Chapter 2
LITERATURE REVIEW

2.1 Introduction

Software Reliability is an important attribute of software quality, together with functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation. Software Reliability is hard to achieve, because the complexity of software tends to be high. For any system with a high degree of complexity, including software, it is difficult to reach a certain level of reliability. System developers tend to push complexity into the software layer, with the rapid growth of system size and ease of doing so by upgrading the software. While the complexity of software is inversely related to software reliability, it is directly related to other important factors in software quality, especially functionality, capability, etc. Emphasizing these features tend to add more complexity to software. Review of literature introduces various prior research work done in the domain of software reliability.

2.2 Reviews on Software Reliability

Software reliability growth can be monitored using software reliability growth models. In this paper authors have prototyped a software reliability modeling tool called CASRE, a Computer-Aided Software Reliability Estimation tool. Implemented with a systematic and comprehensive procedure as its framework, CASRE encourages more widespread use of software reliability modeling and measurement as a routine practice for software project management. The authors navigate through the CASRE tool to demonstrate its functionality and capability [11].

Sufficient work has been done in the area of software fault analysis, prediction and evaluation. At university of Virginia authors have developed Economic Reliability Analysis framework to the software reliability community and fuse it with the Musa/Okumoto NHPP software reliability model to estimate the economic impact of operational software faults. This framework defines a set of economic vectors that are calculated using the software reliability model along with a few additional financial elements. This extended software reliability model is then used to evaluate an
operational software system with three different proposed software upgrades. All the four software systems are analyzed and compared. This analysis provides a more complete method to apply software reliability modeling techniques to existing software systems and proposed design changes [12].

A system must function sufficiently reliably for its application, but it must also reach the market at the same time before its competitors and with a competitive cost. Some systems may be less market-driven than others, but balancing reliability, time of delivery, and cost is always important. One of the most effective ways to do this is to engineer the test process through quantitative planning and tracking. Unfortunately, most software testing is not engineered, and the resulting product may not be as reliable as it should be, and/or it may be too late or too expensive. Software reliability engineering combines the use of quantitative reliability objectives and operational profiles (profiles of system use). The operational profiles guide developers in testing more realistically, which makes it possible to track the reliability actually being achieved. ATandT has been a major user of SRE. Hence in this article, the Author describes SRE in the context of an actual project at ATandT, which he calls Fone Follower. The Author has selected this example because of its simplicity; it in no way implies that SRE is limited to telecommunications systems. SRE is based on the ATandT Best Current Practice of Software Reliability Engineering, approved in May 1991, Qualification as an ATandT best current practice requires use on typically eight to 10 projects with documented large benefit/cost ratios, as well as a probing review by two boards of high-level managers. The work done by Musa was experimented by 70 project managers in specification of their practice. But still very less standard practices are explored till now [15].

Most stringent restriction that is present in most software reliability models is the assumption of independence among successive software failures. In this research work authors have developed the software reliability modeling framework that can consider the phenomena of failure correlation and to study its effects on the software reliability measures. The important property of the developed Markov renewal modeling approach is its flexibility. It allows construction of the software reliability model in both discrete time and continuous time, and (depending on the goals) to base the analysis either on Markov chain theory or on renewal process theory. Thus, their modeling approach is an important step toward more consistent and realistic modeling
of software reliability. It can be related to existing software reliability growth models. Many input-domain and time-domain models can be derived as special cases under the assumption of failure-independence. This paper aims at showing that the classical software reliability theory can be extended to consider a sequence of possibly-dependent software runs, viz. failure correlation. It does not deal with inference or with predictions, per se. For the model to be fully specified and applied to estimations and predictions in real software development projects, the authors need to address many research issues, e.g., the

- Detailed assumptions about the nature of the overall reliability growth,
- Way modeling-parameters change as a result of the fault-removal attempts [19].

Criterion to determine a more appropriate software reliability growth model in the early development phase is very useful for reliability estimation. The criterion with a discrete software reliability growth model determines the absolute worth of a model because the discrete software reliability growth model perfectly reproduces the parameter estimates when the data are used as an exact solution of the equation. The values of the proposed criterion of a discrete software reliability growth model are smaller than those of software reliability growth model in all periods during the test phase with actual data sets. The discrete software reliability growth models are described with different equations which have exact solutions. The models yield accurate parameter estimates in spite of a small amount of input data in an actual software testing. Therefore, the criterion and discrete software reliability growth models enable to predict in the early development phase when software can be released [20].

A number of Markov-based software reliability models have been developed for measuring software reliability. However, the application of these models is strictly limited to software that satisfies the Markov properties. This model shows that it is possible to model software, using a Markovian approach, even though it does not fully satisfy the Markov property. By utilizing a grammar and its derivation rules, the authors group the executions of a set of components into a loop, preserving their execution patterns, yet avoid unlimited state expansion. Furthermore, to capture all the different execution processes, the structures of binomial trees are applied. The
model design by the author greatly increases the applicability of the Markov-based models by relaxing the constraints on the Markov properties. The Author is currently developing a tool that performs automatic state model construction when given a software system or software architecture, and computes software reliability using their architecture-based heterogeneous model. It can be expected that the completion of the tool will significantly promote the state of the art software reliability [24].

