IJSRSET17345 | Software Cognitive Complexity Metrics for OO Design: A Survey

Article · January 2019

CITATIONS
3

READS
671

2 authors, including:

Syed Tanzeel Rabani
Baba Ghulam Shah Badshah University
20 PUBLICATIONS **515** CITATIONS

SEE PROFILE

© 2017 IJSRSET | Volume 3 | Issue 3 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Software Cognitive Complexity Metrics for OO Design: A Survey

Syed Tanzeel Rabani[1], K. Maheswaran[2]

[1]Research Scholar, Department of Computer Science, St. Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India

[2]Assistant Professor, Department of Computer Science, St. Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu, India

ABSTRACT

Software metric is used to measure the quality of a software. The conventional metric may be categorized as procedural and Object-oriented metrics. Object-oriented Programming is widely used for software development from the last three decades. There arises a dire need for metrics to evaluate the quality of software in a better manner. Number of metrics are already proposed for OO design but their implementation is still very less. Cognitive Informatics plays an important role in understanding the fundamental characteristics of software. The cognitive complexity metrics is a better indicator to measure the human effort needed to perform the task and measure the difficulty in understanding the software. The primary objective of this paper is to throw some light on various Software cognitive complexity metrics. The classical and modern metrics of software cognitive complexity are discussed and analysed.

Keywords: Software Metrics, Software Complexity, Cognitive Informatics, Cognitive Complexity

I. INTRODUCTION

Software metric is a quantitative technique to measure the quality of a software. It is essential to measure the software Products, Processes and Professionals. Researchers are working hard to propose various metrics for development stage so that software could be properly assessed.

Object oriented programming(OOP) is very much popular because of its various features as inheritance, interaction, polymorphism, dynamic binding, encapsulation etc. It is also easy to implement, modify, maintain, understand and reuse the software code and modules by using OOP.

Software complexity metrics in object-oriented programming refers to the complexity of software code with respect to understandability, modifiability, maintainability and reusability, etc. According to IEEE, Complexity is "the degree to which a system or component has a design or implementation that is difficult to understand and verify". In software, everything is like un-measurable, as it cannot be touched and visualized, therefore Software engineering community is striving for some technique that can measure the complexity of software more accurately. Cognitive informatics (CI) is a promising area from last three decades in the field of research. It is used in various research fields for obtaining a solution of a given problem such as software engineering, artificial intelligence, and cognitive sciences. The CI based on [1]

found that the functional complexity of a software system depends on three factors: input, output and architectural flow Cognitive complexity plays an important role in software measurement. Measuring the complexity of a software using cognitive approach find a way to fully understand the software in all aspects i.e. data objects: input, output, constant and variables, loops and branches so that it reflects difficulty for the developers to understand the software, can be used to predict the effort required to develop, test and maintain the software. The cognitive complexity takes both internal structure and input/output for the software processing.

Cognitive complexity metrics are used as a measurement to quantify human effort needed to perform a task or difficulty in understanding the software. The aim of this survey is to list out some of the existing Cognitive Complexity Metrics, to make the reader aware of their existence, and to offer references for further reading.

II. SOFTWARE COGNITIVE COMPLEXITY METRICS

Wang et al. [5] calculated the cognitive weight of various Base control structures (BCS's). Researchers are proposing new metrics but use the same cognitive weight for (BCS) as proposed by Wang. The summary of various BCS's along with their categorization and Cognitive weights are described in Table I.

This section presents the description of various existing cognitive complexity measures along with their limitations and advantages. Among the various cognitive complexity metrics discussed below, some metrics are code level and other are class and coupling metrics.

.

Message Complexity

Sanjay Misra et al., [3] proposed a metric Message complexity, which focuses on coupling factor between classes. Two classes are said to be coupled if there is message call from one class to another. The metric not only count the total number of such messages but also add weight of the called methods. Thus, the Complexity due to message calls are the sum of weights of call and the weight of called method as calculated in eq.1.

$$CWC = \sum_{i=1}^{n}(2 + MC_i) \qquad (1)$$

Where the number 2 represents the weight of message to an external method and MC_i is the weight of called method.

Method Complexity

Wang [4] proposed the Method complexity, which is used to measure the cognitive weight of a particular method. It functions by assigning weights to the base control structures (BCS) inside a method. The BCS along with their categorizations and corresponding weights are shown in Table I.

