A COGNITIVE APPROACH TO MEASURE THE COMPLEXITY OF BREADTH FIRST SEARCH ALGORITHM

Esther Isola [1] ; Stephen Olabiyisi [2] ; Elijah Omidiora [2] ; Rafiu Ganiyu [2] ; Olajide Adebayo [1]

[1] Osun State University, Osun State Nigeria, Department of Information & Communication Technology
[2] Ladoke Akintola University of Technology, Oyo State Nigeria, Department of Computer Science & Engineering

Corresponding author: Rafiu Ganiyu, raganiyu@lautech.edu.ng

ABSTRACT: There are different facets of software complexity some of which have been computed using widely accepted metrics like cognitive complexity metric such as Improved cognitive complexity measure (ICCM), Cognitive functional size (CFS), and Cognitive information complexity measure (CICM), Cognitive complexity metric reflects difficulty for programmers to understand the code and the information packed in it. In this research work, the strength and weakness of existing cognitive complexity metric such as Improved Cognitive Complexity Measure (ICCM), Cognitive Functional Size (CFS), and Cognitive Information Complexity Measure (CICM) on breadth first search code implemented in C++, C# and java language were examined.

KEYWORDS: Breadth first search algorithm, Improved Cognitive Complexity Metric, Cognitive Functional Size, Cognitive Information Complexity Metric.

1. INTRODUCTION

Cognitive informatics is an area of studying the internal information processing mechanisms of the brain and the processes involved in perception and cognition. It is an interdisciplinary approach because it is applied in various research fields to search the solution of a given problem such as software engineering, artificial intelligence and cognitive sciences [AK14]. Cognitive complexity measures are derived from cognitive informatics. The key of determining the cognitive complexity relies on the total cognitive of basic control structures (BCS‟s) e.g. sequence, if-else, switch-case, while-loop and for-loop etc. The calculation of the total cognitive weights of the basic control structures of software seems to resemble the counting rules in combinatory i.e. the rule of sum and the rule of products. Determining the complexity of a software using cognitive measure is an approach used by system developers to fully understand the software in all aspects such as input, output, constant and variable, loops and branches.

Proper evaluation of effect of implementation languages on software complexity is of great importance to both the application developers and also the end users. It guides the application developing software of high quality. It also enables the end users have an upper hand in choosing software from a variety on its own needs. Hence, this paper attempts to compare the performance of the cognitive metric using breadth first search algorithm implemented in C++, Java and C# programming language.

2. LITERATURE REVIEW

2.1 Search Algorithm

A search algorithm is the step-by-step procedure used to locate specific data among a collection of data. It is considered a fundamental procedure in computing. In computer science, when searching for data, the difference between a fast application and a slower one often lies in the use of the proper search algorithm. [KSV14]. All search algorithms make use of a search key in order to proceed with the procedure. Search algorithms are expected to return a success or a failure status, usually denoted by Boolean true/false. Different search algorithms are available, and the performance and efficiency of the same depend on the data and on the manner in which they are used. [KSV14]. There are different types of search algorithms such as the Breadth first search algorithm and Depth first search algorithm but in this research Breadth-first search algorithm is the major focus. Both Breadth first search (BFS) and Depth first search algorithms (DFS) are used to solve the same problem of graph traversal but the major difference is that Breadth first search use stack approach while the depth first search use queue approach [Cab89].

2.2 Software Complexity Measure

Complexity measure is divided into code based, Requirement based, and cognitive complexity measure.

2.2.1 Code Based Complexity Measure

Code based complexity metrics are software measures that have been developed to help developers understand where their code needs rework or increased testing. It helps in analyzing the code before its execution [TK14]. Code metrics focus on the individual system components (procedures and modules) and require a detailed knowledge of their internal mechanisms. It evaluates how It evaluates how complex a software is by measuring its source code.

2.2.2 Required Based Complexity

Requirements engineering or software specification is the process of understanding and defining what services are required from the system and identifying the constraints on the system's operation and maintenance [Som11]. Requirements specification is the process of writing down the user and system requirements in a requirements document. The result of requirements engineering process is the software requirements specification. The software requirements specification (SRS) or the software requirements document is an official statement of what the system developers should implement. The IEEE software requirements specification is taken as the foundation for this complexity measure [AK10].

