover enumerates which have not yet been tested. The model is simply and easy explained, and thus can be suitable for industrial use. The LE model is based on time-based logarithmic software-reliability growth model. It considers that: at 100% coverage for a given enumerable, all defects might not yet have been found.

Teng and Pham (2002) presents a NHPP-based SRGM (software reliability growth model) for NVP (N-version programming) systems (NVP-SRGM) based on the NHPP (Non-homogeneous Poisson process). Although many papers have been devoted to modeling NVP-system reliability, most of them consider only the state reliability, i.e., they do not consider the reliability growth in NVP systems due to continuous removal of faults from software versions. The model is the first reliability growth model for NVP systems which considers the error-introduction rate and the error-removal efficiency. During testing and debugging, when a software fault is found a debugging
effort is devoted to remove this fault. Due to the high complexity of the software, this fault might not be successfully removed, and now fault might be introduced into the software. By applying a generalized NHPP model into NVP system, a new NVP SRGM is established in which the multi-version coincident failures are well modeled. A simplify software control logic for a water-reservoir control system illustrates how to apply this new software reliability model. The S-confidence bounds are provided for system-reliability estimation. This software reliability model can be used to evaluate the reliability and to predict the performance of NVP systems. More application is needed to validate fully the proposed of NVP-SRGM for quantifying the reliability of fault-tolerant software systems in a general industrial setting. As we first model of its kind in NVP reliability growth modeling, the proposed NVP SRGM can be used to overcome the shortcomings of the independent reliability model. It predicts the systems reliability more accurately than the independent model and can used to help determine when to stop testing, which is a key question in the testing and debugging phase of the NVP system-development life cycle.

Huang and Kuo (2002) investigates a SRGM (software reliability growth model) based on the NHPP (non-homogeneous Poisson process) which incorporates a logistic testing effort function. SRGM proposed in the literature consider the amount of testing effort spent on software testing which can be depicted an exponential curve, a Rayleigh curve, or a Weibull curve. However, it might not be appropriate to represent the consumption curve for testing effort by one of this curve in some software development environments. Therefore it show that a logistic testing effort function can be expressed as a software development/test effort curve and that it gives a
good predictive reliability based on real failure-data. Parameters are established, and experiments performed on actual test/debug data sets. Results from application to a real data set analyzed and compared with other existing models to show that the proposed model predicts better. In addition, an optimal software release policy for the model, based on cost reliability criteria is proposed.

Bokhari et al. (2002) presents the case where the time development behaviour of testing effort expenditure are described by Exponentiated-Weibull (EW) curves, software reliability growth models (SRGM) are developed incorporating the amount of testing-effort expenditure during the software testing phase. In this testing, the error detection phenomenon is modeled by Non-Homogeneous Poisson Process (NHPP). It is assumed that the error detection rate to the amount of testing effort spent during the testing phase is proportional to the current error content. For the model, the software reliability measures and the estimation methods of the parameters are investigated. It is shown that software reliability growth model previously obtained for exponential (Yamada et al. (1986)) and Weibull curves (Yamada et al. (1993)) become special cases at Exponential Weibull curves. Using this model, the method of data analysis for software reliability measurement with actual software failure data is developed.

Huang, Lo, Kuo and Lyu (2004) investigate an optimal resource allocation problem in modular software systems during testing phase. The main purpose is to minimize the cost of software development when the number of remaining faults and desired reliability objective are given. An elaborated optimization algorithm based on the Lagrange multiplier is proposed and numerical examples are illustrated. Besides, sensitivity analysis is also
conducted. We analyze the sensitivity of parameters of proposed software reliability growth models and show the results in details. In addition, we present the impact on the resource allocation problem if some parameters are either overestimated or underestimated. We can evaluate the optimal resource allocation problems for various conditions by examining the behaviour of the parameters with the most significant influence. The experiment results greatly help us to identify the contributions of each selected parameter and its weights. The proposed algorithm and method can facilitate the allocation of limited testing-resource efficiently and thus the desired reliability objective during software module testing can be better achieved.

2.2 Theory of Software Reliability Growth Modeling Based on NHPP

In this section Yamada and Osaki (1985) investigates the existing software reliability growth models (SRGMs) described by non-homogeneous Poisson process. The SRGM’s are classified in terms of the software reliability growth index of the error detection rate per error. The maximum-likelihood based on the SRGM’s are discussed for software reliability data analysis and software reliability evaluation.

