ABSTRACT

This paper is described quality of software and introduces the criteria of class cohesion. Class cohesion refers to the extent to which the members of a class are related. The purpose of this paper is to check the software quality, class cohesion, class cohesion metrics and mathematical properties of Class Cohesion. Several class cohesion metrics are proposed in the literature to indicate class cohesion. Metrics that violate class cohesion properties are not well defined, and their utility as indicators of the relatedness of class members is questionable. Results show that the metrics differ considerably in satisfying the cohesion properties; some of them satisfy all properties and others satisfy none. Class cohesion is an important object-oriented quality attribute which should be used to improve the ability of the metrics. It refers to the degree of relatedness between the methods and attributes of a class. Several metrics have been proposed to measure the extent to which the class members are related.

Key-Words: object-oriented class, software quality, class cohesion metric, class cohesion.

1. INTRODUCTION

The development of high-quality software is a primary goal in software engineering. Such software are likely to be stable and maintainable. During the development process, developers and managers typically apply quality metrics to assess and improve software quality. Cohesion, coupling, and complexity are common types of such metrics. The cohesion of a module indicates the extent to which the components of the module are related. A highly cohesive module performs a set of closely related actions and cannot be split into separate modules. Such a module is believed to be easier to understand, modify, and maintain than a less cohesive module. Classes are the basic units of design in object-oriented programs. Class cohesion refers to the relatedness of class members, that is, its attributes and methods. These metrics are based on information available during high- or low-level design phases. High-level design (HLD) cohesion metrics identify potential cohesion issues early in the HLD phase. These metrics do not support class refactoring activities that require information about the interactions between the methods and attributes in a class, because these interactions are not precisely known and defined in the HLD phase. Low-level design (LLD) cohesion metrics use finer-grained information than that used by the HLD cohesion metrics. There are several metrics proposed in the quality of object oriented design. These metrics are used to evaluate the quality of software and uses the phases of large software development at fast and low cost. Class cohesion refers to the extent to which the members of a class are related. Several class cohesion metrics are proposed in the literature to indicate class cohesion and a few of them are mathematically validated against the necessary properties of class cohesion. Metrics that violate class cohesion properties are not well defined, and their utility as indicators of the relatedness of class members is questionable. The purpose of this paper is to describe checking of class cohesion productivity and which metrics are to be used.

The major contributions of this paper are as follows:
(i) This paper is describing the quality of software that explore the relationship between these properties and quality attributes such as fault proneness maintainability, effort or productivity are needed to use these metrics effectively.
(ii) This paper introduces criteria for class cohesion that exhibit special cases such as having no attributes, no parameter types for the methods, or fewer than two methods in the class. These cases prevented us using some of the existing cohesion metrics.

2. RELATED WORK

2.1 Software quality of a system can be vague, undefined and attributes. For any software quality of system there should be three specifications can be used. First, what the system is to do is described by the functional specification. Second, how well the functions are to operate is concerned by a quality specification. Third, how much is to be spent on
the system is concerned by a resource specification. To measuring the software quality, quality factors are described in three sets:-

1. Product operation quality factors are correctness, reliability, efficiency, integrity and usability.
2. Product revision quality factors are maintainability, testability and flexibility.
3. Product transition quality factors are portability, reusability and interoperability.

The ISO 9126 identifies six software quality characteristics are functionality, reliability, usability, efficiency, maintainability, portability. These techniques increasing visibility, procedural structure and checking immediate stages through inspections are used to help enhance software quality. Process and Product quality of a developed product is influenced by the quality of the production process. This is very useful in software development and some product quality attributes are hard to assess. The software processes and product quality relationship is very complex and poorly understood. The process based quality containing straightforward link between process and product in manufactured goods. The quality management activities are quality assurance, quality planning, quality control and quality management.

