British Journal of Mathematics & Computer Science
18(2): 1-11, 2016, Article no.BJMCS.28515
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Development of an Improved Cognitive Complexity Metrics for Object- Oriented Codes

O. Isola Esther[1,2*], O. Olabiyisi Stephen[1,2], O. Omidiora Elijah[1,2], A. Ganiyu Rafiu[1,2], T. Ogunbiyi Dimple[1,2] and Y. Adebayo Olajide[1,2]

[1]Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria.
[2]Osun State University, Osogbo, Osun State, Nigeria.

Authors' contributions

This work was carried out in collaboration between all authors. Authors OIE and OOS designed the study, performed the statistical analysis, wrote the protocol and wrote the first draft of the manuscript. Authors OOE and AGR managed the analyses of the study. Authors YAO and TOD managed the literature searches. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/28515
Editor(s):
(1) Jacek Dziok, Institute of Mathematics, University of Rzeszow, Poland.
Reviewers:
(1) Radosław Jedynak, Kazimierz Pulaski University of Technology and Humanities, Radom, Poland.
(2) Hasni Neji, University of Carthage, Tunisia.
Complete Peer review History: http://www.sciencedomain.org/review-history/15845

Original Research Article

Received: 22nd July 2016
Accepted: 8th August 2016
Published: 20th August 2016

Abstract

Cognitive informatics helps in comprehending the software characteristics and its complexity measures can be used to predict critical information about testability of software system. In this paper, a cognitive complexity metric for C++ programming language is formulated. Since C++ is an object – oriented language, the cognitive complexity metric is capable to evaluate any object- oriented language. This paper presents a new cognitive complexity metric named Improved Cognitive Complexity Metric (ICCM) and perform a comparative study of the proposed metric with the existing metric such as NCCOP, CFS, CICM and CPCM. The result shows that the proposed metric performs better than other metrics by giving more information contained in the software and reflecting the understandability of a source code. Also, an attempt has also been made to present the relationship among ICCM, NCCOP, CICM, CFS and CPCM using Pearson correlation coefficient method.

Keywords: Software complexity; cognitive informatics; basic control structure; object-oriented language.

**Corresponding author: E-mail: esther.isola@uniosun.edu.ng;*

1 Introduction

Software complexity is a major feature of computer software and is difficult to be measured accurately. It is defined as "the degree to which a system or component has a design or implementation that is difficult to understand and verify" [1]. High complexity may result in more errors and difficulties in maintenance, understandability, modification and testing effort [2,3]. Therefore, there has been a great deal of interest in defining appropriate metrics to measure the complexity of the software. Although some useful metrics have been proposed to measure the software complexity [4,5], the current solutions are not enough to settle down this rigorous problem. Both computer researchers and the software engineers are looking for more powerful and effective metric of software complexity. There are some complexity measures based on cognitive aspects such as Cognitive Functional Size (CFS) proposed by Wang and Shao [6] to measure the complexity of a software, it depends on input, output parameters and internal control flow. It excludes some important details of cognitive complexity such as information contained in variables and operators.

New Cognitive Complexity of Program (NCCoP) was proposed by Amit and Kumar [7] to measure the cognitive complexity of a program; the metric considered the number of variables in a particular line of code and the weight of Basic Control Structure. NCCoP fails by not distinguishing between the variables in a program, which only provide the information contained in a software. Nevertheless, to solve such a problem, Arbitrarily Named Variables (ANV), Meaning Named Variables (MNV) and the weight of Basic Control Structures (BCS) is employed to improve the performance of NCCoP. Hence, this research formulated a cognitive complexity metric using ANV, MNV and BCS to provide more information contained in a software and measure the difficulty of code comprehension.

2 Review of Some Cognitive Complexity Measures

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

2.1 Code based complexity measures

Code complexity metrics are used to locate complex code. To obtain a high quality software with low cost of testing and maintenance, the code complexity should be measured as early as possible in coding. Developer can adapt his code when recommended values are exceeded [2]. Code based complexity measure comprises Halstead Complexity Measure and Mac Cabe's Cyclomatic Complexity and Lines of Code Metrics.

2.2 Cognitive complexity measures

Cognitive complexity measures quantify human difficulty in understanding the source code [8]. Some of the existing cognitive complexity measures are KLCID Complexity Metrics, Cognitive Functional Size (CFS), Cognitive Information Complexity Measure (CICM), Modified Cognitive Complexity Measure (MCCM), Scope Information Complexity Number of Variables (SICN), Extended Structure Cognitive Information Measure (ESCIM) and Unified Complexity Measure (UCM).