This section presents the useful methods of software reliability analysis based on SRGM’s described by non-homogeneous Poisson processes (NHPP’s) [Ross (1983) and Asher and Feingold (1984)]. Sub-section 2.2.1 discusses the general concept on SRGM’s described by NHPP’s. In particular, the software reliability growth index of the error detection rate per error, which characterizes the software reliability growth process during the software testing phase, is defined, and the quantitative measures for software reliability evaluation are derived. Sub-section 2.2.2 summarizes the
existing SRGM’s and classifies them by using the error detection rate per error and sub-section 2.2.3 discusses the maximum-likelihood estimations for SRGM’s based on the observed software data.

2.2.1 Software Reliability Growth Model (SRGM)

A software failure is based as an unacceptable departure of program operation caused by a software error remaining in the system. The following usual assumptions in the area of software reliability growth modeling are introduced.

- A software system is subject to software failures at random times caused by software errors.
- Each time a software failure occurs, the software error which caused it is immediately removed, and no new errors are introduced.

The testing time such as the calendar time or the machine execution is generally used as the unit of error detection period for describing the time-dependent behavior of the cumulative number of errors detected by software testing. Let \( \{N(t), t \geq 0\} \) be counting process representing the cumulative number of errors (or failures) detected in the intervals \((0,t)\). Then, the expected value of \( N(t) \), called a mean value function of an NHPP, is defined by \( H(t) \).

An SRGM based on an NHPP can usually be formulated as:

\[
P_r\{N(t) = n\} = \frac{(H(t))^n}{n!} \exp[-H(t)], \quad t \geq 0 \quad (n=0,1,2,\ldots)
\]

where \( H(t) = \int_0^t h(x) dx \)

then \( h(t) \) is called an intensity function of an NHPP, which means the instantaneous error detection rate.
Defining \( H(\infty) \) as the expected cumulative number of errors to be eventually detected, i.e., the expected initial error content to be estimated, we can easily show that:

\[
\lim_{t \to \infty} \Pr\{N(t) = n\} = \frac{a^n}{n!} e^{-a} \quad (n=0, 1, 2, \ldots)
\]

This implies that \( N(t) \) obeys a Poisson distribution with mean \( a \) after the testing of infinity long duration. As a useful software reliability growth index, the error detection rate per error (per unit time) at testing time \( t \) is given by:

\[
d(t) = h(t)/[a - H(t)]
\]

We have the relationship between \( d(t) \) and \( H(t) \) as:

\[
H(t) = a \left[ 1 - \exp \left( - \int_0^t d(u) \, du \right) \right]
\]

The following definitions characterizing a software reliability growth aspect in software testing can be introduced Yamada and Osaki (1985).

**Definition 1:** \( H(t) \) is an increasing error detection rate (IEDR) (mean value) function if \( d(t) \) is non-decreasing in \( t, t \geq 0 \).

**Definition 2:** \( H(t) \) is a decreasing error detection rate (DEDR) (mean value) function if \( d(t) \) is non-increasing in \( t, t \geq 0 \).

**Definition 3:** \( H(t) \) is a constant error detection rate (CEDR) (mean value) function if \( d(t) \) is constant \( (t \geq 0) \).

A software reliability growth process characterized by the IEDR (DEDR) function indicates increasing (decreasing) test efficiency.

The following random variables are defined for deriving the quantitative measures for software testing evaluation:

\( \bar{N}(t) \): number of error remaining in the system at testing time \( t \), i.e.,

\( N(\infty) - N(t) \)
\( X_k \): time interval between (k-1) and \( k^{th} \) failures (k-1, 2, ..., n)

\( S_k \): the \( k^{th} \) failure occurrence time, i.e., \( \sum_{i=1}^{k} X_i \)

Then, the expected and variance of \( N(t) \) are given by

\[
\begin{align*}
N(t) &= \mathbb{E}[N(t)] \\
&= ae^{\int_0^t du} \\
&= \text{var}[N(t)]
\end{align*}
\]

The so-called software reliability is conditional survival probability of \( X_k \) given that \( S_{k-1} = t \), and is given by:

\[
R(x/t) = \Pr\{X_k > x / S_{k-1} = t\} = \exp \left[ a \left( \exp \left( - \int_0^t d(u)du \right) - \exp \left( - \int_0^t d(u)du \right) \right) \right]
\]

which is independent of \( k \). The software reliability presents the probability that a failure does not occur in \( (t, t+x] \)

### 2.2.2 Existing SRGM’s

A software reliability growth curve representing a relation between the time span of software testing and the cumulative number of detected errors is observed in a software error detection process during the software testing phase. There are two types of shapes for the observed software reliability growth curves: exponential and S-shaped software reliability growth curves are called the exponential and S-shaped SRGM’s, respectively. Several existing SRGM’s based on NHPP’s briefly summarized in the following.

