Chapter 1

Introduction

1. Introduction

Success of any software organization depends on total customer satisfaction which in turn depends on the development of quality software. Software Engineering methodology enables the production of quality software. One of the important characteristics of quality software is that it should be defect-free. The aim of defect detection and prevention is to provide quality software that will reduce the cost and time involved in fixing a defect, increase productivity and enable to achieve total customer satisfaction.

This chapter begins with a brief explanation of software and its diverse definitions on quality from the perception of users, developers and business. The need for development of quality software is emphasized by providing several definitions of defects and the various techniques of preventing these defects while developing software. This chapter also gives the details concerning the use of soft computing techniques in defect prediction, practical challenges and research oriented issues in order to produce high quality software.

1.1.1 Software

Software is a collection of programs that are non-tangible and are necessary for the computer to perform any function. Software is a unique entity that has a strong impact on every conceivable field. This includes medical, scientific, business, educational, defence, transport, telecommunication to name only a few.

Institute of Electrical and Electronics Engineers(IEEE Std.) [1] defines software as (a) a condition or a capability needed by a user to solve a problem (b) a condition or a capability that must be met or possessed by a system to satisfy a contract, standard, specification or other formally imposed document [IEEE-Std-610.12-1990].The term software quality has various definitions based on the viewpoints of the customer, developer and user. From the customer’s perspective Stephan H. Kan 2003 [2] defines quality as a vague concept that can be discussed and judged [2]. Definition of quality includes fitness for use when the customer’s requirements and expectations are considered. From the developer’s perspective, the term quality has a different definition. Gerald M 1997 [4] defines quality as the existence of characteristics of a product that can be assigned to meet requirements [4][5]. Also quality should meet the requirements [6].From the user’s perspective, Watts Humphrey: 1995 [3] states that the principal focus of software quality is the needs of users [3]. However, quality
adds value to business too. Quality also means satisfaction of the requirements of the stakeholders such as clients, customers and sponsors of the product, developers and auditors. Another popular definition of quality is development of defect-free product. He further emphasizes that customer satisfaction can be improved by reducing defects [34].

Definition of quality includes deployment of the product within the constraints of time, cost and resources. High quality software refers to the development of defect-free products which are capable of producing predictable results and are deliverable within time and cost constraints. It should serve the fitness of purpose that is the software should serve the customer’s purpose of getting it developed and it should meet all the functionalities desired by them. A systematic approach towards development of high quality software is necessary due to the increased competitiveness in today’s business world, technological advances, increase in hardware complexity and frequently changing business requirements. Production of high quality software is an innovative task that reduces the business risk through reduction in cost of development, time for rework and is an indicator of how much an organization has matured. It is the prerogative of every organization to have efficient software.

1.1.2 Need for effective software

On a daily basis, software development teams must deal with time, money, expertise and technology constraints to produce quality software. This practical perspective forces software engineering researchers to aim for development of efficient software which is not only useful but also accessible to the software engineering industry.

Barry W Boehm:1991 [31] attributed the importance of software economics to three reasons namely the alteration of the dynamics of technology innovation, the increased impact of software-enabled change in organizations and the identification of value creation as the key to success [31]. The quality and reliability of software products are of paramount importance and hence software testing is a key component in the software development life-cycle. The increased use of software has only increased the cost of debugging defective software. Existing software testing methods are still unable to provide high-quality software products. This is due to the fact that the development of a software product is an inherently complex venture. Defect prediction itself is a multifaceted area which demands rigorous research [7].
1.1.3 Software Engineering

Software Engineering is a discipline that aims at producing high quality software through systematic, well-disciplined approach of software development. Gursimran Singh Walia and Jeffrey C. Carver states software engineering encompasses processes, management techniques, practical methods, best practices, use of tools and standards to achieve the objective [7].

To deliver defect free software, it is imperative to predict and fix all possible defects before the product is delivered to the customer. Software repositories have huge amount of information which is useful while assessing software quality. Since software life cycle is a human activity, it is just not possible to produce any software without defects. The phases of Software Development Life Cycle (SDLC) include requirement analysis, design, implementation, testing and maintenance.

