Original Research Article

AN EXPLORATORY STUDY OF COGNITIVE BASE COMPLEXITY MEASURES OF ONLINE ALGORITHMS

***Abstract*— Measuring the complexity of software has been an insoluble problem in software engineering. Complexity measures can be used to predict critical information about testability of software system from automatic analysis of the source code. In this paper, Improved Cognitive Complexity Metric (ICCM) is applied on C programming language. since C is a procedural language, the cognitive complexity metric is capable to evaluate any procedural language. This paper presents a cognitive complexity metric named ICCM. First, the metric is analytically evaluated using Weyuker's properties for analyzing its nature. Secondly, perform a comparative study of the metric with the existing metric such as NCCOP, CFS, CICM and CPCM, and the result shows that ICCM do better than other metrics by giving more information contained in the software and reflecting the understandability of a source code. Also, an attempts 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, online algorithms**

I. INTRODUCTION

Many well known software complexity measures have been proposed such as [1], Halstead programming effort [2] Oviedo's data flow complexity measures [3], Basili's measure [4] and Wang's cognitive complexity measure [5]. All the reported complexity measures are supposed to cover the correctness, effectiveness and clarity of software and also to provide good estimate of these parameters. Out of the numerous proposed measures, selecting a particular complexity measure is again a problem, as every measure has its own advantages and disadvantages. There is an ongoing effort to find such a comprehensive complexity measure, which addresses most of the parameters of software. Reference [6] suggested nine properties, which are used to determine the effectiveness of various software complexity measures. A good complexity measure should satisfy most of the Weyuker's properties. For measuring the complexity of a code, one must consider most of the internal attributes responsible for complexity.

Complexity is a difficult concept to define. It can be found in relation to software development, software metrics, software engineering for safety, reverse engineering, configuration management and empirical studies of software engineering [7]. So far, there is no exact understanding of what is meant by complexity with various definitions still being proposed. High complexity of a system usually means that the complexity cannot be represented in a short and comprehensive form. Reference [8] stated that complexity (of a modular software system) is a system property that depends on the relationships among elements and is not a property of any isolated element. Reference [9] defined software complexity as "the degree to which a system or component has a design or implementation that is difficult to understand and verify". Therefore, complexity relates both to comprehension complexity as well as to representation complexity. There are some complexity measures based on cognitive aspects such as Cognitive Functional Size (CFS) proposed by [5] 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 [10] 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.

II. REVIEW OF RELATED WORKS

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

A.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 [11] Code based complexity measure comprises Halstead Complexity Measure and Mac Cabe's Cyclomatic Complexity and Lines of Code Metrics.

B. Cognitive Complexity Measures

Cognitive complexity measures quantify human difficulty in understanding the source code [12]. 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).

C. Klcid Complexity Metrics

Klemola and Rilling (2004) proposed KLCID based complexity measure. Defined identifiers as programmer defined variables and based on identifier density (ID).

$$ID = \frac{Total\ number\ of\ identifiers}{Line\ of\ Code} \quad (1)$$

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

$$\text{KLCID} = \frac{Number\ of\ Identifier\ in\ the\ set\ of\ unique\ lines}{Number\ of\ unique\ lines\ containing\ identifier} \quad (2)$$

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 same.

E. Cognitive Functional Size

Reference [5] 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:

$$\text{CFS} = \left(N_i + N_O\right) * W_c \quad (3)$$

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, total cognitive weight is defined as:

$$\text{Wc} = \sum_{j=1}^{q}\left[\prod_{k=1}^{m}\sum_{i=1}^{n} W_c\,(j,k,l)\right] \quad (4)$$

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.

F. 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 [13]. The CICM can be expressed as:

$$\text{CICM} = \text{WICS} * \text{SBCS} \quad (5)$$

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:

$$\text{Ei} = \frac{ICS}{LOCS} \quad (6)$$

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.

G. Modified Cognitive Complexity Measure

Reference [14] modified CFS into Modified Cognitive Complexity Measure (MCCM) by simplifying the complicated weighted information count in CICM as:

MCCM =

(7)

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 [13]. Based on these arguments, CPCM was thus defined as:

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

(8)

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

H. Improved Cognitive Complexity Metric

Improved Cognitive Complexity Metric is defined as the product of the number of variables and Cognitive weight of Basic Control Structure of the software [17]. The ICCM can be expressed as:

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

(9)

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 WC is the weight of BCS as shown in Table 1 corresponding to the particular structure of line.