Goel and Okumoto (1979) first proposed an SRGM based on an NHPP. This model is called the exponential SRGM, which describes a software failure
detection phenomenon. The mean value function showing an exponential
growth curve is given by:

\[ H(t) = m(t) = a[1 - \exp(-bt)], \quad b > 0 \tag{2.8} \]

Where \( b \) represents the error detection rate per error at an arbitrary testing
time. It is clear that \( m(t) \) is a CEDR function since:

\[ d(t) = d_{\alpha}(t) = b, \quad t \geq 0 \tag{2.9} \]

In contrast to the homogeneous error detection rate of (2.9) for the
exponential SRGM, the detestability of an error is considered to be non-
homogeneous over the testing period since the errors detected early in the
software testing are different from those detected later on. Then, Yamada
and Osaki (1984) proposed a non-homogeneous error detection rate model
on the assumption that there exist two types of errors: Type 1 (Type 2) errors
are easy (difficult) to detect. This NHPP model, called the modified
exponential SRGM, has a mean value function of:

\[ H(t) = m_{\alpha}(t) = a \sum_{i=1}^{3} p_{i} [1 - \exp(-b_{i} t)], \quad 0 < b_{2} < b_{1} < 1 \tag{2.10} \]

\[ \sum_{i=1}^{3} p_{i} = 1, \quad 0 < p_{i} < 1, \quad (i=1, 2) \tag{2.11} \]

Where

\( b_{i} \) is error detection rate per Type \( i \) error \( (i=1,2) \)

\( p_{i} \) is content proportion of Type \( i \) errors, i.e., \( p_{i} \) is the expected initial
error content of Type \( i \) errors \( (i=1,2) \)

For the error detection rate per error at testing time \( t \) given by:
\[ d(t) = d_p(t) = \sum_{i=1}^{2} \left[ \frac{P_i \exp(-b_i t)}{P_1 \exp(-b_1 t) + P_2 \exp(-b_2 t)} \right] b_i \]

It can be shown that \( m_p(t) \) is a DEDR function.

In a software error removal phenomenon it should be assumed that a testing process consists of not only a software failure detection process, but also a software error isolation process. Yamada et al. (1983) offered the delayed S-shaped SRGM for such an error detection process, in which the observed growth curve of the cumulative number of detected errors is S-shaped. This NHPP model has mean value function of

\[ H(t) = M(t) = a \left[ 1 - (1 + bt) \exp(-bt) \right], \quad b > 0 \]

which shows an S-shaped growth curve. The parameter \( b \) represents the failure detection rate (and the error isolation rate). It can be shown that \( M(t) \) is an IEDR function since

\[ d(t) = d_M(t) = b^2 t / (1 + bt) \]

is monotonically increasing in testing time \( t \).

Another S-shaped SRGM was proposed by Ohba (1984). The model is called the inflection S-shaped SRGM, which describes a software failure detection phenomenon with a manual dependence of detected errors. In the error detection process, the more failures we detect, the more undetected failures become detectable. This NHPP model has a mean value function of

\[ H(t) = I(t) = d \left[ 1 - \exp(-bt) \right] / \left[ 1 + c \exp(-bt) \right], \quad b > 0, \quad c > 0, \]

which shows an S-shaped growth curve. The parameters \( b \) and \( c \) represent the failure detection rate and the inflection factor, respectively. It can be shown that \( I(t) \) is an IEDR function since

\[ d(t) = d_I(t) = b t / \left[ 1 + c \exp(-bt) \right], \]
is monotonically increasing in testing time $t$.

