 Rita Silva 57960

**Martin packaging metrics**

This metric is used to measure interdependence between classes.

It is characterized by:

* **Afferent couplings(Ca)**

Measures the total number of external classes coupled to classes of a package because of incoming coupling.

All classes are counted only once.

If the package does not contain any class or external classes do not use any of the classes of the package, then the value of Ca is zero.

Preferred values for the metric Ca are in the range of 0 to 500.

In the graph it’s seen that the number of incoming dependencies isn’t unexpected, since the hight number suggests high stability.

* **Efferent couplings (Ce)**

Number of classes within the package that depend on external classes.

That is Afferent Couplings are Arriving references, Efferent Coupling are Exiting or Escaping references.

References to test classes and library classes are not included.

If the classes in the same package do not use any external class linkage, then the value of Ce is zero.

The high value of the metric indicates instability of a package, that meaning that any change change in one of the numerous external classes can cause the need for changes to the package.

Ce should be between 0 and 20.

With the graph we can see that there are a considerable number of classes that do depend on external ones to their package in comparation with the values of the graph related to afferent coupling.

* **Abstractness(A)**

Ration between number of abstract classes (interfaces included) in the package being analysed and the total number of classes in that package.

This ranges from 0 to 1. 0 indicates a completely concrete package while 1 na abstract one.

In the graph calculated we can see that there are more packages that are concrete than there are abstract ones.

* **Instability (I)**

Ration between Ce and Ce + Ca.

This indicates the flexibility of the package in relation to change.

Ranges from 0 to 1, where 0 indicates a completely stable package while 1 a unstable one.

From the graph we can conclude that there are a large number of unstable packages, going over 60 even. This means that all those classes have a reason to change, they are very unstable and that can be caused by the dependencies that the classes have between one another, in other words, if one class changes those that depend on it also need to change, therefore the packages are unstable.

* **Distance from main sequence(D)**

Calculated by the formula |A + I –1|.

This indicates the balance between A and I.

This metric has a range of [0,1], where the closer D is to 0, the better.

A=0 and I=0 is a highly stable and concrete category, as such it is not desirable because it is rigid. It cannot be extended because it is not abstract. And it is very difficult to change because of its stability. A=1 and I=1 is also undesirable (perhaps impossible) because it is maximally abstract and yet has no dependents. It, too, is rigid because the abstractions are impossible to extend.

In this case we can see that althought we see a lot of packages having D closer to zero (or even zero), there are still many closer to 1 (or even reaching one). This means that all of those need to be reexamined and recosntructed.

**Conclusion**





Overall this is a “good” pattern for packages, it would be better if we solved the packages with the D very close to 1 (from 0.6 to 1) althought in average its value is closer to 0, we should also minimize the value of Ce, it’s average is 317,68 (way above 20), this can be achieved if we dealt with the two shotgun surgery code smell we found ( because this kind of code smell prevents change and therefore has a higher instability), and the smells known as message chains too ( this code smell contrinutes to excess coupling between classes).