CHAPTER 1

INTRODUCTION

1.1 Overview

Finding and fixing software defects is difficult, and needs significant effort. Software defects can cause a range of problems, ranging from minor glitches to catastrophic glitches resulting in loss of life [1]. For example, the destruction of prototype Ariane 5 rocket just after its launch is due to software defect, resulting in loss of European Space Agency’s US $1 billion. Similarly a software defect caused the Therac-25 radiation therapy device behave abnormally for certain input sequence resulting in the delivery of lethal radiation doses to patients [26].

The projects that lack proper planning are liable to defects and the time spent to find defects and fix them is more than the actual code development time. Zero defects are not practicable. In spite of intensive testing for defects, many defects still exist, leading to unpredictable behavior of the software and sometimes becoming catastrophic or unusable.

Typically a software solution is a product with known and unknown defects [23]. This leads to a continuous process of evolving software by removing known defects and uncovering new defects over
time. In this mode as new defects are uncovered, fixing defects of highest priority is the task of the software development team.

Code review, unit testing and system testing integration are the traditional process to identifying defects. It also encompasses a software defect prevention process that accompanies software evaluation and design processes. The developers of software record defects found during evaluation and testing operations. The well organized software development also analyzes defects using statistical processes; systems like Six Sigma are used effectively to reduce defects.

Reliability assessment on upgradation of the existing systems is a major predicament in IT software management. Existing systems need to be constantly maintained, to add new components and to remove defects; hence the demands of upgradation. Reliability assurance after an upgrade is assessed by observing failure behavior of the upgraded system in operation with the prior knowledge of the old system.

Traditionally software metrics are used to define the program complexity, to estimate programming time. Extensive research has also been carried out to calculate the number of defects in a module using software metrics. Basically, software metrics are used for obtaining quantitative measurements of the software or its specifications. Mostly value of the metrics is represented as an
interval, ordinal or nominal scale. Metric values in ratio scale are preferred for the mathematical equations used in software process as it allows for easy and meaningful computation.

The software metrics is widely researched. The first and most extensive report of software metrics program containing key guidelines forming a base for subsequent metrics program was done by Grady and Caswell. Goal Question Metric (GQM) was proposed, by Basili and Rombach, using ideas from the total quality management to ensure that the metrics activities are always goal-driven. The GQM ensured that the metrics program should collect only those metrics which are relevant to a particular goal. The works of Briand et al., 1996 highlights the need of scale types to define measurement for software metrics activities [39].

An emerging approach for defect prediction is the use of data mining techniques to predict the problematic areas in the software. These techniques extensively rely on software metrics associated with the software. Bayesian inference under simplifying assumptions formed the base of the reliability assessment studies. The accuracy of predictions achieved due to reliability is the subject that is going to be discussed. The use of different data mining techniques for practical reliability assessment is open for future research.
Defect prediction models are used to find the reliability of the system, estimate the efficiency of the design and testing process over a number of defects. The byproduct of the above process is the reduction of staffing required and the maintenance cost. Defect prediction models use machine learning methods to learn and predicting potentially defected modules within the software. Cyclomatic complexity and size of the software are the most commonly used metrics for defect models [52]. The testing metrics that are created in the test phase prove to be useful to estimate the sequence of defects [44]. Metrics from all the steps of the software process such as design, implementation, testing should be utilized and connected to specific dependencies. For designing an effective prediction model metrics from a single process will not be sufficient.

Machine learning algorithms are applied effectively in problem domains with changing values and regularities. Software metrics is a good example of the above statement. The software metric data of modules or software combined with defect data forms the input of the machine learning algorithms. These algorithms help execute a probability distribution and analyze errors [35]. The techniques popularly used for software defect prediction problems are decision trees, neural network and Bayesian belief network [38; 35; 40].
1.2 Problem Statement

The goal of this research is to help developers identify defects based on existing software metrics using data mining techniques and thereby improve software quality which ultimately leads to reducing the software development cost in the developing and maintenance phase. This research focuses on identifying defective modules and so the scope of software that needs to be examined for defects could be prioritized. This allows the developer to run test cases in the predicted modules. The proposed methodology helps in identifying modules that require immediate attention and so the reliability of the software can be improved faster as the higher priority defects can be handled first. In particular, the goals of this research are:

- To propose a data mining approach to the software defect prediction process based on software metrics.

- To present neural network architecture for software defect prediction process model based on the available metrics.