Besides stochastic SRGM’s decided above, deterministic SRGM’s by fitting logistic and Gompertz growth curves, have been widely used to estimate the error content of software systems Yamada et al. (1983). In Japan, some computer manufactures and software houses actually apply the logistic and Gompertz growth curve models. The growth curves were originally developed to predict demand trend, economic growth, or future population. The expected cumulative numbers of errors detected up to testing time is given for the logistic growth curve model as

$$n_t(t) = k / [1 + m \cdot \exp(-pt)], \quad m>0, p>0, k>0,$$

and for the Gompertz growth curve model as

$$n_t(t) = k \cdot a^e^{-b \cdot t}, \quad 0<a<1, 0<b<1, k>0$$

where $k$, $p$, $m$, $a$ and $b$ are constant parameters to be estimated by regression analysis. The parameter $k$ in the both models is the expected initial error content of a software system.

### 2.2.3 Maximum-Likelihood Estimation

We assume that mean value function $H(t)$ includes $N$ model parameters $w_i (i = 1, 2, ..., N)$ as well as parameter $a$ where $w = (w_1, w_2, ..., w_N)$.

Suppose that the data set on $n$ failure occurrence times $s_k (k = 1, 2, ..., n; 0 \leq s_1 \leq s_2 \leq ... \leq s_n)$ is observed during the software testing phase where $s = (s_1, s_2, ..., s_n)$.

Then, the likelihood function for the $(N+1)$ unknown parameters $a$ and $w_i (i = 1, 2, ..., N)$ in the NHPP model with $H(t)$, given $s$, is given by

$$L(a, w/s) = \exp[-H(s_n)] \prod_{k=1}^{n} h(s_k).$$
Taking natural logarithm of the likelihood function yields
\[
I(a, w | s) = \sum_{i=1}^{n} \ln \frac{h(s_i)}{H(s_n)}.
\]

2.21

Then, the (N+1) maximum-likelihood estimates and \( w_i \) \( (i = 1, 2, \ldots, N) \) can be obtained by solving the likelihood equations.

\[
\frac{\partial I(a, w | s)}{\partial a} = \frac{\partial I(a, w | s)}{\partial w_i} = 0 \quad (i = 1, 2, \ldots, N).
\]

2.22

Suppose that the data set on the cumulative number of detected errors, \( y_k \), in a given interval \((0, t_j) \) \( (k = 1, 2, \ldots, n; 0 < t_1 < t_2 < \ldots < t_n) \) is observed where \( t = (t_1, t_2, \ldots, t_n) \) and \( y = (y_1, y_2, \ldots, y_n) \).

Then, the likelihood function for the (N+1) unknown parameters \( a \) and \( w_i \) \( (i = 1, 2, \ldots, N) \) in the NHPP model with \( H(t) \), given \( (t, y) \) is given by

\[
L(a, w | t, y) = \prod_{k=1}^{n} \frac{\{H(t_k) - H(t_{k-1})\}^{y_k - y_{k-1}}}{(y_k - y_{k-1})!} \exp[-\{H(t_k) - H(t_{k-1})\}],
\]

2.23

Where \( t_0 = 0 \) and \( y_0 = 0 \). Taking the natural logarithm of the likelihood function yields

\[
I(a, w | t, y) = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[H(t_k) - H(t_{k-1})] - H(t_n) - \sum_{i=1}^{n} \ln[y_i - y_{i-1}]!.
\]

2.24

Then, the (N+1) maximum-likelihood estimates \( \hat{a} \) and \( \hat{w}_i \) \( (i = 1, 2, \ldots, N) \) can be obtained by solving the likelihood equations:

\[
\frac{\partial I(a, w | t, y)}{\partial a} = \frac{\partial I(a, w | t, y)}{\partial w_i} = 0 \quad (i = 1, 2, \ldots, N).
\]

2.25

For the software error data \((t, y)\), the distribution of the estimated model parameters for large samples can be derived. That is, if the sample size \( n \) is sufficiently large, the maximum-likelihood estimates \( \hat{a} \) and \( \hat{w}_i \) \( (i = 1, 2, \ldots, N) \) follow an asymptotic joint normal distribution. Then, the true asymptotic covariance matrix \( \Sigma \) of the \( \hat{a} \) and \( \hat{w}_i \) is given by the inverse matrix of the true Fisher information matrix \( F \). The estimated \( \hat{\Sigma} \) is given by
which is an \( (N+1) \times (N+1) \) symmetric matrix.

