

A Complexity Measure Based on Cognitive Weights

Sanjay Misra

*Department of Computer Engineering
Atilim University, Ankara, Turkey
smisra@atilim.edu.tr*

Abstract

Cognitive Informatics plays an important role in understanding the fundamental characteristics of software. This paper proposes a model of the fundamental characteristics of software, complexity in terms of cognitive weights of basic control structures. Cognitive weights are degree of difficulty or relative time and effort required for comprehending a given piece of software, which satisfy the definition of complexity. An attempt has also been made to prove the robustness of proposed complexity measure by comparing it with the other measures based on cognitive informatics.

Keywords: Software complexity, cognitive weights, basic control structures.

1. Introduction

Many well known software complexity measures have been proposed such as McCabe's cyclomatic number [9], Halstead programming effort [4], Oviedo's data flow complexity measures [10], Basili's measure [2,3], Wang's cognitive complexity measure [16], Knot complexity [12] and others [1,5, 6, 7]. All the reported complexity measures are supposed to cover the correctness, effectiveness and clarity of software and 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 [11].

The complexity measures based on cognitive informatics is in developing phase. Cognitive complexity measures are the human effort needed to perform a task or difficulty in understanding the software. In this paper, an attempt has been made to develop a very simple method for calculating the complexity of code in terms of

cognitive weights. This method is the most suitable due to not only its simplicity but also it provides the complete information about the information contents of a program.

In section 2, we discussed the other complexity measure based on cognitive informatics. In section 3, we propose a new complexity measure with its formulation. The comparison of the proposed measure with the others has been done in section 4. The conclusion is given section 5.

2. Cognitive Weights and Cognitive Informatics

In cognitive informatics, it is found that the functional complexity of software in design and comprehension is dependent on internal architecture of the software. Basic control structures (BCS), sequence, branch and iteration [13, 14, 15] is the basic logic building blocks of any software. The cognitive weight of software [16] is the extent of difficulty or relative time and effort for comprehending a given software modeled by a number of BCS's. There are two different architectures for calculating W_{bcS} : either all the BCS's are in a linear layout or some BCS's are embedded in others. For the former case, sum of the weights of all n BCS's; are added and for the latter, cognitive weights of inner BCS's are multiplied with the weights of external BCS's.

The cognitive weights for Basic Control Structures are as under:

Category	BCS	Weight
Sequence	Sequence (SEQ)	1
Branch	If-Then-Else (ITE)	2
	Case	3
Iteration	For-do	3
	Repeat-until	3
	While-do	3
Embedded Component	Function Call (FC)	2
	Recursion (REC)	3
Concurrency	Parallel (PAR)	4
	Interrupt (INT)	4

Table 1: Basic control structures and their Cognitive Weight

Kushwaha and Misra [7] has proposed a complexity measure, which includes the information contents of software. They consider the theory of Wang [17], which explains that software obeys the laws of Informatics and the Cognitive Science based on the following assertions:

- Software represents computational information.
- Software is a mathematical entity.
- Software is the coded solution to a given program.

- Software is a set of behavioral instructions to computer.

According to Wang [17], Information is the third essence in modeling the natural world supplement to matter and energy. Wang [18] defines software, as “Software in cognitive informatics is perceived as formally described design information and implementations instructions of computing application”. In other words, Wang proved that complexity of any software is in the form of complexity of understanding of the information contained.

Hence, the cognitive complexity of the software should be based on the measure that takes into account the total amount of information contained in the software.

3. Cognitive Weight Complexity Measure (CWCM)

By considering the above theories, the author is in favor of that, although cognitive functional size approach is good but one can find the same conclusion only by the consideration of cognitive weights. Cognitive weights itself provide the sufficient information about the information contained in the software. In the next section, it is proved by comparing this approach with the cognitive functional size approach. There is no need to add more information in cognitive weight measurement for complexity value calculation. Kushwaha and Misra [7] has tried to modify the functional size approach, but it is shown in the next section that there are very much similarity of their approach with functional size approach. Therefore, his proposal also does not contribute very much.

In this proposal, we consider that our cognitive weight complexity measure depends upon:

The Cognitive Weights of Basic Control Structures

In fact, cognitive weights correspond to the number of executed instructions. For example, if in a simple program without any loop, the processor executes only once at the run time. So the weights assigned to such code is one. Cognitive weights of basic control structures are basic building blocks of software and the standard weights for different control structures are given in Table-1. The total cognitive weight of a software component W_c is defined as the sum of cognitive weight of its q linear blocks composed in individuals BCS'S. Since each block may consists of m layers of nesting BCS's, and each layer with n linear BCS's, the total cognitive weight, W_c can be calculated by:

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

The unit of cognitive weight complexity measure is defined as the cognitive weight of the simplest software component i.e. a linear structured BCS i.e.

$$\text{CWCM} = f(W_{\text{bcs}}) = 1 \text{ Cognitive Weight Unit (CWU)}$$

The above measure has been illustrated with the help of an example below.

