ISSN No. 0976-5697

Volume 3, No. 4, July- August 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Cognitive Weighted Response for a Class: A New Metric for Measuring Cognitive Complexity of OO Systems

A. Aloysius*
Assistant Professor, Department of Computer Science
St.Joseph's College (Autonomous)
Tiruchirappalli, Tamil Nadu, India
aloysius1972@gmail.com

L. Arockiam
Associate Professor, Department of Computer Science
St.Joseph's College (Autonomous)
Tiruchirappalli, Tamil Nadu, India
larockiam@yahoo.co.in

Abstract: Various new techniques, methods and metrics are being developed by researchers for calculating the complexity of the class in Object Oriented (OO) software. Chidamber and Kemerer (CK) have proposed a metric suite for measuring the class complexity of OO design. CK metrics are well known and widely accepted suites of OO metrics. Among those set of metrics, Response For a Class (RFC) is one of the metrics, which is nothing but the number of methods that can be potentially executed in response to a message received by an object of a class. In RFC, each function call statement value is considered to be 1. The cognitive feature is not included in RFC metric which is felt as a major negative aspect of this metric. So here, we are proposing a new metric namely Cognitive Weighted Response For a Class (CWRFC). In CWRFC, the cognitive weights have to be assigned for the function call statement based on the effort needed to understand their type of function calls due to message passed by an object of that class. The proposed metric has been proved to be a better measure of cognitive complexity of class with function call statement through the case studies and experiments.

Keywords: Cognitive Weight, Software Complexity, Cognitive Weighted Response For a Class, Response For a Class, Message Passing.

I. INTRODUCTION

In modern era, the biggest challenge that the software industries are facing, is the upcoming of new technologies. This inevitable change has swept across the corporate world and has changed the demands of the corporate world. This phenomenon has made the software engineers to gear up themselves to meet and manage the change in large software system. And in addition to it the difficulties of software cognitive complexities too should be dealt with [1] [2] [3]. Cognitive informatics is a trans-disciplinary enquiry of cognitive and information sciences that investigate the internal information processing mechanisms and processes of the brain and natural intelligence, and their engineering applications via an interdisciplinary approach [4]. Software complexity deals with the psychological complexity of the programs [5]. These measures serve both as an analyzer and a predictor in quantitative software engineering [6]. The Identification of complex modules is very important as it requires exact testing so as to develop a better quality software system. Additionally, this identification may help during maintenance. Source code metrics can be used to locate such modules. OO technologies have also been increasingly used in organizations these days to identify the complex modules. It is theorized that the structural properties such as coupling, cohesion, functional complexity and inheritance have an impact on the cognitive complexity of the system [7] [8]. And it even places a "mental burden" on developers, inspectors, testers and maintainers to understand the system [9].

Software metrics play a vital role in the software industry to assure the quality of the software. Though the reusability function of OO Paradigm has enriched the capability of several software industries, it has considerably increased the complexity [10]. So, there is an increasing need for introducing new complexity measures. A new metric namely CWRFC is proposed for an OO systems which is an extension of the RFC proposed by CK [11]. CWRFC includes the cognitive complexity due to message passing by an object to the Function Call Statement (FCS) and is a better indicator of complexity of OO systems.

Calling the function is an indispensable part of an OO programming language. Functions are called by its name and the messages may be passed through lists of arguments. Based on the message passing, the FCS is divided into two categories such as Default Function Call Statement (DFCS) and Argumentum Function Call Statement (AFCS). Commonly messages are passed in two ways namely: Pass By Value (PBV) and Pass By Reference (PBR). It is proven that an AFCS may be represented as the combination of PBV and PBR. In PBV, values are passed from the actual arguments of a calling function to the formal arguments of a called function. In PBR, addresses are passed from the actual arguments of a calling function to formal arguments of a called function. It is known that, in PBR the changes made in the called function will be reflected in the calling function, whereas in PBV the changes will not be reflected.

