Structural Complexity

Introduction

We can think of program structure as having two parts:

- Control-flow structure
- Data-flow structure

Control-flow looks at the way that programs address the sequence in which instructions are executed; the step-by-step, iterative, and looping make-up of programs.

Data-flow looks at the trail of a data item as it is read, written, manipulated, or created by a program.

Structural Complexity

The complexity of program structure is intuitively connected to effects such as readability of code, understandability, effort, testability, reliability and maintainability. McCabe proposed that program complexity could be measured by the 'cyclomatic number'.

Cyclomatic number - many refer to it as *cyclomatic complexity* - is derived from a program's control flow graph. Cyclomatic complexity is equal to the number of linearly independent paths through the program - in practice, the metric is usually equivalent to one plus the number of decisions in the program.

Despite their widespread use, McCabe's metrics have been criticised on both empirical and theoretical grounds. Empirically it has been claimed that they are no better indicators of complexity than LOC since they are no better at predicting effort, reliability, or maintainability. Theoretically, it has been argued that the metrics are too simplistic; for example, McCabe's metric is criticised for failing to take account of data-flow complexity or the complexity of unstructured programs. This has led to numerous efforts that try to characterise different views of complexity, such as metrics involving the modelling both control flow and data flow.

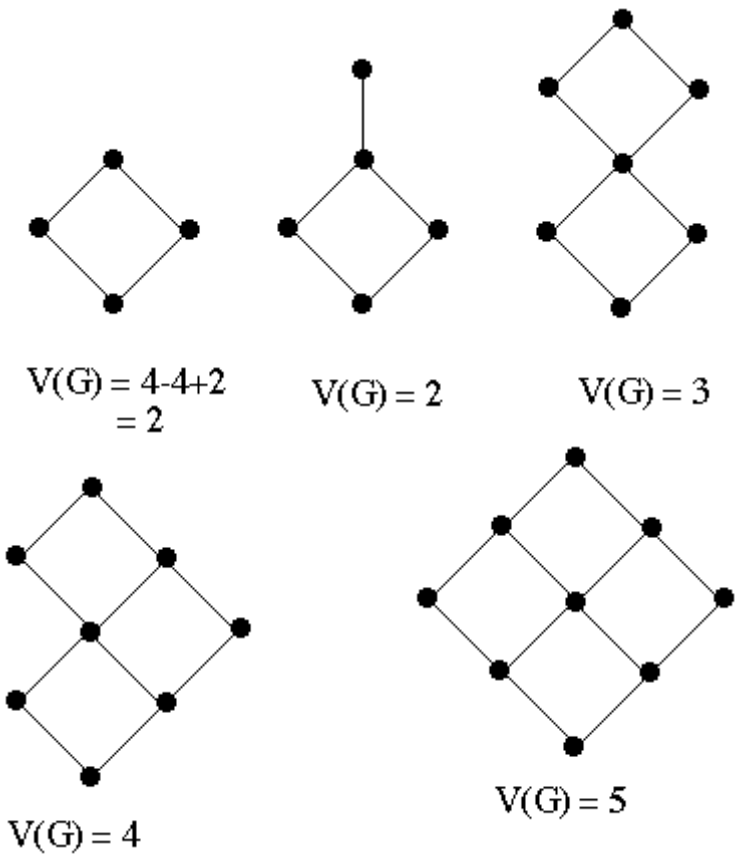
Cyclomatic Complexity

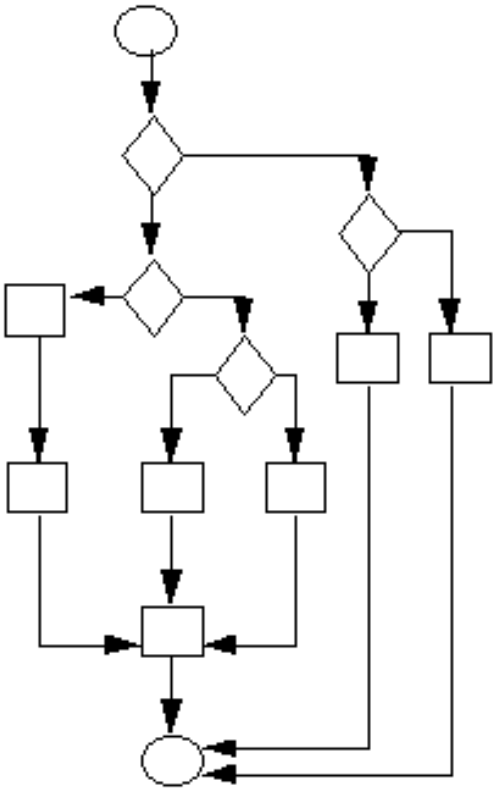
The cyclomatic complexity is measured from a program's control flow graph:

no. edges - no. nodes +2

Let G be a connected, directed acyclic graph (DAG) with e edges and n nodes
The cyclomatic complexity (or cycle rank) of G is defined as:

$V(G) = e - n + 2$





If G is the control flowgraph of program P and G has e edges (arcs) and n nodes

$$v(P) = e-n+2$$

v (P) is the number of linearly independent paths in G

here e = 16 n =13 v (P) = 5

More simply, if d is the number of decision nodes in G then

$$v(P) = d+1$$

In the diagram - the flow graph - above;

- the number of decision nodes is 4, hence cyclomatic complexity is 5,
- the number of edges - lines going from one node to another - is 16, and the number of nodes is 13, hence the cyclomatic complexity is (e - n + 2) 16 -13 + 2 = 5.