The aim of this research is to propose a new method for software defect identification using machine learning methods. The data set of software metrics used for this research is acquired from NASA’s Metrics Data Program (MDP). The goal in this research focuses on improving the classification accuracy of the Data mining algorithm. The existing classification algorithms are evaluated initially, based on
its extensive study, a novel Neural Network algorithm with a degree of fuzziness in the hidden layer is proposed.

1.3 Background of the problem

In the current business scenario, integration and restructuring of the existing software used in business applications due corporate take-over, change in corporate strategy, a conversion of models (data processing model to service oriented model) or even more reliance on the company's IT structure over the previously existing manual process. The consequence of this incessant modification process is that older software systems are required to collaborate with newer software systems in a way that was not expected in their original design. Such existing large-scale software applications are called legacy applications. A legacy system when designed and implemented was done on the then environment; the current IT infrastructure is a drastically different environment.

The software quality is gauged by the attributes like functionality, performance, ease of maintenance, documentation and reliability. Software reliability is highly regarded as it is hard to achieve with complex software. To achieve a definite level of reliability, the software layers are complicated during designing process by the system developers to accommodate up gradation of the software system in the later stages and to manage the escalation in system
size. The software reliability is inversely related to the degree of complexity of software, but complexity is directly related to functionality, capability qualities of the software system. Thus achieving the balance of complexity and the reliability are very important. Emphasis on the qualities of the software is likely to add more complexity to the software [1].

Though extensive testing in various phases of the software reduces defects, it cannot be eliminated totally. The Software configuration management process in the organization provides a wealth of data on previous defects detected and how it was handled. Researchers worked on predictive models using the metrics available in the repository and various papers have been published on software defect prediction which is part of the process of software defect prevention. Though the software never changes the severity of the software defects, depending upon the system in which the software is run the defect’s severity varies. Defining the severity levels of the software defect presents another major problem because of the different systems the software is run. Errors are classified into five categories, namely catastrophic, severe, major, minor and no effect. The defects or errors identified in any software come under anyone of the above categories.
1.4 Significance of the study

Software defect identification and locating the module with defect in software projects is difficult work. The larger the project, the task of defect identification gets tougher with sophisticated testing and evaluation mechanism, thus increasing the cost of the task. The advantages of evaluating the software continuously and in a regimented manner are that an accurate estimation of project costs and schedule and significant improvement in the product and process qualities. Software metric data analysis becomes handy at discovering locations of possible defects in the programming code.

This study helps in identifying defects in existing legacy systems using software metric data analysis done using various tools available in the market for reducing time significantly for the actual metric consolidation. The cost in maintaining the software comes down significantly, because of the capability to identify problematic modules and their severity.

1.5 Research Issues

1. Examining for better means of reducing project cost and time.

2. Enabling the project manager in better managing the project.

3. Benchmarking the software defect prediction accuracy.

4. Augmenting the existing software quality assurance systems.
1.6 Scope of the study

This study focuses on the data mining classification algorithms with respect to the domain of software defect prediction. The data set used in this research is provided by the NASA KC1 Metrics Data Program. The data set consists of software metrics and its associated error data at the function level. The data warehouse contains the data which has been gathered and authenticated by the NASA’s Metric Data Program.

The objectives of the thesis can be broadly classified into

- Investigate existing data mining algorithms and understand the behavior of classification algorithm on the software defect dataset.
- In the second phased investigate preprocessing algorithms to improve the classification algorithms used in the first phase.
- Based on the outcome of investigations based on existing algorithms and pre processing techniques, propose a novel neural network classification algorithm to improve the accuracy of the software defect prediction.
1.7 Limitations of the study

This work focuses on the KC1 data set from NASA’s metric data program. The proposed algorithm has not been validated on other available data set or on any real software project. This work provides an indication of modules which has a high probability of defect, but does not mention the location of the defect.

1.8 Thesis overview

The thesis is organized into six chapters. The First chapter is an introduction to the subject of software metrics and defect prediction, the problems associated with it, background of the problem, and significance of the study, scope and limitations of the thesis.

Second chapter deals with Literature survey on Software reliability, Data mining and different Classification methods. In the third chapter, software reliability, software quality metrics and data mining concepts and classification techniques used in this research are studied.

The fourth chapter proposes a mathematical model to preprocess the NASA KC1 data set to improve the software reliability and it is compared with the data without preprocessing.
In fifth chapter, in order to improve the software’s reliability further, a novel Bell Fuzzy membership function is proposed to the neuron of Multilayer perception neural network and it is applied on data set, and the results are compared with methods available in literature. The sixth chapter concludes this work with suggestions for future work.