2.3 Klcid complexity metrics

Klemola and Rilling proposed KLCID based complexity measure. Defined identifiers as programmer defined variables and based on identifier density (ID).

$$ID = \frac{Total\ number\ of\ identifiers}{Line\ of\ Code} \qquad (2.1)$$

For calculating KLCID, number of unique lines of code was found, lines that have thesame type and kind of operands with same arrangements of operators considered equal. KLCID is defined as:

$$\text{KLCID} = \frac{Number\ of\ Identifier\ in\ the\ set\ of\ unique\ lines}{Number\ of\ unique\ lines\ containing\ identifier} \tag{2.2}$$

This method can become very time consuming when comparing a line of code with each line of the program. It also assumes that internal control structures for the different software’s are thesame.

2.4 Cognitive functional size

Wang and Shao [9] proposed functional size to measure the cognitive complexity. The measure defines the cognitive weights for the Basic Control Structures (BCS). Cognitive functional size of software is defined as:

$$\text{CFS} = (N_i + N_O) * W_c \tag{2.3}$$

Where Ni= Number of Inputs, No= Number of Outputs and Wc= Total Cognitive weight of software.

Wc is defined as the sum of cognitive weights of its q linear block composed in individual BCS’s. Since each block may consist of m layers of nesting and each layer with n linear BCS [10], total cognitive weight is defined as:

$$\text{Wc} = \sum_{j=1}^{q}\left[\prod_{k=1}^{m}\sum_{i=1}^{n} W_c\ (j,k,l)\right] \tag{2.4}$$

Only one sequential structure is considered for a given component.

Now difficulty with this measure is the inability to provide an insight into the amount of information contained in software.

2.5 Cognitive information complexity measure

Cognitive Information Complexity Measure (CICM) is defined as product of weighted information count of the software and sum of the cognitive weights of Basic Control Structure (SBCS) of the software [11]. The CICM can be expressed as:

$$\text{CICM} = \text{WICS} * \text{SBCS} \tag{2.5}$$

This establishes a clear relationship between difficulty in understanding and its cognitive complexity. It also gives the measure of information contained in the software as:

$$\text{Ei} = \frac{ICS}{LOCS}$$

where Ei represents Information Coding Efficiency.

The cognitive information complexity is higher for the programs, which have higher information coding efficiency. Now the problem with these measures is that, they are code dependent measures, which itself is a problem as stated earlier. Various theories have been put forward in establishing code complexity in different dimensions and parameters.

2.6 Modified cognitive complexity measure

CFS was modified by [11] into Modified Cognitive Complexity Measure (MCCM) by simplifying the complicated weighted information count in CICM as:

$$\text{MCCM} = (Ni1 + Ni2) * Wc$$

where N_{i1} is the total number of occurrences of operators, Ni_2 is the total number of occurrences of operands, and Wc is the same as in CFS.

However, the multiplication of information content with the weight Wc derived from the whole BCS's structure remains the approach's drawback. Also, [12] proposed Cognitive Program Complexity Measure (CPCM) based on the arguments that the occurrences of inputs/output in the program affect the internal architecture and are the forms of information contents. The computation of CFS was also critized such that the multiplication of distinct number of inputs and outputs with the total cognitive weights was not justified as there was no reason why using multiplication.

Besides, it was established that operators are run time attributes and cannot be regarded as information contained in the software as proposed by [11]. Based on these arguments, CPCM was thus defined as:

$$\text{CPCM} = S_{io} + W_c \tag{2.6}$$

where S_{io} is the total occurrences of input and output variables and Wc is as in CFS.

3 Proposed ICCM (Improved Cognitive Complexity Measure)

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 and Meaningfully Named Variables are considered.

3.1 Number of arbitrarily named variables

Following [13], it was suggested that the name of the variables should be chosen in a way which is meaningful in programming. If the variable names are taken arbitrarily, there is no problem if the developer himself is evaluating the code. Fig. 1 shows the code written in ANV vs MNV, it was observed that effort required for comprehending the syntax written with Arbitrarily Named Variable (ANV) increases the difficulty of understanding three times more than the one written with Meaningfully Named Variable (MNV) [14] as shown in Fig. 1. In the formulation of the proposed metric, the weights of the ANV are considered to be three times greater than MNV. Therefore, the weight of ANV is assigned as three units.

