Martin Packaging Metrics

Collected Metrics Explanation:

Abstractness (A): $\frac{\text{number of abstract classes and interfaces in a package}}{\text{total number of classes and interfaces in a package}}$

This value ranges between 0 and 1. The closer it is to 1, the more abstract the package is.

Afferent Coupling (Ca):

The number of classes in other packages that are dependent on classes from this package. The higher the number, the higher the responsibility this package has.

Efferent Coupling (Ce):

The number of classes in this package that are dependent on classes from other packages. The higher the number, the more dependent on other packages this package is.

Distance from the main sequence (D): $|\text{abstractness} + \text{instability} - 1|$

This value ranges between 0 and 1. A package is optimal (on the main sequence) if it's balanced between abstractness and instability. The closer it is to 1, the further from the main sequence the package is.

Instability (I): $\frac{\text{efferent coupling}}{\text{efferent coupling} + \text{afferent coupling}}$

This value ranges between 0 and 1. The closer it is to 1, the more unstable the package is (regarding its adaptability to change; for example, losing access to other packages).

JabRef's packages' ***abstractness***:

Seeing as JabRef's average abstractness is 0.06, on a range from 0.00 to 1.00, we can conclude JabRef used a very small percentage of abstract classes and interfaces throughout the whole project (6%).

JabRef's packages' ***afferent and efferent couplings and instability***:

JabRef's afferent and efferent couplings total number is the same for both, 58621. Therefore, we can conclude JabRef's instability on each package is, on average, 0.5, making JabRef as a whole as stable as it is unstable. This number could be improved.

JabRef's packages' ***distance from the main sequence***:

Analyzing JabRef's distance from the main sequence, we can understand its packages are, in general, somewhat balanced between abstractness and instability, since, on average, optimality is 0.3 (the best possible value being 0.0).

These metrics do not seem to be related to any code smell we identified.

Francisco Pires, nº 58208