A lot has been said and published about the merits and limitations of reliability predictions as contrasted to reliability testing and assurance techniques from a product development standpoint. It also attempts to answer questions such as: are MIL-based reliability prediction methods useful? At what stages of the product development process? Which elements of the prediction can be practically used, and which should be discounted? How can the accuracy of reliability predictions be improved? Every method offers a certain benefit at a certain cost, is limited by a time element. No single answer exists in accurately predicting and demonstrating reliability. Balancing cost, benefit and time, the essential elements of a new product reliability and quality assurance program, provide a framework for selecting the methods. Specific, theoretical and practical examples are used to demonstrate the concepts and illustrate the methods that have been successfully used with encouraging results. In addition, useful interpretations of reliability predictions are presented, as it appears many popular misconceptions exist in the electronics industry [25].

Software reliability models are used for the estimation and prediction of software reliability. Selection of a software reliability model for use in a particular case has been an area of interest for researchers in the field of software reliability. Software reliability models are classified according to Software Development Life Cycle (SDLC) phases. The authors have identified and defined a number of criteria (with importance level) for software reliability model selection. In this work, the authors have proposed an algorithm based on these criteria for the selection of software reliability models along with an example. This algorithm can be used in different phases of SDLC and applies on various classes of software reliability models [26].

Software reliability assessment methods for concurrent distributed system development can be done by using the Analytic Hierarchy Process. Also, in this work authors make a comparison between the inflection S-shaped software reliability
growth model and the other models based on a non-homogeneous Poisson process applied to reliability assessment of the entire system composed of several software components. Moreover, authors analyze actual software fault count data to show numerical examples of software reliability assessment for the open source project. Furthermore, the authors investigate an efficient software reliability assessment method for the actual open source system development [30].

International petrochemical, chemical, refining and petroleum industries are trying to implement reliability programs to improve plant safety while trying to maintain plant availability. In this paper authors have addressed plant safety and availability thru the eye of reliability engineering and reliability data collection. The paper explains what reliability engineering is, how reliability models can be made and what kind of data needs to be collected. It demonstrated through practical examples how reliability data can be collected, what problems may arise and how plants can benefit from good reliability data [45].

2.3 Work with empirical approach

Software reliability engineering must develop beyond statistical analysis of data and analytic models which frequently require unrealistic assumptions. This work deals with the development of a viable discipline of simulation to aid experimental and industrial application of software reliability engineering. This will require developing standard modeling components, connections, tools, and a body of knowledge to interpret the phenomena that are being modeled. The authors explain how and why simulation models of software reliability can help to characterize the testing and debugging process more accurately. Examples are given to illustrate possible application of software reliability simulation [13].

Software reliability growth models (SRGMs) have been developed to estimate software reliability measures such as the number of remaining faults, software failure rate, and software reliability. The objective of this research work is to incorporate fault removal efficiency into software reliability assessment. Fault removal efficiency is a useful metric in software development practice and it helps developers to evaluate the debugging effectiveness and estimate the additional workload. In this paper, imperfect debugging is considered in the sense that new faults can be introduced into the software during debugging and the detected faults may not be removed completely. A
model is proposed to integrate fault removal efficiency, failure rate, and fault introduction rate into software reliability assessment. In addition to traditional reliability measures, the proposed model can provide some useful metrics to help the development team make better decisions. Software testing data collected from real applications are utilized to illustrate the proposed model for both the descriptive and predictive power. The expected number of residual faults and software failure rate are also presented [14].

Karama Kanoun et.al have presented a method which is based on the analysis and evaluation of software reliability by processing failure data collected on a software product during its development and operation. Traditionally, system reliability efforts have not focused on software reliability. The authors believe that failure prediction can be improved if software reliability modeling is integrated into an overall approach. The method authors propose is based on the combined use of descriptive analyses, trend analyses, and reliability models to control testing activities, evaluate software reliability, and plan maintenance [16].

Yu-Shen Su et.al have proposed an artificial neural network-based approach for software reliability estimation and modeling. The authors first explain the networks from the mathematical viewpoints of software reliability modeling. That is, the authors show how to apply neural network to predict software reliability by designing different elements of neural networks. Furthermore, the authors use the neural network approach to build a dynamic weighted combinational model. The applicability of proposed model is demonstrated through four real software failure data sets. From experimental results, authors can see that the proposed model significantly outperforms the traditional software reliability models [17].