The F has the elements given by

\[
E[-\partial^2 l(a, w | t, y) / \partial a^2],
\]

\[
E[-\partial^2 l(a, w | t, y) / \partial w_i^2], \quad (i = 1, 2, \ldots, N),
\]

\[
E[-\partial^2 l(a, w | t, y) / \partial a \partial w_i] \quad (= E[-\partial^2 l(a, w | t, y) / \partial w_i \partial a]) \quad (i = 1, 2, \ldots, N),
\]

and

\[
E[-\partial^2 l(a, w | t, y) / \partial w_i \partial w_j] \quad (i \neq j; \ i, j = 1, 2, \ldots, N),
\]

Using the \( (N+1) \) maximum likelihood estimators \( \hat{a} \) and \( \hat{w}_i \) \( (i = 1, 2, \ldots, N) \) together with their asymptotic properties, the approximate point and interval estimations of the quantitative measures for software reliability evaluation, such as \( n(t) \) of (2.6) and \( R(x|t) \) of (2.7), can be performed. Let

\[
f(a, w_1, w_2, \ldots, w_N) \] denote a function of the model parameters \( a \) and \( w_i \) \( (i = 1, 2, \ldots, N) \).

Then, the maximum-likelihood estimates \( \hat{f}(a, w_1, w_2, \ldots, w_N) \) of \( f(a, w_1, w_2, \ldots, w_N) \) is given by

\[
\hat{f}(a, w_1, w_2, \ldots, w_N) = \hat{f}(\hat{a}, \hat{w}_1, \hat{w}_2, \ldots, \hat{w}_N)
\]

For large samples, if \( f(a, w_1, w_2, \ldots, w_N) \) is continuously differentiable, then

\( \hat{f}(a, w_1, w_2, \ldots, w_N) \) is asymptotically normally distributed. The true asymptotic variance is given by
Thus, the 100\(\gamma\) percent confidence bounds of \(\hat{f}(a,w_1,w_2,...,w_n)\) are given by

\[
\hat{f}(a,w_1,w_2,...,w_n) \pm K\varphi \sqrt{\text{Var}\left[\hat{f}(a,w_1,w_2,...,w_n)\right]} \tag{2.29}
\]

Where \(K\varphi\) is the 100(1+\(\gamma\))/2 percent point of the standard normal distribution (see Nelson (1982)). Applying \(n,(t)\) and \(R(x|t)\) to \(f(a,w_1,w_2,...,w_n)\) in (2.29), we obtain the asymptotic confidence bounds.

### 2.3 Software Reliability Growth Model with a Weibull Test-effort Function

In this section Yamada et al. (1993) develop software reliability measurement during the testing phase is essential for examination the degree of quality or reliability of a developed software system. We develop software reliability model incorporating the amount of the effort expended during the software testing phase. The time-dependent behaviour of test-effort expenditures is described be a Weibull curve. Assuming that the error detection rate to the amount of test effort spent during the testing phase is proportional to the current error content, the model is formulated by a non-homogeneous poisson process. This model is applied to the prediction of additional test-effort expenditure achieve the objective number of errors.
detected by software testing and the determination of the optimum time to stop software testing for release.

Section 2.3.1 proposes the test-effort function described by the Weibull curve. A software reliability growth model with the Weibull test-effort function is discussed in section 2.3.2 where the quantitative measures for software reliability and the maximum likelihood estimators are provided. Subsections 2.3.3 & 2.3.4 present the prediction of additional test-effort expenditures to achieve the objective number of detected errors and the optimal software release time as applications of the model to software reliability management, respectively.

Assumptions
1. A software system is subject to failure at random time caused by errors remaining in the system.
2. Each time a failure occurs, the error which caused it is immediately removed, and no new errors are introduced.
3. Test effort is described by a Weibull curve.
4. The mean number of errors detected in the time interval \((t, t + \Delta t)\) to the current test-effort is proportional to the mean number of remaining errors.
5. The error detection phenomenon in software testing is modeled by an NHPP.

