An Analysis of the Mc Cabe Cyclomatic Complexity Number

Miguel Lopez

Alain Abran

Naji Habra

Agenda

- Problem Statement
- Cyclomatic Number in Graph Theory
- Application of the Cyclomatic Number in Software
- Analysis of McCabe’s Number
- Conclusion

Problem Statement

- Some measurement methods used in the software industry are still not well understood.
- Although these measurement methods are correctly applied by practitioners, there remain ambiguities in their design and corresponding interpretation.

Problem Statement

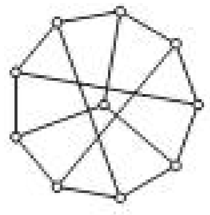
- McCabe Cyclomatic Number is one of these misunderstood measure.
- McCabe Cyclomatic Number is often applied in the industry, but it remains some misconception in the design measurement itself.

Cyclomatic Number in Graph Theory

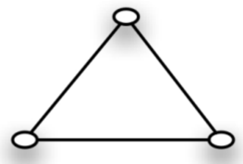
- Some definitions of Graph theory are necessary to explain the McCabe Cyclomatic Number.
- Indeed, McCabe attempts to apply some concepts of Graph theory into Software Measurement.
- We propose to better analyze the McCabe Cyclomatic Number.

Cyclomatic Number in Graph Theory

- *A Simple Graph* is a (usually finite) set of vertices V (or nodes) and a set of unordered pairs of distinct elements of V called edges.

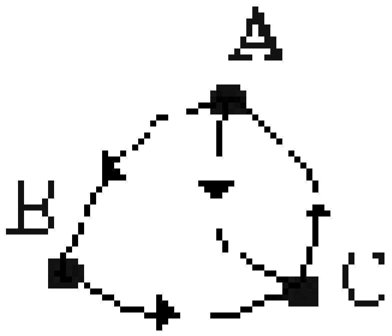


- *A Cycle Graph* is a path that begins and ends with the same vertex.



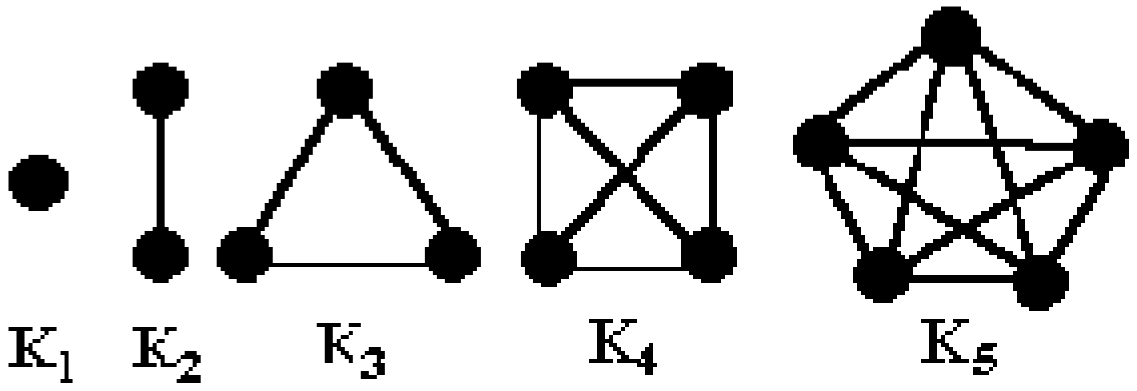
Cyclomatic Number in Graph Theory

- *A Directed Graph* (also called a digraph or quiver) is a (usually finite) set of vertices V and set of ordered pairs (a,b) (where a, b are in V) called edges. The vertex a is the initial vertex of the edge and b the terminal vertex.
- A graph in which the edges are directed.



Cyclomatic Number in Graph Theory

- *A Strongly Connected Graph* is a directed graph that has a **path** from each vertex to every other vertex.



Cyclomatic Number in Graph Theory

- *The Cyclomatic Number* of a strongly connected directed graph is equal to the maximum number of linearly independent cycles. Equation 1 gives the Cyclomatic Number, v(G):

$$v(G) = e - n + p (1)$$

where there are e edges, n vertices and p separate components.

Application of the Cyclomatic Number in Software

- McCabe suggests to consider the program as a directed graph.
- The program is modeled as a ***control flow graph***.
- Each **vertex** in the graph represents a **basic block**. **Directed edges** are used to represent **jumps** in the control flow.

Application of the Cyclomatic Number in Software

- There are two specially designated blocks:
  - the **entry block**, through which control enters the flow graph
  - the **exit block**, through which all control flow leaves.
- But, program control flow graphs are not strongly connected, but they become so when a virtual edge is added connecting the exit node to the entry node.

Application of the Cyclomatic Number in Software

- So, Equation 1 becomes:

$$v(G) = e - n + p + 1 \quad (2)$$

- The 1 added is the virtual edge.