Alan Wood [18] has collected defect occurrence times during system test and statistically correlated the test data with known mathematical functions, called software reliability growth models. If the correlation is good, then the function can be used to predict future failure rates, or the number of residual defects in the code. The Author found that the correlation with a simple exponential model was good and that this model can reasonably predict the number of residual defects in their delivered software.
Michael R. Lyu and Allen Nikora have described the construction of a CASE tool for a systematic and automatic application of software reliability modeling for real-world projects. Instead of proposing more new models, the authors focus on the practical project applications of existing software reliability models for better software reliability estimations. The authors build CASE tool on top of a number of existing software reliability models, called component models, as the baseline for software reliability measurement. The main advancement of this tool over other similar tools is highlighted by its capability in constructing various reliability estimations under a new paradigm to linearly combine the component models. Moreover, this tool features its enhanced graphical user-interface which greatly facilitates the potentially tedious application procedure for software reliability estimation [21].

With software being used in more critical applications, software reliability engineering (SRE) is an increasingly important field of software engineering. Collecting data for software reliability research, however, is difficult. Software projects do not always collect data that is complete or consistent enough for SRM research. In this paper author James M. Keables has developed a data collection package that can be used by people who may be interested in donating software reliability data. In addition, the Author is developing a simulation environment for use in software reliability research. The environment is, in essence, a laboratory which conducts software reliability experiments [22].

Chin-Yu-Haung et.al have proposed a generalized logistic testing-effort. That function relates work profile directly to the natural flow of software testing the actual consumption of resources during software testing which provides more accurate information for reliability modeling purpose. The authors also describe the effects of applying new tools and techniques for increased efficiency of software testing and studied the related optimal software release time problem from the cost-benefit viewpoint. New reliability problems are formulated to incorporate software testing effort and efficiency. Finally, numerical examples are provided to demonstrate these new approaches [23].

Chin-Yu Huang et.al have provided a simple but useful approach to measure and assess operational software reliability. Their approach of describing the working status of various software operational environments is very flexible as he can model
various environments ranging from an exponential NHPP to an S-shaped growth curve. Based on the integrated theoretical foundation, the technologies and approaches presented in this paper offer a consistent, quantitative software reliability evaluation scheme during both the testing phase and the operational phase [27].

Carol Smidts has provided Software reliability which has progressively become critical issues due to the number and the nature of the fields invaded. Software development companies need to be provided with the capability to improve their software development process and decrease its related costs. The Software Reliability Engineering Curriculum and in particular a graduate level software reliability modeling course at the University of Maryland was created to respond to this need. The contents of the course in the context of a particularly diverse environment are discussed [28].

Many studies have been performed on the subject of software reliability but few have explicitly considered the impact of software testing on the reliability process. Chin-Yu Huang and Michael R. Lyu have presented two important issues in software reliability modeling and software reliability economics: testing effort, and efficiency. First, the authors propose a generalized logistic testing-effort function that enjoys the advantage of relating work profile more directly to the natural flow of software development, and can be used to describe the possible testing-effort patterns. Furthermore, the authors incorporate the generalized logistic testing-effort function into software reliability modeling, and evaluate its fault-prediction capability through several numerical experiments based on real data. Secondly, the authors address the effects of new testing techniques or tools for increasing the efficiency of software testing. Based on the proposed software reliability model, the authors present a software cost model to reflect the effectiveness of introducing new technologies. Numerical examples and related data analyses are presented in detail. From the experimental results, the authors obtain a software economic policy which provides a comprehensive analysis of software based on cost and test efficiency. Moreover, the policy can also help project managers determine when to stop testing for market release at the right time [29].

Xiaolin Teng and Hoang Pham have presented a new methodology for predicting software reliability in the field environment. Their work differs from some existing
models that assume a constant failure detection rate for software testing and field operation environments, as this new methodology considers the random environmental effects on software reliability. Assuming that all the random effects of the field environments can be captured by a unit-free environmental factor, which is modeled as a random-distributed variable, the authors establish a generalized random field environment (RFE) software reliability model that covers both the testing phase and the operating phase in the software development cycle. Based on the generalized RFE model, two specific random field environmental reliability models are proposed for predicting software reliability in the field environment: the gamma-RFE model, and the alpha-RFE model. A set of software failure data from a telecommunication software application is used to illustrate the proposed models, both of which provide very good fittings to the software failures in both testing and operation environments. This new methodology provides a viable way to model the user environments, and further makes adjustments to the reliability prediction for similar software products. Based on the generalized software reliability model, further work may include the development of software cost models and the optimum software release policies under random field environments [32].

Reliable software should be correct, intact, unanimous and robust. However, software reliability could only be measured by software testing. If there are reliability faults in the completed software system, it would be too late to change the software design. Bing Chao et.al have proposed that the strict testing and managing of software reliability should be carried out at initial stage of software engineering - software requirements analysis (SRA). In this paper, the authors introduced two methods to implement SRM according with the different SRA models; one is based on structured analysis and the other is based on object-oriented analysis (OOA). By using evolutionary prototype, the software reliability quotas (SRQ) can be tested at the stage of requirements analysis. In conclusion authors give some advice about how to select evaluation model of software reliability [33].

