Cyclomatic complexity: The nesting problem

Conference Paper · September 2013

DOI: 10.1109/ICDIM.2013.6693981

CITATIONS
12

READS
9,133

3 authors, including:

Sara Shahzad
University of Peshawar
59 PUBLICATIONS **545** CITATIONS
SEE PROFILE

Ibrar Ahmad
Beijing University of Posts and Telecommunications
16 PUBLICATIONS **197** CITATIONS
SEE PROFILE

Cyclomatic Complexity: The Nesting Problem

Mir Muhammd Suleman Sarwar, Sara Shahzad, Ibrar Ahmad
Department of Computer Science
University of Peshawar
Peshawar, Pakistan
mirsuleman1@gmail.com, sara@upesh.edu.pk, ibrar@upesh.edu.pk

***Abstract*—Cyclomatic complexity is a metric for the measurement of complexity of a software. This metric although widely cited has many limitations. Many authors criticized cyclomatic complexity in many ways but still it is the most widely accepted idea regarding software complexity. One of the problems in cyclomatic complexity is the nesting problem. A nested construct is more complex than a simple construct, but cyclomatic complexity calculates same complexity for both types of constructs. Many authors addressed this problem but their solutions also has limitations. One of the limitations is that same solutions cannot be applied over nested-loop. In this paper we propose a solution to differentiate between a nested loop and a simple loop.**

***Index Terms*—Complexity, cyclomatic complexity, nesting.**

I. Introduction

Cyclomatic complexity is a metric for the measurement of complexity for a program's source code. This metric was originally devised by Thomas J. McCabe in 1976 for testing purpose [1][2][4]. It is also used as a benchmark for comparison of different source codes. A program with high complexity is considered more error prone and hence less efficient [3][5][6][7].

Cyclomatic complexity is calculated by first of all drawing a flow chart from a source code. For example consider the following flow chart of a code.

.

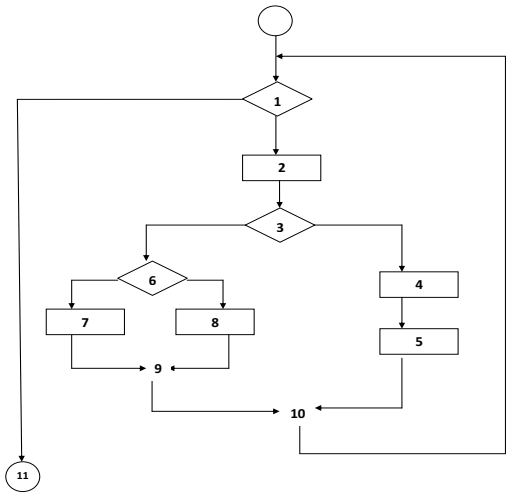


Fig. 1. Flow chart for an example source code G

Then draw a Control Flow Graph(CFG) from the flow chart. In this CFG the procedural statements in sequence are represented by a single node. CFG for above flow chart is as follows

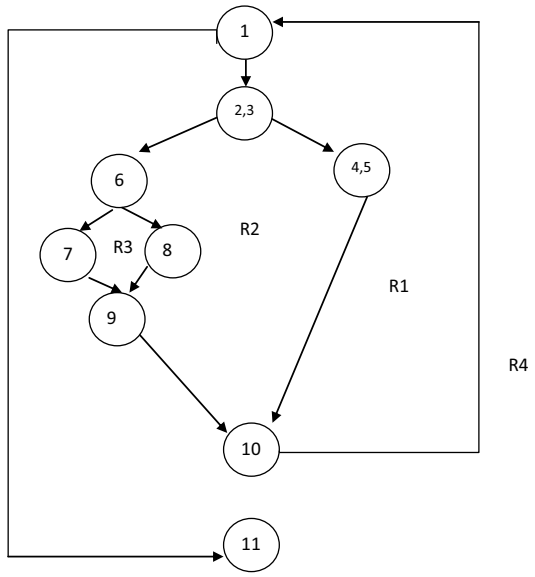


Fig. 2. Control flow graph(CFG) of the source code G

Then cyclomatic complexity "V" is calculated in one of the following three ways [1][2][4][17].

1. The number of regions(R).

$$V(G) = R \quad (1)$$

Where V(G) represents cyclomatic complexity for graph G.
Areas bounded by edges and nodes are called regions. When counting regions, we include the area outside the graph as a region.
The CFG has 4 regions.