2.2.3 Cognitive Complexity Metric

Cognitive Complexity refers to the human effort needed to perform a task or the difficulty experienced in understanding the code or the information packed in it [Iso17]. Understandability of the code is known as program comprehension and is a cognitive process and related to cognitive complexity. In other words, the cognitive complexity is the mental burden on the user who deals with the code, for example the developer, tester and maintenance staff. Some of cognitive complexity measures includes: CFS (Cognitive Functional Size) of program, CICM (Cognitive Information Complexity Measure), MCCM (Modified Cognitive Complexity Measure), CPCM (Cognitive Program Complexity Measure) and ICCM (Improved Cognitive Complexity Measure).

Improved Cognitive Complexity Measure (ICCM)

The names of variables used in the code plays a very important role in increasing or decreasing the understandability of the code. For calculating the understandability and information contained in the software in this research, Arbitrarily Named Variables (ANV), Meaningfully Named Variables (MNV) and Cognitive weight of Basic Control Structure as shown in Table 1 are considered. [I+16]

$$\boldsymbol{ICCM} = \sum_{\boldsymbol{K=1}}^{\boldsymbol{LOCs}} \sum_{\boldsymbol{V=1}}^{\boldsymbol{LOCs}} (\boldsymbol{3ANV + MNV}) * \boldsymbol{W_c(K)}$$

Table 1: Basic Control Structure and Cognitive Weight Unit

Category	BCS	CWU
Sequence	Sequence	1
Condition	If-else / Switch	2
Loop	For / For-in While/do…While	3
Functional activity	Functional-call Alert/ prompt throw	2
Exception	try-catch	1

Cognitive Information Complexity Measure (CICM)

Cognitive Information Complexity Measure (CICM) is defined as product of weighted information count of the software (WICS) and sum of the cognitive weights of Basic Control Structure (SBCS) of the software [Kus06]. The CICM can be expressed as:

$$CICM = WICS * SBCS$$

Cognitive Functional Size (CFS)

[SW03] proposed Cognitive Functional Size (CFS) which depends on three fundamental factors: inputs, outputs, and internal processing (Basic Control Structure). It satisfied 8 properties of Weyuker's,

$$CFS = (N_i + N_o) + BCS$$

3. METHODOLOGY

The study investigates the effect of implementation languages on software complexity measure of breadth first search algorithm implemented in Java, C++ and C# program. The metrics used are Improved Cognitive Complexity Measure (ICCM), Cognitive Functional Size (CFS) and Cognitive Information Complexity Measure (CICM). Table 2 shows the sample of how ICCM was computed for breadth first search algorithm implemented in C#.

4. RESULT AND DISCUSSIONS

The cognitive complexity values for different cognitive measures for Breadth First Search (BFS) algorithms written in three different object oriented languages (C++, C#, JAVA) are shown in Table 3. The graph for relative comparison between the cognitive complexity measures and object oriented

languages are shown in Figs. 1 to 4. The statistics collected from the metrics used are compared with the values obtained from BFS implemented in C++, C# and JAVA to investigate the comprehensibility, usefulness and robustness of the evaluated measures. A relative graph which shows the comparison between CFS, CICM, and ICCM in C++, C# and JAVA program is plotted in Fig 1. A close inspection of this graph shows that ICCM is closely related to CICM but not CFS in which ICCM reflect similar trends with CICM. Also, both ICCM and CICM considers variables in there formulation while CFS did not. In other words, high ICCM values are due to the fact that ICCM includes most of the parameters of different measures and measures the effort required in comprehending the code.

Fig 2 shows the comparison graph of ICCM, CFS and CICM for C++ program. BFS algorithm written in C++ has the complexity value for CFS has 143, CICM is 125 and ICCM complexity value is 160, the three metrics have different complexity values because what they considered were different. CFS measures the cognitive size of the program while CICM measures the information contained in a program and ICCM measures the effort required in understanding the program.

Figure 3 shows that for BFS algorithm implemented in C#, CFS has the least complexity value of 40 and ICCM has the highest complexity value of 129. This is due to the fact that CFS only considers the size of the program and ICCM considers the effort required in understanding the program.