IT (information technology) advanced with steady steps from 1970’s is essential in our daily life. As the results of the advances in high-speed data-transfer network technology, software development environment has been changing into new development paradigm. Yoshinobu Tamura and Shigeru Yamada have proposed a new approach to software reliability assessment by creating a fusion of neural
network and software reliability growth model. In this paper, the authors show application examples of software reliability assessment based on neural network and software reliability growth model for open source software. Also, the authors analyze actual software fault count data to show numerical examples of software reliability assessment for the open source software. Then, the authors consider the efficiency and effectiveness of the software reliability assessment method for the actual open source software [34].

Swapna S. Gokhale has provided a detailed discussion regarding the input parameters required by each model, and how these parameters may be estimated from the different software artifacts. Depending on the software artifacts that are available during a given phase of the software life cycle, and the parameters that can be estimated from these artifacts, author provides guidance regarding which model may be appropriate to predict the reliability of an application during each phase of its life cycle. The future research is focused along two dimensions: to enable the application of these models to real life software applications and different parameters of the models need to be estimated from various artifacts. The author is presently developing techniques for parameter estimation based on different software artifacts. As discussed earlier, hierarchical methods provide reliability (MTTF, and MNTF) predictions which are approximations to the reliability (MTTF, and MNTF) predictions produced by composite models. However, hierarchical methods enable a concise representation of application reliability in terms of the failure behavior of the components, and architectural behavior which is captured by the expected number of visits to each component. This concise representation facilitates sensitivity analysis to identify the impact of individual components on application reliability. To exploit the advantage offered by hierarchical methods, namely ease of sensitivity analysis, the accuracy of the reliability predictions obtained using the hierarchical methods needs to be improved. Developing methods to enhance the accuracy of the reliability predictions obtained using hierarchical methods is another area of future research [35].

It is very important to select and use appropriate software reliability metrics in software reliability engineering. This paper proposes a framework for selecting software reliability metrics based on analytic hierarchy process (AHP) and expert judgment. Selecting criteria and the metrics for selection are identified. In each development phase, the
grading of metrics according to every criterion are given by experts qualitatively, and then analyzed synthetically to calculate the weights of metrics using AHP. A preliminary application is practiced, and the metrics whose weights are top-ranked are recommended and analyzed. Sensitivity and consistency of this method are also analyzed. Compared with general selecting criteria, the method studied in this paper can be used to select appropriate metrics correctly, stably and systemically. Furthermore, the final selection results are accordant with engineering experience, and using the metrics recommended make software reliability evaluation more reliable and effective [36].

At present, most of software reliability models have to build on certain presuppositions about software fault process, which also brings on the incongruence of software reliability models application. Zhang Yongqiang and Chen Huashan have adopted Genetic Programming (GP) evolution algorithm to establish software reliability model based on mean time between failures’ (MTBF) time series. The evolution model of GP is then analyzed and appraised according to five characteristic criteria for some common-used software testing cases. Meanwhile, the authors also select some traditional probability models and the Neural Network Model to compare with the new GP model separately. The result testifies that the new model evolved by GP has the higher prediction precision and better applicability, which can improve the applicable inconsistency of software reliability modeling to some extent [37].

In general, the software-testing time may be measured by two kinds of time scales: calendar time and test-execution time. In this research work authors have developed two-dimensional software reliability models with two-time measures and incorporate both of them to assess the software reliability with higher accuracy. Since the resulting software reliability models are based on the familiar non-homogeneous Poisson processes with two-time scales, which are the natural extensions of one-dimensional models, it is possible to treat both the time data simultaneously and effectively. The authors investigate the dependence of test-execution time as a testing effort on the software reliability assessment, and validate quantitatively the software reliability models with two-time scales. The authors also consider an optimization problem when to stop the software testing in terms of two-time measurements [38].
Hiroyuki Okamura and Tadashi Dohi have developed EM (Expectation-Maximization) algorithms for the phase-type software reliability models with two types of fault data: fault-detection time data and grouped data with arbitrary time intervals. In numerical examples, the authors compare the EM algorithms with the quasi-Newton’s method and illustrate the effectiveness on the unified model and parameter estimation method [39].

Louis J. Gullo et.al have presented a reliability capability assessment process that can assist OEMs and system integrators in assessing prospective suppliers for their ability to design and manufacture reliable products before they are delivered for use, and on an ongoing basis, help a company in identifying shortcomings in its reliability program, which can be rectified by subsequent improvement actions. The assessment can also help to establish reliability management practices for use by designers, suppliers, customers, and independent authorities. The assessment method may be used to evaluate the reliability capability of all types of electronics-related industries that perform activities influencing the reliability of a product. It can produce increased customer satisfaction, provide competitive opportunities, and shorten the product development cycle. In summary, a reliability capability assessment can be used for:

- Specifying or planning reliability practices if product development is implemented internally;
- Evaluating reliability practices to determine the extent to which a supplier is capable of providing a product that meets the reliability requirements/needs; and
- Improving reliability practices if the current reliability practices have been evaluated and improvement is desired or required [40].

