“Cohesion Techniques & Its Analysis using LCOM4”

Dr. Ashish Jolly¹, Mr. Shakti Kumar²
¹Department of Computer Science, Govt. College Barwala (Panchkula), Haryana, India
e-mail:ashishjolly76@gmail.com
²Department of Computer Science, Govt. P. G. College Ambala Cantt, Haryana, India
e-mail:shaktikumarsanjay@gmail.com

Abstract—designing complex software at once is difficult but if we divide the problem into parts or modules then it will be easier to manage the whole project. Once the module are considered it is important to understand inter as well as intra module dependency. Cohesion refers to the degree to which the elements of the modules belong together. Coupling defines the dependency of one module on one or more other modules. In a good design methodology more cohesion is required while coupling should be as low as possible. Different types of coupling are available from which some are desirable while the design of the software.

Keywords—software coupling; cohesion; Complexity Analysis; Dependency; Comparison.

INTRODUCTION
When we develop the software at once is difficult and time consuming. So in order to reduce complexity we build the project in parts or modules, such a design technique is known as modular design techniques. When we design software using modules reduces the overall complexity of software development. Once the modules are being designed they are connected together to form the overall system. The connectivity is done in such a manner so that coupling is less and cohesion is more.

COHESION AND ITS TYPES
Cohesion is the degree to which elements inside a module belong to each other. In cohesion we consider the statements inside the given module and the way they all are working together to achieve a specific goal. Types of cohesion are given below.

Coincidental cohesion
It occurs due to arbitrary grouping of parts for module. This kind of cohesion is considered as worst, it is because the elements are just kept together inside the module without considering their contribution to module.

Logical cohesion
In Logical cohesion the parts of module are grouped together because logically they are performing same job even when they are different by nature for example functionality of keyboard and mouse vary but they both can be logically grouped inside the module designed to accept input from the user.

Temporal cohesion
Temporal cohesion occurs when the parts of a given module are processed at a particular instance of time when the program is executing in the memory. For example when an exception is caught then error is reported to user and error log is also created.

Procedural cohesion
The parts or statements of a program which are grouped under this cohesion have to follow a given sequence. For example before displaying the contents of a file we must first ensure that the read permissions are provided for the same.

Communicational cohesion
Only those parts which operate the same data are kept under communicational cohesion.

Sequential cohesion
This cohesion groups those parts together in which output of one part is considered as input of the other part.

Functional cohesion
This cohesion is considered as best because all the parts of the module contribute to a single well defined task.

LCOM4 A COHESION METRIC
Cohesion metrics helps us to find out how well the methods of a class are related to each other. A cohesive class performs one function while non cohesive class performs more than one function which are totally unrelated.

A. Lack of cohesion of methods (LCOM)
There are several LCOM ‘lack of cohesion of methods’ metrics. There are four variants: LCOM1, LCOM2, LCOM3 and LCOM4. The LCOM1, LCOM2 & LCOM3 are used for various object oriented languages while LCOM4 is used for Visual Basic
systems. LCOM4 is considered best because it considers property accessors.

B. LCOM4 (Hitz&Montazeri) recommended metric

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. There should be only one such a component in each class. If there are two or more components, the class should be split into so many smaller classes.

In the measure of LCOM4 Methods a and b are related if:

1. they both access the same class variable, or
2. a calls b, or b calls a.

After determining the related methods, we draw a graph linking the related methods to each other. LCOM4 equals the number of connected groups of methods.

- LCOM4=1 indicates a cohesive class, which is the "good" class.
- LCOM4>=2 indicates a problem. The class should be split into so many smaller classes.
- LCOM4=0 happens when there are no methods in a class. This is also a "bad" class

For Example Consider a class DEMO with A(),B(),C(),D() & E() as five methods and x,y as two data members.

| DEMO |
| +x:int |
| +y:int |
| +A():void |
| +B():void |
| +C():void |
| +D():void |
| +E():void |

**Fig1. class diagram**

In the DEMO class method A() access method B() which in turn access data member x. Method C() access the data member y while method D() access both method E() and data member y as shown in figure 2.

break it into two parts and they are {A,B,x} and {C,D,E,y} as given below so that both the parts will have value of LCOM4=1

**Fig.3 Improving LCOM metric by splitting the class**

**Fig.4 providing more cohesive design for the given class**

Consider a class that encapsulates 3 variables and provides 3 properties to access each of these 3 variables. Such a class displays low cohesion, even though it is well designed. The class could well be split into 3 small classes, yet this may not make any sense.

**CONCLUSION**

After analysis of various cohesion techniques we observed that High cohesion is desirable since it promotes encapsulation. As a drawback, a highly cohesive class has high coupling between the methods of the class, which in turn indicates high testing effort for that class. Low cohesion indicates inappropriate design and high complexity. It has also been found to indicate a high likelihood of errors. The class should probably be split into two or more smaller classes.

**REFERENCES**