Application of the Cyclomatic Number in Software

- Furthermore, in a McCabe transposition, only individual modules are taken into account, instead of the whole software.
- So, Equation 2 becomes:

$$v(G) = e - n + 2 \quad (3)$$

- There is always one disconnected components. So, p equals 1.

Analysis Framework

- Analysis framework is made up of 5 steps:
  - Definition of measured Concept
  - Complexity Attribute
  - Units Problem
  - Definition of the measured Entity
  - Interpretation in the Industry

Analysis: Measured Concept

- Finally, based on Equation 3, McCabe suggested a measure of a program complexity, *i.e.* cyclomatic complexity, which he interpreted as *the amount of decision logic in a single software module.*
- Moreover, the Cyclomatic Number of a control flow graph is considered as a Cyclomatic *Complexity* Number.

Analysis: Measured Concept

- Now, while using the term ‘complexity’, a definition of it, of the attribute itself, or of his direct characterization is not provided.
- This approach is basically a mapping of the concepts selected from graph theory into a certain view of software as a control flow graph.

Analysis: Complexity Attribute

- By adding the label ‘complexity’ to the expression ‘Cyclomatic Number’, McCabe leads the reader to believe that the attribute he considered is the complexity of a source code program, but does not explicitly document this claim by association.
- Is this claim by association relevant, and valid?

Analysis: Complexity Attribute

- It is necessary to **explain** the assumption related to the **applicability** of graph theory concepts, such as cyclomatic complexity, in the software measurement.
- It could be of interest to **ensure** that this assumption does not **imply** some **risks** when the measurement results are used in the context of planning a testing effort or of estimating error rate.

Analysis: Units Problem

- A proper rewriting of (2) taking the units into consideration would lead to (4) instead of (3), that is:

$$v(G)^{independentcycle} = e^{edge} - n^{nodes} + p^{connectedcomponents} + 1^{virtualedge} \quad (2)$$

becomes:

$$v(G)^{independentcycle} = (e+1)^{edge+virtualedge} - n^{nodes} + p^{connectedcomponents} \quad (4)$$

Analysis: Units Problem

- Which is the common concept between the items of Equation (4) that allows to add them each others?
- Of course, adequate interpretation of units in equation (4) remains an issue.

Analysis: Definition of the measured Entity

- The entity measured by the Cyclomatic Complexity Number is a control flow graph.
- According to McCabe, the measured **entity** is the **source code** of a given module, which corresponds to a **function** or a **subroutine**.

Analysis: Definition of the measured Entity

- But, do graphs correctly represent the source code entity in order to measure its Cyclomatic Number?
- In other words, is the **assumption** concerning the **one-to-one relation** of a given module source code and its corresponding graph **verified**?

Analysis: Definition of the measured Entity

- One source code of one module is related to one and only one graph. But the contrary is not necessarily true; that is, one **graph** can be **related to one or many source codes**.
- So, it is not obvious that the **final source code** corresponds to the measured graph.
- Moreover, McCabe suggests to use this Cyclomatic Number in order to plan the testing effort.

Analysis: Definition of the measured Entity

- Another point discussed is the ‘**virtual edge**’ added to the control flow graph in order to obtain a strongly connected graph.
- But doesn’t adding a virtual edge modify the nature of the entity considered, *i.e.* the source program ?

Analysis: Definition of the measured Entity

- The justification of this virtual edge is as follows:
  - *It is not just a numerical convenience. Intuitively, it represents the control flow through the rest of the programming in which the module is used* [WAT96].

Industry Interpretation

- *Complexity can be used directly to allocate testing effort by leveraging the connection between complexity and error to concentrate testing effort on the most error-prone software* [WAT96].
- This assertion by McCabe has led to generalizations such as: ‘The higher the Cyclomatic Complexity Number, the higher the error rate’ derived from the McCabe assertion that a relation exists between the Cyclomatic Number, relabeled ‘complexity’, and the testing effort. Of course, the expressions ‘the higher the error rate’ and ‘the most error prone’ are clearly not placed on a ratio scale, but at best on an ordinal scale.

Industry Interpretation

- *A complexity measure correlates with errors in software modules* [WAT96].
- Again, this statement has led users of McCabe’s Number to associate a small number of errors with a low Cyclomatic Number.
  - However, a coefficient of correlation (r) between two given variables X & Y does not measure any causality relation between those variables. A coefficient close to 1 does not mean that one variable implies the other, it simply expresses the fact that the two variables vary in the same direction.

Industry Interpretation

- *Maintainers can keep maintenance changes from degrading the maintainability of software by limiting the Cyclomatic Complexity Number during a modification* [WAT96].
- The same comments as those above apply.

Conclusion

- Artificially labeling the Cyclomatic Number as a ‘complexity’ concept has led to considerable ambiguity on the use of this Number as a measurement number rather than as a qualitative empirical model which varies according to the empirical contexts.
- This paper highlighted a key problem of measurement units:
  - Related concepts have not been either adequately explored or adequately explained. Without such knowledge and insights, it is difficult to improve such a design.

Questions

- ?