The reliability is one very important parameter of electronic devices, hardware, and applications software. Zuzana Krajcuskova has tried to explain basic principles of software reliability models and their practical usage in technical activities. The author has attempted to illustrate the basic principles of software reliability models, their practical use in technical activities. The author has also attempted to show the key assumptions on which software reliability models are based, specific assumptions related to each specific software reliability model and some examples of concrete software reliability models [41].
Software reliability is one of the important attributions of dependable systems. However, for the time being, the assumptions that software reliability measurement makes do not deal with some important factors that influence the accuracy of the software reliability models such as the complexity of the software and the effectiveness of test suite, resulting in inaccuracy or even incorrectness in evaluating the software reliability. The idea of the strategies presented in this paper is to incorporate the important factors related to the precision of software reliability measurement into reliability models, which mainly include complexity of software and the test effectiveness, making the software reliability models more adequate and accurate to the real measurement. Experimental results based on five programs selected have shown the feasibility and usefulness of the new approach [42].

It is necessary for the software system to work in an acceptable degree of reliability and quality. Stochastic modeling, queuing systems and network models, neural networks models, wavelet models, etc are some of interpretative methods to forecast the reliability of software system. But these approaches contain some limitations. Denghua Mei has presented theoretical concepts of grey system theory that has been studied widely, but techniques for the design of grey system model in software system reliability have still been in waiting for further investigation and development. The author discusses grey forecasting model in software system reliability in the paper [43].

Shiyi Xu has defined the strategies of aiming at making the reliability models more accurate proposed in the paper are of critical importance for the further study of dependable software, although it is just an initial result of research in this area. In this paper, author tries to incorporate the most important factors which influence the estimation of software reliability into the consideration of software reliability model, such as software complexity and test effectiveness, so that the reliability model will become relatively precise in the software’s characteristics, and making the model more accurate in quantity. For more convinced approach some experimental results are given to demonstrate the principles and the idea of proposed methods [44].

Generalized methods for software reliability growth modeling have been proposed so far. But, most of them are on continuous-time software reliability growth modeling. Many discrete software reliability growth models (SRGM) have been proposed to
describe a software reliability growth process depending on discrete testing time such as the number of days (or weeks); the number of executed test cases. Shinji Inoue and Shigeru Yamada have discussed generalized discrete software reliability growth modeling in which the software failure-occurrence times follow a discrete probability distribution. These generalized discrete SRGMs enable to assess software reliability in consideration of the effect of the program size, which is one of the influential factors related to the software reliability growth process. Specifically, the author develops discrete SRGMs in which the software failure-occurrence times follow geometric and discrete Rayleigh distributions, respectively. Moreover, authors derive software reliability assessment measures based on a unified framework for discrete software reliability growth modeling. Additionally, the authors also discuss optimal software release problems based on our generalized discrete software reliability growth modeling. Finally, the authors show numerical examples of software reliability assessment by using actual fault-counting data [46].

Michael R. Lyu has reviewed the history of software reliability engineering, the current trends and existing problems, and specific difficulties. Possible future directions and promising research problems in software reliability engineering have also been addressed. The author laid out the current and possible future trends for software reliability engineering in terms of meeting industry and customer needs. In particular, the author has identified new software reliability engineering paradigms by taking software architectures, testing techniques, and software failure manifestation mechanisms into consideration. Some thoughts on emerging software applications have also been provided [47].

Tomotaka Ishii and Tadashi Dohi have performed the goodness-of-fit test for several NHGP-based software reliability models (SRMs) and compare them with the existing NHPP-based ones. Throughout a numerical example with a real software fault data, it is shown that the NHGP-based SRMs can provide the better goodness-of-fit performances in earlier testing phases than the NHPP-based ones, but approach to them gradually as the testing time goes on [48].

Testing-time when a characteristic of a software failure-occurrence or fault-detection phenomenon is notably changed is ordinarily called change-point. Shinji Inoue and Shigeru Yamada have discussed a framework for software reliability growth modeling
with change-point as one of the solutions to incorporate the effect of the change-point into software reliability assessment, and also discuss its application to an optimal software release problem with change-point. That is one of the interesting issues for project management of software development. Finally, the authors show numerical examples of their model and derived software release policy by using actual data [49].

Aiming at the prediction precision and applicability problem for the traditional software reliability prediction models, from the point of nonlinear time sequence, Yunlong Teng et.al have presented a novel software reliability prediction model using RBF neural network based on empirical mode decomposition theory. In this paper, the fault data series obtained from software reliability test phase is decomposed into a series of intrinsic mode functions and a residue signal. Then a RBF network is constructed for an intrinsic mode function or the residual signal. Finally output of every prediction model is integrated into one output with equal weightage. Experimental results showed that the proposed model had higher precision of prediction and better applicability, compared with traditional software reliability models [50].