Table 1. Basic Control Structure (Kushwaha and Misra, 2006)

Category CWU		**BCS**
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

III. Materials and Method

A.The metrics are applied on some online algorithm codes which are written in C language. Ten(10) different types of online algorithms codes were considered. These programs were different from each other in their architecture, the calculations of ICCM for these online algorithms are given in Table 2. The structures of all the 10 programs are as follows: The second column of the tables shows the C codes. The sum of Arbitrarily Named Variables (ANV), the Meaningfully Named Variables (MNV) and the operators in the line is given in the third column of the table. The cognitive weights of each C codes lines are presented in the forth column. The C complexity calculation measure for each line is shown in the last column of Tables 2 and Table 3 shows the ICCM, CICM, CFS, CPCM and NCCOP results of the ten (10) different online algorithm codes.

B.. Analytical Evaluation of ICCM using Weyuker's Property

The ICCM metric was verified to satisfy all nine Weyuker's properties. Weyuker (1988) properties have been suggested as a guiding tool in identification of a good and comprehensive complexity measure by several researchers.

Property 1: $(\exists P)(\exists Q)(|P| \neq |Q|)$ Where P and Q are program body.

This property states that a measure should not rank all programs as equally complex.

ICCM for least recently used (LRU) and least frequently used (LFU) algorithm are considered. LRU contains seven iterations and six branches, LFU contains seven iterations and five branches. The complexity of LRU (ICCM = 405) and LFU as ICCM = 427. It is clear that the complexity of LRU and LFU are different, so this property is satisfied by the proposed measure

Property 2: Let C be a non-negative number then there are only finitely many programs of complexity C.

Calculation of ICCM depends largely on the number of arbitrarily named variables, meaningfully named variables and cognitive weight of Basic Control Structures. Also all the programming languages consist of finite number of BCS's. Therefore ICCM holds for this properly.

Property 3: There are distinct programs P and Q such that /p/ = /Q/

Transpose algorithm has the ICCM value of 416, also considering Move to Front algorithm, the ICCM is 416. These examples showed that the two different programs can have the same complexity, that is 416. So ICCM hold for the third property.

Property 4: $(\exists P)(\exists Q)(P \equiv Q \ \& \ |P| \neq |Q|)$

This property states that the two programs implementing with different algorithm should have different complexity. FIFO program, the 'if 'condition have been replaced by the sequential formula " frame [i] [0] = 0 and frame [i] [1] = -1, in LRU program . With this change ICCM of FIFO is 333 and for LRU is 405. It is clear that the two programs with same objects have different complexity. Hence ICCM holds this property.

Property 5: $(\forall P)(\forall Q)(|P| \leq |P;Q| \text{ and } |Q| \leq |P;Q|)$.

This property states that if the combined program is constructed from class P and class Q, the value of the program complexity for the combined program is larger than the value of the program complexity for the class P or the class Q.

The program body of page replacement algorithm, this program consist of three program body, one for calculating FIFO, the other for LRU and the third program is for calculating the Optimal. FIFO program contains six alterations and 6 branches, LRU program contains seven iterations and four branches. The total cognitive weight of the complete program (FIFO, LRU and OPTIMAL) body is = 1096 ICCM. The complexity of FIFO is 333, LRU = 405, optimal = 315. The cognitive complexity of Page replacement algorithm (FIFO + LRU + Optimal) is greater than FIFO, LRU and Optimal; that is ICCM of FIFO (333) is less than Page replacement (1096) and ICCM of LRU (405) is less than 1096 and ICCM of Optimal (315) is less than 1096. Hence ICCM holds this property.

Property 6(a): $(\exists P)(\exists Q)(\exists R)(|P| = |Q|) \ \& \ (|P;R| \neq |Q;R|)$

Let P be the Transpose program and Q be the MTF program. The ICCM of both the programs is 416. Appending R to P didn't give Q program. Hence property 6(a) is not satisfied by the ICCM.

Property 6(b): $(\exists P)(\exists Q)(\exists R)(|P| = |Q|) \ \& \ (|R;P| \neq |R:Q|)$

This property states that if a new program is appended to two programs which have the same program complexity, the program complexities of two new combined program are different or the interaction between P and R can be different than interaction between Q and R resulting in different complexity values for P + R and Q + R. If any numbers of statements are added into programs p and program Q the complexity will changes. So ICCM hold this property.

Property 7: There are program bodies P and Q such that Q is formed by permutting the order of the statement of p and (/p/ ≠ /Q/).

This property states that permutation of elements within the item being measured can change the metric values. The intent is to ensure that metric values due to permutation of programs. Since variables is dependent on the number of Arbitratily and meaningfully named variable in a given program statement and the number of statements remaining after this very program statement, hence permuting the order of statement in any program will change the value of variables. Also cognitive weights of BCS's depend on the sequence of the statement. Hence ICCM will be different for the two programs. Thus ICCM holds for this property.