Table 2: Evaluation of the Implementation Language using ICCM

S/N	CODE	ANV+MNV	CWU	ICCM
1	Public class tree<k,v> class,	8	1	8
2	Where k: IComparable<K>	7	1	7
3	Where V:class	4	1	4
4	{	0	1	0
5	Private Node <K,V> root;	8	1	8
6	Public V BFS(K key)	8	1	8
7	{	0	1	0
8	Queue <Node<K, V>>queue = new Queue<Node<K,V>>0;	16	1	16
9	While (queue.Any)	2	3	6
10	{	0	1	0
11	Var Node = queue.Dequeue();	4	1	4
12	If(node.key == key)	3	2	6
13	{	0	1	0
14	Return node. value;	3	3	3
15	}	0	1	0
16	For each (var child node.children)	4	3	12
17	{	0	1	0
18	Queue.Enqueue(child);	3	1	3
19	}	0	1	0
20	}	0	1	0
21	Return default(V);	4	1	4
22	}	0	1	0
23	Private class Node<K,V>	8	1	8
24	Where K : Class, Icomparable<K>	8	1	8
25	Where V: class	4	1	4
26	{	0	1	0
27	Public K key;	4	1	4
28	Public V value;	4	1	4
29	Public Node <K,V> [] children;	8	1	8
30	}	0	1	0
31	}	0	1	0
	TOTAL			129

Table 3: The complexities of Object Oriented Programming languages for Breadth First Search Algorithm

ALGORITHM	PROGRAMS	CFS	CICM	ICCM
BFS	C++	143	125	160
BFS	C#	40	91.96	129
BFS	JAVA	160	188.37	223

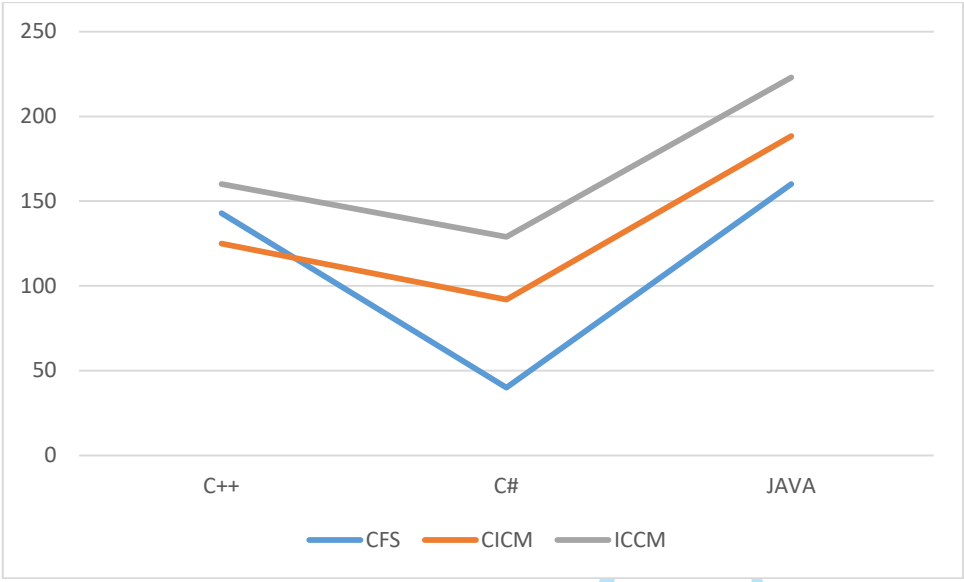

Figure 1: Relative graph between ICCM, CFS, and CICM for C++, C#, and JAVA Programs