2.2 Class Cohesion Metrics

Classes are related to each other in Cohesion metrics which are measured products. Only one functions can be performed in a cohesive class and in a non-cohesive class which can be performed two or more unrelated functions. A non-cohesive class may need to be restructured into two or more smaller classes. The assumption behind the following cohesion metrics is that methods are related if they work on the same class-level variables. Methods are unrelated if they work on different variables altogether. In a cohesive class, methods work with the same set of variables. In a non-cohesive class, there are some methods that work on different data.

There are four mathematical properties of Class Cohesion Metric.

(i) If a cohesion measure belongs to a specific interval 0 to Max then it is called non negativity and normalization and Normalization allows for easy comparison between the cohesion of different classes.
(ii) If cohesion of a class equals 0 and if the class has no cohesive interactions while the cohesion of a class is equal to Max if all possible interactions within the class are present that is called null value and maximum value.
(iii) If the addition of cohesive interactions to the module cannot decrease its cohesion then it is called monotonicity.
(iv) If the merging of two unrelated modules into one module does not increase the module’s cohesion. Therefore, given two classes, c1 and c2, the cohesion of the merged class c’ then it is called cohesive modules. These properties were widely used to support the theoretical validation for several proposed class cohesion metrics.

2.3 Some Metrics

1. Coupling between Objects (CBO):- CBO for a class is count of the number of other classes to which it is coupled.
2. Coupling between Objects (CBO1):- Same as CBO, except that inheritance based coupling is not counted.
3. Lack of Cohesion (LCOM1):- It counts number of null pairs of methods that do not have common attributes.
4. Lack of Cohesion (LCOM2):- It measures the dissimilarity of methods in a class by looking at the instance variable or attributes used by methods.
5. Number of Children (NOC):- The NOC is the number of immediate subclasses of a class in a hierarchy.
6. Depth of Inheritance (DIT):- The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.
7. Weighted Methods per Class (WMC):- The WMC is a count of sum of complexities of all methods in a class.
8. Response for a Class (RFC):- The response set of a class (RFC) is defined as set of methods that can be potentially executed in response to a message received by an object of that class.

3. CLASS COHESION

3.1 The Method behind used in cohesive class: - A cohesive class performs one function. Lack of cohesion means that a class performs more than one function. This is not desirable. If a class performs several unrelated functions, it should be split up. There are two methods used in cohesive class. One method stated that the high cohesion is desired if it promote encapsulation. But the drawback consists that a highly cohesive class having high coupling between the methods of classes and which indicate high testing effort of that class. The other method stated that low cohesion is desired inappropriate design and consider high complexity. But it has been found high likelihood errors, And the cohesive class should be split into two or more smaller classes. The Project Analyzer supports in several ways to analyze cohesion:-

3.2 LCOM Metrics: - LCOM metrics aims is used to detect problem classes. It is lack of cohesion of methods. A high LCOM value means low cohesion. There are four LCOM that is lack of cohesion of method variants that are LCOM1, LCOM2, LCOM3 and LCOM4. For the Visual basic systems we use LCOM4 variant and the other variants may be of scientific interest or says LCOM4 is the lack of cohesion metric we recommend for Visual Basic programs. LCOM4 measures the number of connected components in a class. A connected component is a set of related methods and class-level variables. There should be only one such a component in each class. If there are 2 or more components, the class
3.3 TTC and LCC metrics: Tight and Loose Class Cohesion. The TTC and LCC metrics aims to tell the difference of good and bad cohesion. These metrics are describe large values are good and low values are bad. The metrics TTC and LCC are closely related to the idea of LCOM4 even though there are some differences. The higher TTC and LCC the more cohesive and thus better the class. For TTC and LCC we can only consider visible methods whereas the LCOM4 considers all methods. A method is visible unless it is Private. A method is visible also if it implements an interface or handles an event. In other respects, we use the same definition for a method as for LCOM4. The components are related in two methods that are if they both access the same class-level variable or the call trees starting at a and b access the same class-level variable. For the call trees we consider all procedures inside the class, including Private procedures. If a call goes outside the class, we stop following that call branch. When 2 methods are related this way, we call them directly connected. When 2 methods are not directly connected, but they are connected via other methods, we call them indirectly connected. Example: A - B - C are direct connections. A is indirectly connected to C (via B).

