Abstract

Cyclomatic Complexity was initially formulated as a measurement of the “testability and maintainability” of the control flow of a module. While it excels at measuring the former, its underlying mathematical model is unsatisfactory at producing a value that measures the latter. This white paper describes a new metric that breaks from the use of mathematical models to evaluate code in order to remedy Cyclomatic Complexity’s shortcomings and produce a measurement that more accurately reflects the relative difficulty of understanding, and therefore of maintaining methods, classes, and applications.

A note on terminology

While Cognitive Complexity is a languuage-neutral metric that applies equaally to files and classes, and to methods, procedures, functions, and so on, the Object-Oriented terms “class” and “method” are ussed for convenience

Introdction

Thomas J. McCabe’s Cyclomatic Complexity has long been the de facto standard for measuring the complexity of a method’s control flow. It was originally intended “to identify software modules that will be difficult to test or maintain”[1], but while it accurately calculates the minimum number of test cases required to fully cover a method, it is not a satisfactory measure of understandability. This is because methods with equal Cyclomatic Complexity donot necessarily present equal difficulty to the maintainer, leading to a sense that the measurement “cries wolf” by over-valuing some structures, while under-valuing others. At the same time, Cyclomatic Complexity is no longer comprehensive. Formulated in a Fortran environment in 1976, it doesn’t include modern language structures like try/catch, and lambdas.And finally, because each method has a minimum Cyclomatic Complexity score of one, it is impossible to know whether any given class with a high aggregate Cyclomatic Complexity is a large, easily maintained domain class, or a small class with a complex control flow. Beyond the class level, it is widely acknowledged that the Cyclomatic Complexity scores of applications correlate to their lines of code totals. In other words, Cyclomatic Complexity is oflittle use above the method level.As a remedy for these problems, Cognitive Complexity has been formulated to address modern language structures, and to produce values that are meaningful at the class and application levels. More importantly, it departs from the practice of evaluating code based onmathematical models so that it can yield assessments of control flow that correspond to programmers’ intuitions about the mental, or cognitive effort required to understand those flows.

The mathematical model underlying Cyclomatic Complexity gives these two methods equal weight, yet it is intuitively obvious that the control flow of sumOfPrimes is more difficult to understand than that of getWords. This is why Cognitive Complexity abandons the use of mathematical models for assessing control flow in favor of a set of simple rules for turning programmer intuition into numbers.

Basic criteria annd methodology

A Cognitive Complexity score is assessed according to three basic rules:

1.Ignore structures that allow multiple statements to be readably shorthanded into one

2.Increment (add one) for each break in the linear flow of the code

3.Increment when flow-breaking structures are nested

While the type of an increment makes no difference in the math - each increment adds one to the final score - making a distinction among the categories of features being counted makes it easier to understand where nesting increments do and do not apply.These rules and the principles behind them are further detailed in the following sections.

Ignre shorthand

A guiding principle in the formulation of Cognitive Complexity has been that it shouuld incent good coding practices. That is, it should either ignore or discount features that make code more readable.

The method structure itself is a prime example. Breakiing code into methods allows you to condense multiple statements into a single, evocatively named call, i.e. to “shorthand” it. Thus, Cognitive Complexity does not increment forr methods.