Requirements phase is the foundation phase for the entire development life cycle. Requirement analysis phase defines customer’s problems that require feasible and optimal solutions. Specification activity of requirement analysis phase produces a technical document. This document is popularly called System Requirement Specification (SRS). It specifies what is to be built. SRS is the deliverable of requirement analysis phase. Validation activity of requirement analysis phase examines the SRS to ensure that the requirements are unambiguous, complete and consistent. Requirements management activity controls and manages the modifications to the requirement. It maintains a traceability table known as traceability matrix to control and manage the changes [7].

An analysis team is responsible for all the activities at requirements analysis phase. The elicited requirements in addition provide a basis for planning and estimating the cost, time and resources for the further development of the product. Studies made by National Aeronautics and Space Administration (NASA) reveal that the major problems in the software industry are due to poor requirements elicitation, inadequate requirement specification and inadequate management of changes to requirements [6].Design phase of software development begins with the collection of SRS. Vasudevan S: 2005[9] represents ‘how’ phase of software development. It consists of two major activities namely high-level design and low-level design. The high-level design includes tasks such as architectural design, abstract specification, interface design and component design. Low-level design includes tasks such as component design, data structures design and algorithm design. Architectural design maps the SRS. Architectural design specification, high-level design
specification and low-level design specification are the deliverables of the design phase. NASA Studies have shown the need for process improvement at the design phase. NASA Software Engineering suggests the use of specification languages, graphical representations. Implementation of improved design techniques prevent and eliminate errors early in the life cycle [9].

Testing is defect detection activity which is followed in all phases of development. It verifies the code against the requirement specification and design. The aim of verification is to identify and eliminate defects. Testing validates the final product against the needs of the customer and its operation in the intended environment. It includes product-level testing and process-level testing. Product-level testing consists of unit testing, integration testing, system testing and acceptance testing. Maintenance phase begins when the final product is deployed to customers. Software maintenance activities maintain the product for its successful operation in the intended environment until the product becomes absolute [9]. To deploys high quality software, it is essential to develop defect-free deliverables in each phase of software development. Automation of testing activities through tools is one of the promising techniques for effective defect management.

### 1.2 Defects in software

Defect is imperfection or undesired behaviour that occurs either in the deliverable or in the final product. A defect also refers to any flaw or imperfection in a software process. Anything related to defect is a continual process and not a state. Mukesh Soni 2006 [10] defects in an application can lead to harmful situations in all phases of software development process [10]. Some of the popular definitions of defect are as follows:

**Definition 1**

General: Shortcoming that prevents an item from being complete, desirable, effective, safe, meritorious, or simply that which makes it to malfunction or fail in its purpose [11].

**Definition 2**

**Law:** Lack of legal sufficiency that it does not meeting the legal requirements due to incorrect or incomplete following of a required procedure [11].

**Definition 3**

**Manufacturing:** Non-conformance of a product with the specified requirements or non-fulfilment of user expectations (including the safety aspects) [11].
From the software perspective, defect can be either an error or a failure or a fault. IEEE Standard [1] states the following:

Error - Defect made in the human thought process while trying to understand given information, solve problems, or while trying to use methods and tools.

Fault - Concrete manifestation of an error within the software. One error may cause several faults and various errors may cause identical faults.

Failure - Departure of the operational software system behaviour from user expected requirements. A particular failure may be caused by several faults and some faults may never cause a failure Anand Prasadi etl 2010 [12]. Thus a causal relationship can be achieved among error, failure and fault.

Identifying and locating defects in software projects for measuring the success of projects is a difficult task. When project size grows, this task becomes expensive since sophisticated testing and evaluation mechanisms will be required. On the other hand, measuring software in a continuous and disciplined manner brings several advantages such as accurate estimation of project costs and schedules and helps in improving product and process qualities. Detailed analysis of software metric data also gives significant clues about the possible locations where defects in a programming code might occur.

