2.1. SOFTWARE RELIABILITY GROWTH MODELS

The Software Reliability Growth Model (SRGM) is the tool, which can be used to evaluate the software quantitatively, develop test status, schedule status and monitor the changes in reliability performance. There have been many Software Reliability models developed in the last two decades. Most of these are based upon historical failure data collected during the testing phase. These models have been utilized to evaluate the quality of the software and for future reliability predictions. They have further been used in many management decision-making problems that occur during the testing phase. But none of these models can claim to be the best and hence there is a need for further research. In the following sections we describe each class of software reliability models with a representative reliability model.

2.1.1. SINGLE SYSTEM SOFTWARE

This model considers the whole software as a single monolithic system and the structure of the model is not considered in the process of reliability estimation of the software system. The reliability is solely estimated based on the failure history. Popular reliability estimation models include the Goel-Okumoto [13] model, and the Jelinski-Morandamodel [14]. Single system software failure behaviour is usually modelled using Software Reliability Growth Model and classified as times between failure and failure count models.
2.1.2. FAILURE RATE MODELS

Inter failure times are the main modelling parameter for these models. Generally it is expected that the time for next failure increase as more bugs are detected and corrected. This may not be always correct as inter failure times are random variables and subjected to statistical fluctuations. The failure rate is a modelling tool for type I models. The reliability function is a non-decreasing function of mission time. This demonstrates an increase in the software credibility.

2.1.3. MARKOVIAN MODELS

When a Markov process represents the failure process, the resultant model is called Markovian model. The software can attain many states at any particular time with respect to number of faults remaining or number of faults already removed. The transition between states depends on the current state of the software and the transition probability. The memory less property of the Markovian process implies that the time between failures follows an exponential distribution. Numerous attempts have been made to develop Markovian models; especially in earlier days due to similar theory in hardware reliability was already well developed. One of the popular reliability models developed by Jelinski and Moranda [14] is a Markov process model. Littlewood [15, 16, 17] proposed a model based on semi-Markovian process to describe the failure phenomenon of software with module structure. Cheung [18] has also proposed a Markovian model to describe module-structured software. Kremer [19] proposed birth-death

2.1.4. FAULT COUNTING MODELS

This category includes the models, which describe the failure phenomenon by stochastic processes like Homogeneous Poisson Process (HPP), Non-Homogeneous Poisson Process (NHPP), and Compound Poisson Process (CPP) etc. The majority of these failure count models are based upon the NHPP. Schneidewind [20] proposed a fault detection model based on NHPP. Different NHPP models are distinguished by their unique mean value functions. Yamada et al. [21] proposed the delayed S-Shaped model. Ohba [22] proposed the inflection S-Shaped model. Musa et al. [8] have proposed the basic execution time model and Log-Poisson model. Goel [7] modified Musa’s original model by introducing the test quality parameter. Yamada et al. [23] also proposed a discrete time model. The effect of testing effort on failure process was taken into consideration by Yamada et al. [24]. Kapur et al. [6] modified Goel-Okumoto [13] model by introducing the concept of imperfect debugging. Kareer et al. [25] proposed a model with two types of faults where each fault type is modelled as by an S-Shaped curve model. Xie and Zhao [26] have illustrated how the Schneidewind model can be modified to result in many of the above SRGMs.
Zeephongsekul et al. [27] proposed a model describing the case when a primary fault introduces secondary faults. Attempts as listed above for new models were made with the primary intention of getting flexible models that could describe a range of failure count curves or reliability growth curves like exponential curves and highly S-Shaped curves. Models with such property are termed as flexible SRGMs [6]. Recently efforts have been directed towards development of general SRGMs [6]. General SRGMs are flexible models and many of the above models can be derived from them.

Recently reliability modelling for distributed development environment has caught the attention of many researchers. Large software systems have modular design. A system is said to be modular when each activity of the system is performed by exactly one component, and when the inputs and outputs of each component are well defined [10]. Often different development teams develop such components of software separately. With availability of communication networks at cheaper rates some software components are developed at separate geographic locations also. Software developed under this distributed development environment has proved to be economical. Many times components from other software projects are also reused. Therefore SRGMs for software developed under distributed environment needs to have different approach. But very few attempts have been made in this regard [28].
2.1.5. MODELS BASED ON BAYESIAN ANALYSIS

In the previous two categories the unknown parameters of the models are estimated either by the least squares method or by the maximum likelihood method. But in this category of models, the Bayesian analysis technique is used to estimate the unknown parameters of the models. This technique facilitates the use of information obtained by developing similar software projects. Based on this information the parameter of given model are assumed to follow some distribution (known as priori distribution). Given the software test data and based on a priori distribution, a posterior distribution can be obtained which in turn describes the failure phenomenon. Littlewood and Verral proposed the first software reliability model based on Bayesian Analysis [29]. Singpurwalla [30, 31] have proposed a number of Bayesian software reliability models for different testing environments.

The following four categories of Software Reliability models do not make any dynamic assumptions of the failure process. These are defined very briefly here.

2.1.6. STATIC MODELS

These models use statistical techniques to evaluate software reliability and can be evaluated only if complete failure data is available. These models were developed in the initial stage of reliability model evolution and are now seldom used, as it cannot incorporate the structure of the software.
2.1.7. THE INPUT DOMAIN AND FAULT SEEDING MODELS

In fault seeding models, a known number of bugs are seeded (planted) in the program. The software is then tested for faults. The bugs detected would have a combination of inherent faults and seeded faults. The number of inherent faults in the software is calculated based on the inherent and seeded faults detected using maximum likelihood estimation and combinatorial. The drawback of this approach is that the seeded faults and the inherent faults must have the same detection probability. This is difficult to achieve. The basic approach in input domain based model is to generate a set of test cases from an input distribution. The reliability measure is calculated from the number of failures observed during symbolic or physical execution of sampled test cases. The test cases selected from the representative input space are executed recording the results. The probability of the success can be evaluated using statistical techniques. It is generally difficult to estimate the input distribution (operational profile); generally the input distribution is obtained based on the different paths that exist in the software.

2.1.8. SOFTWARE METRICS MODELS

The models in this class relate the fault content in the software to some features of the software program such as program length, complexity, volume etc. These models are empirically built and the result obtained by a model is dependent on the software development process environment, which may not be the same in
the other projects.