Two different scenarios are possible to calculate the Weight in each method. Either all BCSs are in sequential manner or it contains one into another, the latter scenario is calculated below in eq. 2:

$$MC = \sum_{j=1}^{q}\left[\prod_{k=1}^{m}\sum_{i=1}^{n} W_c(j,k,i)\right] \qquad (2)$$

Where W_c is the sum of q linear blocks comprises of individual BCSs. Every block can consist of m nesting BCSs, and each layer contains n linear BCSs. The Method complexity excludes some important details of cognitive complexity such as information that is contains in identifiers and the operators.

TABLE I

COGNITIVE WEIGHTS FOR BCS's

Category	**BCS'S**	**Weight(Wc)**
Sequence	Sequence(SEQ)	1
Branch	If-then-else(ITE)	2
	Case(CASE)	3
Iteration	For-do(Ri)	3
	Repeat-until(R1)	3
	While-do(R0)	3
Embedded Component	Function call(FC)	2
	Recursion(REC)	2
Concurrency	Parallel(PAR)	4
	Interrupt(INT)	4

Cognitive Information Complexity Measure

Kushwaha et. al [9] proposed Cognitive information complexity measure(CICM) which used to calculate cognitive complexity of each method in a class. This measure is computationally simple. CICM is defined in eq. 3

$$CICM = WICS * W_c \quad (3)$$

Where WICS stands for Weighted Information Count of Software and W_c is the weight of BCS's. WICS is defined below in eq. 3.1

$$WICS = \sum_{k=1}^{locs} WICL_k \quad (3.1)$$

Software is a mathematical entity and information contained in the program is a function of identifiers, and the operators are used to perform operations on information.

Information = f (Identifiers, Operators). Information is supplied to the entire program.

$$ICS = \sum_{k=1}^{locs} (I_k) \quad (4)$$

In eq. 4 ICS is the information Contained in Software, LOCs are number of lines in the software. Weighted Information Count of k^{th} LOC (WICL) of a program, which is a function of identifiers, operands and LOC. It is represented in eq. 5

$$WICL_K = ICS_K/[LOCS - K] \quad (5)$$

The weighted information is calculated of each line making it very complex to calculate. It is observed that the information is the function of operators and operands, but the information is only contained in the operands. The operators are only used to perform some operation on operands.

Class Complexity

Mishra [12] suggested a type of metric using cognitive weights to determine the class complexity in object oriented system. It functions by associating a particular weight with each method. After assigning weight it adds the weights of all methods. Thus, complexity of a whole class is determined. If the complexity of whole system is to be determined then weights of classes are added in the same level. But if they are child classes, then the weights are multiplied.

Class Complexity (CC) of a system is calculated in eq. 6

$$CC = \prod_{j=1}^{m} \left[\sum_{k=1}^{n} W_{cjk} \right] \quad (6)$$

Where, W_c is the weight of the concerned class, M represents the depth in object oriented code and n represents the number of classes.The Calculation of class complexity was easy and also language independent, but a well-defined metric not only considers the number of methods, classes and subclasses but should also consider the internal structure of the method.

Weighted Class Complexity

Mishra [13] extended the CC metric and proposed a new metric as Weighted class Complexity (WCC). Object oriented program structure depend upon the attributes and methods in a class. The complexity of a class is calculated by the sum of weights of methods and attributes as in eq. 7

$$WCC = Na + \sum_{p=1}^{s} MCp \qquad (7)$$

Where Na is the number of attributes in a class. MC refers to the method complexity as calculated in eq. 2
Weight of individual classes are added as follows

$$TWCC = \sum_{x=1}^{y} WCC_x \qquad (8)$$

TWCC stands for total weighted Class Complexity. It includes the complexity due to internal structure of methods and attributes. But a well-defined Object-oriented metrics must also consider the powerful concepts of Object Oriented programming like Inheritance, Encapsulation, Overloading and Polymorphism.