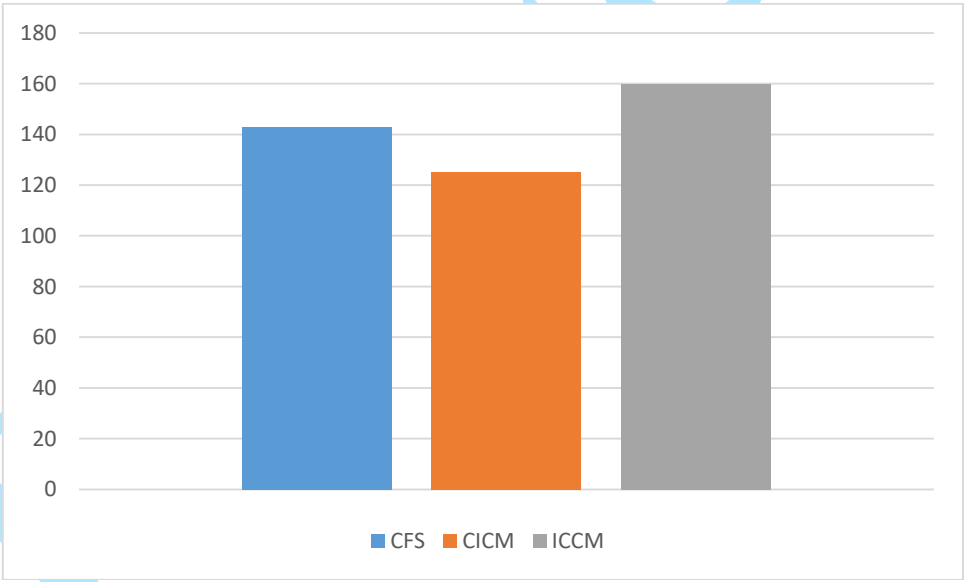

Figure 2: Comparison Graph of ICCM, CFS and CICM against C++ programs

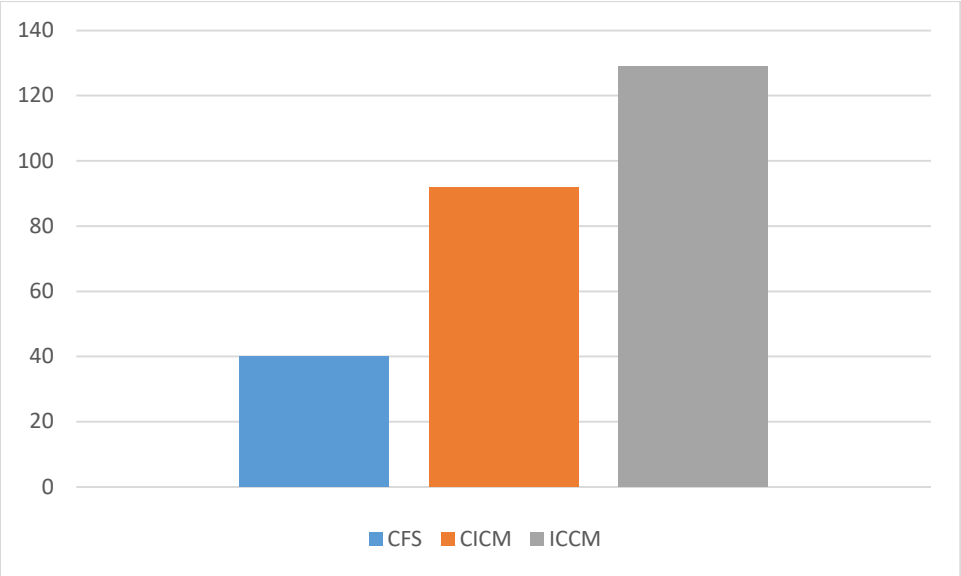

Figure 3: Comparison Graph of ICCM, CFS and CICM against C# programs

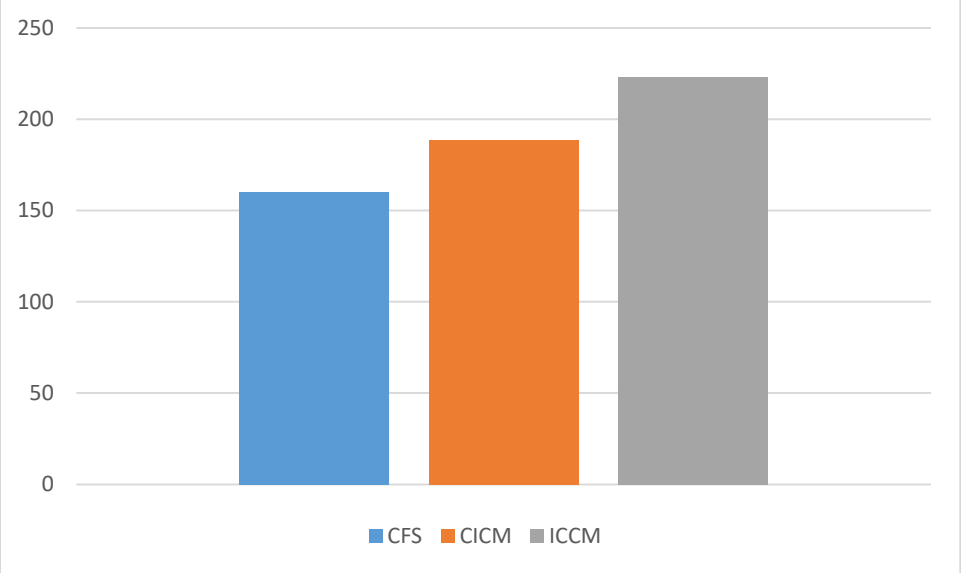

Figure 4: Comparison Graph of ICCM, CFS and CICM against Java programs

Comparative graph of ICCM, CFS and CICM for Java program is shown in Fig 4. CFS has the least complexity value of 160 followed by CICM with the value of 188.37 and ICCM has the highest complexity value of 223. This is because ICCM considers most of the parameters considered by CFS and CICM.

5. CONCLUSION

Software complexity is one of most important factors in software engineering. Programs having high complexity in their code will not be easy to comprehend. To reduce risks related to software complexity, most software engineers measure the complexity of program as a feasibility study before start any project and also find the suitable metrics to be used. In this paper, Breadth first search(BFS) algorithm was considered by computing ICCM, CFS, and CICM metrics using some object oriented languages (C++, C# and JAVA). The results showed that BFS implemented in Java programming language has the highest cognitive complexity value and BFS implemented in C# has the lowest value for all the metrics used (ICCM, CFS and CICM). Which implies that for any of the considered metrics, BFS algorithm implemented in C# is the best because it has the lowest complexity value and therefore it will require less effort to understand.

REFERENCES