3.2 Number of meaningfully named variables

It is clear that Meaningfully Named Variables are more understandable than Arbitrarily Named Variables, as shown in Fig. 1. The weight of MNV is assigned one unit. It should be noted that discriminating the ANV and MNV is subject to developer’s choice.

3.3 Cognitive weights of basic control structures (BCS)

The complexity of a program is directly proportional to the cognitive weights of Basic Control Structures (BSC). The cognitive weight of software is the extent of difficulty or the relative time and effort for comprehending the given software modelled by a number of BCS. BCS are basic building blocks of any software and their weights are one, two and three as given in Table 1. These weights are assigned on the classification of cognitive phenomenon as discussed by [4].

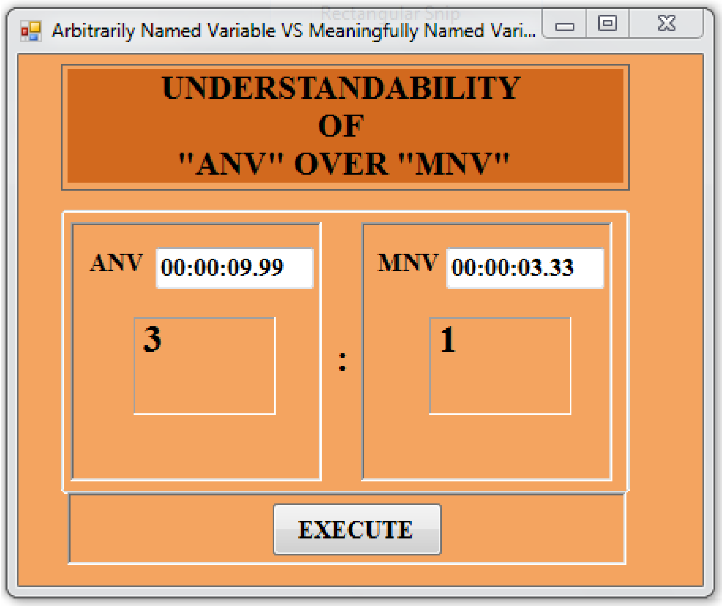
Arbitrarily Named Variable VS Meaningfully Named Vari...
UNDERSTANDABILITY OF "ANV" OVER "MNV"
ANV
00:00:09.99
3
:
MNV
00:00:03.33
1
EXECUTE

Fig. 1. Understandability of ANV over MNV

Table 1. Basic control structure [4]

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

Using the above considerations, we propose the following metric to reflect the information contained in the software and code comprehensibility.

$$ICCM = \sum_{K=1}^{LOCs} \sum_{V=1}^{LOCs} (3ANV + MNV) * W_C(K) \qquad (2.7)$$

Where, the first summation is the line of code from 1 to the last Line Of Code (LOC), Arbitrarily Named Variables (ANV) and Meaningfully Named Variable (MNV), are the number of variables in a particular line of code and W_C is the weight of BCS as shown in Table 1 corresponding to the particular structure of line.

3.4 Demonstration of ICCM

The proposed cognitive complexity metric given in equation (2.7) is demonstrated with First in First Out Algorithm, given by the following Table 2.

Table 2. First - In - First – Out (FIFO) algorithm for C++ programming language

S/N	Code	ANV + MNV	CWU	ICCM
1.	# include (iostream)	0	1	0
2.	# include (stdio.h)	0	1	0
3.	# include (stdlib.h)	0	1	0
4.	# include (ctype.h)	0	1	0
5.	using name space std,	2	1	2
6.	int found (int x, int 1, int max)	12	1	12
7.	{	0	1	0
8.	for (int i = 0, I<max, 1++)	11	3	33
9.	if (1 [i] = = x) (return (i) ;)	4	2	8
10.	return (-7);	1	1	1
11.	}	0	1	0
12.	int main ( )	1	1	1
13.	{	0	1	0
14.	cout<<"\n\n enter maximum number of frames in the main memory: \t";	4	1	4
15.	int max;	2	1	2
16.	cin > max;	1	1	1
17.	int ∞ 1 = new int [max];	4	1	4
18.	for (int I = 0; I < max; i++) 1 [i] = - 1,	14	1	14
19.	int a, x;	7	1	7
21.	cout <<" \n \n enter the sequence of page request (enter -1 to stop) : \ t	1	1	1
22.	while (1)	0	3	0
23.	{	0	1	0
24.	cin >> x;	3	1	3
25.	if (x = = -1) {	3	2	6
26.	cout << " \n \n" ; break; }	1	1	1
27.	else {	0	1	0
28.	if (k < max)	4	2	8
29.	{	0	1	0
30.	if (nes = found (x, 1, max) ! = -1)	6	2	1
31.	cout << " \n \n page " already exists frame " << res << in MM 1	1	1	
32.	cout << " \n \n Next page: \ t " ;	3	1	3
33.	}	0	1	0
34.	else	0	1	0
35.	{	0	1	0
36.	if (cres = found (x, 1, max) e = -1)	9	2	18
37.	{	0	1	0
38.	cout <<" \n \n page " << " already exists in frame	1	1	1
39.	" << res << " in MM;	4	1	4
40.	cout << " \n \n Next page : \ t " ;	3	1	3
41.	}	0	1	0
42.	else	0	1	0
43.	{	0	1	0
44.	cout << " \n \n page fault has occured " ;	1	1	1
45.	cout << " \n \n page " << x <<" has been allocated	3	1	3
46.	from "<< c <<" in MM by replacing pace " << 1 [c]	7	1	7
47.	1 [c] = x;	6	1	6
48.	c = (c + 1) % max;	7	1	7
49.	cout << " \n \n Next page : \ t " ;	4	1	4