Shinji Inoue and Shigeru Yamada have discussed a parameter estimation method and a software reliability assessment measure of their model. Further, the authors show numerical examples for software reliability analysis by using actual data. Finally, they conduct goodness-of-fit evaluation of model to the actual data, and see that their model has a better performance than representative one-dimensional SRGMs in terms of mean squared errors [51].

Various OSS (Open Source Software) s are being modified and adopted into software products with their own quality level. However, it is difficult to measure the quality of an OSS before use and to select the proper one. In this paper authors have presented an approach to software reliability assessment of OSS adopted software system in the early stage. The authors identified the software factors that affect the reliability of software system when a large software system adopts OSS and assess software reliability using those factors. They are code modularity and code maintainability in software modules related with system requirements. The authors used them to calculate the initial fault rate with weight index (correlated value between requirement and module) which represents the degree of code modification. The authors apply the
proposed initial fault rate to reliability model to assess software reliability in the early stage of a software life cycle. Early software reliability assessment in OSS adoption helps to make an effective development and testing strategies for improving the reliability of the whole system [52].

Leslie Cheung et.al have addressed the challenges in developing a software component reliability prediction framework. The authors have done this by exploiting architectural models and associated analysis techniques, stochastic modeling approaches, and information sources available early in the development lifecycle. The authors extensively evaluate their framework to illustrate its utility as an early reliability prediction approach [53].

Shaik.Mohammad Rafi et.al have proposed a SRGM incorporating the Logistic-exponential testing effort function that is completely different from the logistic type Curve. The authors observed that most of software failure is time dependent. By incorporating testing-effort into SRGM authors made realistic assumptions about the software failure. The experimental results indicate that proposed model fits fairly well [54].

Arvin C.H. Wu and Kwoting Fang have provided another view from the three concepts, technological frames (people), social phenomena (organization), and implementation of information systems (technology) by way of a robust empirical and theoretical study. The authors have done that by looking at the literature of IS, by conducting a case study and by formulating a set of theoretical propositions. The authors found that social phenomena such as language, symbolic power, and communication processes are fundamental for understanding how technological interpretations are framed. Accordingly, the authors learned that technological frames are not static but dynamic. This inherent quality of frames may be intriguing not only for researchers but also for practitioners interested in finding new techniques for systems implementation. This is relevant as one cannot understand how individuals make sense of technology and IS without looking at their social interactions and cultural contexts [61].

Michael R. Lyu et.al have developed a set of new combination models that combine the results of single, or component, models. As measured by statistical methods for determining a model's applicability to a set of failure data, a combination model tends
to have more accurate short-term and long-term predictions than a component model. After evaluating these models using both historical data sets and data from recent Jet Propulsion Laboratory projects, authors have found that they are consistently satisfactory. To make it easier to apply reliability models and to form combination models, authors are developing a tool to automate many reliability-measurement tasks [75].

Generally, software reliability models can provide accurate reliability measurement in the later phase of testing. However, predictions in the early phase of software testing are useful as cost-effective and timely feedback. Q.P. Hu et al. have proposed to “reuse” failure data from past projects/releases with ANN models to improve early reliability for current project/release. To illustrate the proposed approach, two numerical examples are developed. Better prediction performance is observed in early phase of testing compared with original ANN model without failure data reuse. Furthermore, the optimal switching point from proposed approach to original ANN model in the whole testing phase is studied, with specific analysis on the two examples [76].

### 2.4 Reviews on Enterprise Resource Planning

S. Parthasarathy et al. have done the validation of crucial technical factors of ERP projects found from the literature than the usual managerial factors. Organizations should change their business processes to fit the ERP system, which provides the best practices and avoid customization. The review of case studies of ERP system implementation indicates that, in reality the practice is otherwise. Many articles have come up addressing the managerial issues in ERP implementation. It is necessary to mention here that managerial issues are manageable and ERP failure and its performance upgradation have to be viewed from the other side. This opened the door for this research study focusing the crucial technical factors for successful ERP implementation and enhancing its performance. Though ERP is a managerial concept, basically it is packaged software and its failure and performance require consideration from technical perspective. An exploratory case study was conducted to study this research issues and data analysis done through statistical techniques. The literature review was done meticulously and the crucial technical factors identified from the literature for performance enhancement of ERP projects agree with the factors cited...
by the ERP team in the industry. The outcome of this paper is establishment of three
crucial technical factors to the ERP vendors for performance enhancement of ERP
projects. ERP vendors and organizations in the process of launching ERP should
remember to integrate these crucial technical factors with other managerial factors to
deliver a quality ERP system on time as well as within budget that will reap the full
benefits of packaged software. Validating their results by a multi-organizational case
study, considering other technical aspects of ERP implementation like operating
system, client-server technology, mainframe technology, reliability and future
technology integration are the other areas for future research [55].