1.2.1 Need for defect prediction

Developing defect free software system is very difficult. Most of the time, unknown bugs or unforeseen deficiencies might occur even in software projects in which the principles of software development methodologies were applied carefully. Due to the presence of defective software modules, the maintenance phase of software projects could become really painful for the users and costly for the enterprises. Therefore, predicting the defective modules or files in a software system prior to project deployment is a very critical and significant activity. It will lead to decrease in the total cost of the project and an increase in overall project success rate. Foad Marzoughi [8] Defect prediction will give chances to the development team to retest the modules or files in which the probability of defect is high. By spending more time on the defective modules and less time on the non-defective ones, the resources of the project could be utilized better and as a result, the maintenance phase of the project will be easier for both the customers and the project owners [8]. When budget does not allow for complete testing of an entire system, software managers can use such predictors to focus the testing on parts of the system that seem defect-prone. These potential defect-prone trouble spots can then be examined in more detail using model checking, intensive
testing etc. Humphrey:1989 [13] stated that quality is measured in terms of defects and software development process should be capable of fulfilling the needs of the customer at a low cost. According to the author, defects can be captured close to their origin through personal reviews which might reduce the cost of detection [13].

Geoff Dromey [14] has suggested curative and preventive approaches to tackle defect related issues for the development of quality software. Curative approach focuses on testing where both developers and users identify defects and predict them. In preventive approach, three aspects are emphasized. They are i) anticipation of defects, ii) use of formal inspections and prototypes to discover defects early in the development cycle and iii) use of tools for detection of defects. According to the author, prediction of defects is a better approach when compared to curative approach. The consideration of quality requirements similar to functional requirements to facilitate complete benefit of preventive approach has been emphasized by the author [14]. Defect prediction is one of the most significant activities in the software development process. An analysis of defects at an early stage reduces the time, cost and the resources required for rework.

Purushotham Narayan [15] has proved that defect prediction is one of the reliable mechanisms of software quality assurance in any project. The expensive nature of defects when identified late in the project has been indicated by the author. The cost to fix a defect found at requirements phase after deployment of the product is 100 times the cost of fixing the defect at the requirements phase. An enhanced software process which includes defect prevention strategy to achieve high quality and bug-free product is proposed. The significance of defect prevention through a real time scenario as a case study has been explained by the author. It is shown that 60% of defects can be reduced when defect prevention activities are implemented in the development of the product [15]. Early defect detection prevents defect migration from requirements phase to design phase and from design phase to implementation phase [16]. It by adding value to the most important attributes of software such as reliability, maintainability, efficiency and portability, quality is enhanced [17].

Gursimran and Jeffrey [7] emphasize on the defect detection and removal of both early and late faults in the software development as a primary focus for quality. Regardless of advancement in the quality research, the development of quality product is still a challenge. Lack of understanding of the source of problems by the developers, inability to learn from mistakes, lack of effective tools and incomplete verification process are some of the reasons
for the introduction of defects. Hence the need for research to provide more insight into the understanding of sources of the faults rather than just the faults themselves [7].

Vasudevan [9] emphasizes on defect prevention in software industry. Identification and classification of defects play vital roles in the measurement-based process and in the product improvement. Severity level is assigned to the classified defects. Defect prevention activities which include commitment from the management, creation of an action plan related to defect prevention activities, periodic review, defect measurement and causal analysis of the defects are recommended by the author [9].

Adeel et al., [19] conducted a survey to study the impact of DP in industries. The survey results have shown that defect detection at the early stages is less expensive than detection at later stages. They have recommended several techniques like prototyping, use of CASE tools, training techniques, Quality Function Deployment (QFD) technique and Joint Application Development (JAD) to avoid defects in the requirement analysis phase [19]. According to Mukesh Soni [20], developers in United States spend nearly 60% of the development time to fix defects. Therefore, when a software module is identified as defect-prone, more and more testing efforts are applied to improve its quality and reliability. The amount of testing effort required for a defective module is relative to its amount of defect-proneness. In other words, it is undesirable to apply equal amount of testing efforts to all the software modules. The testing efforts should therefore be allocated on the basis of its degree of defect-proneness.