2. The number of predicates(P) plus one.

$$V(G) = P + 1 \quad (2)$$

978-1-4799-0615-4/13/$31.00 ©2013 IEEE

Each node that contains a condition is called a predicate node and is characterized by two or more edges emerging from it.

V(G) = 3 predicate nodes + 1 = 4.

3. Number of edges(e) minus nodes(n) plus 2.

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

V(G) = 11 edges – 9 nodes + 2 = 4

II. BACKGORUND

Eli Berlinger argued in a research that if we use a particular element of programming in a complex manner than it should contribute more to complexity of program as compared to the same element used in a simple way [8]. For example a nested loop is more complex than a sequence of loops.

Myers addresses the problem of nested IF statements [9].

Consider two example codes "A" and "B". In the first example there are two IF-Else statements in sequence and in the other example we have a nested IF-Else construct. The CFGs of these two codes will be as follows:

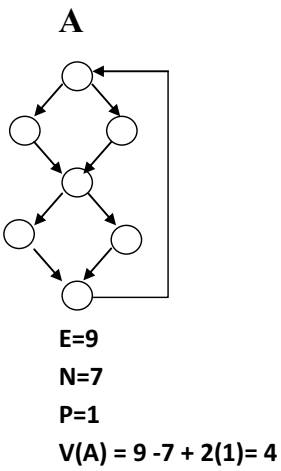


Fig.3. CFG of A and V(A)

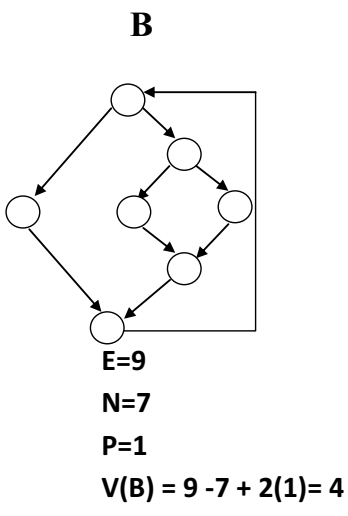


Fig. 4. CFG of B and V(B)

Both of these CFG's has a cyclomatic cxomplexity equal to 4. However Eli argues that the nested IF-Else should be more complex than a sequential IF-Else because several predicates are operative simultaneously [8].

Myers identified the nested IF-Else anomaly. In this study three types of IF-Else examples were given. A simple IF-Else with one condition, two conditions and a nested IF-Else with each IF statement containing one condition. The problem was to differentiate among the three cases. The problem was solved in a ratio manner in such a way, that the lower bound represented number of decision statements plus one and the upper bound represented individual conditions plus one [9]. This is explained in section III.

Knots [10] is a measure to calculate program complexity. Knot is identified as the case in which a line in the control flow structure crosses another line. For example in unstructured sequence loops and unstructured nested loops [10][11][12][13][14].

Other researchers have also suggested that addition of nested control structure adds to the complexity [15][16].

Another study addresses the problem by comparing two programs. One with a sequence of two loops and the other containing two loops in a nested fashion. They proposed a modified form of cyclomatic complexity by adding the number of loops and branches plus the number of nested levels of loops and branches represented by CC* [18][19].

III. THE NESTING PROBLEM

The positioning of a control structure inside another control structure is called nesting e.g. nested-IF and nested-loop.

Cyclomatic complexity fails to calculate complexity for nested structures. Many authors tried to solve this problem, and many quite successfully did to some extent. They argue that a nested-IF or nested-loop is more complex than that of a sequential structure [8 - 10][15 - 19].

A. Nested IF Problem

Consider an IF-statement with two condition:

IF(A=0) &(B=1) THEN…

There could be two ways to draw CFG for this IF statement [9].