With respect to many cognitive processes, the Cognitive complexity of a computer program can be studied. Program comprehension is one of the important cognitive processes involved in programming. In this paper, a new metric CWRFC is defined and validated against the comprehension process

II. LITERATURE REVIEW

There have been several metrics that were proposed for OO systems by researchers. One of the best known metric suites proposed for OO metrics is CK metric suites. The followings are the six metrics proposed by CK: Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT), Response For a Class (RFC), Number Of Children (NOC), Lack of Cohesion of Methods (LOCM) and Coupling Between Objects (CBO) [11] [12]. Parvinder Singh Sandhu and Dr. Hardeep Singh [5] have proposed a paper that gives the evaluation of CK's metric suites and suggests that certain refinements and extensions to these metrics would bring about accurate and precise results for OO based systems. Raed Shatnawi [13] has proposed a paper that identifies the threshold values for CBO, RFC and WMC at two levels of risks using a quantitative methodology based on the logistic regression curve. These threshold values can be used to identify the most error-prone classes.

Classes are the building blocks of any OO program. Class is an encapsulation of attributes and functions (functions are also known as methods). Functions are the self contained block of statements that perform some kinds of tasks. It reduces the complexity and debugging of the large programs by dividing them into smaller functions. It is clear that the function is one of the major factors which will affect the complexity of the class. The use of different types of function call statements will increase the complexity of the programs. There are no specific measures that exist to calculate the complexity arising due to cognitive load in understanding the different FCS. Hence, a new metric CWRFC has been proposed for OO system with the internal architecture of an object.

The proposed metric CWRFC is explained in section 3, Calibration of FCS is discussed in section 4, the experimentation of a new metric and the case study is described in section 5, a comparative study of CWRFC with RFC in section 6 and section 7 presents the conclusion and future work.

III. PROPOSED METRIC: COGNITIVE WEIGHTED RESPONSE FOR A CLASS

CWRFC is used to calculate the complexity of the class using the Response Set Complexity. If there are m numbers of response sets in a class then, the CWRFC of that class can be calculated by using the Equation (1).

$$CWRFC = \sum_{j=1}^{m} RSC_j \qquad (1)$$

where,

RSC is the response set complexity, which can be calculated by adding the set of all methods (M) in a class and set of methods (R) called by any of those methods.

$$\mathrm{RSC} = \mathrm{M} + \forall_i \mathrm{R}_i \qquad (2)$$

Based on the message passing, the Methods are divided into two categories such as Method With Argument (MWA) and Method without Argument (MOA). MOA is also known as Default Function (DF). Commonly in MWA, the arguments are passed in two ways namely: Pass By Value (PBV) and Pass By Reference (PBR). So RSC can be calculated by using the Equation (3).

$$\mathrm{R} = \mathrm{DF} * (\mathrm{CW_f} + \mathrm{WF_d}) + \mathrm{PBV} * (\mathrm{CW_f} + \mathrm{WF_v}) + \mathrm{PBR} * (\mathrm{CW_f} + \mathrm{WF_r}) \qquad (3)$$

Where

DF is the total number of Default Function Call Statements.

PBV is the total number of Pass By Value Function Call Statements.

PBR is the total number of Pass by Reference Function Call Statements.

CW_f is the Cognitive Weights of the Function Call Statements WF_d is the Weighting Factor of the DFC statements.

WF_v is the Weighting Factor of the PBV statements,

WF_r is the Weighting Factor of the PBR statements.

Wang et.al, [14] have proposed cognitive weights of the control structure in a method as 1, 2, 3, and 4 to the sequence, branch, iteration and function call statement respectively. J.Charles et.al [15] have also validated the weights proposed by Wang. Therefore, the cognitive weight of the Function Call Statement holds the value as 2 by Wang [14].

The weighting factor of different type of the FC statement is based on the classification of cognitive phenomenon as described by Wang [16], is as follow

	Weights
Sub-Conscious Cognitive Function Call Statement (DFC)	1
Meta Cognitive Function Call Statement (PBV)	2
Higher Cognitive Function Call Statement (PBR)	3

IV. CALIBRATION

In this chapter, we validate Default Function Call, Pass By Value and Pass By Reference as Sub, Meta and Higher Cognitive Function Call Statement respectively. A comprehension test has been conducted for a group of students to find out the time taken to understand the complexity of OO program with respect to the function call. The group of students who had sufficient exposure in analyzing the OO programs, and who had undergone courses in C++ language, were selected for comprehension test, and in addition the selected 30 students had 65% and above marks in their Semester Examination.

The time taken by students to comprehend the programs was recorded after the completion of each program, as shown in Fig.1. All these program comprehension timings were registered and the mean time to comprehend was calculated. As three different programs have been administered in each case, totally nine different mean timings were recorded. Average time was calculated for each program from the individual time taken by students, which shows in Fig 2.