Property 8: If P is renaming of Q, then /p/ = /Q/

The measure gives the numeric value so renaming the program will not affect the complexity of a program. Hence ICCM holds for this property

Property 9: $(\exists P)(\exists Q)(|P| + |Q|) < (|P;Q|)$ OR $(\exists P)(\exists Q)(\exists R)(|P| + |Q| + |R|) < (|P; Q;R|)$

This property states that the programs complexity of a new programs combined from two programs is greater than the sum of two individual programs complexities. In other words, when two programs are combined, the interaction between programs can increase the complexities metric value.

For the program Page Replacement Algorithm, if we separate the main program by segregating P (FIFO), Q (LRU) and R (Optimal), we have the program Page replacement algorithm. Where the cognitive complexity of individual are FIFO (333), (LRU) 405 and (Optimal) 315. The combination of the three programs into one program has the complexity of 1053, while the complexity for Page Replacement Algorithm is 1096. Hence 1053 <1096. This proves that ICCM holds for this property.

F. Demonstration of ICCM

The cognitive complexity metric given by equation (9) is demonstrated with Frequency Count Algorithm given by the following Table 2.

Table 2. Frequency Count Algorithm

S/N	**CODE**	**ANV+ MNV**	**CWU**	**ICCM**
1.	# Include<stdio.h>_	0	1	0
2.	int main ( )	1	1	1
3.	{	0	1	0
4.	int arr[100],freq ,[100]	3	1	3
5.	int size,i,j,count,	9	1	9
6.	/* Read size of array and elements in array*/	1	1	1
7.	Printf (“Enter size of array:”),	1	1	1
8.	Scanf (”%d”, &size),	4	1	4
9.	Printf (“Enter elements on array:”),	1	1	1
10.	For (i=o,i<size,i++)	10	3	30
11.	{	0	1	0
12.	Scanf (“%d” ,&arr[i])	7	1	7
13.	Freq[i]=-l,	4	1	4
14.	}	0	1	0
15.	/* counts frequency of each element/*	1	1	1
16.	For(I =o, I <size, I ++)	10	3	30
17.	{	0	1	0
18.	Count =I,	1	1	1
19.	For(j =I + I, j < size, j++)	13	3	39
20.	{	0	1	0

21.	if(arr[i] = = arr [j]	8	2	16
22.	{	0	1	0
23.	Count++,	1	1	1
24.	Freq [j] = 0,	4	1	4
25.	}	0	1	0
26.	}	0	1	0
27.	if (freaq [i]!=0)	4	2	8
28.	{	0	1	0
29.	Freq[i]=count,	5	1	5
30.	}	0	1	0
31.	}	0	1	0
32.	Printf("\n Frequency of all elements of array:\n"),	1	1	1
33.	For (i =0,i<size , i++)	10	3	30
34.	{	0	1	0
35.	If (freq [1] 1 = 0)	4	2	8
36.	{	0	1	0
37.	Print f ("% d occurs % d times in", arr [1], freq [1]) }	10	1	10
38.	}	0	1	0
39.	}	0	1	0
40.	return 0	1	1	1
41.	}	0	1	0
			58	**2582**

IV. COMPARATIVE STUDIES BETWEEN ICCM AND SOME COGNITIVE MEASURES

The cognitive complexity values for different existing cognitive measures and ICCM measure are shown in Table 3 and also the table for pearson correlation coefficient among the measures are shown in Table 4. The graphs for comparison between the existing cognitives measures and ICCM measure are shown in Figure 2 and Figure 3.

Table 3.Cognitive Complexity Values of CICM, CFS, CPCM, NCCOP and ICCM

ALGORITHM	**CFS**	**CICM**	**CPCM**	**NCCOP**	**ICCM**
FC	78	90	55	97	258
OPTIMAL	132	128	91	127	315
FIFO	72	112	74	136	330
LRU	87	93	89	173	405
TRANSPOSE	85	82	60	141	416
LFU	98	102	100	194	427
MTF	92	120	93	238	416