1.2.2 The success of software projects through defect prediction

Software industry is a multibillion dollar contributor to the economic growth and technological advancement of modern society [23][25]. Bohem attributed the importance of software economics to three factors namely the alteration of the dynamics of technology innovation, increased impact of software-enabled change in organizations and the identification of value creation [31].

According to a survey carried out by the Standish Group, an average software project exceeded its budget by 90 percent (Chaos Chronicles, 1995). Software Projects are not risk-free ventures. Standish Group's study report in 2004 indicates that only 29% of projects are successful, 53% were challenged (delivered late, over budget and/or with less than the required features and functions) and 18% failed (cancelled prior to completion or delivered and never used) [33]. This survey took place in mid 90s and contained data from about 8000
projects. However, the Standish Group report conducted between 2002 and 2011 indicates that most of the projects are either challenged or deemed to be complete failures. Accordingly, 37% of survey projects were classified as successful, 21% was classified as failed and 42% were classified as challenged. These statistics show the importance of measuring the software quality early in its life cycle and taking the necessary precautions before failure occurs. An extensive metrics programs are being used by the industries in software projects to satisfy some external assessment body which are seen as unnecessary. This can be attributed as one of the major rationales for the aforementioned survey status of project success or failure rates. Hence, it is required to have complete awareness about the project status based upon the retrospection made on the empirical data of the historically developed projects. This necessitates gaining a complete knowledge on mining of valuable information available in the data repositories of the software industry.

1.2.3 Need for mining software engineering data

Mining software engineering data has emerged as one of the successful research directions since few decades. Software defect prediction is an important research focused in the software engineering field for more than three decades. Software repositories are often used in practice as recordkeeping repositories, they are rarely used to support decision-making processes. For example, historical repositories are used to track the history of a bug or a feature, but are not commonly used to determine the expected resolution time of an open bug based on the resolution time of previously-closed bugs.

1.2.4 Software defect prediction by soft computing techniques (DM, FL, GA, NN)

Software defects are expensive in terms of quality and cost. The cost of capturing and correcting defects is one of the most expensive software development activities. It will not be possible to eliminate all defects but it is possible to minimize the number of defects and their severe impact on the software quality. This is achievable by implementing a defect prediction process that focuses on improving software quality by decreasing the defect density. A little investment in defect management process can yield significant profits [20].

Defect management is one of the major issues prevailing in software industry. The main objective of defect management is to achieve complete customer satisfaction. One of the most important steps towards total customer satisfaction is generation of nearly zero-defect products. Both industry personnel and researchers are continuously working in this
direction. Hence, continued existence of software organization lies in the hands of their potential customers. Implementing emerging software systems can be risky if the significant success factors have been documented incorrectly. Software repositories have lots of information which are useful in assessing software quality. Hence, soft computing techniques have proved to be a valuable addition to existing statistical techniques normally used in prediction / estimation models.

Soft computing techniques such as data mining and machine learning algorithms can be applied on these repositories to extract useful information. Numerous studies have applied soft computing techniques (e.g. data mining (DM), fuzzy logic (FL), neural network (NN) and genetic algorithm (GA)) to the software defect prediction problem [26]. This will enable the prediction of modules which are likely to experience failure during operation based on software metrics.

Data mining generally called knowledge discovery in databases (KDD) is defined as the non trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data [17]. Data mining has been the main concern of many researchers and is being widely used in various applications. Data mining refers to extracting or mining knowledge from large amounts of data. It is used to determine characteristics of association, classification, clustering, prediction and estimation within data sets [18]. DM methods are gaining enormous recognition and popularity because they have been found to be beneficial in areas such as performance improvement and cost reduction in many industrial and business applications [29].