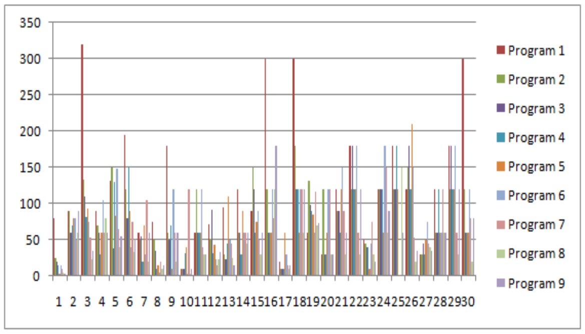

Figure 1. Individual time taken by the student to solve programs.

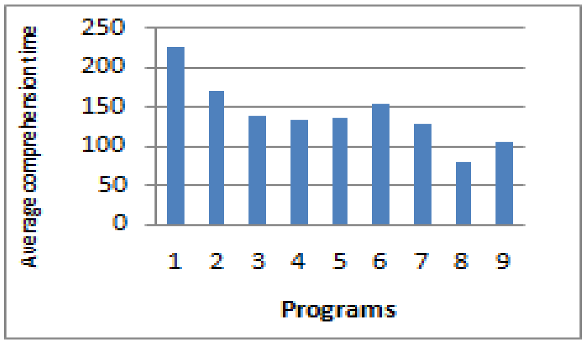

Figure 2. Average comprehension time of each program

Table 1. Average Time of Each Program and Category

Programs	***Average comprehension Time***	***Category***	***Average comprehension category time***
1	227.4483	PBR	179.1494
2	170.1379		
3	139.8621		
4	133.4483	PBV	142.2299
5	137.7241		
6	155.5172		
7	129.1724	DFC	104.9655
8	79.86207		
9	105.8621		

In Table1, the average comprehension time of programs is listed. As these programs are based on OO programming concepts, the mean times are also calculated for each category of the programs and are tabulated. From the above table, it’s clear that, the mean time of PBR is higher than PBV which in turn is higher than DFC. This implies that the cognitive load to understand the PBR is greater than PBV and DFC. Sub concisions cognitive phenomena are used to understand the DFC. Meta cognitive phenomena are needed to understand the PBV. Higher cognitive phenomena is needed to understand the PBR.. The same is reflected in the comprehension time required to understand the program as shown in table 1. Hence, it is concluded that PBR, PBV and DFC belong to Higher, Meta and Sub concisions Cognitive Function Call respectively. The graphical representation of Fig.3 gives a better understanding of the complexity of PBR, PBV and DFC.

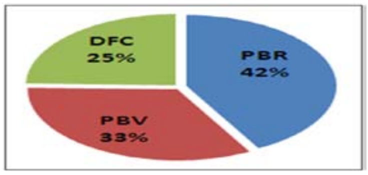

Figure 3. Graphical representation of the categories

V. EXPERIMENTATION AND A CASE STUDY

The proposed complexity metric given by equation 1 is evaluated with the following example program namely PROGRAM 1. PROGRAM 1(with both types of FCS):

```
#include <iostream.h>
using namespace std;
class student
{
protected:
 int roll_number;
public:
  void get_number(int a)
  {
    roll_number = a;
        }
  void put_number(void)
  {
    cout << "Roll No: " << roll_number << endl;
         }
};
class test : public student
{
protected:
  float  part1,part2;
public:
  void get_mark(float  &x, float  &y)
  {
      part1=x;
   part2=y;
        }
  void put_mark(void)
  {
    put_number();
    cout << "Marks obtained: "  << endl;
    cout << "Part1: "  << part1 << endl;
    cout << "Part2 "  << part2 << endl;
        }
};
class sports
{
protected:
  float  score;
public:
  void get_score(float s)
  {
    score = s;
        }
  void put_score(void)
  {
        cout << "Score wt: " << score << endl;
          }
    };
    class  result : public test, public sports
    {
       float  total;
     public:
      void calculate(void)
       {
         total  = part1 + part2 + score;
         void grade(total);
       }
      void grade(float &tot)
       {
         total1=tot;
         if(total1>=40)
            cout<<  "Pass " <<endl;
        else
            cout<<  "Fail " <<endl;
            }
      void display(void);
    };
    void result :: display(void)
    {
       put_mark();
       put_score();
    }
    int main()
    {
       result student1;
         student1.get_number(110);                  //RS1
         student1.get_mark(27.5,30.0);             //RS2
       student1.get_score(6.0);                   //RS3
       student1.calculate();                      //RS4
       student1.display();                        //RS5
    return 0;
    }.
```