S/N	Code	ANV + MNV	CWU	ICCM
50.	}	0	1	0
51.	}	0	1	0
52.	}	0	1	0
53.	}	0	1	0
54.	delete [ ] 1;	0	1	0
55.	return (0);	1	1	1
56.	}	0	1	0
				187

Line 1 to 4: There is no variable. 0
Line5: There are 2 MNV. 2
Line6: There are 2 ANV and 6 MNV. 3(2) + 6 = 12
Line7: There is no variable. 0
Line8: There are 3 ANV and 2 MNV. 3(3) + 2=11
Line9: There are 3 ANV and 1 MNV. 3(1) + =4
Line10: There is 1 MNV.1
Line11: There is no variable. 0
Line12: There is 1 MNV.1
Line13: There is no variable. 0
Line14: There are 1 ANV and 1 MNV. 3(1) + 1= 4
Line15: There are 2 MNV. 2
Line16: There is 1 MNV.1
Line17: There are 4 MNV. 4
Line18: There are 4 ANV and 2 MNV. 3(4) + 2 = 14
Line19: There are 2 ANV and 1 MNV. 3(2) + 1= 7
Line20: There are 2 ANV and 2 MNV. 3(2) + 2 = 8
Line21: There is 1 MNV. 1
Line22&23: There is no variable. 0
Line24: There is 1 ANV. 3(1) = 3
Line25: There is 1 ANV. 3(10) = 3
Line26: There is 1 MNV. 1
Line27: There is no variable 0
Line28: There are 1 ANV and 1 MNV. 3(1) + 1 = 4
Line29: There is no variable. 0
Line30: There is 1 ANV and 3 MNV. 3(1) + 3 = 6
Line31: There is 1 MNV. 1
Line32: There is 1 ANV and no MNV. 3(1) = 3
Line33&35: There is no variable. 0
Line36: There are 2 ANV and 3 MNV. 3(2) + 3 = 9
Line37: There is no variable. 0
Line38: There is 1 MNV. 1
Line39: There are 1 ANV and 1 MNV. 3(1) + 1= 4
Line40: There is 1 ANV. 3(1) = 3
Line41&43: There is no variable. 0
Line44: There is 1 MNV. 1
Line45: There is 1 ANV. 3(1) = 3
Line46: There are 2 ANV and 1 MNV. 3(2) + 1 = 7
Line47: There are 2 ANV. 3(2) = 6
Line48: There are 2 ANV and 1 MNV. 3(2) +1 =7
Line49: There is 1 ANV. 3(1) + 1 = 4
Line50&53: There is no variable. 0
Line54: There is no variable. 0
Line55: There is 1 MNV. 1
Line56: There is no variable. 0

4 Comparative Studies between Proposed and Existing Cognitive Measures

The cognitive complexity values for different existing cognitive measures and proposed measure for online algorithms (Frequency Count (FC) algorithm, Optimal (OPTIMAL) algorithm, First in First Out (FIFO) algorithm, Least Recently Used (LRU) algorithm, Transpose algorithm, Least Frequently Used (LFU) algorithm) are shown in Table 3 and also the table for Pearson correlation coefficient among the measures are shown in Table 4. The graph for comparison between the existing cognitive measures and the proposed measure are shown in Figs. 2 and 3.

