Cyclomatic Complexity

UCC v.2014.08

Version History

Date	Author	Version	Changes
08/12/2014	Anandi Hira	2014.08	Explanation of the Cyclomatic Complexity output

Table of Contents

1 Introduction

Thomas McCabe developed a measure of program complexity in 1976, referred to as Cyclomatic Complexity. Cyclomatic Complexity counts the number of linearly independent paths within a program. Effort required in code construction is affected by complexity, hence, a valuable metric to incorporate in the Unified Code Count (UCC) tool. Additionally, developers are able to determine the number of independent path executions and baseline unit tests required for validation. Being aware of the Cyclomatic Complexity, developers find that they can assure that all paths have been tested at least once. The details of how the McCabe Cyclomatic Complexity metric is measured in UCC is described with examples. This standard applies across all of the languages for which Cyclomatic Complexity has been implemented.

2 McCabe Cyclomatic Complexity

As mentioned previously, Cyclomatic Complexity is a count of the number of linearly independent paths within a program. For instance a simple linear program that has no decision points has a complexity of 1, whereas if it contained an IF statement, then there are two separate paths through the code and so it would have a complexity of 2.

An easy way to calculate Cyclomatic Complexity is to take the number of decisions being made in the code, and adding 1. The following keywords usually identify decision points within code, the specific word and syntax depending on the programming language:

- IF
- ELSE IF
- REPEAT-UNTIL
- WHILE
- FOR
- CASE
- Database exception clause (except for when successful)

3 Cyclomatic Complexity Rings and Implementation

Information Engineering Technology did a study of the Cyclomatic Complexity for a specific domain of projects, and has described Cyclomatic Complexity in 4 “rings” CC1 – CC4. This categorization of Cyclomatic Complexity has allowed for the implementation to be broken down.

1. CC1: The original McCabe method treated each branch as a count.
2. CC2: A variation of the original method that counts Boolean operators within the decision point
    a. For example, the statement

IF a==1 AND b==1

Would receive a Cyclomatic Complexity value 1 for CC1, but received a value of 2 in compliance with CC2.

3. CC3: Increments Cyclomatic Complexity for each CASE OF clause, and ignores the individual CASE clauses.
4. CC4: Counts distinct IF/ELSE IF statements. Hence, if an IF statement is present multiple times within a function or file, it is only counted once..

The following table summarizes the differences between the four Cyclomatic Complexity rings:

Statement Type	**CC1**	**CC2**	**CC3**	**CC4**
IF/ELSE IF	+1	+1 for IF/ELSE IF and +1 for each AND/OR clasue	+1	+1 for each distinct IF/ELSE IF clause
REPEAT-UNTIL	+1	+1	+1	+1
WHILE	+1	+1	+1	+1
FOR	+1	+1	+1	+1
CASE OF	0	0	+1	0
CASE	+1	+1	0	+1
Database exception clause	+1	+1	+1	+1

The Cyclomatic Complexity measurement for CC1, CC2, and CC3 are reported as separate columns in the UCC Cylcomatic Complexity Output report "*outfile_cyclomatic_cplx.csv*". The results are reported by file, as well as function/module. CC4 is the only ring of complexity that has not yet been implemented within UCC.

4 Risk Evaluation

SEI has identified a risk evaluation or level with Cyclomatic Complexity ranges – this risk evaluation is provided in the UCC Cylcomatic Complexity Output report "*outfile_cyclomatic_cplx.csv*":

Cyclomatic Complexity	**Risk Evaluation**
1 - 10	A simple program, without much risk
11 – 20	More complex, moderate risk
21 – 50	Complex, high risk program
> 50	Untestable program (very high risk)

5 References

Complexity Metrics. Aivosto Oy. http://www.aivosto.com/project/help/pm-complexity.html. Accessed 4/23/2012.