Models other than those falling under above categories exist in the literature and more are being proposed. A number of review papers have been written, to name among others Littlewood [16], Goel [7], Xie [9], Kapur et al. [6]. In the following section we briefly describe some Software Reliability Growth models based upon NHPP assumptions. Estimation procedures and Criteria for comparison among Software Reliability Growth Models are also discussed.

**2.1.9. MULTI COMPONENT MODELS**

White Box model or Multi component system considers the Software Architecture of the system, its interactions with the different modules. White box models are used to model component based software. Such models seek to explicitly incorporate the testing method used during the testing phase, as well as the structure of the software being tested. Modular systems have a general framework in which the system reliability is calculated. All modular systems are grouped as State Based, Path Based and Additive models.

**2.1.10. STATE BASED MODELS**

Architecture based models assume that components fail independently and that a component failure will ultimately lead to a system failure. Unlike hardware reliability every component is always in use, software components need a utilization factor in State Based Models. State Based Models are generally modelled using Markov models like CTMC (Continuous time Markov chain),
DTMC (Discrete time Markov chain) or semi Markov models. System reliability estimates are obtained using both architecture and failure model. This is achieved using two methods, Composite and Hierarchical solution approach.

2.1.11. COMPOSITE-HIERARCHICAL APPROACH

The State Based Models are further classified into composite and hierarchical based on the solution approach to obtain the reliability of the system. Composite method combines the architecture model with the failure model and then solved for reliability prediction. If the architecture model is first solved first and then superimposed on the failure behaviour on the architecture model solution to predict reliability.

2.1.12. PATH BASED MODELS

Similar to State Based Models, the Path Based Models consider software architecture with components and interfaces. Initially the different paths in system are obtained either experimentally or algorithmically. Path reliability is the product of all component reliabilities along the path. The system reliability is average of all the path reliabilities. State based models analytically account for the infinite loops in a path but path-based models terminate the loop to one or to an average execution time of the path. Shooman Model [32] considers reliability of modular software introducing the path based approach by using the frequencies with which different paths are run. Krishnamurthy and Mathur [33] developed a method to combine architecture and failure process by estimating
the path reliabilities based on the sequence of components executed for a single test run and the average over all test runs to obtain the system reliability. State Based Models are an important category of models for modular systems.

2.1.13. ADDITIVE MODELS

Additive models consider software testing phase and each component reliability is modeled by NHPP. This implies system failure rate is also NHPP with cumulative number of failures and failure intensity functions that are the sums of corresponding function of each component. Additive model do not consider architecture of the software. Xie & Wohlin [34] developed an additive based architecture model. Semi Markov and Markov regenerative models attempt to relax the Markovian assumption of exponential failure and repair times description, in a restrictive manner. They are also exposed to the state space explosion problem. Discrete-event simulation on the other hand offers an attractive alternative to analytical models as it can capture a detailed system structure and facilitate the study of influence of various factors such as reliability growth, various repair policies. All software development teams have to answer an obvious question of when to stop testing and release the software, whether the software is single component or multi-component.

2.2. NHPP BASED SOFTWARE RELIABILITY GROWTH MODELS

Stochastic processes are used for the description of a system’s operation over time. There are two main types of stochastic processes: continuous and discrete.
Among discrete processes, counting processes in reliability engineering are widely used to describe the appearance of events in time (e.g., failures, number of perfect repairs, etc.). The simplest counting process is a Poisson process. The Poisson process plays a special role to many applications in reliability engineering.

As a general class of well-developed stochastic process model in reliability engineering, NHPP models have been successfully used in studying hardware reliability problems. They are especially useful to describe failure processes which possess certain trends such as reliability growth and deterioration. Therefore, an application of NHPP models to software reliability analysis is then easily implemented. The model provides the expected number of faults/failures at a given time.

two classes of discrete time models. One class describes an error detection process in which the expected number of errors detected per test case is geometrically decreasing while the other class is proportional to the current error content. Yamada, Ohtera and Narihisa [24] further proposed a testing-effort dependent model which assumes the testing-effort to follow either exponential, Weibull or Rayleigh distribution. Kapur, Garg and Kumar [6] proposed a discrete time model based on the concept that the testing phase has two different processes namely, fault isolation and fault removal. Kapur and Younes [38] further proposed a discrete time model based on the assumption that the software consists of \( n \) different types of faults and on each type of fault a different strategy is required to remove the cause of the failure due to that fault. Ohba [22] proposed the hyper-exponential model to describe the fault detection process in a module structured software. Khoshgoftaar [39] proposed the K-Stage Erlangian model. Kapur and Garg [40] modified Goel-Okumoto model by introducing the concept of imperfect debugging. Kareer, Grover and Kapur [25] proposed two types of faults models where each fault type is modelled by an S-shaped curve.

In the real life software development projects, the non-uniform testing is more popular and hence the S-shaped growth curve has been observed in many software development projects. The cause of S-shapedness has been attributed to different reasons. Ohba and Yamada [41] and Yamada, Ohba and Osaki [21] attributed it to time delay between the fault removal and the initial failure
observation which is result of the unskilledness of the testing team at the early stages of the test. Also, Ohba [22] attributed it to the mutual dependency between software faults. Yamada, Ohtera and Narthisa [24] described it to the non-uniform distribution of the testing-effort. Bittanti et al. [42] accrued it to the increased fault detection rate later in the testing phase. Kapur, Garg and Kumar [6] described it to the presence of different types of faults in a software system.

The earlier SRGMs were developed to fit an exponential reliability growth curve and they are known as exponential SRGMs such as Jelinski-Moranda model [14], basic execution time model [8] and Goel-Okumoto model [13]. Later, few SRGMs were developed taking into account causes of the S-shapedness [21, 22, 42, 43].