**TCC and LCC definitions**

- Maximum number of possible connections are denoted by \( NP \) that is \( =N*(N-1)/2 \) where \( N \) is the number of methods.
- Number of direct connections or number of edges in the connection graph is denoted by \( NDC \).
- Number of indirect connections denoted by \( NID \).
- Tight class cohesion \( TCC = NDC/NP \) and TCC is in the range of 0..1.
- Loose class cohesion \( LCC = (NDC+NIC)/NP \) and LCC is in the range 0..1. TCC<=LCC.
- The higher TCC and LCC, the more cohesive the class is.
- What are good or bad values? According to the authors, TCC<0.5 and LCC<0.5 are considered non-cohesive classes. LCC=0.8 is considered "quite cohesive". TCC=LCC=1 is a maximally cohesive class: all methods are connected.

3.4 Cohesion diagrams: Cohesion diagrams visualize class cohesion and it displays the procedures of a module and use that variables which having same module. The diagram lets you understanding the procedure-to-procedure and procedure-to-variable relationships within a single module. The Cohesion diagram uses the same analysis as the LCOM4 cohesion metric. The module is split into related groups which can be based on procedure calls and variables shared. The more groups is less cohesion. The LCOM4 value [1], [2] etc. is displayed next to each module. Even though LCOM4 is a class metric and less useful for standard modules, forms or Structures, Cohesion diagrams are available for all module types.
3.5 Non-cohesive classes: The Non-cohesive class report suggests which classes should be split and how. This specialized report detects classes that could be split into smaller classes. The report lists class of low cohesion. These classes consist of 2 or more unrelated parts or sub-components. The parts don't call each other nor do they access common class-level variables. As they have no common functionality, they could be in separate classes. For each sub-component in a non-cohesive class, the report lists the methods and variables belonging to that sub-component. Consider splitting the class into these sub-components. This is not always possible. You can read more about the logic behind non-cohesive classes and components under the cohesion metrics. To visualize cohesion, try the Cohesion diagrams in Enterprise Diagrams.

4. PROBLEM FORMULATION
This paper introduces criteria for assigning cohesion values to classes that exhibit special cases, such as having no attributes, no parameter types for the methods, or fewer than two methods in the class. These cases prevented us using some of the existing cohesion metrics. We used our value-assignment criteria to assign values to the special cases for some cohesion metrics. This value-assignment allows applying these metrics for all classes in object-oriented software systems. In the case of cohesion metrics there are several points where these metrics show undefined values. This study will help in improving the applicability of metrics on classes of special cases including the inheritance. This problem will include four scenarios:
(i) Both inherited methods and attributes are excluded,
(ii) Only inherited attributes are excluded,
(iii) Only inherited methods are excluded, and
(iv) Both inherited methods and attributes are included. By assigning values to special cases including the factor of inheritance can improve the applicability of metrics to much more extent.

5. RESULT & DISCUSSION
The results show that our assigned values for the undefined cases do not violate the key cohesion properties and considerably improve the ability of the metrics to explain the presence of faulty classes and may therefore improve their ability to indicate the quality of the class design. The result is determined the quality of software that the properties and quality of attributes. This paper explains the criteria for class cohesion that contain the metrics and properties. Our metric does not distinguish between attributes and methods of different accessibility levels i.e., public, private, and protected. These cases prevented us using some of the existing cohesion metrics. There are several approaches to validate cohesion metrics. In this paper, we followed a widely used approach in which the cohesion is studied in terms of metric capability and which is an indirect way to assess them as quality indicators.

REFERENCES