Attribute Weighted Class Complexity

Aloysius [15] extended the work of EWCC metric [14] and proposed a new metric as Attribute Weighted Class Complexity (AWCC). As cognitive load is different for different attributes, so in AWCC different cognitive weight are assigned to attributes. These weights were assigned based on the efforts needed to understand the different data types. AWCC is calculated below by eq. 9 as under

$$AWCC = \sum_{i=1}^{n} AC_i + \sum_{j=1}^{m} MCj + \sum_{k=1}^{m} ICC_k \qquad (9)$$

Attribute complexity is derived as:

$$AC = (PDT * W_p) + (DDT * W_d) + (UDDT * W_u) \qquad (9.1)$$

Where PDT is the number of Primary Data Type attributes; DDT is the number of Derived Data Type attributes; UDDT is the number of User Defined Data Type attributes. W_p is the weight of PDT which is 1; W_d is the weight of DDT which is 2; W_u is the weight of UDDT which is 3. AWCC only adds the Attribute complexity to the Extended Weighted Class Complexity and does not add other object-oriented features.

Cognitive weighted coupling between objects

Aloysius et al., [18] proposed cognitive weighted coupling between objects (CWCBO) to overcome the limitation of Coupling between object (CBO) by C.K. CBO doesn't consider the different type of coupling. It only considers the no. of objects to which the given class is coupled. Each couple is assigned as a weight 1. It considers the different types of coupling involved. It is calculated as follows:

$$CWCBO = (CC * WFCC) + (GDC * WFGDC) + (IDC * WFIDC) + (DC * WFDC) + (LC * WFLCC) \qquad (10)$$

Where CC is the total number of modules that contains Control Coupling; WFCC is the Weighting Factor of Control Coupling; DC is the count of Global Data Coupling; WFGDC is the Weighting Factor of Global Data Coupling and its weight is given as 1. IDC is the count of Internal Data Coupling; WFIDC is the Weighting Factor of Internal Data Coupling and its weight is given as 2; DC is the count of Data Coupling; WFIDC is the Weighting Factor of Internal Data Coupling and its weight is given as 2; WFDC is the Weighting Factor of Data Coupling and its weight is given as 3; LCC is count of Lexical Content Coupling; WFLCC is the Weighting Factor of Lexical Content Coupling and its weight is given as 4.

Entire Cognitive Code Complexity

Jakhar et.al [20] proposed cognitive class complexity for object oriented code. The proposed metric first calculates the method complexity by using cognitive weights as per eq. 4. After calculating method complexity, complexity in all methods is added to get class complexity. ECCC was calculated by summing the complexity of all classes in object-oriented code as shown in eq. 11

$$ECCC = \sum_{s=1}^{x} CC \qquad (11)$$

EWCC represents the Entire cognitive code complexity. CC is the class complexity defined by eq. 12

$$CC = \sum_{i=1}^{n} (information + W_c + RASP)_i \qquad (12)$$

Information represents the number of operands and operators. W_c is the cognitive weight of BCS's. as shown in Table I. RASP is the ratio of Accessing similar parameters which is calculated as fallows in eq. 13

$$RASP = \sum M_i \bigcap M_j / na \qquad (13)$$

Where, M represents the methods of a class, i, j are the method numbers and na is the number of attributes in the same class.

New weighted method complexity

Jhakhar et al., [19] proposed a metric as New Weighted Method complexity (NWMC).

The main aim of this metric is to measure the cognitive complexity, analyse the development time and measure the understandability of a program. NWMC is defined in eq. 14

$$NWMC = N_{parameters} * W_c \qquad (14)$$

$$N_{parameters} = N_i + N_o + N_{lp} + N_{fp} \qquad (14.1)$$

Where N_i is the individual number of inputs of the main program and some other program which is called from the main program and some other program; N_o is the individual number of outputs of the main program and some other program which is called from the main program and some other program; N_{lp} is the number of local parameters other than N_i; N_{fp} is the number of formal parameters during function and recursive call; W_c is the cognitive weight of base control structures as calculated in eq. 4. Understandability finds the relationship between the NWMC and difficulty. The understandability is calculated in eq.14.2

$$UA = (NWMC^a) * b \qquad (14.2)$$

Where a and b are constants that are empirically derived using regression.

