Saudi Journal of Engineering and Technology (SJEAT)
Scholars Middle East Publishers
Dubai, United Arab Emirates
Website: http://scholarsmepub.com/

ISSN 2415-6272 (Print)
ISSN 2415-6264 (Online)

Cognitive Weighted Number of Children for Measuring Coupling in Aspect Oriented Software

Mr. K.R. Martin[1], Dr. E. Kirubakaran[2], Dr. E. George Dharma Prakash Raj[3]
[1]Assistant Professor Department of Computer Science St. Joseph's College (Autonomous), Trichy-2
[2]Professor & Head, Department of Computer Sciences Technology, Karunya University, India
[3]Asst Professor in School of Computer Science, Engineering and Applications, Bharathidasan University, Tiruchirappalli, India

Original Research Article

***Corresponding author**
Mr. K.R. Martin

Article History
Received: 02.02.2018
Accepted: 12.02.2018
Published: 30.03.2018

DOI:
10.21276/sjeat.2018.3.3.4

Abstract: Metrics are an important technique in quantifying desirable software and software development characteristics of aspect oriented software development (AOSD). Theoretical and empirical validation of metrics and of their relation to software attributes is a cumbersome and long process. It is of paramount importance that we validate the utility of metrics we use in order to enable others to use them, too. Aspect oriented programming is an efficient approach to improve the software program at the time of software maintenance for modularizing crosscutting concerns. However, in order to take the advantages of AOP, there is a need for supporting the systematic refactoring of crosscutting concerns to aspects. This paper presents a new cognitive complexity metric namely Cognitive Weighted Number of Children for measuring coupling in Aspect Oriented Software.
Keywords: Metrics, software program, Cognitive Weighted Number, Children.

INTRODUCTION

Software engineering is a difficult and complex task. Software metrics are one way to predict quality within a system, pointing to problem areas that can be addressed prior to software release. Metrics attempt to measure a particular aspect of a software system. Several approaches to estimate complexity of software but none of them have been accepted as a true measure of complexity of an Aspect. Aspect oriented perspective is one of the most significant ways to quantify reliability of software by controlling aspect oriented constructs.

Aspect Oriented Programming (AOP) is a new technology for separating crosscutting concerns that are usually hard to do in object-oriented programming. As AOP has better capability to handle crosscutting concerns than object-orientation it helps to write more modularized and more maintainable code. AspectJ is a general-purpose, aspect-oriented extension to the Java programming language. Given that AspectJ is an extension to Java, every valid Java program is also a valid AspectJ program. In AspectJ has CWNOC metric to measure the different type of inheritance proposed by various researchers. NOC metric is didn't prove their metric according to the statistical approach and data are not accurate. So, there is a need for cognitive Weighted, Number of Children (NOC) and prove given data according to the statistical approach for the Aspect level measurement. Hence our main goal is to define a CWNOC metric to measure the Complexity of coupling.

LITERATURE REVIEW

Ceccato and Tonella[10] introduced many aspect oriented metrics which included aspect oriented coupling metrics as well. The metrics that the study used was the extension of the metrics suite from objects oriented metrics. Similarly to the related OO metric, NOC measures the scope of the properties. The work also collected the value for the metrics from software using the developed tool.

A. Aloysius and G. Arockia Sahaya Sheela [5] studied about aspect oriented metrics. This paper addresses the development and implementation of various metrics for AOP design paradigm and outlines the future directions.

Kumar *et al.* [9] extended their framework for coupling measures from [16-18] frameworks and proposed new coupling metrics for generic AO systems. They identified the connections that cause coupling in aspect oriented system and defined six coupling measures namely coupling on attribute type (CoAT), coupling on parameter type (CoPT), coupling on attribute reference (CoAR), coupling on operation invocation (CoOI), coupling on inheritance (CoI) and coupling on high level association (CoHA).

Ananthi Sheshasaayee and Roby Jose [7] studied about Aspect Oriented Coupling and Cohesion Measures for aspect oriented systems. This study is planned to frame an idea about the coupling, cohesion measures and framework all along with tool support for the coupling measures.

Mandeep Kaur and Rupinder Kaur [1] analysed Improving the Design of Cohesion and Coupling Metrics for Aspect Oriented Software Development. This study focuses on developing metrics for better calculation of coupling and cohesion values.