2.3.1 Test Effort Function

Resources such as time, money, and manpower are spent in developing software system that meet deadlines. Commonly, a Rayleigh curve has been used to estimate and predict the time-dependent behaviour of resources consumed testing appreciably affect software reliability. Approximately 40-
50 percent of the total amount of software development resources are spent in testing. Yamada et al. (1986) proposed software reliability growth models incorporating the amount of test-effort spent on software reliability assuming that both the test-effort during the testing and the software development effort can be described by the Rayleigh curve. They also assumed an experiment curve as an alternate to the Rayleigh curve. However, in many software testing situations it is difficult to describe the test-effort by an exponential of Rayleigh curve since actual test-effort data shows various patterns. We use a Weibull curve as the test-effort function to describe the test-effort patterns at testing time t (calendar time).

\[ w(t) = \alpha \beta t^m \exp[-\beta t^m], \quad \alpha > 0, \beta > 0, m > 0 \]  

where \( \alpha \) is the total amount of test-effort expenditures required by software testing, and \( \beta \) and \( m \) are scale and shape parameters respectively.

When \( m=1, m=2 \), exponential and Rayleigh test-effort functions are obtained, respectively. If \( m>1 \), the scale parameter is

\[ \beta = 1/\left[ \left( \frac{m}{m-1} \right) t_{w_{\text{max}}}^m \right] \]  

Notation

\( t_{w_{\text{max}}} \): time when the amount of test effort \( w(t) \), reaches a maximum.

The integral form of (2.30) is

\[ w(t) = \alpha \left[ 1 - \exp[-\beta t^m] \right], \quad 2.32 \]

Which represents the cumulative amount of test-effort in \((0,t]\). From (3), when \( t = t_q \)

\[ w(t_q) = 0.63\alpha \]

The \( \alpha, \beta, m \) in the Weibull test-effort function (2.30) or (2.32) can be estimated by the method of least-squares Draper &Smith (1981). The
estimates for \( \alpha, \beta, m \) are determined for the \( n \) observed data pairs in the form \((t_k, w_{k})\) \((k = 1, 2, \ldots, n)\). To apply the method of least-squares to the observed test-effort data is transformed

\[
\ln w(t) = \ln \alpha + \ln \beta + \ln m + (m - 1) \ln t - \beta t^n
\]

From (2.33), the least-squares estimates \( \hat{\alpha}, \hat{\beta}, \hat{m} \) can be obtained by minimizing \( S(\alpha, \beta, m) \)

\[
S(\alpha, \beta, m) = \sum \left( \ln w_{k} - \ln \alpha - \ln \beta - \ln (m - 1) \ln t_{k} + \beta t_{k}^{n} \right)^2
\]

2.3.2 Software Reliability Growth Modeling

2.3.2.1 Model Description

During the software development testing phase errors in the system are detected and corrected. A software failure is an unacceptable departure of program operation caused by a software error remaining in the system. Following the usual assumptions (6) and assumption (4), we obtain the following equations.

\[
\frac{dm(t)}{dt} / w(t) = r[a - m(t)]
\]

\[
m(t) = a(1 - \exp[-rW(t)]),
\]

\[
W(t) = \frac{1}{r} \ln \left[ \frac{a}{a - m(t)} \right]
\]

For stochastic modeling of software-error detection phenomenon, defining the mean value of \( N(t) \) based on an NHPP by \( m(t) \) in (2.36) yields a software reliability growth model incorporating the Weibull test-effort function under the assumptions of Goel et al, (1979) & Yamada (1991)

\[
\Pr\{N(t) = n\} = poim(n;m(t)), \quad n=0,1,2,\ldots
\]

The NHPP intensity function is
\[ \lambda(t) = a \cdot r \cdot w(t) \cdot \exp[-r W(t)] \]  

2.39

From (2.38) we show that the limiting distribution of \( N(t) \) is Poisson with mean.

\[ m(\infty) = a \cdot (1 - \exp[-r \alpha]) \]  

2.40

Eqn. (2.40) implies that even if a software system is tested during an infinity long duration, all errors in the system can not be detected Yamada et al. (1986).

Then mean number of undetectable errors is \( a \cdot \exp[-r \alpha] \).