[AK10] **Ashish S., Kushwaha D. S.** – *A complexity measure based on Requirement Engineering Document*. Journal of Computer Science and Engineering, Vol.1 Issue1 pp.112-119, 2010.

[AK14] **Amit K. J., Kumar R.** - *A New Cognitive Approach to Measure the Complexity of Software*. International Journal of Software Engineering and its Applications. Vol. 8 No.7, pp. 185-198, 2014.

[Cab89] **McCabe T. A.** - *Complexity measure*. IEEE Trans. Software Engineering (se-2, 6) pp. 308-320, 1989.

[Iso17] **Isola E. O.** - *Development of an improved cognitive based complexity metric*. Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria; 2017.

[I+16] **Isola E., Olabiyisi S. O., Omidiora E. O., Ganiyu R. A., Ogunbiyi D. T., Adebayo O.** - *Development of an Improved Cognitive Complexity Metrics for Object Oriented Codes*. British Journal of Mathematics & Computer Science Article no. BJMCS.28515, 3-6, 2016.

[Kus06] **Kushwaha D. S.** - *Robustness analysis of cognitive information complexity measure using Weyuker properties*. India: ACM SIGSOFT, 2006.

[KSV14] **Kaur A., Sharma P., Verma A.** - *A appraisal paper on Breadth first search, Depth first search and Red black tree*. International Journal of Scientific and Research Publications., Volume 4, Issue 3, page 2-4, 2014.

[Som11] **Sommerville I.** - *Software Engineering*, 7th Edition, Addison Wesley, 2011.

[TK14] **Tiwar U., Kumar S.** - *Cyclomatic complexity metric for component based software*, ACM, vol. 39, no. 0004-5411, p. 6, 2014.

[WS03] **Wang Y., Shao J.** - *Measurement of the Cognitive Functional Complexity of Software*. Proc. of 2nd IEEE Int'l Conference on Cognitive Informatics, 67, 2003.