METRICS ANALYSIS

Existing Metric

Number of Children (NOC)

Similarly to DIT, NOC measures the scope of the properties, but in the reverse direction with respect to DIT proposed by Ceccato *et al*. [10]. The number of children of a module indicates the proportion of modules potentially dependent on properties inherited from the given one.

Proposed Metric

Cognitive Weighted Number of Children (CWNOC)

By adding the weighting factor of different types of inheritance is multiplied by aspect and java's number of children in the hierarchy root.

$$\mathbf{CWNOC = CWNOC_C + CWNOC_A}$$

Here,

CWNOC is over all Cognitive Complexity of Number of Children.

$CWNOC_C$ is Cognitive Complexity of Number of Children for classes.

$CWNOC_A$ is Cognitive Complexity of Number of Children for aspects.

Empirical Metric Data Collection & Evaluation Criteria

This section discusses the CWNOC metric, empirical data, collection statistics, analysis and its implication.

NOC Metric

For empirical analysis, NOC metric is selected for AO software. This metric used to find the complexity of various types of inheritance using Cognitive Approach.

Calibration

In this section, an experiment is conducted to assign cognitive weight to the various types of inheritance. A comprehension test has been conducted for a group of students to find out the time taken to understand complexity of aspect oriented program and java program with respect to different types of inheritance. The group of students selected had sufficient exposure in analysing the Aspect and object oriented programs, as they had undergone courses in AspectJ language. 30 students taken from Rural, 30 students taken from Urban were selected to participate in the comprehension test.

The time taken by students to comprehend the programs was recorded after the completion of each program. The time taken for comprehension of all these programs was noted and the mean time to comprehend was calculated. Five different programs have been administered in each case, totally twenty five different mean timings were recorded. Average time and mean was calculated for each program from the individual time taken by students which shows in Table-1, 2.

Table-1: Categorized Averages Comprehension time for java (in Minutes)

Program	Single	Multilevel	Hierarchical	Hybrid
P1	17	23.7	29.02	33.4
P2	16.62	23.55	28.02	33.43
P3	16.1	23.3	27.98	33.5
P4	16.7	23.6	27.1	33.3
P5	16.25	22.9	27.1	33.5
Average	16.533	23.41	26.83	33.42

Table-2: Categorized Average Comprehension Time for aspect (in Minutes)

program	Single	Multiple	Mixin based
P1	16.7	29.2	33.8
P2	16.6	27.63	33.8
P3	15.97	27.4	34.1
P4	16.82	27.9	34.1
P5	16.2	27.4	33.4
Average	16.45	27.92	33.83

STATISTICAL ANALYSIS

For each inheritance, mean was selected as a measure of central tendency. Table -3 and Table 4 illustrate statistical computation of different types of inheritance.

Table-3: Mean values of different types of inheritance for java (in hours)

Programs	Single	Multilevel	Hierarchical	Hybrid
P1	0.283	0.4	0.484	0.557
P2	0.277	0.393	0.467	0.557
P3	0.27	0.39	0.466	0.56
P4	0.278	0.39	0.45	0.555
P5	0.271	0.38	0.45	0.56
Average	0.271	0.39	0.464	0.557
STDEV	0.006	0.005	0.014	0.001

Table-4: Mean values of different types of inheritance for aspect (in hours)

Programs	Single	Multiple	Mixin-based
P1	0.28	0.49	0.56
P2	0.28	0.461	0.56
P3	0.266	0.46	0.57
P4	0.28	0.46	0.57
P5	0.27	0.46	0.56
Average	0.2742	0.4653	0.5638
STDEV	0.0061	0.0126	0.0053

A standard derivation close to 0 indicates that the data points tend to be very close to the mean of the set.

CWNOC

The proposed metric called CWNOC, which considers the cognitive complexity of the different types of inheritance such as single, multilevel, hierarchical and hybrid inheritance. The exiting NOC metric proposed by Ceccato *et al*. [10] uses the count of children in the given module. It's an equivalent of the NOC metric from CK Metrics suite [11]. This metric does not considered the various types of inheritance. CWNOC can be calculated by using the Equation as follows,

$$\mathbf{CWNOC = CWNOC_C + CWNOC_A} \text{ ----------> } \mathbf{(1)}$$

Here,

CWNOC is over all Cognitive Complexity of Number of Children.

