John McCormack

Brian Padilla

Dharmin Shah

SchoolBud MS 4: Metrics

1. *McCabe's Cyclomatic Complexity*: This metric is an indication of the number of 'linear' segments in a method (*i.e.*sections of code with no branches) and therefore can be used to determine the number of tests required to obtain complete coverage. It can also be used to indicate the psychological complexity of a method. A method with no branches has a Cyclomatic Complexity of 1 since there is 1 arc. This number is incremented whenever a branch is encountered. In this implementation, statements that represent branching are defined as: 'for', 'while', 'do', 'if', 'case' (optional), 'catch' (optional) and the ternary operator (optional). The sum of Cyclomatic Complexities for methods in local classes is also included in the total for a method. Cyclomatic Complexity is a procedural rather than an OO metric. However, it still has meaning for OO programs at the method level. The maximum value for the metric is 10.
2. *Efferent Couplings:* This metric is a measure of the number of types the class being measured 'knows' about. This includes: inheritance, interface implementation, parameter types, variable types, thrown and caught exceptions. In short, all types referred to anywhere within the source of the measured class. A large efferent coupling can indicate that a class is unfocussed and also may indicate brittleness, since it depends on the stability of all the types to which it is coupled. When used to measure couplings between packages (coming in a later release) this measure can be used together with others to calculate 'abstractness', 'stability' and 'distance from the main line'. Efferent coupling can be reduced by extracting classes from the original class (*i.e.* decomposing the class into smaller classes).
3. *Lack of Cohesion in Methods:* Cohesion is an important concept in OO programming. It indicates whether a class represents a single abstraction or multiple abstractions. The idea is that if a class represents more than one abstraction, it should be refactored into more than one class, each of which represents a single abstraction. Despite its importance, it is difficult to establish a clear mechanism for measuring it. This is probably due to the fact that good abstractions have deep semantics and a class that is clearly cohesive when viewed from a semantic point of view may not be so when viewed from a purely symbolic point of view. As an aside, the somewhat inelegant name is due to the wish to have lower metric values representing a 'better' situation.

4. *Weighted Methods Per Class:* This metric is the sum of complexities of methods defined in a class. It therefore represents the complexity of a class as a whole and this measure can be used to indicate the development and maintenance effort for the class. In this implementation, the measure of complexity of methods is McCabe's Cyclomatic Complexity. Classes with a large Weighted Methods Per Class value can often be refactored into two or more classes.
5. *Depth of Inheritance Tree:* This metric provides for each class a measure of the inheritance levels from the object hierarchy top.