**Codebase Metrics**

Martin Package Metrics

Values (Average package):

Ce:239,98

Ca:239,98

I:0,50

A:0,24

D:0,30

Explanation:

Ce (Efferent Coupling): Used to measure interrelationships between classes, gives the number of classes in other packages that classes from a specific package depend upon (outgoing dependencies), allows us to measure the vulnerability of a package to changes in packages it depends on. High values indicate that the package is unstable because there could be changes in the packages it depends on.

Ca (Afferent Coupling): Similar to Ce, this measures the number of classes from other packages that depend on classes from a specific package (incoming dependencies), also allows us to measure the sensitivity of the packages to changes in the package they depend on. High values indicate stability of the package, for it cannot have any substantial changes.

I (Instability): Measures the relative susceptibility of a package to changes, it is defined according to the formula: Ce / (Ce + Ca). Values close to 1 mean that the package has more outgoing dependencies and, therefore, is more susceptible to changes. Values close to 0 mean that the package has more incoming dependencies and, therefore, is less susceptible to changes. Preferred values for I are 0 – 0,3 and 0,7 – 1 (packages should either be very stable or very unstable).

A (abstractness): Measures the degree of abstraction of a package, it is defined by the following formula: Ta / (Ta + Tc), where Ta is the number of abstract classes and Tc is the number of concrete classes. Preferred values are close to the extremes (0 and 1), packages that are stable (metric I close to 0) should also be abstract, while unstable packages (metric I close to 1) should be concrete.

D (Normalized Distance from Main Sequence): Used to measure the balance between stability and abstractness, it is calculated using the following formula: D = |A + I -1|. The value of this metric should be as low as possible, so that the components are located close to the main sequence.

Find potential trouble spots

The spots we want to look out for are packages with a lot of dependencies and a relatively high value for D, these packages don’t have a good balance of abstraction and instability and therefore will require a lot of work if changes are needed:

* Net.sourceforge.ganttproject.util.collect (Ca = 199 D = 1)
* biz.ganttproject.core.chart.canvas (Ca = 766 D = 0,79)
* net.sourceforge.ganttproject.language (Ca = 940 D = 0,68)
* biz.ganttproject.core.time (Ca = 1181 D = 0,58)
* net.sourceforge.ganttproject.task (Ca = 4434 D = 0,53)
* biz.ganttproject.core.option (Ca = 1491 D = 0,53)

There are other packages whose D value is high (1 or close), but these contain a relatively small number of dependencies, so the potential trouble these could generate is not as high as for the ones mentioned above.

Relate with code smells

Large classes normally have a lot of dependencies, because of this, these classes end up being very unstable for a lot of changes could be necessary, and they are not very abstract.