**Martin Packaging Metrics**

**Rodrigo Monteiro Suzana 63069**

**(suzana2314)**

By calculating this metric, using the plugin in Idea IntelliJ, we obtain an csv file with the following columns:

* Abstractness;
* Afferent Couplings;
* Efferent Couplings;
* Distance from the main sequence;
* Instability;

**Efferent Coupling (Ce)**

This metric measures the relationship between classes. I.e., the number of classes in a package that depend on classes located in other packages.

A diagram of a diagram

Description automatically generated

Figure 1

The metric Ce of this example would be 2, because class X has 2 dependencies to other 2 classes.

**Afferent Coupling (Ca)**

This metric measures incoming dependencies between classes. It enables the measurement of the sensitivity in the remaining packages to changes in the analyzed package.

A diagram of a diagram

Description automatically generated

The metric Ca of this example would be 1, because class X has 1 incoming dependency (from *Example Class*).

**Instability**

This metric measures the susceptibility of class to changes. This metric is defined according to the formula:

In our example shown in Figure 1, the instability would be around 0.67. This metric helps identify two types of components:

1. Packages with lots of connections going out and few coming in (value closer to 1) are less stable because they can be easily changed.
2. Packages with lots of connections coming in and few going out (value closer to 0) are harder to change because they have more responsibilities.

**Abstractness**

This metric measures how abstract a package is, somewhat similar to instability. Abstractness refers to the ratio of abstract classes to all classes within the package, and can be easily calculated using the following formula:

Where is the number of abstract classes in a package and is the number of concrete classes in a package.

**Distance from the main sequence**

This metric measures the balance between stability and abstractness, using the following formula:

The goal is to keep this metric as low as possible to keep components close to the main sequence. There are two extreme, unfavorable cases:

1. When A = 0 and I = 0, a package is extremely stable and concrete. This is undesirable because the package is rigid and cannot be extended.
2. When A = 1 and I = 1, it's a nearly impossible situation. A completely abstract package must have some connection to the outside so that an instance implementing the functionality from the abstract classes in this package could be created.

The data extracted from the code base using the plugin makes it easy for us to spot certain code issues. For instance, when we analyze the instability column and notice values closer to 1, it suggests a high number of "connections to the outside," which indicates strong interdependencies among classes. This may imply that classes within a package are deviating from their core purpose.

Conversely, when we encounter an instability value closer to 0, indicating a package with low visibility (fewer connections to the "outside"), it might signify that some classes within the package serve as simple data classes or have little relevance to the codebase.

However, it's essential to exercise caution when relying solely on instability values. For instance, there's a utils package with an instability value of 0. This doesn't necessarily imply issues with the code; it simply means that the classes within the package lack external connections.