CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Object Oriented (OO) approach is a popular choice for building large stable systems. The motivation behind choosing OO approach is modularity, reusability, reduced maintenance and real world modeling of the problem. Ensuring the quality of OO systems in the early phases of development saves time, effort and money. To control and improve the quality of software, periodic measurement and analysis of the underlying software artifacts is required. OO design metrics are quantitative indicators of the structural design properties of the software system in the early stage of development. Quality models can then be used to describe the design properties (independent variables) as measurable external quality attributes of the system (dependent variables) such as reliability, maintainability or reusability.

Many OO design measures and quality prediction models have been proposed in the literature. The measures and models are the result of continuing attempt in improving the quality prediction process. Once the appropriate set of measurements and prediction models are in place, large systems can be assessed in a short period of time. The entire process can be automated without any human intervention.
Replicated studies covering several domains and development environments are required to draw conclusions on the effectiveness of the metrics and models to assess the software during the early stages of development.

1.2 MOTIVATION

Many OO design metrics have been proposed in the literature (El-Wakil et al 2004), (George et al 2014). There is a need for studying the appropriateness of metrics across the programming languages. The main benefit of such an analysis would be that the metrics and prediction techniques need not be modified every time the underlying programming language changes in a development environment. This served as a motivation to investigate the suitability of OO metrics across three widely used OO languages - C++, Java and C#.

Lincke et al (2008) have performed a study on ten commercial, freely downloadable tools available for metric computation for three software systems and concluded that the tools interpret the metrics differently and support for all metrics across OO programming languages is not available. The issues discussed in this work are the motivation behind the design and development of a generic automated metrics computation tool.

Replicated metrics have been proposed to capture each underlying property of OO paradigm. An analysis is performed to find the appropriate metric that captures the cohesion property. Briand et al (1998b) in their proposal of the cohesion framework have analyzed the existing cohesion metrics and listed their limitations. A similar analysis was performed by Al Dallal (2012). These analyses indicate limitations in existing cohesion metrics. LCOM1 and LCOM2 are inverse cohesion metrics and their values are not normalized. LCOM3 considers access methods for computing
cohesion values. This leads to an artificial decrease in cohesion value. LCOM4 could give higher value due to a single common attribute being shared among the methods of the class. LCOM5 gives high cohesion values only in the case of a questionable design where all methods access all attributes. TCC and LCC consider constructors for computing cohesion value. This results in high values for cohesion, as the constructor typically initializes all class members. ICH is a method level cohesion metric scaled for use as class level metric. When unrelated classes are combined ICH gives higher value for cohesion. Class Cohesion (CC) and Sensitive Class Cohesion Metric (SCOM) have limitations in assessing similarity of methods. The limitations in similarity based cohesion metrics served as the motivation behind the proposal of a new cohesion metric that overcomes the limitations of existing cohesion metrics.

In a prediction model, providing maximum prediction accuracy is a desirable property. Artificial Neural Network (ANN) is a widely used prediction model for classification in various fields (Kanmani et al 2007) and (Civicioglu 2012). The ANN model could be improved by finding optimal values for its parameters. ANNs are trained to map the input values to the expected output from the dataset. Training is the process of adjusting the weight values of the ANN. A well trained ANN has the ability to adapt to new inputs. There are many algorithms existing for training neural networks. Gradient descent, Differential Evolution algorithm (Slowik & Bialko 2008), Simulated Annealing (Abbasi & Mahlooji 2012) and Genetic algorithm (Zhong, et al 2011) are some of the widely used training algorithms. Research work on Swarm intelligence algorithms for training ANN to predict faults in OO systems is limited. This prompted the investigation of swarm intelligence based algorithms for training ANN.
1.3 OBJECTIVE OF THE THESIS

The objectives of this research work are

- To investigate the appropriateness of OO design metrics across the OO languages - C++, Java and C#.

- To develop an automated metric computation tool to support all OO metrics, which is also extensible to compute any new OO metric in any language.

- To investigate the appropriate metric to capture the cohesion property accurately and validating it.

- Enhancing the performance of fault prediction model using Artificial Neural Network with swarm intelligence based training algorithms.

1.4 ORGANIZATION OF THE THESIS

To meet the objectives stated above, the research work is carried out and presented as described below.

Chapter two elaborates on the OO paradigm and gives details on the earlier research contributions carried out in the field of OO metrics. The evaluation model used to analyze metrics for their availability on different OO languages is described. The results of the analysis of coupling, cohesion, inheritance, size, complexity and interface metrics are listed.

Chapter three introduces the automated tool developed for computing metrics from source code of OO languages. The design of the tool
is presented, highlighting its extensibility. The metric values computed using the automated tool and existing metric tools are compared.

Chapter four explains a new cohesion metric, High Precision Cohesion Metric and its significance. This chapter discusses the mathematical validation of the metric. The results comparing the fault prediction ability of the various cohesion metrics is presented.

Chapter five describes the theoretical and empirical validation of the proposed cohesion metric. The relationship of the metric with other class characteristics is analyzed. The improvement in fault prediction using the proposed metric instead of LCOM in the CK metric suite is explored.

Chapter six discusses the software quality prediction models using ANN trained by swarm intelligence algorithms. The various swarm intelligence algorithms considered, are detailed and the modifications required to adapt the swarm intelligence algorithms to train ANN are explained. The fault prediction accuracy and time taken by the various models are compared.

Chapter seven summarizes the work done and the research contributions made. Future enhancements are suggested that could be taken up as a continuation of this research work.