2.2.1. SUMMARY OF NHPP BASED SOFTWARE RELIABILITY GROWTH MODELS

A very large number of Continuous Time Models has been developed in the literature to monitor the fault removal process and measure and predict the reliability of the software systems. During testing phase, it has been observed that the relationship between the testing time and the corresponding number of faults removed is either exponential or S-shaped. There exists another category of models, classified as flexible models. These models can depict both exponential and S-shaped failure growth phenomenon depending upon the values of the parameters. The following are some of the well-established models.
1. Model due to [Goel&Okumuto, 1979] (purely exponential) [13]
3. Model due to [Ohba, 1984] (Flexible) [22]
4. Model due to [Bittanti at al., 1988] (Flexible) [42]
5. SRGM based on error generation (Obha and Chou 1989) [44]
6. SRGM with imperfect Debugging [kapur and Garg, 1990] [40]
7. SRGM for an Error Removal Phenomenon (Kapur and Garg 1992) [43]
8. Generalized SRGM due to [Kapur, Younes and Agarwala 1995](Model for faults of Different severity) [45]

Some of the general assumptions are as follows:
1. Software system is subject to failure during execution caused by faults remaining in the system.
2. Failure rate of the software is equally affected by faults remaining in the software.
3. The number of faults detected at any time instant is proportional to the remaining number of faults in the software.
4. On a failure, repair effort starts and fault causing the failure is removed with certainty.
5. All faults are mutually independent from failure detection point of view.
6. The proportionality of failure detection / fault isolation / fault removal is constant.
7. The fault detection / removal phenomenon is modelled by NHPP.

2.2.2. NOTATIONS

\[ m(t) : \text{ Expected number of faults identified in } (0,t] \]

\[ a, b : \text{ Constants, representing initial fault content and rate of Fault removal per remaining faults for software.} \]

\[ p, q : \text{ Proportionality constants} \]

The brief description of above mentioned models is given in subsequent section.

I. GOEL-OKUMUTO MODEL [13]

Following differential equation results from assumption-3

\[ \frac{d}{dt} m(t) = b[a - m(t)] \quad (2.1) \]

The above first order linear differential equation when solved with the initial condition \( m(0) = 0 \) gives the following mean value function for NHPP

\[ m(t) = a(1 - e^{-bt}) \quad (2.2) \]

The mean value function is exponential in nature and doesn't provide a good fit to the S-Shaped growth curves that generally occur in Software Reliability. But the model is popular due to its simplicity.

Now we briefly discuss below some S-Shaped SRGMs.

II. DELAYED S-SHAPED SRGM [21]

Fault detection in this model is assumed to be a two-phase process consisting of failure detection and its eventual removal by isolation. It takes into account
the time taken to isolate and remove a fault and so it is important that the data
to be used here should be that of fault isolation. It is further assumed that the
number of faults isolated at any time instant is proportional to the number of
faults remaining in the software.

Thus,

\[
\frac{d}{dt} m_f(t) = b[a - m_f(t)] \\
\frac{d}{dt} m(t) = b[m_f(t) - m(t)]
\]  

(2.3)  

(2.4)

\( m_f(t) \) is the expected number of failures in \((0,t]\). Solving (2.13) and (2.14), we
get the mean value function as

\[
m(t) = a\left\{1 - (1 + bt)e^{-bt}\right\}
\]  

(2.5)

Alternately the model can also be formulated as one stage process directly as
follows:

\[
\frac{d}{dt} m(t) = \left(\frac{b^2t}{1 + bt}\right)(a - m(t))
\]  

(2.6)

It is observed that \( \frac{b^2t}{1 + bt} \rightarrow b \) as \( b \rightarrow \infty \). This model was specifically developed
to account for lag in the failure observation and its subsequent removal. This kind
of derivation is peculiar to software reliability only.
III. INFLECTION S-SHAPED SRGM [22]

The model attributes S-Shaped-ness to the mutual dependency between software faults. Other than assumption-3 it is also assumed that the software contains two types of faults, namely mutually dependent and mutually independent. The mutually independent faults are those located on different execution paths of the software, therefore they are equally likely to be detected and removed. The mutually dependent faults are those faults located on the same execution path. According to the order of the software execution, some faults in the execution path will not be removed until their preceding faults are removed.

Let \( r \) denote the ratio of independent faults to the total number of faults in the software. This ratio is called the inflection parameter \((0 < r \leq 1)\). If all faults in the software system are mutually independent \((r = 1)\) then the faults are randomly removed and the growth curve is exponential. According to the assumptions of the model, the fault removal intensity per unit time can be written as:

\[
\frac{d}{dt} m(t) = b(t)[a - m(t)] \quad (2.7)
\]

\( b(t) \), the fault removal rate at time \( t \) is defined as

\[
b(t) = b\phi(t) \quad (2.8)
\]

Where, \( \phi(t) \) the inflection function is defined as

\[
\phi(t) = r + (1 - r) \frac{m(t)}{a}, \quad \phi(0) = 0 \text{ and } \phi(\infty) = 1 \quad (2.9)
\]
$b$ is the fault removal rate in the steady state. Solving (2.17) under the initial condition $m(0)=0$ we get

$$m(t) = a \left[ \frac{1-e^{-bt}}{1+r-e^{-bt}} \right]$$

(2.10)

If $r=1$, the model reduces to the Goel-Okumuto model [13]. For different values of $r$ different growth curves can be obtained and in that sense the model is flexible.

### IV. FLEXIBLE SRGM [42]

The model is based on the following differential equation:

$$\frac{d}{dt} m(t) = k(m)(a - m(t))$$

(2.11)

Where $k(m) = k_i + (k_f - k_i) \frac{m(t)}{a}$

(2.12)

Here $k_i$ and $k_f$ are initial and final values of Fault Exposure Coefficient. If $k_i = k_f$, then it reduces to Exponential model. If $k_f > k_i$; the failure growth curve takes S-shape. If $k_f$ is very small as compared to $k_i$ that it is almost equal to zero, the failure growth curve becomes flat at the end. The solution of equation (2.11) with initial condition $m(t = 0) = 0$ is:

$$m(t) = a \left[ \frac{1-e^{-k_f t}}{k_f - k_i \frac{e^{-k_f t}}{k_i}} \right]$$

(2.13)
For different values of $k_f$ and $k_i$, it describes different growth curves.

V. SRGM BASED ON ERROR GENERATION [44]

Obha and Chou [44] proposed first SRGM incorporating the effect of error generation. They assumed that a constant error generation rate $\alpha$ and rate of introduction of error is proportional to the failure intensity. The following differential equation describes the failure phenomenon of the model

$$m'(t) = b \left[ a - \alpha m(t) - m(t) \right]$$  \hspace{1cm} (2.14)

Solving the above equation (2.14) under the initial condition $m(0) = 0$ we get

$$m(t) = \frac{a}{1 - \alpha} \left( 1 - e^{-b(1-\alpha)t} \right)$$  \hspace{1cm} (2.15)

VI. SRGM WITH IMPERFECT DEBUGGING [40]

Imperfect debugging is the area of essential study in software reliability literature to mark the difference between the failure observation and fault removal processes. In real development scene, the number of failure observed can be less than or more than the number of error removed. Kapur and Garg [43] has discussed the first case in their error removal phenomenon flexible model which shows as the testing grows and testing team gain experiences, additional number of faults are removed without them causing any failure. But if the number of failure observed is more than the number of error removed then we are having the case of imperfect debugging. The concept of imperfect debugging was first
introduced by Goel [7]. He introduced the probability of imperfect debugging in Jelinski and Moranda [14].