$CWNOC_C$ is Cognitive Complexity of Number of Children for classes.

$CWNOC_A$ is Cognitive Complexity of Number of Children for aspects.

$$\mathbf{CWNOC_C = WFIT*NOC_C}$$
$$\mathbf{CWNOC_A = WFIT*NOC_A}$$

Here,

WHI is Weighting Factor of Inheritance Type.

The Weighting Factor of each type of Inheritance is calibrated in Table 5, Table 6 using the method discussed in the Empirical Metric Data Collection. The weight value is calculated based on the mean value of both java and aspect. Average mean value of each type of java Inheritance is SI=0.271, MLI=0.39, HCL=0.464, HBI=0.557 and aspect inheritance is SI=0.2742, MLP=0.4653, MB=0.5638. To normalize the mean value to get appropriate weight value. The following table explained the rounded values of each type of inheritance that is called weighting factor of each type of inheritance.

Table-5: Weight Value of Each Type of Inheritance for java

Inheritance type	Weight Value
WFSI	0.3
WFMLI	0.4
WFHCI	0.5
WFHBI	0.6

Table-6: Weight Value of Each Type of Inheritance for aspect

Inheritance type	Weight Value
WFSI	0.3
WFMLP	0.5
WFMB	0.6

Comparative Studies

A comparative study has been made with most widely accepted the metric proposed by Ceccato *et al*. [10] is NOC. It's an equivalent of the NOC metric from CK Metrics suite [13]. NOC is a Number of immediate subclasses or sub-aspects of a given module. The current CWNOC metric is one step ahead of existing NOC metric, because it includes the complexity that arises due to the various types of inheritance. Another advantage of CWNOC metric is that, it takes cognitive weights into consideration and data collection satisfies the fenton *et al.* [15] properties. In order to compare the proposed metric a comprehension test was conducted to rural and urban degree students. There were sixty students who participated in the test; the students were given five different programs in AspectJ for the comprehension test. The test was to find out the output of the given programs. The time taken to complete the test in minutes is recorded. The average time taken by all the students is calculated. In the following Table 7, a comparison has been made with NOC, CWNOC and the comprehension test result.

Table-7: Complexity Metric Values and Mean Comprehension Time

Program#	Existing Metric Value (NOC)	Proposed Metric Value (CWNOC)	Mean Comprehension Time
1	1	0.6	13.5
2	1	1.1	13.6
3	2	1.8	16.7
4	2	2	21.5
5	5	6	30.164

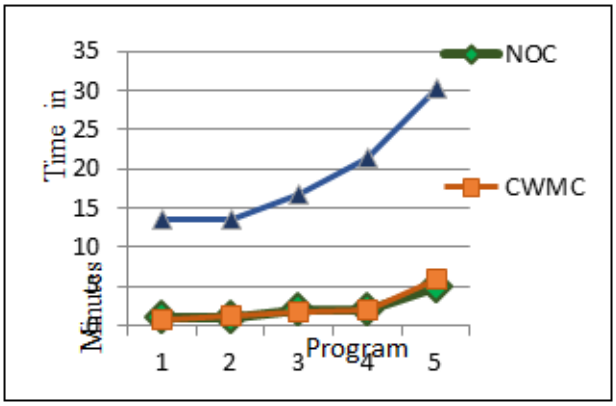

Fig-1: Complexity Metric Values Vs Mean Comprehension Time

Complexity of the NOC is calculated by adding the weighting factor of different types of inheritance is multiplied by aspect and java's number of children in the hierarchy root. This is better indicator than the existing NOC. The weight of each type of inheritance is calculated by using cognitive weights and weighting factor of type of the inheritance similar to which is suggested by Wang *et al*. [14]. It is found that the resulting value of CWNOC is larger than the NOC. This is because, in NOC, the weight of each inheritance is assumed to be one. However, including cognitive weights for calculation of the CWNOC is more realistic because it considers different types of inheritance. The results are shown in the Table 8.8. A correlation analysis was performed between NOC Vs Comprehension Time with r = 0.960262 and CWNOC Vs Comprehension time with r = 0.963971. CWNOC has more positively correlated than NOC. From the Table 8.8, it is observed that CWNOC value is larger than NOC value which concludes that CWNOC is a better indicator of complexity of the classes with various types of inheritance.