Example 1. An algorithm to calculate the average of a set of numbers as shown in Figure-1 is used to illustrate the application of CWCM to measure the complexity

```

# define N 10
main ()
{
int count;
float sum, average, number;
sum=0;
count=0;
while(count<N)
{
scanf(“%f”,&number);
sum = sum +number;
count = count +1;
}
average= sum/N;
printf(“N=%dsum = %f”, N,sum);
printf(“average = %f”,average),
}

```

Figure-1: An algorithm to calculate the average of a set of n numbers

We illustrate the CWCM to calculate the complexity of the above program as under:

$$\begin{aligned}
 \text{BCS (sequence)} \quad W_1 &= 1 \\
 \text{BCS (iteration)} \quad W_2 &= 3 \\
 W_c = W_1 + W_2 = 1 + 3 &= 4 \\
 \text{CWCM} &= W_c \\
 &= 4 \text{ CWU}
 \end{aligned}$$

Then, the cognitive complexity measure value of the algorithm is 4 CWU.

4. Comparative Study of Cognitive Weight Complexity Measures with Others

In this section, we have taken different ‘C’ program from [8] for analysis of the result.

We calculated the Cognitive Weight Complexity Measure (CWCM) for different programs. Then, we compared Cognitive Weight Complexity measure with cognitive functional size. The value of Cognitive weight complexity measure and cognitive functional size are given in the table 2.

The CWCM for all the programs gives lower complexity values when we compare it with cognitive functional size approach. It can be easily seen that CWCM already includes the considerations of information contained in terms of cognitive weights. It is also worth mentioned that lower complexity value in terms of number is considered better measure in comparison of measures, which gives higher complexity value.

CWCM has also been compared with cognitive information complexity measure as illustrated in table below, table 3.

No.	Cognitive Weights (CW)	Cognitive Functional Size(CFS)	Ref of source code
1	7	21	Fig 7
2	4	8	Fig 1
3	7	14	Fig 6
4	10	30	Fig 8
5	15	30	Fig 5
6	3	9	Fig 2
7	3	9	Fig 3
8	21	42	Fig 4

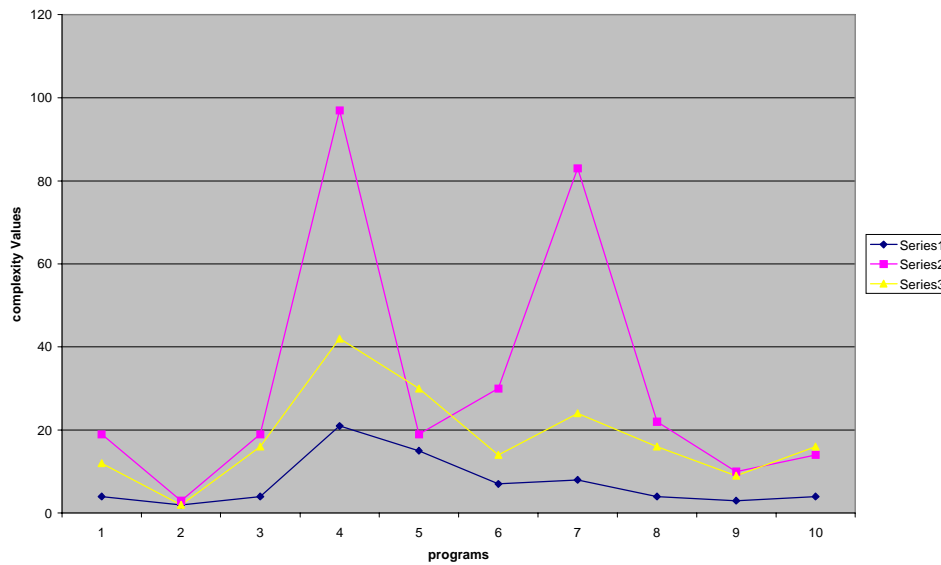
Table 2: Complexity values for CWCM & CFS

No	Cognitive weight complexity measure(CWCM)	Cognitive Information complexity measure(CICM)	Software functional size (CFS)
1	4	19	12
2	2	3	2
3	4	19	16
4	21	97	42
5	15	19	30
6	7	30	14
7	8	83	24
8	4	22	16
9	3	10	9
10	4	14	16

Table 3: Complexity values for CWCM , CICM and CFS.