Kapur and Garg [43] introduced the imperfect debugging in Goel and Okumoto [13]. They assumed that the FRR (Fault Reduction Rate) per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed by the time infinity is more than the initial fault content.

Let $p$ be the probability of removing a fault upon failure.

Then, 
\[
\frac{d}{dt} m_r(t) = bp\left[a - m_r(t)\right] \tag{2.16}
\]

On solving equation (2.16) with initial condition $m_r(0) = 0$, we get

\[
m_r(t) = a\left(1 - e^{-bpt}\right) \tag{2.17}
\]

The number of failures corresponding to removal of $m_r(t)$ errors is

\[
m_f(t) = \int_0^t b\left(a - m_r(x)\right)dx \tag{2.18}
\]

\[
m_f(t) = \frac{a}{p}\left(1 - e^{-bpt}\right) \tag{2.19}
\]

Here number of failures as the testing is continued for infinitely long time

\[
m_f(\infty) = \frac{a}{p}
\]

Which is greater than actual fault content ‘a’ (as $0 \leq p \leq 1$).
VII. SRGM FOR AN ERROR REMOVAL PHENOMENON (43)

This model is based upon the following additional assumption: On a failure observation, the fault removal phenomenon also removes proportion of remaining faults, without their causing any failure.

Based on the assumption the fault removal intensity per unit time can be written as

$$\frac{d}{dt} m(t) = p[a - m(t)] + q \frac{m(t)}{a} [a - m(t)]$$

(2.20)

Solving equation (2.20) with the usual initial condition, the expected number of faults detected in $[0,t]$ is given as

$$m(t) = a \left[ 1 - e^{-(p+q)t} \right]$$

$$\left[ 1 + \frac{q}{p} e^{-(p+q)t} \right]$$

(2.21)

This is similar to equations (2.10), though they have been derived under different assumptions.

VIII. GENERALIZED SRGM FOR FAULTS OF DIFFERENT SEVERITY [45]

So far it has been assumed that all the faults in the software are of the same type.

In reality, it has been observed that any software system contains different type of faults and each type of faults requires different strategy and different testing effort to remove it.
In this SRGM, it is assumed that the testing phase consists of three different processes, namely failure observation, fault isolation and fault removal. The time delay between the failure observation and subsequent removal is assumed to represent the severity of the fault. The more severe the fault, more the time delay. The faults are classified as simple, hard and complex. The fault is classified as simple if the time delay between failure observation, isolation and removal is negligible. If there is a time delay between failure observation and isolation, the fault is identified as a hard fault. If there is a time delay between failure observations, isolation and removal, the fault is classified as a complex fault. The total removal phenomenon is the superposition of these three processes.

Assuming \( a_1, a_2, a_3 \) to be number of simple, hard and complex faults in a software system \((a_1 + a_2 + a_3 = a)\), the simple fault removal process is modelled as the following:

\[
\frac{d}{dt} m(t) = b_1 [a_1 - m(t)] \tag{2.22}
\]

Solving, we get \( m(t) = a_1 (1 - e^{-b_1 t}) \) \( (2.23) \)

The hard fault removal process is modelled as a two-stage process

\[
\frac{d}{dt} m_{2f}(t) = b_2 \left( a_2 - m_{2f}(t) \right) \tag{2.24}
\]

\[
\frac{d}{dt} m_2(t) = b_2 \left( m_{2f}(t) - m_2(t) \right) \tag{2.25}
\]

Solving, we get \( m_2(t) = a_2 \left( 1 - (1 + b_2 t) e^{-b_2 t} \right) \) \( (2.26) \)
The Complex fault removal process is modelled as a three-stage process

\[
\frac{d}{dt} m_{3f}(t) = b_3 \left( a_3 - m_{3f}(t) \right) \tag{2.27}
\]

\[
\frac{d}{dt} m_{3i}(t) = b_3 \left( m_{3f}(t) - m_{3i}(t) \right) \tag{2.28}
\]

\[
\frac{d}{dt} m_{3}(t) = b_3 (m_{3i}(t) - m_3(t)) \tag{2.29}
\]

Solving, we get

\[
m_3(t) = a_3 \left(1 - (1 + b_3 t + \frac{b_3^2 t^2}{2})e^{-b_3 t} \right) \tag{2.30}
\]

The mean value function of the SRGM is given by

\[
m_r(t) = m_1(t) + m_2(t) + m_3(t) \tag{2.31}
\]

Assuming \(a_2 = p \cdot a, a_3 = q \cdot a\) and \(a_1 = (1 - p - q)a,\)

\[
m(t) = a[(1 - e^{-b_3 t})(1 - p - q) - p(1 + b_3 t)e^{-b_3 t} - q(1 + b_3 t + \frac{b_3^2 t^2}{2})e^{-b_3 t}] \tag{2.32}
\]

Assuming \(b_1 = b_2 = b_3 = b,\) we have

\[
m(t) = a[1 - e^{-bt} \left(1 + (p + q)bt + \frac{b^2 t^2}{2}\right)] \tag{2.33}
\]

The mean value function obtained in (2.33) can be generalized to include \(n\) different types of faults depending upon their severity. We may write

\[
m(t) = \sum a_i [1 - e^{-b_i t} \left(\sum_{j=0}^{i-1} \frac{(bt)^j}{j!}\right)] \tag{2.34}
\]
2.3. SRGM WITH TESTING COVERAGE FUNCTION [46]

Testing coverage is an important measure for both software developers and customers of software products. It can help software developer to evaluate the quality of the tested software and determine how much additional effort is needed to improve the reliability of the software. Testing coverage, on the other hand, can provide customers with a quantitative confidence criterion when they plan to buy or use the software products. In this thesis some new SRGMs with Testing Coverage Function have been proposed.

Pham et al. [46] have proposed an SRGM based upon NHPP as below:

\[
\frac{dm(t)}{dt} = \frac{c'(t)}{1 - c(t)}[a(t) - m(t)]
\]

(2.35)

c(t), testing coverage function, being non-negative and non-decreasing function of testing time t can be assumed as \( c(t) = 1 - (1 + bt)e^{-bt} \).