Data mining methodologies describe a data mining project as consisting of a sequence of phases and the tasks to be performed and the corresponding activities during each of the phases are highlighted. KDD process as presented in [17] [18] considered the following five stages in order to extract deemed knowledge from database. The significance of data mining is gaining acceptance in business as a means of earning higher profits at lower costs. Data mining and machine learning algorithms are useful in the prediction of software bug estimation. Machine learning models and Data mining techniques can be applied on the software repositories to extract the defects in a software product. Some common techniques include random forest, decision tree learning, Naïve Bayesian classification and neural networks etc. Tracking and detecting potential software defects in the early stages is very critical in many high assurance systems. However, building accurate quality estimation models is a challenge [12]. Therefore, investigation of problematic software modules before calibrating any software quality estimation models is desirable [17].
Fuzzy Logic is an effective soft-computing technique for solving uncertainties which arise due to imprecise inputs in order to generate linguistic and/or quantitative outputs. Fuzzy Logic has gained popularity in recent times as a sensible technique to achieve improved estimation accuracy of variables in any process. These variables may range from software cost estimation to activity scheduling for a software project. There is ample evidence in the literature to claim that a Fuzzy Logic model is the most appropriate estimation technique to handle imprecise and incomplete numerical data and linguistic knowledge. It is extremely difficult to enumerate linguistic knowledge using conventional mathematics such as rules, expert information and design requirements [24].

Fuzzy logic model makes use of fuzzy logic concepts introduced by Lofti A. Zadeh [24]. Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. A fuzzy set expresses the degree to which an element belongs to a set. The characteristic function of a fuzzy set is allowed to have values between 0 and 1 which denotes the degree of membership of an element in a given set. Fuzzy sets and Fuzzy logic have been employed in various real life applications. Fuzzy logic modelling techniques such as Fuzzy c-means clustering (FCM) and fuzzy inferences have been found to be useful additions to the existing statistical and machine learning techniques used for modelling software development [27].

A significant motivation for using fuzzy logic is its ability to estimate required defect much earlier in the development process. However, the above-mentioned description does establish the applicability of fuzzy c-means clustering (FCM), random forest methods to face the challenges of the software defect testing. Recent papers in software defects literature have shown that such machine learning formulations lead to significantly better performance than the existing methods. However, the utilization of these methods as a research paradigm for formulating software defect prediction challenges needs further investigation. The goal is to stimulate interest in the software quality testing community and to promote the use of FCM, random forest and machine learning techniques for formulating in the field of software project success prediction. Fuzzy logic is successful methodology to solve the realistic problems which include the simplicity and flexibility. It handles the problems with imprecise and incomplete data which is based on the fuzzy set theory. It is used to represent linguistic values like low, old and complex. Genetic programming (GP) is an evolutionary computation (EC) technique that automatically searches for an optimal solution of a problem without requiring the user to know or specify the form or structure of the solution in advance [28],
[29]. GP technique has been successfully applied to solve large number of difficult problems such as modelling of industrial processes [28], [29].

The general idea behind genetic algorithms is as follows. (Mitchell, 1998): a population of abstract representations (denominated chromosomes) of possible solutions (individuals) to an optimization problem evolves to produce better solutions. The evolution process begins with a population of randomly produced individuals. In each generation, all individuals are evaluated using a fitness function, randomly selected and modified (by mutation and crossover) to form a new population that replaces the previous one. This process is repeated until a termination criterion is satisfied (Michalewicz and Fogel, 2004; Mitchell, 1998).

