Cyclomatic Complexity in Software Development

Surendra Kalagara
Assistant Professor
Bharat Institute of Engineering and Technology,
Hyderabad

***Abstract*: Cyclomatic complexity is software metric used in software developments as White box testing and structural testing. The purpose of the paper is to describe the Use and Analysis on Cyclomatic complexity in Software development with an example. The Cyclomatic complexity is computed using the flow graph of the program. The nodes of the graph correspond to one or more code statement and the edges connect two nodes. Based on the flow graph, finding cyclomatic complexity is discussed in this paper.**

1. INTRODUCTION

Cyclomatic complexity is software metric (measurement). It was developed by Thomas J. McCabe, in 1976 and is used to indicate the complexity of a program. It is a quantitative measure of the complexity of programming instructions. It directly measures the number of linearly independent paths through a program's source code. It is one of the metric based on not program size but more on information/control flow. The Cyclomatic Complexity is software metric that provides quantitative measures of logical complexity of a program.

1.1 What is Cyclomatic Complexity?

Cyclomatic complexity is a software metric used to measure the complexity of a program. These metric, measures independent paths through program source code. Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths. Cyclomatic complexity can be calculated with respect to functions, modules, methods or classes within a program.

This metric was developed by Thomas J. McCabe in 1976 and it is based on a control flow representation of the program. Control flow depicts a program as a graph which consists of Nodes and Edges.

1.2 How to Calculate Cyclomatic Complexity?

The Cyclomatic Complexity is computed in one of five ways:

- The number of regions of the flow graph corresponds to the Cyclomatic complexity.
- The Cyclomatic complexity, V(G), for a graph G is defined as **V(G) = E – N + 2**
- Where E is the number of flow graph edges and N is the number of flow graph nodes.
- The Cyclomatic complexity, V(G), for a graph G is defined as **V(G) = E – N + 2P**
- Where E is the number of flow graph edges, N is the number of flow graph nodes and P is connected components.
- The Cyclomatic complexity, V(G), for a graph G is also defined as **V(G) = P + 1**
- Where P is the number of predicate nodes contained in the flow graph G. The predicate node is
  a node that has of out degree two i.e. Binary node.
- The Cyclomatic complexity, V (G), for a graph G is also defined as total number of independent path of flow graph.

2. PROPERTIES OF CYCLOMATIC COMPLEXITY:

Following are the properties of Cyclomatic complexity:

1. V (G) is the maximum number of independent paths in the graph
2. V (G) >=1
3. G will have one path if V (G) = 1
4. Minimize complexity to 10

2.1 How this metric is useful for software testing?

Basis Path testing is one of White box technique and it guarantees to execute at least one statement during testing. It checks each linearly independent path through the program, which means number test cases, will be equivalent to the Cyclomatic complexity of the program.

This metric is useful because of properties of Cyclomatic complexity (M) -

1. M can be number of test cases to achieve branch coverage (Upper Bound)
2. M can be number of paths through the graphs. (Lower Bound)

Steps to be followed:

The following steps should be followed for computing Cyclomatic complexity and test cases design.

Step 1 - Construction of graph with nodes and edges from the code
Step 2 - Identification of independent paths
Step 3 - Cyclomatic Complexity

Calculation

Step 4 - Design of Test Cases

Once the basic set is formed, TEST CASES should be written to execute all the paths.

More on V (G):

Cyclomatic complexity can be calculated manually if the program is small. Automated tools need to be used if the program is very complex as this involves more flow graphs. Based on complexity number, team can conclude on the actions that need to be taken for measure.

Complexity Number	Meaning
1-10	Structured and well written code High Testability Cost and Effort is less
10-20	Complex Code Medium Testability Cost and effort is Medium
20-40	Very complex Code Low Testability Cost and Effort are high
>40	Not at all testable Very high Cost and Effort

3. TOOLS FOR CYCLOMATIC COMPLEXITY CALCULATION:

Many tools are available for determining the complexity of the application. Some complexity calculation tools are used for specific technologies. Complexity can be found by the number of decision points in a program. The decision points are if, for, for-each, while, do, catch, case statements in a source code.

Examples of tools are

- **OCLint** - Static code analyzer for C and Related Languages
- **devMetrics** - Analyzing metrics for C# projects
- **Reflector Add In** - Code metrics for .NET assemblies
- **GMetrics** - Find metrics in Java related applications
- **NDepends** - Metrics in Java applications

4. USES OF CYCLOMATIC COMPLEXITY:

Cyclomatic Complexity is a very common buzz word in the Development community. This technique is mainly used to determine the complexity of a piece of code or functionality. The technique was developed by MaCabe and helps to identify the below 3 questions for the programs / features

- Is the feature / program testable?
- Is the feature/ program understood by every one?
- Is the feature / program reliable enough?

- As a QA we can use this technique to identify the "level" of our testing. It is a practice that if the result of Cyclomatic complexity is more or a bigger number, we consider that piece of functionality to be of complex nature and hence we conclude as a tester; that the piece of code / functionality requires an in-depth testing. On the other hand if the result of the Cyclomatic Complexity is a smaller number, we conclude as QA that the functionality is of less complexity and decide the scope accordingly.

- **Cyclomatic Complexity can prove to be very helpful in**
  - Helps developers and testers to determine independent path executions
  - Developers can assure that all the paths have been tested atleast once
  - Helps us to focus more on the uncovered paths
  - Improve code coverage in Software Engineering
- Evaluate the risk associated with the application or program
- Using these metrics early in the cycle reduces more risk of the program

5. CONCLUSION.

Cyclomatic Complexity is software metric useful for white box and structural testing in Software Development. It is mainly used to evaluate complexity of a program. If the decision points are more, then the complexity of the program is more. If the decision points are more then the complexity of the program will be more.

[1] A Review on Cyclomatic Complexity by Ramesh Patelia and Shipan Vyas
[2] Roger.S.Pressman.Software Engineering-A Practitioners Approach-Tata Mcgrawhill Publications.
[3] Somerville, Software Engineering, Addison-Wesley
[4] https://www.softwaretestinghelp.com/ Cyclomatic-complexity/