Table 4. Pearson Correlation of Complexity Values for Different Measure in C

		CFS	CICM	CPCM	NCCOP	ICCM
CFS	Pearson Correlation	1	.602	.547	.057	-.005
	Sig. (2-tailed)		.152	.203	.904	.992
	N	7	7	7	7	7
CICM	Pearson Correlation	.602	1	.609	.283	-.149
	Sig. (2-tailed)	.152		.146	.538	.749
	N	7	7	7	7	7
CPCM	Pearson Correlation	.547	.609	1	.717	.492
	Sig. (2-tailed)	.203	.146		.070	.262
	N	7	7	7	7	7
NCCOP	Pearson Correlation	.057	.283	.717	1	.784*
	Sig. (2-tailed)	.904	.538	.070		.037
	N	7	7	7	7	7
ICCM	Pearson Correlation	-.005	-.149	.492	.784*	1
	Sig. (2-tailed)	.992	.749	.262	.037	
	N	7	7	7	7	7

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

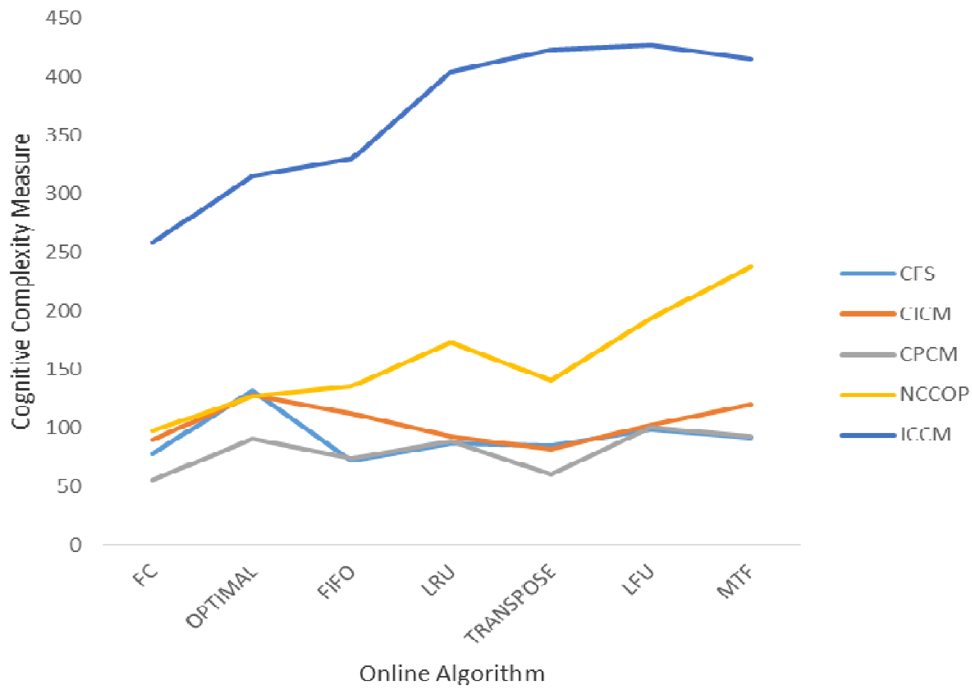

Figure 2. Relative graph between ICCM, NCCOP, CFS, CPCM and CICM for C Programs

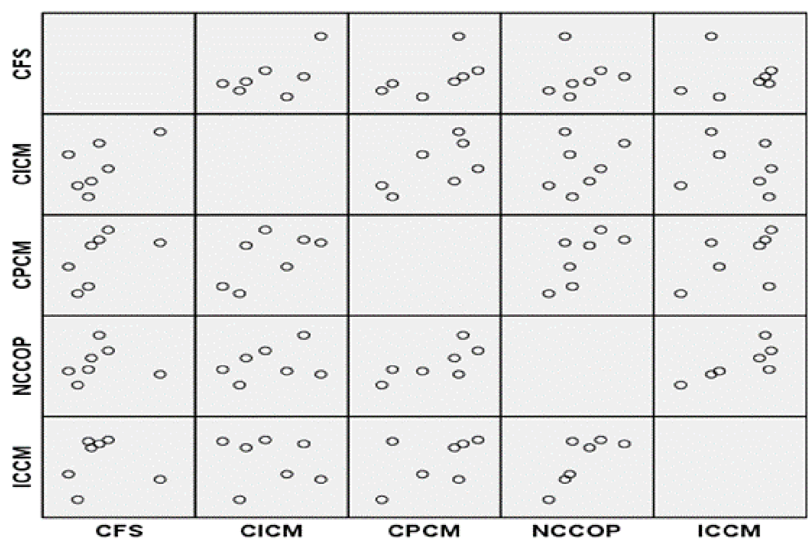

Figure 3: Scatter Plots of Complexity Values for Different Measure

A.Discussion

In this research, series of experiments were conducted to show the effectiveness of the ICCM. The results as shown in Table 3, shows that ICCM gives accurate result compared to the other existing cognitive complexity measures. ICCM for FC algorithm has the lowest value of 258 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 also observed that FC algorithm has the lowest information packed in the program but were not able to reflect code comprehensiveness. LFU algorithm has the highest value of complexity which is (ICCM = 427), which indicates that LFU has the highest complexity information packed in the software. NCCOP, CICM, CFS and CPCM was not able to show that because 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 Figure 3. 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 measure the effort required in comprehending the software. For example, ICCM has the highest value for LFU (427) 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 Figure 3 that their exist positive linear relationship between the pairs of different measurement.