Significance of GP: Evolutionary algorithms have been found ‘experimentally’ efficient in finding solutions to the modelling problems. GP is a very good example of an evolutionary algorithm. It has all the advantages offered by evolutionary algorithms and much more. The advantages offered by GP for modelling can be summarized as:

i) GP is a global search technique that makes use of hyper plane search which makes it less likely to get stuck in the local optimum. This is different from other techniques such as neural networks and gradient descent which are influenced by local optimal values.

ii) GP enjoys the benefit of variety in solution structures unlike most of the evolutionary algorithms which have fixed size solutions such as genetic algorithms or fixed architectures such as neural networks [20].

iii) GP can automatically eliminate unrelated attributes of the modelling problem while performing feature extraction [20]. Important attributes appear near the root while less important ones would appear deeper in the tree [21].

iv) GP has the capability to operate on portions of data for extracting significant rules. There is no need to use all of the training data to develop models [20].

v) GP are like white boxes that clearly sketch the relationships between attributes as opposed to many other black box solutions like neural networks [22].

vi) GP has the ability to operate on the data in its original form. No pre-processing or data transformations are usually required to apply GP for modelling tasks.
### 1.3 Practical challenges

Data mining methods, using genetic algorithm and fuzzy clustering techniques have the capability to transform the major challenges into probabilistic solutions in the field of software defect prediction analysis. However, some practical challenges need to be overcome in order to enable the widespread use of genetic fuzzy data mining methods. Most software engineering data mining studies rely on well known, publicly available tools. Many such tools are general purpose and should be adapted to assist the particular task at hand.

- Software engineering researchers may lack the expertise to adapt or develop mining algorithms or tools while data mining researchers may lack the background to understand mining requirements in the software engineering domain.
- One way of reducing the above gap is to foster close collaborations between software engineering community (requirement providers) and data mining community (solution providers). Our effort is directed towards achieving such collaboration.
- Another major challenge is that in real-world software projects, software fault measurements (such as fault proneness labels) might not be available for training a software quality-estimation model when an organization is dealing with a software project type it’s never dealt with before.
- A software organisation might not have recorded or collected software fault data from a previous system release. So, how is it possible for the quality assurance team to predict the quality of a software project without the collected software metrics?
- The team can’t take a supervised learning approach without software quality metrics such as the risk-based class or number of defects or faults. The estimation task then falls on the analyst (expert) who must decide the labels for each software module. Cluster analysis, an exploratory data analysis tool, naturally addresses these two challenges.

From the point of view of project managers or experts, software projects can be categorized into two groups namely defect or defect free (successful and unsuccessful) [30]. Any project which is completed on time and within budget with all features and functions originally specified is a successful project. A project which is completed and functioning but exceeds budget or time estimate or a project with fewer features and functions than originally specified or a project which is cancelled before completion or never implemented is deemed to be an unsuccessful project. Some of the issues in data mining are: What types of software engineering data are available for mining? How they should be customized to fit the
requirements and characteristics of software engineering data? Which software engineering tasks can benefit from mining software engineering data? How are data mining techniques used in software engineering? What are the challenges in applying data mining techniques to software engineering data? Which data mining techniques are most suitable for specific types of software engineering data? What are the freely available data sources and data mining and analysis tools (e.g. WEKA [17])? What software repositories and datasets should be mined for defect prediction? How can we resolve the problem of ceiling effects as well as imbalanced and highly skewed datasets? How can we get better results in identifying defects from large features and high level software modules?

1.4 Research oriented issues (RQ)

Although software defect prediction and correction has evolved rapidly in the past few years, many important issues still remain. First, detection systems must be more effective, detecting wide range of attacks with fewer false positives. Second defect detection must keep pace with modern software which is of complex, increased in size and critical in nature. There is a need for analysis techniques of defects. The issues connected with the current systems are covered in detail in chapter 2 is to develop a system that detects close to 100% of attacks, but this is still far from achieving this goal.

RQ1 what is the context of the defect prediction model?

To understand the environment or to which environment the prediction model was developed. Examine context primarily in terms of the origin of systems and the programming language using which the model was developed and tested. This contextual information allows us to discuss the applicability and generalisability of models. NASA’s publicly available software metrics data have proved very popular in developing fault prediction models. It also has the advantage that researchers are able to replicate and compare results using different approaches based on the same data set. However, although the repository holds many metrics and is publicly available, it is not possible to explore the source code or trace back how the metrics were extracted. It is also not always possible to identify if any changes have been made to the extraction and computation mechanisms over time.
RQ2  what variables have been used in defect prediction models?