A plot for CWCM, CICM and CFS is shown in fig 2. (Instead of below).

The plot in fig 2 shows the trends of cognitive weight complexity measure with cognitive information complexity measure. It is seen that the trends of each graph is almost similar, if the complexity value is high for some program, then it reflects in all the graphs. This comparative study proves the similarity between all the complexity measures. Once, we are getting the appropriate information by a small number and by simple calculation, there is no need to adopt the complex method for the same information. It also proves the robustness of this measure.



Series 1: Cognitive weight complexity measure, Series 2 : CICM Series 3: CFS

Figure-2: Graph for CWCM, CICM and CFS

5. Conclusion

A complexity measure based on cognitive weight is proposed. It is found that cognitive weight complexity measure is the most suitable measure, when it is compared with other similar measures. The most important feature of this measure is that it is simple to understand, easy to calculate, less time consuming, gives the complexity value in terms of small number, and language independent i.e. it satisfy most of the property of a good measure. It will aid the developers and practitioners in evaluating the software complexity due to its simple ness, which serves both as an analyzer and as a predictor in quantitative software engineering.

6. Acknowledgement

I am highly thankful to Prof. Ibrahim Akman, chair of the department for his encouragement and support during the work. I am also thankful to my collogues Dr. Alok Mishra and Dr. Hurevren Kilic for useful discussion.

References

- [1] Baker, A.L., and Zweben, S.H. (1980), A comparison of Measures of control flow Complexity, IEEE Transaction on Software Engineering, 6, 506-511.
- [2] Basili. V.R.,(1980), Qualitative software complexity models: A summary in tutorial on models and methods for software management and engineering. IEEE Computer Society Press, Los Alamitos,CA.

- [3] Basili, V.R., Selby, R.W and Phillips, T.Y., (1983) Metric analysis and data validation across fortran projection. *IEEE Transactions Software Engineering*, SE-9(6):652-663.
- [4] Halstead. M.H. (1997), *Elements of software science*, Elsevier North-Holland, New York
- [5] Harrison, W., (1992) An entropy-based measure of software complexity. *IEEE Transactions on Software Engineering*, 18(11): 1025-1029.
- [6] Kearney, J.K., Sedlmeyer, R.L., Thompson, W.B., Gray, M.A. and Adler. M.A., (1986) *Software complexity measurement*. ACM Press, New York, 28:1044-1050.
- [7] Kushwaha, D.S. and Misra, A.K. (2006). Robustness Analysis of Cognitive Information Complexity Measure using Weyuker Properties, *ACM SIGSOFT Software Engineering Notes*, 31, 1, 1-6.
- [8] Misra S. and Misra. A. K. (2004), Evaluating Cognitive Complexity Measure with Weyuker's properties. *Proceedings of third IEEE International Conference on Cognitive Informatics*, 103-108.
- [9] McCabe. T.H. (1976), A complexity measure. *IEEE Transactions Software Engineering*, (SE-2,6):308-320.
- [10] Oviedo, E.I. (1980). Control flow, data and program complexity. *Proc. IEEE COMPSAC*, Chicago, IL, pages 146-152.
- [11] Weyuker. E.J., (1988), Evaluating software complexity measure. *IEEE Transaction on Software Complexity Measure*, 14(9): 1357-1365.
- [12] Woodward, M. R., Hennel. M. A., David . H., (1979) A measure of control flow complexity in program text. *IEEE Transaction on Software Engineering*, SE-5, Vol. 1, pages 45-50.
- [13] Wang. Y. (2002). Component Based Software Measurement in F. Barbier ed. *Business Component - Based Software Engineering*. 247-262.
- [14] Wang. Y. (2002). The real-time process algebra (RTPA). *Annals of Software Engineering An International Journal*, 14:235-274.
- [15] Wang. Y. and Shao J. (2002) Y Wang. On cognitive informatics, Keynote Lecture. *Proceeding of the 1st IEEE International Conference on Cognitive Informatics*, pages 34-42.
- [16] Wang. Y. and Shao J. (2003). A new measure of software complexity based on cognitive weights, *Can.J.Elect. Comput. Eng.*, 28, 2, 69-74.
- [17] Wang. Y. and Shao J. (2004). On cognitive informatics: Foundation of Software Engineering. *Proceeding of the 3rd IEEE International Conference on Cognitive Informatics (ICCI'04)*, IEEE CS Press. 22-31.
- [18] Wang. Y. (2004). On the Informatics Laws of Software. *Proceeding of the 1st IEEE International Conference on Cognitive Informatics (ICCI'02)*, IEEE CS Press. 132-141.