Assuming that faults can be introduced during the debugging phase with a constant fault introduction rate \( \alpha \). Therefore \( a(t) \) is a linear function of the testing time i.e. \( a(t) = a(1 + \alpha t) \). Solving, We get

\[
m(t) = a(1 + \alpha t - \frac{bt + 1}{e^{bt}}) - \frac{a\alpha(1 + bt)}{be^{bt+1}}[\ln(bt + 1) + \frac{a}{i+1}(1 + bt)^{i+1} - 1]
\]

(2.36)

2.4. CHANGE-POINT PROBLEM IN SOFTWARE RELIABILITY

In the software reliability growth phase, the software testing process in a sense, determines the nature of the failure data. There are many factors that affect
software testing. These factors are unlikely to be kept stable during the entire process of software testing, with the result that the underlying nature of the failure process is likely to experience major changes. The fault detection rate strongly depends on some parameters like skill of testing team, program size, software testability, defect density and resource allocation. The fault detection rate for the all the faults lying in the software differs on the basis of their severity. In most of the NHPP based Software Reliability Growth Models, the fault detection rate is assumed to be constant. During a software testing process, there is a possibility that the underlying fault detection rate is changed at some time moment called Change Point. This would result in a software failure intensity function either increasing or decreasing monotonically [47, 48]. The position of the Change Point can be judged by the graph of actual failure data. The work in this area started with Zhao [49] who introduced the Change Point analysis in Hardware and Software reliability. Shyur [50], Huang [47], Kapur et al. [51] also made their contributions in this area.

2.4.1. **GOEL-OKUMOTO MODEL USING CHANGE POINT [47]**

Let the parameter $\tau$ be the Change Point that is considered unknown and is to be estimated from the data. The fault detection rate function is defined as:

$$b(t) = \begin{cases} 
  b_1 & \text{when } 0 \leq t \leq \tau, \\
  b_2 & \text{when } t > \tau
\end{cases}$$  

(2.37)
Under the assumptions described above, the fault removal process can be described by the following differential equation:

\[
\frac{dm(t)}{dt} = b(t)(a - m(t)) \tag{2.38}
\]

\[
m(t) = \begin{cases} 
  a\left(1 - e^{-b_1t}\right) & \text{when } 0 \leq t \leq \tau \\
  a\left(1 - e^{-b_2(\tau - t)}\right) & \text{when } t > \tau 
\end{cases} \tag{2.39}
\]

2.5. FLEXIBLE GENERALISED SRGM FOR FAULTS OF DIFFERENT SEVERITY [52]

It is a fact that complex software system consist of different types of faults, and different types of faults need different treatment. Different types of faults are depicted by different types of curve. We also assume that the removal growth of type 1 fault which is simple in nature follows exponential curve. For other fault, which are more, severe in nature, we incorporate logistic learning during removal phenomenon and these faults are depicted by different types of S-shaped curves.

In the beginning, we assume that only three types of faults exist in software type 1, 2 and 3 (simple, hard and complex namely) and later, we extended our modelling to n types of fault.

Assuming \(a_1, a_2\) and \(a_3\) to be simple, hard and complex faults in a software system \((a_1 + a_2 + a_3 = a)\), the simple fault removal process is modelled by the following

\[
\frac{dm_1}{dt} = b_1\left(a - m_1(t)\right) \tag{2.40}
\]
Solving the equation \(2.40\) with initial condition \(m_1(0) = 0\), we get:

\[
m_1(t) = a_1(1 - \exp(-b_1t))
\]  

(2.41)

The hard faults removal process is modelled as a two stage process,

\[
\frac{dm_{z_f}}{dt} = b_2\left(a_2 - m_{z_f}(t)\right)
\]  

(2.42)

\[
\frac{dm_z}{dt} = \frac{b_2}{1 + \beta_2 \exp(-b_2t)}\left(m_{z_f}(t) - m_z(t)\right)
\]  

(2.43)

Here we assume that the learning phenomenon of removal team grows as testing progresses and follows logistic removal rate.

Solving equation \(2.42\) and \(2.43\) with initial condition \(m_{z_f}(0) = 0, m_z(0) = 0\), 

we get: 
\[
m_z(t) = a_2 \frac{1 - (1 + b_2t) \exp(-b_2t)}{(1 + \beta_2 \exp(-b_2t))}
\]  

(2.44)

Here \(m_{z_f}(t)\) represents the number of failure observed in time \(t\) whereas \(m_z(t)\) represents the number of faults removed in time \(t\).

The complex fault removal process is modelled as a three-stage process,

\[
\frac{dm_{z_f}}{dt} = b_3\left(a_3 - m_{z_f}(t)\right)
\]  

(2.45)

\[
\frac{dm_{z_s}}{dt} = b_3\left(a_3 - m_{z_s}(t)\right)
\]  

(2.46)

\[
\frac{dm_s}{dt} = \frac{b_3}{1 + \beta_3 \exp(-b_3t)}\left(m_{z_s}(t) - m_s(t)\right)
\]  

(2.47)
Here we assume that the learning phenomenon of removal team grows as testing progresses and follows logistic removal rate.

Solving equation (2.45) (2.46) and (2.47) with initial conditions

\[ m_{3_f}(0) = 0, m_{3_a}(0) = 0 \text{ and } m_3(0) = 0, \]

we get:

\[ m_3(t) = a_3 \frac{1 - \left(1 + b_3 t + \frac{b_3^2 t^2}{2}\right) \exp(-b_3 t)}{(1 + b_3 \exp(-b_3 t))} \]  

(2.48)

It may be noted that \( m_2(t) \) and \( m_3(t) \) are expressed by delayed S-shaped and 3-stage Erlang growth curves with logistic removal rates. The removal rates per remaining fault for simple, hard and complex faults are given by

\[
\begin{align*}
    b_{1_i}, d_2 &= b_2 \left[ \frac{1}{(1 + \beta_2 \exp(-b_2 t))} - \frac{1}{(1 + \beta_2 + b_2 t)} \right] \quad \text{and} \\
    d_3 &= b_3 \left[ \frac{1}{(1 + \beta_3 \exp(-b_3 t))} - \frac{(1 + b_3 t)}{(1 + \beta_3 + b_3 t + b_3^2 t^2)} \right] \quad \text{Respectively.}
\end{align*}
\]

Note that \( d_2(t) \) and \( d_3(t) \) increases monotonically with \( t \) and tend to \( b_2 \) and \( b_3 \) respectively, as \( t \to \infty \). Thus, in steady-state delayed S-shaped and 3-stage Erlang with logistic removal rate of growth curves behave similar to the exponential growth curves and hence there is no loss of generality in assuming steady state rates \( b_2 \) and \( b_3 \) equal to \( b_1 \). He rates for three types of fault become \( b_1 \).