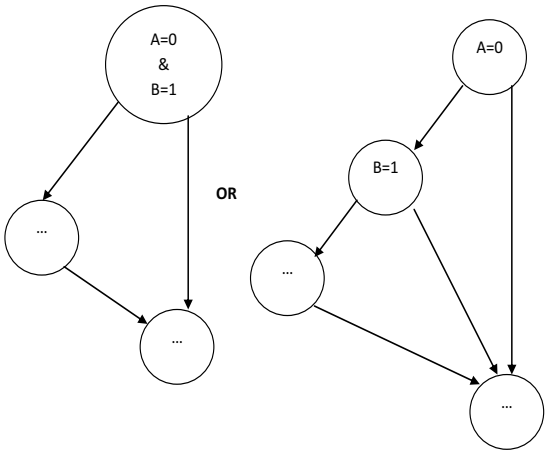


Fig. 5. CFGs for IF statement with two conditions[9]

Now IF we have these two ways to draw CFG and we have three examples of IF statements [9].

A: IF(X=0) THEN…..
ELSE…..

B: IF(X=0) & (Y>1) THEN….
ELSE…..

C: IF(X=0) THEN
IF(Y>1) THEN……
ELSE…….
ELSE..... [9].

It is clear from the above examples that the nested-IF statement C is more complex than statement B and statement B is more complex than statement A. If we use the first way to draw graph for statement B than we won't be able to differentiate between statement A and statement B since there graphs and complexity will be same. And if we use the second way to draw graph for statement B than we won't be able to differentiate between statement B and statement C. The problem is to find out a solution in which statement B has more complexity than statement A and the nested-IF statement C has more complexity than statement B[8][9].

$$A<B<C \quad (4)$$

B. Nested Loop Problem

Similarly a nested loop is considered to be more complex than a sequence of loops. Consider two example codes D and E. Code D has three loops connected in sequence and Code E has three nested loops. All of the loops in both examples executes three times.

CODE D

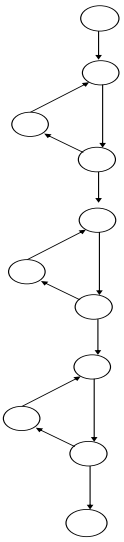
```
For(int i=1;i<=3;;i++)
{
}
For(int k=1;k<=3;k++)
{
}
For(int m=1;m<=3;m++)
{
}
```

CODE E

```
For(int i=1;i<=3;;i++)
{
For(int k=1;k<=3;k++)
  {
    For(int m=1;m<=3;m++)
     {
     }

  }
}
```

Figure 6 and Fig. 7 represents CFGs of code D and E



E – N + 2 = 13-11 + 2= 4

Fig. 6 CFG of Code D and V(D)

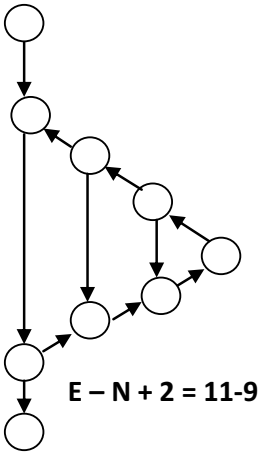


Fig. 7. CFG of Code E and V(E)

Cyclomatic complexity for both of these graphs equals to 4, which is highly inappropriate. Firstly because, we can see that the nested-loop graph is more complex than the sequential graph of loops. Secondly and most importantly if we count the total number of iterations, code D has total 9 iterations and code E has 27 iterations. So, the nested loop structure will always have more number of iterations as compared to their sequential structure, given that the lower and upper bounds will be the same.

IV. EXISTING SOLUTIONS FOR COMPLEXITY OF NESTED CONSTRUCTS

The objective of this paper is to find out a solution in which we can differentiate between the complexity of a nested construct and a simple construct.

A. Solution for the Nested IF Problem

Myers [9] proposes solution to the nested-IF problem by a ratio manner technique (Section III-A). Two values are assigned to each type of statement for example X : Y. The lower bound(X) represented number of decision statements plus one and upper bound(Y) represented individual conditions plus one. So statement A is assigned a value 2 : 2. Statement B is assigned a value 2:3 and statement C is assigned the value 3:3. This may be written as:

$$2:2 < 2:3 < 3:3 \quad (5)$$

The above equation satisfies the condition in Eq. 4.
Myers proposed a solution to the nested-IF problem. However, researchers consider this solution to be a solution for all types of nesting problems including the nested-loop problem.

B. Solution for the Nested Loop Problem

Piwowarski [16] proposed his solution to calculate complexity of a nested-loop, represented by N. The value of N is given by:

$$N = V^*(G) + N^* \quad (6)$$

Where V*(G) represents adjusted cyclomatic complexity, in which case/switch statements are considered a single predicate. N* represents the effect of nested control structures and is given by

$$N^* = i\ P^*(i) \quad (7)$$

Where P*(i) is the nesting depth of the ith predicate [16].