### 2.3.2.2 Software Reliability Measures

Based on the NHPP model with \( m(t) \) in (2.36), two quantitative measures for software reliability assessment can be derived Goel et al. (1979) and Yamada (1991). The mean number of errors remaining in the system at testing time \( t \), its variance, and reliability are

\[ r(t) = a(\exp[-r W(t)] - \exp[-r W(\infty)]) \]

\[ = \text{var}\{\bar{N}(t)\} \]  

2.41

\[ R(x/t) = \exp[-a(\exp[-r W(t)] - \exp[-r W(t + x)])] \]  

2.42

### 2.3.2.3 Maximum Likelihood Estimations

The reliability growth parameters \( a \) and \( r \) in the NHPP model with \( m(t) \) in (2.36) can be estimated by the method of maximum-likelihood. Let the estimated parameters \( \hat{a}, \hat{r}, \hat{\mu} \) in the Weibull test-effort function in (2.30) or (2.32) have been obtained by the method of least-squares. The \( \hat{a} \) and \( \hat{r} \) are determined for the \( n \) observed data pairs \( (t_k, w_k) \ (k = 1, 2, \ldots, n) \). Then, the joint pmf, the log-likelihood function, for the unknown parameters \( a \) and \( r \) in the NHPP model with \( m(t) \) in (2.36), is
\[ \ln L = \sum_{i=1}^{n} (y_i - y_{i-1}).\ln a \]

\[ + \sum_{i=1}^{n} (y_i - y_{i-1}).\ln \exp([-rW(t_i)] - \exp[-rW(t_i)]) \]

\[ - a(1 - \exp[-rW(t_n)]) - \sum_{i=1}^{n} \ln((y_i - y_{i-1})!), \quad 2.43 \]

\[ t_0 = 0 \text{ and } y_0 \equiv 0. \]

The usual calculus methods for an interior maximum result in

\[ y_n = a f_n, \quad 2.44 \]

\[ a g_n = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(g_i - g_{i-1})}{(f_i - f_{i-1})}, \quad 2.45 \]

\[ f_i = 1 - \exp[-rW(t_i)], \quad 2.46 \]

\[ g_i = W(t_i)\exp[-rW(t_i)], \quad (k = 1, 2, ..., n). \]

which can be solved numerically.

If the sample size \( n \) of the observed data is sufficient large, the maximum-likelihood estimators \( \hat{a} \) and \( \hat{r} \) asymptotically follow a bivariate \( \mathcal{N} \) normal distribution.

\[ \begin{pmatrix} \hat{a} \\ \hat{r} \end{pmatrix} \sim \mathcal{BN}\left(\begin{pmatrix} a \\ r \end{pmatrix}, \Sigma\right), \quad (n \to \infty) \quad 2.47 \]

The \( \Sigma \) in the asymptotic properties of (2.47) is useful in qualifying the variability of the estimated parameters \( \hat{a} \) and \( \hat{r} \) Nelson (1982), and is the inverse of \( F \)

\[ F = \begin{bmatrix} E\left(-\frac{\partial^2 \ln L}{\partial a^2}\right) & E\left(-\frac{\partial^2 \ln L}{\partial a \partial r}\right) \\ E\left(-\frac{\partial^2 \ln L}{\partial a \partial r}\right) & E\left(-\frac{\partial^2 \ln L}{\partial r^2}\right) \end{bmatrix} = \begin{bmatrix} f_n/a & g_n \\ a \sum_{i=1}^{n} (g_i - g_{i-1})^2/(f_i - f_{i-1})^2 \end{bmatrix} \quad 2.48 \]

Where,
\[ g_k = W(t)e^{-\lambda W(t)} \]  
\[ f_k = 1 - e^{-\lambda W(t)} \]

Substitute the value of \( a \) and \( r \) in (2.49) and calculate \( F^{-1} \). The estimated asymptotic variance-covariance matrix is:

\[ \Sigma = F^{-1} = \begin{pmatrix} \text{Var}(\hat{a}) & \text{Cov}(\hat{a}, \hat{r}) \\ \text{Cov}(\hat{a}, \hat{r}) & \text{Var}(\hat{r}) \end{pmatrix} \]

2.3.3 Prediction of Additional Test-Effort

To estimate the additional test-effort expenditures required to satisfy a specified reliability of software system which has been tested, i.e., to achieve an objective number of errors detected by software testing. Musa (1975) and Okumoto (1985) presented estimation methods for the additional software testing time required to achieve mean time to failure (MTTF) and failure intensity objectives, respectively.

If the objective number of detected errors is \( m_G \), then the cumulative test-effort required to reach the objective \( m_G \) can be obtained from (2.37) as

\[ W_G = \frac{1}{r} \ln \left[ \frac{\hat{a}}{\hat{a} - m_G} \right] \]

Thus, the additional test-effort to reach \( m_G \) is \( W_G - W_G \).