This question identifies the independent and dependent variables of the model. Results show that researchers are still struggling to find reliable and useful metrics as an input to their fault prediction models. Researchers continue to search for predictor variables that correlate to faults independent of project. However, it is still not clear which model is most general, i.e., one which works in most situations.

RQ3  what modelling methods have been used in the development of defect prediction models?

This question identifies whether models are based on regression, machine learning or other approaches. It allows us to discuss the popularity and effective use of particular modelling methods. However, despite the substantial level of research activity and the several models developed in the area of fault prediction, there is still no consensus in the research community as to which approach is most applicable to specific environmental circumstances.

RQ4  how do study measure the performance of models?

Understanding how prediction studies measure and validate model performance gives an indication of how confident we can be in these results. Analyse several aspects of model performance (e.g. significance levels, data balance, accuracy) to discuss how well models are able to predict faults in code.

The aim of our work is the estimation and prediction of defect distribution as well as to improve the quality of the product using defect prediction mechanism. In this work, we are addressing defect prediction mechanism using fuzzy data mining techniques.

1.5 Motivation

The motivation behind the present work was the realization of that with the increasing demand for software, increasing size of the software, complexity of the software and critical nature of the software, none of the present day detection systems can meet the high demands for very high detection rate and prediction of success rate of software. Also most of the detection and prediction systems available in literature survey show distinct preference for detecting a certain class of software projects and moderately for other classes of projects.
1.6 Problem Statement

The review of the existing success rate prediction systems by the defects has posed the following problems which are to be considered for research.

Problem 1: What are the issues connected with existing prediction system?
Problem 2: Is there any system available at present which has complete detection coverage.
Problem 3: Is it possible to improve the detection performance of the existing prediction system.
Problem 4: What are the existing different metrics available for measure the prediction of success rate?
Problem 5: What are the limitations in the current FCM system?
Problem 6: What are the limitations associated with existing RF classifier algorithm.
Problem 7: How modified FCM & RF integrated with GA gives effective prediction model.

This thesis tries to find solutions to the above raised problems and the proposed solutions are available in the chapters from 3 to 6 with the related work in the above problem areas introduced in the respective chapters. In essence the problem can be stated as given a set of heterogeneous project success rate systems how should these individual results assimilated in order to enhance the prediction of project success.

1.7 Major contributions of the thesis

This thesis contributes integrated algorithms for the effective enhancement of prediction of success rate of a project. This research also incorporates a theoretical basis for improvement in enhancement of prediction.

X.1 Theoretical Formulation

1. Issues identified with FCM
2. Issues identified with RF
3. Improvement in performance of the modified system over the existing independent systems has been proved.

X.2 Experimental Validation

1. The NASA data set for prediction of success rate of project has been evaluated against existing systems
2. Improved detection rate using modified FCM has been validated

3. Improved detection rate using modified RF has been validated.

1.8 Research Goal

Given the resource parameter such as number of people, skill set, experience level, time and cost. We believe that a unique processing module using the proposed modified system will open the better than the best of existing prediction systems.

1.9 Organization

The rest of the thesis is organized as follows. Chapter 2 presents state-of-the-art data mining, machine learning technologies from areas concerned with traditional defect management systems. Chapter 3 proposes effective study and management of various data mining clustering technologies for defect prediction analysis to determine the success of software projects. The chapter leverages existing technologies and provides additional services for defect distribution and aggregation besides discussing the use of real-world models and strategies, availability of various clustering techniques and the strength and weakness of each technique. An attempt is also made to determine suitable method for analysing the defect patterns. Chapter 4 presents architecture and framework of cascading clustering and classification. It is observed that the accuracy of random forest classifier is better as compared to other classifiers. Chapter 5 discusses the design and implementation of integrated approach of fuzzy c means clustering, neural network and random forest algorithm with a view to analyse the defect patterns in software projects. Chapter 6 briefly presents the results, analysis and their performance evaluation. Chapter 7 concludes the thesis and provides directions for future work.