V. PROPOSED SOLUTION FOR THE NESTED LOOP PROBLEM

Both of these solutions presented by Myers [9] and Piwowarski [16] appropriately differentiate between the nested-IF statements and simple IF-statements. However we argue that these proposed solutions also have limitations. No matter how much complex a nested-IF is, it either executes once only or does not execute at all.
The same solutions cannot be applied to nested-loops. The problem with loops is that the number of iterations may vary a lot and it might execute more than once, which makes loops more complex than IF statements. For example the total number of iterations in code D is 9 and that of code E is 27 (section III-B) although every loop has a count equal to three. Consider another example with two codes J and F. These codes also has 3 loops in both codes. In code J we have three loops connected in sequence and in code F we have a nest of three loops. Same example as in section III-B but this time with a difference that every loop has a single iteration. Every loop executes once only. The graph for both of these codes will be same as that of fig.6 and fig.7. However the total number of iterations in code J will be equal to 3 and that of code F will be 1 only. In this case the sequence structure is more complex than the nested-loop. But interestingly cyclomatic complexity for both codes will be equal to 4.

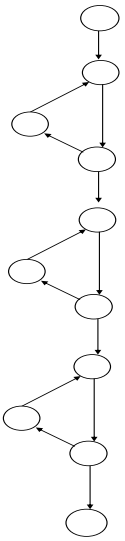
CODE J

```
For(int i=1;i<=1;;i++)
{
}
For(int k=1;k<=1;k++)
{
}
For(int m=1;m<=1;m++)
{
}
```

CODE F

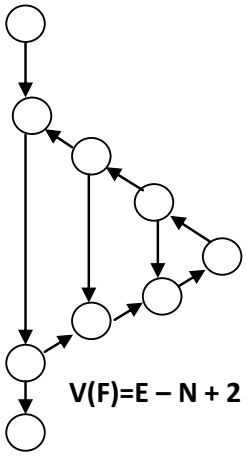
```
For(int i=1;i<=1;;i++)
{
For(int k=1;k<=1;k++)
  {
    For(int m=1;m<=1;m++)
     {
      }

  }
```



V(J)=E – N + 2 = 13-11 + 2= 4

Fig. 8. CFG of Code J and V(J)



V(F)=E – N + 2 = 11-9 + 2 = 4

Fig. 9. CFG of Code F and V(F)

And the level of depth and ratio technique will also calculate same complexity for code D, E, J and F. But the fact is that all of these codes differ greatly.

It is clear from the above four example codes that the above mentioned solutions have limitations in addressing the problem of complexity of nested-loop. Also the significance of number of iterations of loops cannot be ignored including the level of depth of nested-loop.

Consider the first example code E which consists 3 nested loops and every loop executing 3 times. So, the total number of iterations will be as:

$$3 * 3 * 3 = 27 \tag{8}$$

We can write the above equation as:

$$3^3 \tag{9}$$

If we fix the number of iterations per loop equal to 3 then the total number of iterations will be equal to 3 power level of depth of nest.

But if we don't know the number of iterations per loop than the total number of executions is calculated by multiplying number of iterations of every loop at every level of depth in the nest. Suppose "n" are the total number of level of depth and "P" represents number of iterations at every level.

P1= total iterations at level 1

P2= total iterations at level 2

.

.

.

.

Pn=total iterations at level n

Then total iterations will equal to:

$$\text{Total iterations} = P1*P2*P3\ldots\ldots Pn \tag{10}$$

$$\text{Total iterations} = \prod_{i=1}^{n} \text{Pi} \tag{11}$$

Pi can be calculated as:

$$Pi = Ui - Li + 1 \tag{12}$$

Where,

Pi = No. of iterations of ith loop.

Ui = upper bound of ith loop.

Li = lower bound of ith loop.

Adding Eq.11 with Eq.3 we get:

$$V(G) *= V(G) + \prod_{i=1}^{n} Pi \tag{13}$$

Where,

V(G)* = adjusted cyclomatic complexity for any control flow graph "G".