We also note that
$b_1 > b_i \left[ \frac{1}{1 + \beta_2 \exp(-b_2 t)} - \frac{1}{1 + \beta_2 + b_2 t} \right]$

$$b_i \left[ \frac{1}{1 + \beta_3 \exp(-b_3 t)} - \frac{(1 + b_3 t)}{1 + \beta_3 + b_3 t + b_3^2 t^2} \right]$$

This is in accordance with the severity of faults.

The mean value function of the proposed SRGM is

$$m(t) = m_1(t) + m_2(t) + m_3(t)$$

Where, $m_1(t), m_2(t)$ and $m_3(t)$ are the mean value functions of simple, hard and complex faults respectively in the time $[0,t]$.

$$m(t) = a_1 (1 - \exp(-b_1 t)) + a_2 \left( \frac{1 - (1 + b_2 t) \exp(-b_2 t)}{1 + \beta_2 \exp(-b_2 t)} \right) + a_3 \left( \frac{1 - \left( 1 + b_3 t + \frac{b_3^2 t^2}{2} \right) \exp(-b_3 t)}{1 + \beta_3 \exp(-b_3 t)} \right)$$

(2.49)

Assuming $b_1 = b_2 = b_3 = b$ we have

$$m(t) = a_1 (1 - \exp(-bt)) + a_2 \left( \frac{1 - (1 + bt) \exp(-bt)}{1 + \beta_2 \exp(-bt)} \right) + a_3 \left( \frac{1 - \left( 1 + bt + \frac{b^2 t^2}{2} \right) \exp(-bt)}{1 + \beta_3 \exp(-bt)} \right)$$

(2.50)

Assuming $a_1 = a p_1, a_2 = a p_2$ and $a_3 = a (1 - p_1 - p_2)$
The mean value function obtained in equation can be generalised to \( n \) different types of faults depending upon their severity, we may write

\[
m(t) = a_1 \left(1 - \exp(-bt)\right) + a_2 \frac{\left(1-(1+bt)\exp(-bt)\right)}{(1 + \beta_2 \exp(-bt))} + a(1-p_1 - p_2) \frac{\left(1-(1+bt+b^2t^2)\exp(-bt)\right)}{(1 + \beta \exp(-bt))}
\]

(2.51)

Above model determines the type of faults present in software with logistic removal rate and is abbreviated as GE-\( n \) (Logistic). If \( \beta_i = 0 \), the above model reduces to equation (2.41)

The analysis followed in equation (2.50) can again be applied. Therefore assume \( b_1 = b_2 = b_3 = \ldots \ldots \ldots \ldots b_n = b \). We also assume that the value of \( \beta \) remains same for different types of fault form estimation view. Accordingly equation can be rewritten as

\[
m(t) = \sum_{i=1}^{n} m_i(t) = a_1 \left(1 - \exp(-bt)\right) + \sum_{i=2}^{n} \frac{a_i}{(1 + \beta_i \exp(-bt))} \left[1 - \exp(-bt) \frac{\sum_{j=0}^{i-1} (bt)^j}{j!}\right]
\]

(2.52)

(2.53)

2.6. SOME DISCRETE TIME MODELS

A very few number of Discrete Time Models has been developed in the literature. The following are some of the well-documented models: models due
to [23] (purely exponential in nature) and [38, 53]) (purely S-Shaped). These models are briefly discussed next.

2.6.1. ASSUMPTIONS

1. Failure observation / fault removal phenomenon follows NHPP with mean value function $m(n)$.

2. Software is subject to failures during execution caused by faults remaining in the software.

3. Each time a failure occurs, an immediate effort takes place to decide the cause of the failure in order to remove it.

4. The debugging process is perfect.

5. Failure rate of the software is equally affected by the remaining faults.

2.6.2. NOTATIONS

$a$ : Initial fault-content of the software.

$b$ : Fault removal rate per remaining fault per test case.

$m(n)$ : The expected mean number of faults removal by the $n^{th}$ test case.

$m_f(n)$ : The expected mean number of failures occurred by the $n^{th}$ test case.

$m_r(n)$ : The expected mean number of removals incurred by the $n^{th}$ test case

I. EXPONENTIAL MODEL [23]

Based on the mentioned assumptions, the expected cumulative number of faults removed between the $n^{th}$ and the $(n+1)^{th}$ test cases is proportional to the
number of faults remaining after the execution of the \( n^{th} \) test run, satisfies the following difference equation

\[
m(n + 1) - m(n) = b(a - m(n))
\]  
(2.54)

Solving the above difference equation using Probability Generating Function (PGF) given as \( P(z) = \sum_{0}^{\infty} z^n m(n) \) with initial condition \( m(n=0)=0 \), one can get the closed form solution as:

\[
m(n) = a(1 - (1 - b)^n)
\]  
(2.55)

The above mean value function is exponential in nature and does not provide a good fit to the S-Shaped growth curves that generally occur in Software Reliability. Next, we briefly discuss below an S-Shaped model.

**II. DELAYED S-SHAPED MODEL [53]**

In this model, the testing phase is assumed to have two different processes namely, fault isolation and fault detection processes. Accordingly, we have two difference equations

\[
m_f(n + 1) - m_f(n) = b(a - m_f(n))
\]  
(2.56)

\[
m(n + 1) - m(n) = b(m_f(n) - m(n))
\]  
(2.57)

Solving the difference equations (2.56) and (2.57) using PGF with initial conditions \( m_f(n=0)=0 \) and \( m(n=0)=0 \) respectively, one can get the closed form solution as:

\[
m(n) = a\left(1 - (1 + bn)(1 - b)^n\right)
\]  
(2.58)
Alternately the model can also be formulated as one-stage process directly as follows. \[ m(n+1) - m(n) = \frac{b^2 n(n+1)}{1+bn} (a-m(n)) \] (2.59)

It is observed that \( \frac{b^2 n(n+1)}{1+bn} \to b \) as \( n \to \infty \).