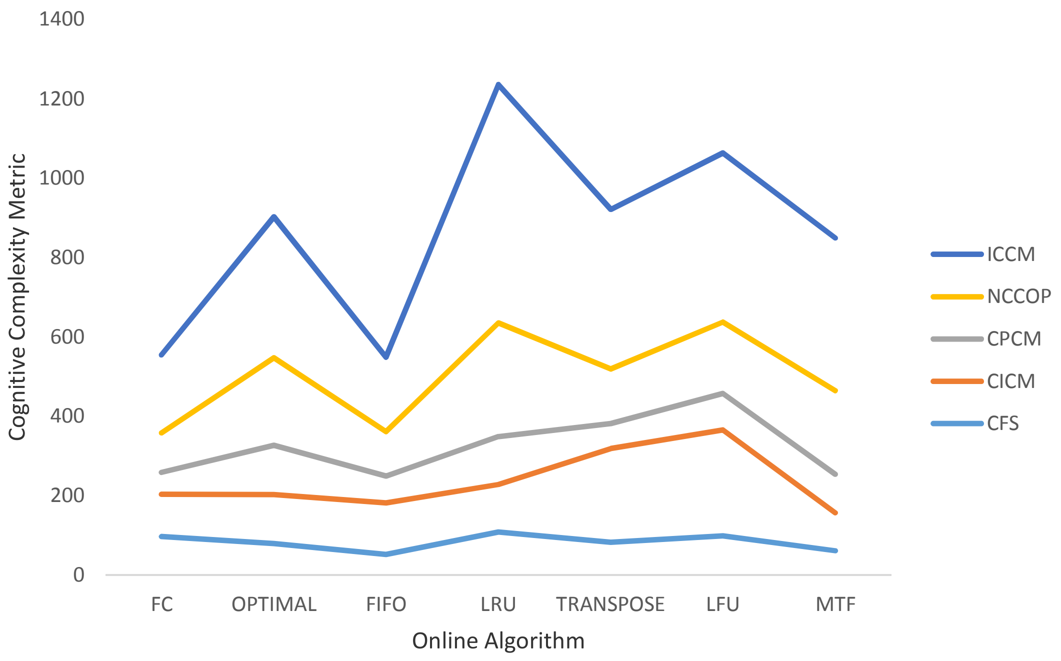

Fig. 2. Relative graph between ICCM, NCCOP, CFS, CPCM and CICM for C++ programs

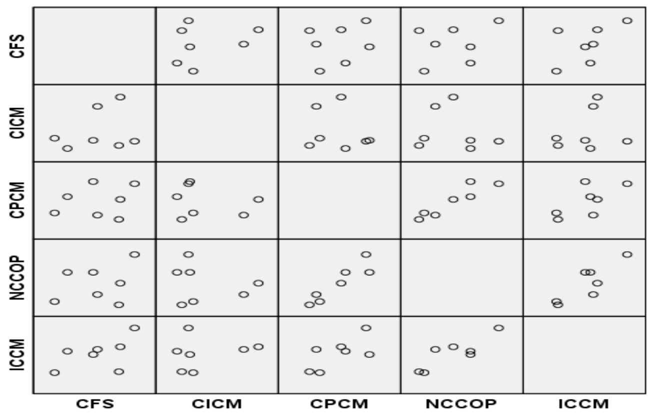

Fig. 3. Scatter plots of complexity values for different measure in C++

Table 3. Complexity values for different measures in C++

ALGORITHM	CFS	CICM	CPCM	NCCOP	ICCM
FC	97	106	55	100	196
OPTIMAL	79	123	125	220	355
FIFO	52	130	67	112	187
LRU	108	120	121	286	600
TRANSPOSE	82	236	63	138	402
LFU	98	267	92	180	426
MTF	61	96	97	220	385

Table 4. Pearson correlation of complexity values for different measures in C++

		CFS	CICM	CPCM	NCCOP	ICCM
CFS	Pearson correlation	1	.268	.220	.319	.546
	Sig. (2-tailed)		.562	.636	.485	.205
	N	7	7	7	7	7
CICM	Pearson correlation	.268	1	-.201	-.187	.232
	Sig. (2-tailed)	.562		.666	.689	.617
	N	7	7	7	7	7
CPCM	Pearson correlation	.220	-.201	1	.924**	.679
	Sig. (2-tailed)	.636	.666		.003	.094
	N	7	7	7	7	7
NCCOP	Pearson correlation	.319	-.187	.924**	1	.860*
	Sig. (2-tailed)	.485	.689	.003		.013
	N	7	7	7	7	7
ICCM	Pearson correlation	.546	.232	.679	.860*	1
	Sig. (2-tailed)	.205	.617	.094	.013	
	N	7	7	7	7	7