CONCLUSIONS

A CWNOC metric for measuring the class & aspect level complexity has been formulated. CWNOC includes the cognitive complexity due to different types of Inheritance. CWNOC has proven that, complexity of the class & aspect getting affected, which is based on the cognitive weights of the various types of Inheritance. The assigned cognitive weight of the various types of Inheritance is validated using the comprehension test. The metric is evaluated through an experimental and proved to be a better indicator of the class level complexity.

REFERENCES

1. Martin, K. R., Kirubakaran, E., & Raj, E. G. D. P. (2017). Extended Metric of Cognitive Weighted Coupling on Method Call.

2. Kirubakaran, E., & Martin, K. R. A Review on Coupling Metrics in Aspect Oriented System.
3. Yang, C. G., Li, J. D., & Tian, Z. (2010). Optimal power control for cognitive radio networks under coupled interference constraints: A cooperative game-theoretic perspective. *IEEE transactions on vehicular technology*, *59*(4), 1696-1706.
4. Sheela, G. A. S., & Aloysius, A. (2017, February). Design and Analysis of Aspect Oriented Metric CWCOAR using Cognitive Approach. In *Computing and Communication Technologies (WCCCT), 2017 World Congress on* (pp. 195-197). IEEE.
5. Aloysius, A., & Sheela, G. A. S. (2015). Aspect Oriented Programming Metrics–A Survey. *The "International Journal of Emerging Trends in Computing and Communication Technology (IJETCCT)*, *1*(3).
6. Sheela, G. A., & Aloysius, A. (2015). Analysis of measuring the complexity of Advice using a Cognitive Approach. *International Journal of Applied Engineering Research, ISSN*, 0973-4562.
7. Sheshasaayee, A., & Jose, R. (2015). A Descriptive Study about Aspect Oriented Coupling and Cohesion Measures. *International Journal of Innovative Science, Engineering & Technology*, *2*(8).
8. Sheela, G. A. S., & Aloysius, A. (2018). Aspect Oriented Programming-Cognitive Complexity Metric Analysis Tool.
9. Kumar, A., Kumar, R., & Grover, P. S. (2009). Generalized coupling measure for aspect-oriented systems. *ACM SIGSOFT Software Engineering Notes*, *34*(3), 1-6.
10. Ceccato, M., & Tonella, P. (2004, November). Measuring the effects of software aspectization. In *1st Workshop on Aspect Reverse Engineering* (Vol. 12).
11. Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on software engineering*, *20*(6), 476-493.
12. Bartolomei, T. T., Garcia, A., Sant'Anna, C., & Figueiredo, E. (2006, November). Towards a unified coupling framework for measuring aspect-oriented programs. In *Proceedings of the 3rd international workshop on Software quality assurance* (pp. 46-53). ACM.
13. Briand, L. C., Daly, J. W., & Wust, J. K. (1999). A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on software Engineering*, *25*(1), 91-121.
14. Arisholm, E., Briand, L. C., & Foyen, A. (2004). Dynamic coupling measurement for object-oriented software. *IEEE Transactions on Software Engineering*, *30*(8), 491-506.
15. Fenton, N., & Bieman, J. (2014). *Software metrics: a rigorous and practical approach*. CRC Press.
16. Shao, J., & Wang, Y. (2003). A new measure of software complexity based on cognitive weights. *Canadian Journal of Electrical and Computer Engineering*, *28*(2), 69-74.
17. Team, A. (2005). The AspectJ programming guide, 2003. *URL http://eclipse. org/aspectj/doc/released/-progguide/index. html*.
18. Laddad, R. (2003). *AspectJ in action: practical aspect-oriented programming*. Dreamtech Press.
19. Miles, R. (2004). *AspectJ Cookbook: Aspect Oriented Solutions to Real-World Problems*. "O'Reilly Media, Inc.".
20. https://eclipse.org/aspectj/doc/released/progguide/startingaspectj.html