This model was specifically developed to account for lag in the failure observation and its subsequent removal. This kind of derivation is peculiar to software reliability only.

2.7. DISCRETE SRGMS FOR FAULT REMOVAL PHENOMENON

[54]

The test team can remove some additional faults in the software, without these faults causing any failure during the removal of identified faults, although this may involve some additional effort. A fault that is removed consequent to failure, is known as leading fault, whereas the additional fault removed, which may have caused failures in future, are known as dependent faults.

Under this assumption that while removing leading faults the testing team may remove some dependent faults, the difference equation for the model can be written as: \[ m_r(n+1) - m_r(n) = b(a-m_r(n)) + \frac{c}{a} m_r(n+1)(a-m_r(n)) \] (2.60)

Where \( b \) and \( c \) are the rates of leading and dependent fault detection, respectively. Solving the above equation by the method of PGF and initial condition \( m(n=0)=0 \), we have
\[ m_r(n) = a \left[ \frac{1 - \left(1 - (b + c)^n\right)}{1 + \frac{c}{b} \left(1 - (b + c)^n\right)} \right] \] (2.61)

2.8. SRGM WITH TESTING EFFORT [24]

The testing efforts (resources) that govern the pace of testing for almost all the software projects are [8]:

a) Manpower that includes

b) Failure identification personnel

c) Failure correction personnel.

d) Computer time.

These resources very closely interact during the system test phase. The function of the failure identification personnel is to run test cases and compare the test results against program requirement to establish the failures that have occurred. The failure correction personnel are the debuggers or developers available to repair the software. The Computer facility represents the computer(s) necessary for the failure identification personnel and failure correction personnel to do their tasks. Computer time is the measure that is used for allocating computer resources.

Various SRGMs have been proposed in the Software Reliability Engineering literature under different sets of assumptions. But most of them do not consider the consumption pattern of resources such as computer time and manpower during testing. More realistic SRGM can result if the reliability growth process
is studied with respect the consumption pattern of the testing efforts, measured by CPU hours, number of executed test cases, man-hours, etc. [8].

As the testing continues, more and more testing efforts are consumed. If testing time becomes quite large, testing effort spent also becomes quite large. The testing effort rate is proportional to the testing resources available. Ahmad et al. [55] proposed an inflection S-shaped Software Reliability Growth Model that incorporates exponentiated Weibull test effort function in both perfect and imperfect debugging environments. Rafi et al. [56] discussed module testing and resource allocation dependent release policy in imperfect debugging environment. Ahmada et al. [57] proposed an inflection S-shaped Software reliability Growth Model considering the log-logistic curve as testing effort function. Pachauri et al. [58] proposed Software reliability Growth Model that incorporates generalized modified Weibull testing effort function in imperfect debugging environment with constant and time varying fault detection rates. Madhu et al. [59] proposed a Software reliability Growth Model with imperfect debugging, Change points and a fault reduction factor. Though various types of testing effort function have been studied in the literature.

The most frequently used function to explain the testing effort are-

- Exponential
- Rayleigh
- Weibull
- Logistic
The first two can be derived from the assumption that, "the testing effort rate is proportional to the testing resources available".

\[ \frac{dW(t)}{dt} = v(t)[\alpha - W(t)] \]  

(2.62)

Where \( v(t) \) is the time dependent rate at which testing resources are consumed, with respect to remaining available resources. Solving equation (2.62) under the initial condition \( W(0)=0 \), we get:

\[ W(t) = \alpha \left[ 1 - \exp \left( \int_0^t v(x) \, dx \right) \right] \]  

(2.63)

**Case 1:** When \( v(t) = v \), a constant:

\[ W(t) = \alpha \left( 1 - e^{-vt} \right) \text{ Exponential} \]  

(2.64)

**Case 2:** If \( v(t) = vt \), we get Rayleigh type curve:

\[ W(t) = \alpha \left( 1 - e^{-v t^2 / 2} \right) \]  

(2.65)

**Case 3:** If \( v(t) = v lt^{l-1} \), we get Weibull function:

\[ W(t) = \alpha \left( 1 - e^{-v t^l} \right) \]  

(2.66)

Exponential and Rayleigh curves become special cases of the Weibull curve for \( l=1 \) and \( l=2 \) respectively.

**Case 4:** If we define

\[ \frac{dW(t)}{dt} = v \frac{W(t)}{\alpha} [\alpha - W(t)] \]  

(2.67)

On solving, the cumulative testing effort consumed in the interval \((0, t)\) is given by
W(t) = \frac{\alpha}{1 + le^{-\nu t}}, \quad \text{where} \quad W(0) = \frac{\alpha}{1 + l} \quad (2.68)

Where \(\alpha, \nu,\) and \(l\) are constants. This is the Logistic testing effort function.

### 2.8.1. GO MODEL USING CHANGE POINT AND TESTING EFFORT FUNCTION [47]

Let the parameter \(\tau\) be the Change Point that is considered unknown and is to be estimated from the data. The fault detection rate function is defined as:

\[
b(t) = \begin{cases} 
  b_1 & \text{when } 0 \leq t \leq \tau, \\
  b_2 & \text{when } t > \tau 
\end{cases}
\]

Under the assumptions described above, the fault removal process can be described by the following differential equation:

\[
\frac{d}{dt} \frac{m(t)}{w(t)} = b(t)(a - m(t)) \quad (2.70)
\]

Here \(w(t)\) is the current testing effort consumption at time \(t\).

\[
m(W(t)) = \begin{cases} 
  a\left(1 - e^{-b_1 w(t)}\right) & \text{when } 0 \leq t \leq \tau \\
  a\left(1 - e^{-b_1 w(\tau)} - b_2 w(\tau - t)\right) & \text{when } t > \tau 
\end{cases}
\]