Kresimir Fertalj and Damir Kalpic have presented the results of an investigation
performed in 2001 under the title Comparative Analysis of Information Systems
Software in Croatia. The focus was set on the comparative analysis of domestic and
foreign Enterprise Resource Planning (ERP) software, which is present in Croatia.
The investigation was performed from the standpoint of ERP applicability, regardless
of the development methods and information technology. In other words, the
evaluation was performed primarily from the standpoint of users rather than
designers, programmers or other persons engaged in the system development and
implementation. System evaluation was performed in several phases and in multiple
steps. A general list of relevant ERP characteristics was established first. This initial
list was updated in co-operation with ERP suppliers and users. They completed the
list by adding the characteristics they regarded as important or accentuated some
features of their solutions that had not been mentioned initially. System evaluation
was performed at the users’ sites, having insight to real applications. To increase the
objectivity and accuracy, the evaluating teams consisted of persons of different profile
independent field experts, e.g. an accounting expert, IT expert, end-user etc. In spite
of the attitude taken not to evaluate the concrete basic technology, some estimation of
the computing architecture and functionality was performed, when it was found
relevant for the estimation of applicability [56].

Maturity assessment frameworks can help ERP adopters identify and understand those
practices which help their ERP processes succeed and those which do not. This paper
deploys a Requirements Engineering Maturity Model to examine variations in
instantiations of a standard ERP requirement engineering (RE) process. The author
draws on their previous results and the lessons learnt from eight years of experience in using ERP RE processes [57].

Software engineering metrics are units of measurement that are used to characterize the software engineering products and processes. This paper deals with a Software Metrics Plan (SMP) containing different software metrics to manage software processes during ERP implementation. Two hypotheses have been formulated and tested using statistical techniques to validate the SMP. The statistical analysis of the collected data from an ERP project supports the two hypotheses, leading to the conclusion that the software metrics are momentous in ERP projects [58].

Xin James He and Wenjie Wu have presented the early trends of enterprise resource planning (ERP) in China via an executive survey and subsequent statistical analysis. While ERP has been contributed to increased efficiency, increased revenue, and reduced costs, companies with horror stories about their ERP implementation are not uncommon. ERP implementation in China is so new that, except for a few large enterprises, the vast majority of Chinese enterprises are still on the sidelines weighing pros and cons. Whereas it is complex at any company, ERP implementation will be especially challenging for Chinese enterprises due to small company sizes, transitional enterprise ownership from planned to market economy, and cultural and linguistic barriers. Thus, this research scrutinizes such ERP pre-implementation concerns as key factors affecting ERP decisions, potential benefits, major obstacles to implementation, and unsettled issues with managerial implications to Chinese enterprises and ERP vendors [59].

An enterprise resource planning (ERP) system is a complex network composed of various business processes. S. G. Chen and Y. K. Lin have proposed methods based on stochastic flow network model to analyze the capacity sensitivity of persons with respect to the performance of an ERP system. The nodes in the network denote the person’s responsibility for the business tasks during the processes. The arcs between nodes denote the process precedence relationships in the ERP system. The performance of an ERP system is then related to the flow of the documents through the network. To analyze the person’s impact against the system performance, the increasing-capacity contributivity analysis and the decreasing-capacity impairment analysis are conducted. Some interesting findings are obtained. For example, the
strategy of increasing capacity will not promise to contribute performance. However, the strategy of decreasing capacity is also not always to impair performance. Numerical examples for these analyses are illustrated in this paper [60].

Enterprise resource planning (ERP) systems are currently involved into every aspect of organization as they provide a highly integrated solution to meet the information system needs. In this paper authors have selected a certain number of papers concerning ERP systems between 1998 and 2006, and this is by no means a comprehensive review. The literature is further classified by its topic and the major outcomes and research methods of each study are addressed. The implications for future research are provided [62].

D. Reuther and G. Chattopadhyay have presented common factors that are crucial for SME in the manufacturing sector embarking on an ERP system are identified in this paper for system selection and implementation. User survey was carried out for a target group of SME companies around Brisbane, Australia. The survey captured both qualitative and quantitative data relating to the ERP system in place at a target group of electrical and electronic equipment manufacturing companies located in South East Queensland, Australia. Quantitative analysis was carried out at the cost of the ERP selection and implementation project, both in terms of capital outlay and human resources. The research provided an overall perception of the success level of ERP systems [65].

Joe Nandhakumar and Matti Rossi, Jari Talvinen have reported the findings of a large-scale case study of implementing Enterprise Resource Planning Systems (ERP) in a long-established multinational company within the telecommunication sector. The company has recently streamlined its operations through an ambitious business process redesign initiative and introduced an ERP system. The study examines the process of change enacted during the implementation of the ERP system over time. The findings indicate that many changes emerged as the project team sought to improvise technological features and changes to the context of use to overcome embedded constraints in the existing system, organizational context and in the ERP system itself. The team also took advantage of evolving capabilities and emerging opportunities to continuously enact changes as an ongoing process of project life. The
paper argues that the process of technology-related change may be seen as a form of a ‘drift’ involving a series of purposeful actions with un-planned outcomes [66].