V. CONCLUSION

The result of ICCM exhibit 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. Also ICCM was evaluated through the most famous Weyuker's property, it was found that eight out of the nine properties have been satisfied by ICCM and 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:

- ICCM makes more sensitive measurement, so it provides information contained in a software and also measure the difficulties in understanding the code.
- CFS excludes some important details of cognitive complexity such as information contained in variables, whereas ICCM includes it.
- CICM includes operators which makes it very complicated to calculate whereas information is only contained in the operands/ variables and operators are just used to perform some operation on operands. ICCM was able to handle those isues.
- CPCM is based on total number of occurences of input and output parameters, counting the number of input and output is not clear and ambiguously interpreted. Whereas ICCM was able to handle those issues.
- NCCoP wasn't able to measure the difficulties of code comprehension, Of a fact empirical validations have shown that ICCM was able to reflect the difficulty level of understandability in a program.

The ICCM could be adopted by programmers in determining the understandability of Procedural languages and also provides the information contained in the program.

References

[1] McCabe, T.A. "Complexity measure". IEEE trans. software engineering (se-2, 6) pp. 308-320, 1976.

[2] Halstead M. "Elements of software science". Elsevier North Holland, New York. pp. 67-78, 1977.

[3] Oviedo E."Control flow, data and program complexity". In Proc. IEEE COMPSAC, Chicago, 1980, pp 146-152.

[4] Basili,V.R., Phillips,"T.Y,Metric analysis and data validation across fortran projection" .IEEE Trans.software Eng., SE –9(6), 1983, pp. 652-663.

[5] Wang Y., Shao J. "Measurement of the Cognitive Functional Complexity of Software." Proceeding of the 3rd IEEE International Conference on Cognitive Informatics. pp. 4-9, 2004.

[6] Visscher B.F. 'Exploring Complexity in Software Systems". Ph.D. thesis. Department of Computer Science and Software Engineering. University of Portsmouth, UK. Pp. 130-138, 2005.

[7] Weyuker E. "Evaluating software complexity measure". IEEE Transaction on Software Complexity Measure, Vol.14 No.9, 1988, pp. 1357-1365.

[8] Briand, L.C., Morasca, S., Basili, V.R. "Property-Based Software Engineering Measurement". IEEE Trans. Software Eng. Vol. 22 No. 1, 1996, pp. 68-86.

[9] IEEE Computer Society "IEEE Standard Glossary of Software Engineering Terminology". IEEE std. 610.12, 1990.

[10] 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, 2014, pp. 185-198.

[11] Mayrhauser A.V., Vans A.M. "From Code Understanding Needs to Revers Engineering Tools Capabilities". In: Proc. of 6th Int. Conference on Computer Aided Software Engineering. pp.19-23,1993.

[12] Sanjay Misra, Ibrahim Akman "A New Complexity Metric Based on Cognitive Informatic".Proceedings of 3rd International Conference on Rough Sets and Knowledge Technology, pp. 620–627, 2008.

[13] Misra S. "Cognitive Program Complexity Measure". In Proc. of IEEE, pp.120–125, 2009.

[14] Kushwaha D.S and Misra A.K. "A Modified Cognitive Information Complexity Measure of Software". Proceeding of the 7th International Conference on Cognitive Systems. pp. 120-131, 2008.

[15] Olabiyisi S.O; Omidiora E.O. and Isola E. O."Performance evaluation of procedural cognitive complexity metric and other code based complexity metrics'. September 2012, IJSER, Volume 3,Issue 9.

[16] Isola E. O; Sotonwa K.A. "Performance evaluation of procedural cognitive complexity metric on imperative programming languages" August 2015, IJRASET, Volume 3. Issue viii.

[17] Isola E. O., Olabiyisi S. O., Omidiora E. O.,Ganiyu R. A., Ogunbiyi D.T. and Adebayo O. Y (2016): ''Development of an Improved Cognitive Complexity Metrics for Object- Oriented Codes.'' British Journal of Mathematics & Computer Science. 18(2): PP: 1-11.