***. Correlation is significant at the 0.01 level (2-tailed), *. Correlation is significant at the 0.05 level (2-tailed)*

5 Discussion

In this research, series of experiments were conducted to show the effectiveness of the proposed metric. The results as shown in Table 3, shows that ICCM gives accurate result compared to the other existing cognitive complexity measures. ICCM for FIFO algorithm has the lowest value of 187 which indicates that lower complexity information were packed in the software and also predict how user can easily understand some functions in the code, NCCOP, CFS and CPCM were not able to observe it. Least Recently Used (LRU) algorithm has the highest value of complexity which is (ICCM = 600), which indicates that LRU has the highest complexity information packed in the software. CFS and NCCOP was able to show that but ICCM considers the effort for comprehending the code and the information contained in software.

A relative graph which shows the comparison between CFS, CICM, CPCM, NCCOP and ICCM in C++ program is plotted in Fig. 2. A close inspection of this graph shows that ICCM is closely related to CFS, CICM, CPCM and NCCOP, in which ICCM reflect similar trends. 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 software. For example, ICCM has the highest value for LRU (600) which is due to having larger size of the code and high cognitive complexity.

The correlation coefficient is a statistical measure that measures the relationship between two variables. If one variable is changing its value then the value of second variable can be predicted. it was shown in Fig. 3 that their exist linear relationship between the pairs of different measurement.

6 Conclusion

The result of ICCM exhibits the complexity of program very clearly and accurate than other existing cognitive measures. The practical applicability of the metric was evaluated by different online algorithm codes written in C++ programming language to prove its robustness and well structureness of the proposed measure and also, that there exist a degree of correlation between the measures. The comparative inspection of the implementation of ICCM versus CFS, CPCM, CICM and NCCoP has shown that:

(i) ICCM makes more sensitive measurement, so it provides information contained in a software and also measure the difficulties in understanding the code.
(ii) ICCM could be adopted by programmers in determining the understandability of Object Oriented languages.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Jayanthi B, KrishnaKumari K. Brief study on Software quality metrics and software complexity metrics in Web application. International Journal of Engineering Sciences & Research Technology. 2014;3(12):441-444.

[2] Watson AH, McCabe TJ. Structured testing: a testing methodology using the cyclomatic complexity metric. Computer Systems Laboratory, National Institute of Standards and Technology; 1996.

[3] Nystedt S, Sandros C. Software complexity and project performance. School of Economics and Commercial Law at the University of Gothenburg; 1999.

[4] Kushwaha DS, Misra AK. A modified cognitive information complexity measure of software. ACM SIGSOFT Software Engineering Notes. 2006;31(5):1-4.

[5] Misra S. A complexity measure based on cognitive weights. International Journal of Theoretical and Applied Computer Sciences. 2006;1(1):1-10.

[6] Shao J, Wang Y. A new measure of software complexity based on cognitive weights. Can. J. Elect. Comput. Eng. 2003;28(2):69-74.

[7] Amit Kumar Jakhar, Kumar Rajnish. A new cognitive approach to measure the complexity of software's. International Journal of Software Engineering and Its Applications. 2014;8(7):185-198.

[8] Misra S, Misra AK. Evaluation and comparison of cognitive complexity measure. ACM SIGSOFT Software Engineering Notes. 2007;32:2.

[9] Wang Y, Shao J. Measurement of the cognitive functional complexity of software. The 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), IEEE CS Press, London, UK. 2003; 67-74.

[10] Wang Y. The Real-Time Process Algebra (RTPA). Annals of Software Engineering: An International Journal, USA. 2003;14:235- 274.

[11] Kushwaha DS, Misra AK. Robustness analysis of cognitive information complexity measure using Weyuker properties. ACM SIGSOFT SEN. 2006;31:1.

[12] Misra S. Cognitive program complexity measure. Proc. of 6th IEEE Int'l Conf. on Cognitive Informatics. 2007;120.

[13] Olabiyisi SO, Omidiora EO, Isola EO. Performance evaluation of procedural cognitive complexity metric and other code based complexity metrics. International Journal of Scientific & Engineering Research. 2012;3:9.

[14] Isola EO. Development of an improved cognitive based complexity metric. Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria; 2016.