# Chidamber-Kemerer metrics

## **Coupling**

A classes coupling is defined by the number of classes that depend on it, or that it depends on. On the collected metrics we can see that, as expected, central classes to the program have a high coupling, while outer classes that are used less have a lower coupling. For example, the GanttLaguage class has the highest coupling, since it’s used by a lot of other classes.

Generally high coupling should be avoided, since it complicates the code, like in the GanttProject class, that has the second highest coupling since it uses a lot of other classes. We can begin to see the code smell “god class” here, and this class could have some responsibilities moved to other parts of the code, reducing the number of classes it uses, and it’s coupling.

## **Inherence depth**

This metric measures the number of inheritance steps between a given class and the java Object class. Highly specialized classes have a high inherence depth, such as the GanttTreeTable, while more standalone classes that don’t extend anything don’t.

## **Number of children**

This metric is closely related to the one before, measuring the number of classes that extend a given class. In the code the most extended class is the GPAction class, that is extended by 86 separate classes.

## **Lack of cohesion in methods**

Cohesion is the “strength” of a given class so that it’s responsibilities can’t be easily split into new different classes. We measure this in the relation of methods the methods in the class, that is, if they share a variable, or call each other. If they all the methods in a class are highly related, then the class Is strongly cohesive, and the value is close to 1.

In the program we can see some classes that have a high lack of cohesion, and once again, the ganttProject class comes up, which is more proof that it might be a "god class” and needs to be separated into smaller classes.

## **Class response**

The class response is the number of methods it can possibly call, be it their methods, or other class methods that it can access. Generally, we want a small class response, since it increases the complexity of the class, and decreases cohesion and stability. Once again, the highest class response in the project belongs to the ganttProject class since it handles so many different classes, and has so many responsibilities, which might be a problem.

## **Method complexity**

This metric measures the cyclomatic complexity of the methods in each class, which is the number of independent paths the program can take, for example an if statement would have a complexity of 2. Lower complexity methods easier to understand and to test. In the program, the higher complexity classes are the ones responsible for tasks, and once again, the ganttProject class.