### 2.9. SOFTWARE RELIABILITY GROWTH MODEL USING UNIFIED APPROACH

During the last three decades many SRGM have been proposed in the literature [6, 8, 60], which relates the number of failure (faults identified/corrected) and execution time. These SRGMs assumes diverse testing and debugging (T&D)
environments like distinction between failure and correction processes, learning of the testing personnel, possibility of imperfect debugging and fault generation, constant or monotonically increasing/decreasing fault detection rate (FDR) or randomness in the growth curve. These SRGMs have been applied successfully in many real life software projects but no SRGM can claim to be the best in general as the physical interpretation of the testing and debugging changes due to numerous factors e.g., design of the test cases, defect density, skills and efficiency of the testing team, availability of testing resources etc. The overabundance of SRGMs makes the model selection a tedious task. To reduce this difficulty unified modeling approach have been proposed by many researchers. These schemes have proved to be successful in obtaining several existing SRGMs by following single methodology and thus provide an insightful investigation for the study of general models without making many assumptions. The work in this area started as early as in 1980s with Shanti kumar [61] proposing a generalized birth process model. Gokhale and Trivedi [62] used Testing coverage function to present a unified framework and showed how NHPP based models can be represented by probability distribution functions of fault – detection times. Dohi et al. [63] proposed a unification method for NHPP models describing test input and program path searching times stochastically by an infinite server queuing theory. Inoue [64] applied infinite server queuing theory to the basic assumptions of delayed S-shaped SRGM [21] i.e. fault correction
phenomenon consists of successive failure observation and detection/correction processes and obtained several NHPP models describing fault correction as a two stage process.

Another unification methodology is based on a systematic study of Fault detection process (FDP) and Fault correction process (FCP) where FCPs are described by detection process with time delay. The idea of modelling FCP as a separate process following the FDP was first used by Schneidewind [20]. More general treatment of this concept is due to Xie et al. [26]. Who suggested modelling of Fault detection process as a NHPP based SRGM followed by Fault correction process as a delayed detection process with random time lag.

Unified modeling approach is one of the recent topics of research in software reliability. Two research directions for SRGM are usually considered: unification and parameter estimation of SRGM. In fact, a unified modeling framework comprising some typical reliability growth patterns should be developed for robust software reliability assessment. There are some, but only a few, model unification schemes in the literature. Langberg and Singpurwalla [65] first showed that several SRGM can be comprehensively described by adopting a Bayesian point of view. Miller [66] and Thompson [67] extended the Langberg and Singpurwalla’s idea and developed a generalized Order Statistic models (GOS). More precisely, they showed that almost all SRGM can be explained within the framework of GOS and record value and claim especially for the
NHPP models that the model selection problem is reduced to a simple selection problem of fault detection time distribution. Based on their result the mean value function in NHPP models that can be characterized by theoretical probability distribution function of fault detection time. Chen and Singpurwalla [68] proved that all SRMs as well as the NHPP models developed in the past literature can be unified by self-exciting point processes. Apart from the probabilistic approach, Huang et al. [69] explained the deterministic behavior of the NHPP models, namely mean value function of time by introducing several kinds of mean operation. Pham et al. [70] solved a generalized differential equation by which the mean value function in the NHPP model is governed and proposed an NHPP with a generalized mean value parameter.

2.10. PARAMETER ESTIMATION

To support the model applicability both the parameter estimation and model validation are the necessary aspects. The proposed models presented in the thesis are non-linear and presents extra problems in estimating the parameters. Technically, it is more difficult to find the solution for non-linear models using Least Square method and requires numerical algorithms to solve it. Statistical software packages such as SPSS helps to overcome this problem. SPSS is a Statistical Package for Social Sciences. It is a comprehensive and flexible statistical analysis and data management system. SPSS can take data from almost any type of file and use them to generate tabulated reports, charts, and plots of
distributions and trends, descriptive statistics, and conduct complex statistical analysis. SPSS Regression Models enables the user to apply more sophisticated models to the data using its wide range of nonlinear regression models. For the estimation of the parameters of the proposed models, Method of Least Square (Non Linear Regression method) has been used. Non Linear Regression is a method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables.

2.10.1. METHOD OF LEAST SQUARES

In this method the square of the difference between observed response and value predicted by the model is minimized. If the expected value of the response variable is given by \( \hat{m}(t) \) (can be a mean value function of an SRGM), then the least square estimators of the parameters of the model may be obtained from \( n \) pairs of sample values \((t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\) by minimizing \( J \) given by

\[
J = \sum_{i=1}^{n} \left[ y_i - \hat{m}(t) \right]^2
\]  

(2.72)

\( t_i \) and \( y_i \) observed values of explanatory and dependent variables respectively. For small and medium size samples least square estimation is preferred [8]. The method has also been applied for estimating parameter of testing effort curves.

2.11. COMPARISON CRITERIA FOR SRGMS

The performance of SRGMs are judged by their ability to fit the past software fault data (goodness of fit) and to predict satisfactorily the future behaviour of the software fault removal process (predictive validity) [6, 8, 11]
2.11.1. GOODNESS OF FIT CRITERIA

The term goodness of fit is used in two different contexts. In one context, it denotes the question if a sample of data came from a population with a specific distribution. In another context, it denotes the question of “How good does a mathematical model (for example a linear regression model) fit to the data”?

The Mean Square Fitting Error (MSE): The model under comparison is used to simulate the fault data, the difference between the expected values, \( \hat{m}(t_i) \) and the observed data \( y_i \) is measured by MSE as follows.

\[
MSE = \frac{1}{k} \sum_{i=1}^{k} (\hat{m}(t_i) - y_i)^2
\]  

(2.73)

Where \( k \) is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [6].

The Akaike Information Criterion (AIC): It is defined as \( AIC = -2(\text{The value of the maximum log likelihood function}) + 2(\text{The number of the parameters used in the model}) \)

This index takes into account both the statistical goodness of fit and the number of parameters that are estimated. Lower values of AIC indicate the preferred model, i.e. the one with the minimum number of parameters that still provides a good fit to the data [6, 71].

Coefficient of Multiple Determinations \( (R^2) \): This Goodness-of-fit measure can be used to investigate whether a significant trend exists in the observed
failure intensity. We define this coefficient as the ratio of the Sum of Squares (SS) resulting from the trend model to that from a constant model subtracted from 1, that is

\[ R^2 = 1 - \frac{\text{residual SS}}{\text{corrected SS}} \]

(2.74)

\( R^2 \) measures the percentage of the total variation about the mean accounted for by the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well [6, 71].

**Sum of Squared Errors (SSE):** The SSE is a mathematical approach to determining the scattering of data points; found by squaring the length between each data point and the line of best fit and then summing all of the squares. The sum of the squared errors, SSE, is defined as follows:

\[ \text{SSE} = \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 \]

(2.75)

Where:

\( Y_i \) is the actual observations time series

\( \hat{Y}_i \) is the estimated or forecasted time series

The less values of SSE indicate the best fit of model to the data set [6, 71].