Improved Cognitive Complexity Measure

Isola esther et al., [2] proposed a cognitive complexity metric for object-oriented code. This metric is named as improved cognitive complexity metric (ICCM). The naming of variables used in code plays a vital role in understanding the code. On evaluating the code, it was found that arbitrarily named variables (ANV) increases the difficulty of understanding three times more than meaningfully named variables (MNV). ICCM is shown in eq. 15

$$ICCM = \sum_{k=1}^{locs} \sum_{v=1}^{locs} (3ANV) * W_c \qquad (15)$$

Where the first summation represents the total lines of code. ANV and MNV represent the arbitrarily and meaningfully named variables. W_C is the weight of BCS as calculated by eq. 2

Code Cognitive Complexity(CCC)

Chhabra [21] proposed Code cognitive complexity (CCC) as a new measure by enhancing Module Cognitive Complexity (MCC). Cognitive complexity doesn't depend alone upon the type of control structures but on various modules, their parameters and return values. The weights for BCS are thus refined for calculating MCC. The two refined categories are shown in Table II.

$$MCC = W_c * Distance + \sum_{i=1}^{N_{ip}+N_{op}} + WP_i \quad (16)$$

In eq. (16) Where Wc represents cognitive weight of control statement from which the module call exists, Distance represents the spatial distance of module call.
It is equal to the absolute difference in number of lines between the module definition and the corresponding call/use. N_{ip} & N_{op} are the number of input and output parameters. WP_i represents the cognitive weight of parameter P_i. Considering the possibility of searching from multiple files for the module's definition, distance is defined as follows.

*Distance = (Distance of call of the module from the top of current file) + (Distance of definition of the module from top of file containing definition) + (0.1 *total lines of code of remaining files) / 2*

The CCC is computed by averaging the MCC values for all calls below in eq. 17

$$CCC = \sum_{j=1}^{m} MCC(MC_j)/m \quad (17)$$

Where m is the count of all module calls in the and MC_j represents the j^{th} call

TABLE II
COGNITIVE WEIGHTS OF MEMBERS NEEDING INTEGRATION WITH SPATIAL DISTANCE

Category	**BCS'S**	**Weight (W_c)**
Constant Data	Constant values	1
	Enumerations and defined constants	1
variables	Atomic	1
	Array (1-d) and structure	2
	Multi-dimensional array and pointer based indirection (single)	3
	Multiple indirection, pointer to structure, etc.	4

III. COMPARATIVE ANALYSIS

This Section briefly analyses various Cognitive Complexity metrics with respect to class, code, inheritance and coupling already discussed in the above section. The comparative analysis of different metrics is briefly discussed and displayed in a tabular format in Table III.

TABLE III
ANALYSIS OF VARIOUS OO COGNITIVE COMPLEXITY METRICS

Object-oriented metrics	**Various aspects of metrics**				
	Class complexity	**Code complexity**	**Inheritance**	**Cognitive**	**Coupling**
CWC	X	X	X	✓	✓
MC	X	X	X	✓	X
CICM	X	✓	X	✓	X
CC	✓	X	X	✓	X
WCC	✓	X	X	✓	X
AWCC	✓	X	✓	✓	X

CWCBO	X	X	X	✓	✓
ECCC	✓	✓	X	✓	X
NWMC	X	✓	X	✓	X
ICCM	X	✓	X	✓	X
CCC	X	✓	X	✓	X

On analysing the table, it can be found that CC, WCC, AWCC. ECCC are the metrics that represent Cognitive complexity at the Class level. Among these metrics, AWCC include Inheritance feature of OO code. CICM, ECCC, NWMC, ICCM and CCC represent cognitive code level metrics. CWC and CWCBO represent cognitive coupling metrics.

IV. CONCLUSION AND FUTURE DIRECTION

This Survey presents various cognitive complexity measures, which consider the weight of different BCSs to measure the complexity. This paper addresses the various complexity metrics involved in method, class, code coupling, inheritance and message passing. These metrics help in finding the cognitive complexities of Object Oriented System. So far, not all the features of the OOPs have been fully addressed like abstraction, packages, etc., The future direction includes some fundamental issues identified from the Section II.

(a) The metric can be developed that include complexity at interface or Package level.
(b) There is a need to develop Standard cognitive complexity measures for Cohesion, modularity, dynamic binding and method overloading.
(c) Some Exiting metrics need to be modified so that results that are more accurate could be achieved. Such as AWCC (Attribute weighted class complexity) could be modified to include Interface and package Complexity.
(d) In ICC part of AWCC, C^L is not properly defined. Complexity at various level of inheritance need to be properly established using well-defined metric.

V. REFERENCES

[1] Y. Wang, "On the Cognitive Informatics Foundations of Software Engineering", Proc. of 3rd IEEE Int'l Conference on Cognitive Informatics,2004.