Liang Zhang et.al their paper aims to improve understanding of critical factors affecting ERP implementation success in China. A scale is developed to test the proposed model; two independent variables of business process reengineering and organizational culture that are assumed to be extremely important factors in ERP implementation in China are examined and supported by empirical data. However, due to the small sample size in the survey, there are some limitations in the generalization of the research results to a larger population. Meanwhile, ERP implementation is not a short-term project lasting only two or three months, but a long-term program which may last for one or several years. Factors affecting ERP implementation are complex and abundant, thus many researchers conduct case study only to find out some specific problems with ERP implementation. Undoubtedly, detailed case study is a powerful tool to solicit important issues disregarding to its disadvantage of generalization problems. Thus, combining detailed case study and a large survey would be an ideal method to researchers in the ERP field [67].

Poor project management can imperil the implementation of an ERP system. Having learnt the lessons from the failure of its first ERP implementation, the company in this case reengineered its project management practices to successfully carry out its second ERP implementation. Many critical project management factors contributed to the failure and success of this company’s ERP system. This study explores and identifies critical elements of project management that contributed to the success of the second ERP implementation. For those organizations adopting ERP, the findings provide a roadmap to follow in order to avoid making critical, but often underestimated, project management mistakes [68].

A key issue in enterprise resource planning (ERP) implementation is how to find a match between the ERP system and an organization’s business processes by appropriately customizing both the system and the organization. Wenhong Luo and Diane M. Strong have developed advanced framework for supporting management decision-making about customization choices and the capabilities required to accomplish them. In this framework, the author identifies various customization possibilities for business processes as well as ERP systems. The author also identifies
technical and process change capabilities required for system and process customization. Combining customization options with change capabilities, this paper present a useful way for managers to identify feasible customization options for their particular organization. Such a framework also helps managers to recognize the gap between desired customization options and change capabilities. A case study is used to illustrate the application of the framework [69].

Dr. Theekshana Suraweera et al have provided an in-depth understanding of ERP implementation processes in Sri Lanka. The major issues associated with KM in ERP implementations are brought into light and possible strategies for overcoming such issues are identified. The research builds on the work of Baskerville et al. (2006), and focuses on the importance of ‘operational level’ knowledge practices of capturing, integrating and sharing during ERP system implementations. Results of multiple case study interviews have been analyzed using cognitive maps to determine the issues deemed to have a major impact on the success of ERP implementations. Some of the key strategies that can be used to overcome the problems associated with KM in ERP implementations are presented [70].

P. Li Y.Z. Tian, et al have pointed out the driving factors of innovation and quality play an important role in companies during the process of ERP implementation. It brings forward the ERP system with four-factor control - time-cost-innovation-quality (TCIQ-driven). On this basis, this paper studies the impact of TCIQ model on the manufacturing performance in ERP companies. Based on the International Manufacturing Strategy Survey (IMSS), the paper compares the difference of ERP implementation among the sub-sample through single factor analysis of variance, and analyzes the relationship between the driving factors and manufacturing performance through multiple regression analysis. This paper points out that companies have been driven by four-factor control from time-cost-innovation-quality in the ERP project of discrete manufacturing companies in developed countries. At the same time, it can improve the manufacturing performance significantly through the TCIQ model [72].

Zuzana Krajcuskova et al have presented reliability growth model is very much available tool used during the development phase of electronic devices, of software products. In this paper the author reviews reliability growth models - a brief survey of hardware reliability growth models, of software reliability growth models [73].
Philip J. Boland et.al have presented an effort to address the continuing demand for high quality software, and enormous multitude of software reliability growth models have been proposed in recent years. In spite of the diversity and elegance of many of these, there is still a need for models which can be more readily applied in practice. One aspect which should perhaps be taken more account of is the complex and challenging nature of the testing of software. A brief overview of software reliability modeling and testing is given [74].

2.5 Summary

Many software reliability models have been developed till date which focus on error counts and do not directly model the software development environment. Exponential error rate was assumed for these models. There are few models that attempt to model software reliability based on certain design factors that exist during the software development process. Fault introduction, fault removal, and the environment were considered to model software reliability. Some of the model are estimators because they used primarily during test phase and are continually updated as data are collected. A few models are predictive in nature, which actually predict reliability before the coding begins.

A good model must have assumptions valid for the particular development process for which it will be used. Before a model is implemented, the assumptions must be verified as consistent with what is expected to be experienced on the project. If it is not known which of the models most closely fits the current projects, it may be wise to implement more than one model and analyze the results or to implement models based on historical data from a similar product. It is good idea to implement more than one model on a given project even if information is available on the product.