ABSTRACT

Object oriented approach of software development is the most commonly used approach for developing large and complex software systems. The most important feature of this approach is the integration of analysis, design and coding in a single framework, using the same concepts. This approach reduced the software development and maintenance cost. It has capability of developing highly reliable products. Abstraction, Encapsulation, Inheritance and Polymorphism are the key concepts of this approach.

Measurement is one of the key activities used in the Object Oriented Software Development (OOSD). Although many traditional software metrics are available but these metrics are not suitable for measuring attributes of object oriented software systems due to different analysis, design, coding and testing techniques. Also features of object oriented approach are not addressed by traditional metrics. Consequently huge numbers of new class of metrics have been proposed and validated by researchers' community to give more effective measures for object oriented attributes. This research work conducts extensive literature survey of existing object oriented metrics by focusing on its purpose, approach, validation and specification. This survey gives in-depth view of existing metrics and provides a direction to rest of the study. By focusing on complexity, fault-proneness, reuse and reusability metrics six objectives are realized in this research work. Entire research work is divided into three parts to achieve the objectives.

First part deals with various views of software complexity and its measurement. In this part new complexity model is proposed to compute complexity of a class in terms of complexity of its methods. The proposed model integrates the complexity due to control flow, method calls and data calls. Consequently two new metrics have been proposed which are named as Total Method Call Complexity (TMCC) and Total Data Call Complexity (TDCC). TMCC and TDCC computes the complexity of a method due to method and data calls respectively. Control Flow Complexity (CFC) is computed using traditional McCabe's Cyclomatic complexity metric. This model measures the difficulty level of understanding the classes and shows the scope of further improvement in the classes. Given model is also compared with four Chidamber and Kemerer's metrics - Weighted Methods per Class (WMC), Response For a Class (RFC), Depth of Inheritance
Tree (DIT) and Coupling Between Objects (CBO), using a case study developed in Java language.

Second part concentrates on measurement of fault-proneness of classes. Various factors that affect the fault-proneness of classes have been discussed and many existing metrics have been suggested to measure these factors. This part also suggests measures used in two machine learning algorithms namely Iterative Dichotomiser-3 (ID3) and Classification And Regression Trees (CART) for selecting best suitable metric(s) for classification of faulty and non-faulty classes. The suggested measures are tested on four Chidamber and Kemerer’s metrics - WMC, CBO, RFC and Lack of Cohesion in Methods (LCOM), using two case studies of Java language. By identifying the categories of classes further fault management activities can be planned for optimal utilization of resources.

Third part explores the concept of reuse and reusability and suggests taxonomy of reuse and reusability metrics for better understanding. In this part five new metrics are proposed which are named as Breadth of Inheritance Tree (BIT), Method Reuse Per Inheritance Relation (MRPIR), Attribute Reuse Per Inheritance Relation (ARPIR), Generality of Class (GC) and Reuse Probability (RP). These metrics are useful for evaluating reuse and reusability using inheritance hierarchy. Validity of proposed metrics are tested using a case study and proposed metrics are also compared with four existing metrics – Depth of Inheritance Tree (DIT), Number of Children (NOC), Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF). Proposed metrics are useful for comparing two or more alternative inheritance hierarchies of same problem. Relationships among dimensions (depth and breadth) of inheritance hierarchy, reuse and complexity are also established in this part. Five design oriented metrics - DIT, NOC, MIF, AIF and RFC are used for establishing relationships.

Overall this research work is helpful for understanding existing object oriented metrics, managing complexity in a new way, planning better fault management activities and improving reuse and reusability of classes at design time.