It is of intersect to obtain the estimate and its confidence intervals. Using \( \hat{a} \) and \( \hat{r} \), the \( \hat{W}_G \) is

\[ \hat{W}_G = \frac{1}{\hat{r}} \ln \left[ \frac{\hat{a}}{\hat{a} - m_G} \right] \]
Then, based on the asymptotic properties of $\hat{a}$ and $\hat{r}$ of (2.47), the asymptotic variance of $\hat{W}_G$ is estimated Nelson (1982)

$$Var[\hat{W}_G] = \left( \frac{\partial W_G}{\partial a} \right)^2 V \{\hat{a}\} + \left( \frac{\partial W_G}{\partial r} \right)^2 V \{\hat{r}\} + 2 \left( \frac{\partial W_G}{\partial a} \right) \left( \frac{\partial W_G}{\partial r} \right) Cov \{\hat{a}, \hat{r}\},$$

2.53

where $a = \hat{a}$ and $r = \hat{r}$ are used to evaluate (2.53). Since $W_G$ in (2.51) is continuously differentiable, the estimated $\hat{W}_G$ asymptotically follows a s-normal distribution with mean $W_G$ and variance $Var[\hat{W}_G]$ for large samples. Therefore, the approximate $\gamma$ s-confidence bounds of $\hat{W}_G$ are

$$W_G \pm K_\gamma \sqrt{Var[\hat{W}_G]}$$

2.54

2.3.4 Optimal Software-release Problem

It is important for the software project manager to determine an optimum time to stop software testing and to deliver the system to the users. This is called an optimal software release Koach et al. (1983), Okumoto et al. (1980) and Yamada et al. (1985). This can be formulated by considering the relationship between the attained reliability of the system and the testing resource expended. Using an evaluation criterion of the total mean software cost, the optimal software release problem is reduced to obtained the cost of test effort expenditures during testing phase and the costs of fixing errors before and after release are counted as software cost factors. The total software cost is

$$C(T) = C_1 m(T) + C_2 \{m(T_{IC}) - m(T)\} + C_3 \int_0^T w(x) dx$$

2.55

Differentiating (2.55) with respect to $T$ and equating it to zero yields

$$A(T) = C_3 / (C_2 - C_1)$$

2.56
The optimum release time \( T^* \) minimizes (2.55). The \( A(T) \) in (2.56) is monotonically decreasing in \( T \). If \( A(0) \leq C_3/(C_2-C_1) \), then \( A(T) < C_3/(C_2-C_1) \) for \( 0 < T < T_{lc} \). Then, \( T^* = 0 \) since \( \partial C(T)/\partial T > 0 \) for \( 0 < T < T_{lc} \). If \( A(0) > C_3/(C_2-C_1) > A(T_{lc}) \), there exists a finite and unique solution satisfying (2.56)

\[
T_0 = \left[ -\frac{1}{\beta} \ln \left( 1 - \frac{\ln[a(C_2-C_1)/C_3]}{\alpha} \right) \right]^{1/m}
\]

(2.57)

\[
A(T) > C_3/(C_2-C_1) \text{ for } 0 < T < T_0 \text{ and } A(T) = C_3/(C_2-C_1) \text{ for } 0 < T < T_{lc} \text{.}
\]

Thus, \( T_0 \) in (2.57) minimizes \( C(T) \), i.e., \( T^* = T_0 \) since \( \partial C(T)/\partial T < 0 \) for \( 0 < T < T_0 \) and \( \partial C(T)/\partial T > 0 \) for \( T_0 < T < T_{lc} \). If \( A(T_{lc}) \geq C_3/(C_2-C_1) \), then \( A(T) \geq C_3/(C_2-C_1) \) for \( 0 < T < T_{lc} \). Thus, \( T^* = T_{lc} \) since \( \partial C(T)/\partial T < 0 \) for \( 0 < T < T_{lc} \). These are summarized in theorem 1.

Theorem-1

TESTTRUE

CASE \( A(0) \leq C_3/(C_2-C_1); T^* = 0; \) ENDCASE

CASE \( C_3/(C_2-C_1) > A(T_{lc}); \) There exists a finite and unique \( T = T_0 \) and optimum release time \( T^* = T_0 \); ENDCASE

OTHERWISE: \( T^* = T_{lc} \). ENDTTEST.