[2] O. I. Esther, O. O. Stephen, O. O. Omidiora, A.G. rafiu, T.O. Dimple and Y.A. Olajide. "Development of Improved Cognitive Complexity Metrics for Object-oriented Code", British Journal of Mathematics & Computer Science, Vol.18, No. 28515, pp. 1-11,2016.

[3] Misra, Sanjay, Murat Koyuncu, Marco Crasso, Cristian Mateos, and Alejandro Zunino. "A suite of cognitive complexity metrics." In International Conference on Computational Science and Its Applications, Vol.7336, pp.234-247,2012.

[4] J. Shao and Y. Wang, "A new measure of Software Complexity based on cognitive Weights", Canadian journal of Electrical and Computer engineering, Vol. 28, No.2,2003.

[5] Y. Wang and J. Shao, "Measurement of the Cognitive Functional Complexity of Software", The 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), IEEE CS Press, pp. 67-74,2003.

[6] C. A. R. Hoare, I. J. Hayes, J. He, C. C. Morgan, A. W. Roscoe, J. W. Sanders, I. H. Sorensen, J. M. Spivey and B. A. Sufrin, "Laws of Programming", Communications of the ACM, Vol. 30, No. 8, pp. 672-686,1987.

[7] Y. Wang, "The Real-Time Process Algebra (RTPA)", Annals of Software Engineering: An International Journal, Vol. 14, pp. 235- 274,2003.

[8] Y. Wang, "Using Process Algebra to Describe Human and Software Behaviors", Brain and Mind: A Trans. Disciplinary Journal of Neuroscience and Neuro philosophy, Vol. 4, No. 2, pp. 199-213, 2003.

[9] D. S. Kushwaha and A. K Mishra, "Robustness Analysis of Cognitive Information Complexity Measure using Weyuker Properties", ACM SIGSOFT SEN, Vol. 31, No. 1, pp. 1-6, 2006.

[10] S. Mishra, "Modified Cognitive Complexity Measure", Computer and Information Sciences – ISCI, pp. 1050-1059, 2006.

[11] S. Mishra, "Cognitive Program Complexity Measure", Proc. of 6th IEEE Int'l Conf. on Cognitive Informatics, pp. 120-125,2007.

[12] Sanjay Misra and K. Ibrahim Akram, "A new Complexity Metric Based on cognitive informatics", Proceedings of 3rd international Conference on Rough Sets and Knowledge Technology, pp.620-627,2008.

[13] Sanjay Misra and K. Ibrahim Akram, "Weighted Class complexity: A Measure of Complexity for Object Oriented System", Journal of Information Science and Engineering, pp.1689-1708,2008.

[14] L. Arockiam, A. Aloysius and J. Charles Selvaraj "Extended Weighted Class Complexity: A new measure of software complexity for objected oriented systems", Proceedings of International Conference on Semantic E-business and Enterprise computing SEEC, pp. 77 – 80, 2009.

[15] L. Arockiam and A. Aloysius, "Attribute Weighted Class Complexity: A new Metric for Measuring Cognitive Complexity of OO Systems", International Journal of Computer, Electrical, Automation, Control and Information Engineering, Vol.5, No.10, pp. 1151-1156, 2011.

[16] T. Francis Thamburaj and A. Aloysius, "Cognitive Weighted Polymorphism Factor: A Comprehension Augmented Complexity Metric", International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:9, No.11, pp 2342-2374, 2015.

[17] T. Francis Thamburaj and A. Aloysius," Cognitive Perspective of Attribute Hiding Factor Complexity Metric", International Journal of Engineering and Computer Science, ISSN: 2319-7242 Volume 4 Issue 11. pp, 14973-14979 Nov 2015.

[18] L. Arockiam and A. Aloysius," Coupling Complexity Metric: A Cognitive Approach", I.J. Information Technology and Computer Science, pp. 29-35,2012.

[19] Jakhar and Kumar Rajnish. "Measuring Complexity Development time and understandability of Program A cognitive approach". International Journal of Information Technology and Computer Science (IJITCS), Vol. 6, No. 12, pp.53-60,2014.

[20] Jakhar and Kumar Rajnish, "A cognitive measurement of Complexity and Comprehension for Object -oriented Code", International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol.10, No.3, pp.643-650,2016.

[21] Chhabra, "Code Cognitive Complexity", Proceedings of the World Congress on Engineering(WCE11), Vol.2, pp 2-6, London, 2011.