

*Software Measurement. SOEN 6611 2014/4 D*

*Dr. Peter Rigby*

**Assignment 3**

**Coupling and Cohesion Measures**

|  |  |
| --- | --- |
| **Team members information (in no particular order)** | |
| **Name** | **SID** |
| Ali Sangari | 6816304 |
| Hojabr Sattari | 6435807 |
| Bhaskar Baddam | 6761038 |

# LCOM (Lack of cohesion in methods)

This metric measures the correlation between the methods and the attributes of a class. High cohesion indicates good class subdivision. Lack of cohesion or low cohesion increases complexity. Classes with low cohesion could probably be subdivided into two or more subclasses with increased cohesion

The LCOM version that we use in Understand API is LCOM4 (Hitz & Montazeri, 1995). It is calculated as the number of disjoint components in the graph that represents:

* each method as a node
* the sharing of at least one attribute between two methods as an edge
* the presence of a method call between two methods as an edge

The cohesiveness of a class can be determined from the LCOM value as follows:

LCOM4 = 1 indicates a cohesive class, which is a "good" class.

LCOM4 >= 2 indicates a problem. The class should be split into smaller classes.

LCOM4 = 0 happens when there are no methods in a class. This is also a "bad" class.

However, SciTools Understand uses the following to define LCOM:

100% minus average cohesion for class data members. Calculates what percentage of class methods use a given class instance variable. To calculate, average percentages for all of that class’es instance variables and subtract from 100%. A lower percentage means higher cohesion between class data and methods. [**Research:** Chidamber & Kemerer – Lack of Cohesion in Methods (LCOM/LOCM)]

To calculate LCOM4 (Hitz & Montazeri, 1995), we need the following metrics,

1. Number of functions/methods in a class (given by Understand as "CountDeclFunction")
2. Number of method pairs in class with at least one instance variable that they commonly use or define in their body.
3. Number of method pairs in class that have at least one instance method that they commonly call in their body.

From the three metrics that are mentioned above, Understand only provides the first item. To perhaps a solution would be to calculate the two metrics (items 2 and 3) by parsing the code using a home grown tool.

The following links show a list of available metrics form SciTools and a stackoverflow question as part of our research.

* List of metrics made available through Understand API: <https://scitools.com/support/metrics_list/>
* Stackoverflow question and reply: <http://stackoverflow.com/questions/29464392/how-to-calculate-cbo-and-lcom-for-a-class-using-python-and-scitools-understand-a>

# CBO (Coupling between object classes)

This metric represents the number of classes coupled to a given class. This coupling can occur through method calls, field accesses, inheritance, return types, and exceptions. High CBO is undesirable, high coupling has been found to indicate fault-proneness.

We used CBO (Chidamber & Kemerer) in Understand API. It is calculated as the number of unique classes a class references excluding base classes and nested classes.