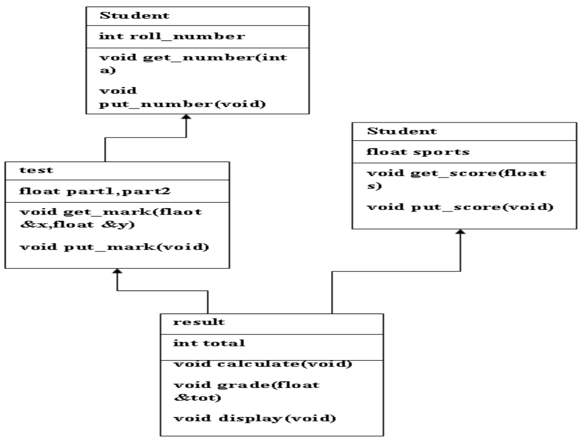

Figure 4. An example of an Object Oriented system with FCS Measurement

In the above program, the number of response set is 5. They are denoted as RS1, RS2, RS3, RS4 and RS5. The complexity measurement of these response sets value are as follows.,

RSC=M+∀_i R_i

R=DF*(2+1)+PBV*(2+2)+PBR*(2+3)

R=DF*(3)+PBV*(4)+PBR*(5)

Calculation for RSC1

M	= 1
R = 0*(3)+0*(4)+0*(5)	= 0
RSC1= 1+0	= 1

Calculation for RSC2

M	= 1
R = 0*(3)+0*(4)+0*(5)	= 0
RSC2= 1+0	= 1

Calculation for RSC3

M	= 1
R = 0*(3)+0*(4)+0*(5)	= 0
RSC3= 1+0	= 1

Calculation for RSC4

M	= 1
R = 0*(3)+0*(4)+1*(5)	= 5
RSC4= 1+5	= 6

Calculation for RSC5

M	= 1
R = 3*(3)+0*(4)+0*(5)	= 9
RSC1= 1+9	= 10

Value of CWRFC

$$\mathrm{CWRFC} = \sum_{j=1}^{m} \mathrm{RSC}_j$$

Here

m=5

$\mathrm{CWRFC} = \mathrm{RSC1} + \mathrm{RSC2} + \mathrm{RSC3} + \mathrm{RSC4} + \mathrm{RSC5}$

= 1+1+1+6+10

= 19

VI. COMPARISON WITH OTHER MEASURES

A comparative study has been made with most widely accepted CK metric suite [11] and has found that RFC metrics proposed by CK et. al did not provide the total complexity of the class by considering the cognitive complexity due to the message passed by an object to the function call of that class. This differentiates our metric from the CK metrics. Mishra et.al suggested that one can calculate the complexity of the class by using cognitive weights of the basic control structure such as sequence, branch, iteration and call structures. The current CWRFC metric is one step ahead of CK's RFC, because it includes the complexity that arises due to the different types of function call statement and internal architecture of an object which passes the message to the functions. Another advantage of our metric is that, it takes cognitive weights into consideration. In the following Table 3, a comparison has been demonstrated with RFC and CWRFC.

We calculated the weight of the class by calculating the response set complexity, in terms of Default Function Call Complexity (DFCC) and Argumentum Function Call Complexity (AFCC). This is a better indicator than the CK's RFC. The weight of each function call statement is calculated by using cognitive weights and weighting factor of type of the message passed to the function call statement by an object which is suggested by Chidamber et al and Wang. We found that the resulting value of CWRFC is higher than the RFC. This is because, in RFC, the weight of each calling statement is assumed to be one. However, including cognitive weights for calculation of the RSC is more realistic because it provides for the complexity of the internal architecture of an object. The results are shown in the Table 2 itself.

