Study of Various Class Oriented Metrics

Prabhjot Kaur
Guest Faculty, Department of Computer Application
SGAD Govt. College Tarn Taran

Abstract
Software metrics relates individual measures in a ways. Software metrics possess various characteristics on which one can evaluate its quality. The role of software metrics is to find significant estimates for software products and directs us in intriguing managerial and technical decisions. Software metrics have become an integral part of software development and are used during every phase of the software development life cycle. In practice, software metric will be used only if it is intuitive and easy to compute otherwise the metric will not be widely adopted. In this paper, I have considered various Class oriented metrics for object oriented programming.

Keywords
Software, Class, Object oriented programming, Metrics.

1. Introduction
Class is the building block of an object oriented system. A class encapsulates operations (functions) and their attributes (data). The class inherits subclasses i.e. parent or base class forms a child or derived class. This subclass inherits the features of its base class plus its own additional features. Basically, there are six class based design metrics for object oriented systems.

2. Objectives of Class Oriented metrics:
   - To measure unique aspects of the OO approach.
   - To measure complexity of the design.
   - To improve the development of the software.

3. Types of Class Oriented Metrics:

   ![Diagram of Class Oriented Metrics]

   - WMC
   - DIT
   - NOC
   - CBO
   - RFC
   - LCOM
3.1 Weighted methods per class (WMC)

It is defined as the sum of the complexity of the methods of a class. If there are ‘n’ methods of complexity c1, c2, c3, ……cn defined for a given class ‘C’, then the complexity factor of these ‘n’ classes should be equal to unity i.e.

\[ WMC = \sum c_i \quad \text{for} \quad i = 1 \text{ to } n. \]

NOM and their complexity tell about the amount of time and effort required to develop and maintain a class. The larger the NOM, the more complex is the inheritance tree. The potential reuse of a class becomes limited when NOM for a specific class increases. WMC must be kept as low as possible.

3.2 Depth of inheritance tree (DIT)

It can be defined as the maximum length of the node from the root of the tree. With increment in DIT, lower level classes will inherit more methods which give rise to problem of predicting class behavior and design becomes complex. The major outcome of this DIT is its Re-usability factor.

Figure 2: A Class Hierarchy

The maximum value for DIT in given figure is 4.

3.3 Number of Children (NOC)

Subclasses are basically referred as children in Inheritance. Each class can possess one or more subclasses depending upon the given situation. When NOC increases, the abstraction of parent class is partially disturbed if subclass is not appropriate member of that class. Moreover, the testing phase gets bit tedious as it is done for each subclass as per inheritance rules. This happens when NOC increases.

3.4 Coupling between object classes (CBO)

The value of CBO is defined using CRC model. It is defined as a count of number of other classes to which it is coupled. More specifically, it is the number of collaborations listed for a class on its CRC index card. Reusability of a class will decrease as CBO increases, or in other words, CBO is indirectly proportional to reusability of a class. Small values of CBO:

- Improve modularity and promote Encapsulation.
- It is easier to maintain and test a class

3.5 Response for a class (RFC)
It is the number of methods in the response set. The response set of a class can be defined as “a set of methods that can potentially be executed in response to a message received by an object of a class”. If a large NOM are invoked from a class, testing and maintenance of a class becomes more complex.

3.6 Lack of cohesion in methods (LCOM)

It measures the dissimilarity between methods of a class using instance variables (attributes). Each method has some data members and member functions for a given class ‘C’. LCOM is the number of methods that access one or more of the same attributes. If no methods access the same attributes, then LCOM=0. If value of LCOM is high, then:

- Complexity is more.
- Encapsulation is inhibited.
- Low-quality design is identified.

4. Conclusion

Metrics can identify potential areas of problems that may lead to problems or errors. Finding these areas in the phase they are developed decreasing the cost and avoids major ripple effects from the changes, later in the development life cycle. From the above discussion we come to conclude that these class oriented metrics measure complexity of design. Moreover the various metrics encountered during the above discussion have varying results. Class oriented metrics like WMC, CBO, RFC, LCOM have lower objective or more specifically the value for these metrics must be kept as low as possible. The rest two metrics i.e. DIT and NOC support Trade-off concept. These two metrics support Reusability factor which is considered the prime factor of Object Oriented system.

References