VI. Conclusion

Complexity is a term with a lot of meanings. Many authors defined complexity in many ways. Some authors even defined types of complexity statement [15]. At the time McCabe [1] devised cyclomatic complexity, he referred to the general term of complexity. McCabes's cyclomatic complexity besides testing was also used as a benchmark for predicting cost, judging efficiency of code and many others. As far as testing is concerned cyclomatic complexity works fine but when it comes to using it as a benchmark for code comparison to judge code efficiency, cyclomatic complexity is just not enough because efficiency of code is related to computational complexity. And two codes with different number of iterations of loops may not have the same computational complexity, even if there graphs are the same. For example, a nested-loop with a billion iterations has more/high computational complexity than a loop with hundred iterations. So, the number of iterations are important while judging computational complexity. As a loop may have more than one number of iterations. Hence a nested-loop's complexity cannot be calculated as that of a nested-IF's complexity. When we add the total number of iterations of nested loop to cyclomatic complexity, this way we can differentiate not only between a nested-loop and a simple loop but we can also differentiate between more complex nested loops and less complex nested loops.

References

[1] T. J. McCabe, "A Complexity Measure," Software Engineering, IEEE Transactions on, vol. SE-2, pp. 308-320, 1976.

[2] Sarwar, Mir Muhammad Suleman, Ibrar Ahmad, and Sara Shahzad. "Cyclomatic Complexity for WCF: A Service Oriented Architecture." Frontiers of Information Technology (FIT), 2012 10th International Conference on. IEEE, 2012.

[3] Gaur, M. (2013). "Software Security-Static Buffer Overflow Analysis in Object Oriented Programming Environment-A Comparative Study." IJCAIT 2(1): 1-8.

[4] R. S. Pressman, Software engineering: a practitioner's approach: McGraw-Hill Higher Education, 2010

[5] Khalid, M., S. ul Haq, et al. (2013). "An Assessment of Extreme Programming Based Requirement Engineering Process." International Journal of Modern Education and Computer Science (IJMECS) 5(2): 41.

[6] Yadav, A. and R. Khan (2012). "Development of Encapsulated Class Complexity Metric." Procedia Technology 4: 754-760.

[7] P.L.M. Kelani Bandara, G.N. Wikramanayake and J.S.Goonethillake, "Software Reliability Estimation Based on Cubic Splines", Proceedings of the World Congress on Engineering

[8] Berlinger, E. (1980). An information theory based complexity measure. Proceedings of the May 19-22, 1980, national computer conference, ACM.

[9] Myers, Glenford J. "An extension to the cyclomatic measure of program complexity." ACM Sigplan Notices 12.10 (1977): 61-64.

[10] Mengel, S. A. and J. V. Ulans (1999). A case study of the analysis of novice student programs. Software Engineering

Education and Training, 1999. Proceedings. 12th Conference on, IEEE.

[11] S.Hung, L.Kwok, and R.Chan."Automatic Program Assessment." Computer and Education, Volume 20, Issue 2, 1993, pp.183-190.

[12] Mengel, S. A. and J. Ulans (1998). Using Verilog LOGISCOPE to analyze student programs. Frontiers in Education Conference, 1998. FIE'98. 28th Annual, IEEE.

[13] R.J. Leach. "Using Metrics to Evaluate Student Programs." SIGCSE Bulletin, Volume 27, Issue 2, June 1995, pp. 41-43.

[14] M.R. Woodward, M.A. Hennell, and D. Hedley. "A Measure Of Control Flow Complexity in Program Text." IEEE Transactions on Software Engineering, Volume SE-5, Issue 1, November 1979, pp. 45 - 50.

[15] Henderson-Sellers, B., Y. R. Pant, et al. (2007). "Cyclomatic Complexity: theme and variations." Australasian Journal of Information Systems 1(1).

[16] Piwowarski, P., (1982), A nesting level complexity measure, ACM SIGPLAN Not., vol. 17, no. 9, pp. 44-50

[17] Jovanović, I. "Software Testing Methods and Techniques." The IPSI BgD Transactions on Internet Research: 30

[18] Hu, E.-W., C. S. Ku, et al. (2006). New DSP Benchmark based on Selectable Mode Vocoder (SMV). Proc. of the 2006 International Conference on Computer Design.DSP

[19] Hu, E.-W., C. S. Ku, et al. (2009). "Performance analysis of digital signal processors using SMV benchmark." International Journal of Signal Processing 5: 3.