Table 2. Complexity Values for Different Programs for the Chosen Metrics

Metrics / Programs	RFC	CWRFC
1	9	33
2	10	29
3	5	17

The RFC and CWRFC values were compared and found that CWRFC measure was larger. According to Chindamber et.al, RFC is an enhanced indicator of complexity of the class. From the table 2, it is observed that CWRFC value is larger than RFC value which concludes that CWRFC is a better indicator of complexity of the classes with function call statement because of the consideration of response set complexity.

VII. CONCLUSION AND FUTURE WORK

A CWRFC metric for measuring the class level complexity has been formulated. The complexity of the class includes the internal complexity of the class and the response set complexity. CWRFC includes the cognitive complexity due to internal architecture of an object, which passes a message to the functions. CWRFC has proven that, complexity of the class getting affected, which is based on the cognitive weights of the different FCS. The assigned cognitive weight of the FCS is validated using the comprehension test and found that the cognitive load to understand the PBR is greater than PBV and DFC. The metric is evaluated through a case study and a comparative study, and proved to be a better indicator of the class level complexity. The proposed metric focuses only on the first level class data. Further, it may be evaluated with the special types of FCS like passing object by reference, object as value, an array of structure to functions, recursive function call and so on., A tool is to be developed for calculating the CWRFC value and to compare it with other metrics. Newer metrics may also be proposed and validated for assessing the cognitive complexity of other OO features.

VIII. REFERENCES

[1] Chiew, V.Wang, Y., “Design of a Cognitive Complexities Measurement and Analysis Tool”, Canadian Conference on Electrical and Computer Engineering (CCECE), 2006, pp.1667–1670.

[2] Jitender Kumar Chhabra,” Cognitive complexity: A New Measure”, Proceedings of the world congress 0n engineering, vol 2 , 2011.

[3] Sanjay Misra,” An approach for the empirical Validation of Software Complexity Measures” Journal of Acta Polytechnica Hungarica , 2011, Vol 8, no 2, ISSN: 17858860.

[4] Wang, Y, “The Theoretical Framework of Cognitive Informatics”, International Journal of Cognitive Informatics and Natural Intelligence , 2007, pp. 1–27.

[5] Parvinder Singh Sandhu and Dr. Hardeep Singh, “A Critical Suggestive Evaluation of CK Metric”, Pacific Asia Conference on Information Systems (PACIS), 2005, pp.183-191.

[6] Basili. VR, Briand. L. C, Melo. WL, “A validation of Object Oriented design metrics as quality indicators”, Technical report, University of Maryland, Department of Computer Science,1995, pp.1-24.

[7] El-Emam, K, “Object-oriented metrics: A review of theory and practice”, Advances in Software Engineering, 2002, pp.23–50.

[8] Sanjay Misra, Ibrahim Akman, Murat Koyuncu, “ An inheritance complexity metric for object-oriented code: A cognitive approach”, Indian Academy of Science, 2011, Vol 36, part 3, pp. 317-337.

[9] Vivanco, R., “Use of a Genetic Algorithm to Identify Source Code Metrics Which Improves Cognitive Complexity Predictive Models”, IEEE international Conference on Program Comprehension, 2007, pp.297–300,.

[10] Ranjeeth. S, Ramu Naidoo “An Investigation Into The Relationship Between The Level Of Cognitive Maturity And The Types Of Errors Made By Students In A Computer Programming” College Teaching Methods & Style Journal-Second Quarter , 2007, pp.31-40

[11] Chidamber. S. R and Kemerer. C. F, “A Metric Suite for Object-Oriented Design”, IEEE Trans. on Software Engineering, 1994, pp.476-493.

[12] Mc Quillan. J. A and Power. J. F, “On the application of software metrics to UML model,” Lecture Notes in Computer Science, 2007, Vol. 4364, pp.217-226.

[13] Raed Shatnawi “An Investigation of CK Metrics Thresholds” ISSRE Supplementary Conference Proceedings, 2006, pp.12-13.

[14] Wang. Y and Shao. J, “A new measure of software complexity based on cognitive Weights.” IEEE Canadian Journal of Electrical and Computer Engineering, 2003, pp.69-74.

[15] Charles Selvaraj. J, Aloysius. A, and Arockiam. L , “A Comparision of Proposed Cognitive weights for control structures and Object Oriented programming languages”, Proceedings of International Conference on Advanced Computing ICAC09, 2009, pp.380-385.

[16] Wang. Y, “On Cognitive Informatics.” IEEE International Conference on Cognitive Informatics, 2002, pp.69-74.