# Codebase Metrics

## Metrics Set

**Average values of metrics for the *ganttproject*:**

* CBO = 10,94
* DIT = 0,57
* LCOM = 2,46
* NOC = 0,56
* RFC = 15,42
* WMC = 13,21

## Explanation

**CBO (Coupling Between Object Classes):** Used to measure coupling between Classes, two classes are coupled when methods declared in one class use methods or instance variables defined by the other. The use relationship can go either way, i.e. both use or be used cases are taken into account, but only once. In addition a high CBO indicates excessive coupling between objects, which is detrimental to modular design and prevents reusability, since to change the behavior of the object implies having to change the coupled objects.

**DIT (Depth of Inheritance Tree):** This metric calculates the maximum inheritance path from class to root class, the deeper a class is in the hierarchy, the more methods and variables it is likely to inherit, making it more complex. In other words, the higher the DIT value, the deeper the tree is, which indicates greater design complexity and consequently can increase the appearance of bugs and decrease quality.

**LCOM (Lack of Cohesion of Methods):** LCOM is used to measure cohesiveness in a class. It works by taking a pair of methods in a class and adding 1 to Q if those methods share at least one variable access, or adding 1 to P if they don’t, so that the final value of LCOM is:

LCOM = P – Q if P > Q

LCOM = 0 Otherwise

If LCOM is 0 it indicates that the class is cohesive, if it’s not it usually means that the class can be split into two or more classes. A high LCOM value indicates disparateness in a class, meaning that it might be attempting to achieve many different objectives, resulting in less predictable behavior. Such classes are usually more prone to errors and are harder to test.

**NOC (Number of Children):** Used to calculate the immediate child classes derived from a base class i.e., unlike DIT which measures depth, this measures the breadth of a class hierarchy. A high NOC can indicate several things:

* High reuse of base class
* Improper abstraction of the parent class
* Misuse of sub-classing

**RFC (Response for a Class):** This metric calculates the number of methods in the response set of a given class, which is a set of methods that can potentially be executed in response to a message received by an object of that class. A higher value here usually indicates more bugs, since classes tend to be more complex and harder to understand.

**WMC (Weighted Methods Per Class):** To put it simply this metric counts the methods of a class, a high value here should bring the focus of our attention to the given class which may have a greater impact on derived classes and consequently the whole system, it also tends to have more bugs.

## Potential Trouble Spots

The **net.sourceforge.ganttproject.language.GanttLanguage** has the maximum CBO value (133) in the project, which can be a problem later on because due to its large coupling number changing this class or one of its couple classes can cause problems in all the others, thus decreasing modularity.

Theclass **net.sourceforge.ganttproject.export.ConsoleUIFacade** holds the maximum LCOM value (34) and even though we know it is still incomplete it is important to give it attention since LCOM represents lack of cohesion between methods, which may suggest that we could break this class into several classes with more cohesion.

Given the high number of direct NOC children (86), and a fair value of WMC (65), it indicates to us that **net.sourceforge.ganttproject.action.GPAction** should be reviewed as it is a relatively complex class that can affect many other classes and is therefore a potential trouble spot.

The **net.sourceforge.ganttproject.GanttProject** is another trouble spot, it has the highest RFC value in the project (282) and also the second highest WMC value (156), making it a very complex class with an excessive size, factors that make it difficult to understand and promote a build-up of bugs. It should be broken down into several simpler classes.

Last but not least, we also have the **net.sourceforge.ganttproject.task.taskManagerImpl** which holds the maximum WMC value in the project (173), given its excessive number of methods this class has, it becomes much more difficult to understand than necessary and consequently promotes an accumulation of bugs and errors.

***Relatability to Code Smells***

**CBO** – a high value of CBO is usually related to a Inappropriate Intimacy code smell since this metric measures the coupling that is happening between classes.

**LCOM** – usually a project should strive to have a low value of LCOM since this likely means that the class is very cohesive, however sometimes cohesive classes might be related to God Classes (Large Class code smell) which are meant to be avoided.

**DIT and NOC** – both metrics can be related to the Shotgun Surgery code smell since they both measure, in two different ways, the amount of inheritance that is happening in the project.

**WMC** – a high value for WMC can be related to the Large Class code smell since this metric is used to measure how many methods each class has. If the metric value is high then it likely means that the class is too big (i.e has too many methods) and could potentially split into two or more classes.