International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

Aspect Oriented Programming - Cognitive Complexity Metric Analysis Tool

G. Arockia Sahaya Sheela*[1], Dr. A. Aloysius[2]

[1*]Assistant Professor, Department of Computer Science, Holy Cross College, Trichy, Tamil Nadu, India

[2]Assistant Professor, Department of Computer Science,St. Joseph's College, Trichy, Tamil Nadu, India

ABSTRACT

Metrics extent assured properties of a software system by representing them to numbers (or to other symbols) rendering to well-defined, objective measurement rules. Examining aspect-oriented systems in order to estimate their quality advances its significance as the paradigm remains to increase in popularity. Subsequently, numerous aspect-oriented metrics have been proposed to estimate different aspects of these systems. Aspect Cognitive Complexity Metric are the newly arisen area of complexity metric. This paper describes AOP-CCMAT (Aspect Oriented Programming - Cognitive Complexity Metric Analysis Tool) that is meant for automatically calculate the Cognitive Complexity metrics. Evaluating the AO metrics based on cognitive complexity is supportive for expecting the maintainability of the software. The measurement of Cognitive Complexity is also necessary to calculate the software quality.

Keywords: Aspect Cognitive Complexity Metric Analysis Tool, Cognitive Complexity, Software metrics, Software quality.

I. INTRODUCTION

Aspect-oriented paradigm has developed a protruding software development technology these days. The quality of AO software is measured by means of numerous AO metrics proposed by various researchers in the past. The existing metrics do not reveal the real complexity of AO systems, because, they did not reflect the cognitive complexity in computing such metrics. Hence, it leads to the development of cognitive complexity metrics in AO system. Many researchers have proposed various cognitive complexity metrics in AO system. In order to use these metrics, a new tool has to be proposed for data collection, data analysis, and metrics validation [4]. Unfortunately, there is a lack of such tools. Thus, effort must be devoted to development such tool [6]. Manual implementation of software metrics data collection development is a time consuming and laborious task for software engineers. Hence, to overcome the above problem, a new tool is developed, namely, Aspect Cognitive Complexity Metrics Analysis Tool (AOP-CCMAT). The AOP-CCMAT is used to measure the cognitive complexity of AO software.

II. EXISTING TOOLS

Growth of software initiates from data collection, data analysis, design and implementation. Quality of software can be measured using metrics. These metrics can be calculated through various metric tools. This section provides a set of AO metric tools that are already available. In a cloud environment, scheduling of resources and security of data are the vital disputes. Researchers challenge to resolve these issues by designing efficient algorithms. However, they are supposed to satisfy the cloud software metrics.

A. Java NCSC

It is a simple command line utility that processes two standard source code metrics for the Java programming language [Met, 03]. Metrics are:

- LOC (Lines Of Code)
- NOC (Number Of Classes)

The metrics are composed globally for each class and function. JavaNCSC can optionally present its output with a little graphical user interface.

B. CCMAT - Cognitive Complexity Metrics Analysis Tool

CCMAT is a mindless computational tool to regulate the quality of software. Firstly, the tool parses the given project file to accumulate the metric data. Secondly, the composed metrics data are deposited in the general repository. Thirdly, the object oriented and cognitive complexity metrics are calculated in a builtin framework. Finally, user interface is used to view the results graphically.

III. PROPOSED TOOL

Aspect Oriented Programming - Cognitive Complexity Metric Analysis Tool (AOP-CCMAT)

A. Broad Architecture

The broad architecture of the proposed tool involves of three important components, namely java parser, built-in framework and user interface as shown in Figure1. The three components are briefly explained in the following sub sections.

B. Java Parser

In software engineering, several metrics are used to estimate the quality of software. Data collection is a significant aspect to determine any software metrics. The parse engine is used to mine the information and it aids to calculate the metrics of the given aspect oriented system. In the proposed system, a new parse engine namely, Java parser is developed. The Java parser is used to assemble the metrics data for any Java project file.

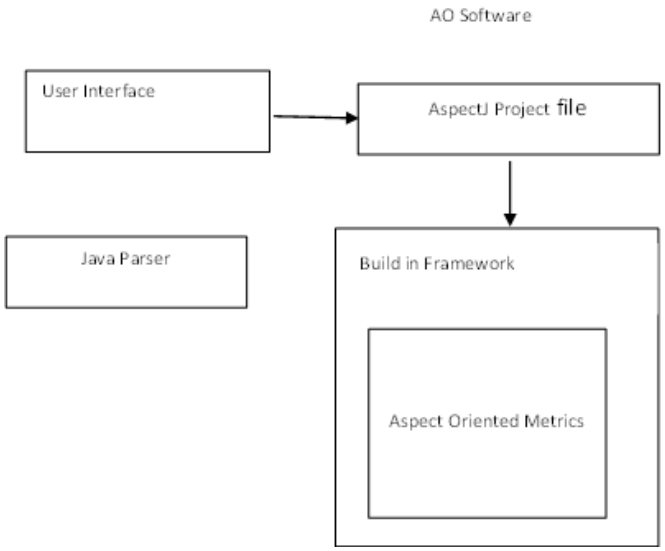

Figure 1. General Architecture of AOP-CCMAT

C. Built-in Framework

The built-in framework is used to estimate the aspect-oriented metrics, existing cognitive complexity metrics and the proposed cognitive complexity metrics. This framework encompasses two modules, namely, existing AOP metrics, and proposed AOP-Cognitive Complexity metrics which are portrayed in Figure 2. The Metric tool encloses methods to compute the following:

Existing AOP metrics

- Weighted Method per Class (WMC)
- Coupling on Advice Execution (CAE)
- Weighted Pointcut per Aspect (WPA)
- Coupling on Attribute Reference (CoAR)

Proposed AOP-Cognitive Complexity metrics

- Cognitive Weighted Method per Class (CWMC)
- Cognitive Weighted Coupling on Advice Execution (CAE)
- Cognitive Weighted Pointcut per Aspect (CWPA)
- Cognitive Weighted Coupling on Attribute Reference (CWCoAR)

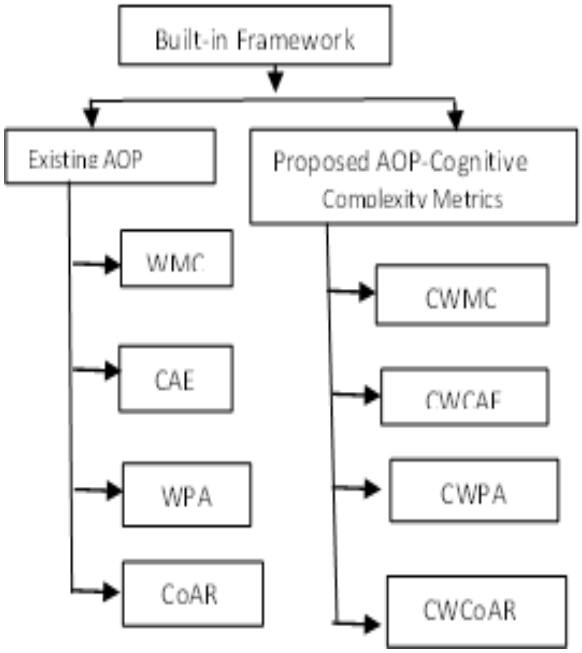

Figure 2. Metrics Built-in Framework

D. User Interface

The graphical user interface is used to show the computed metric outcomes in the graphical form. It assembles all the metrics value from the built-in framework. This interface supports the user to know the complexity of the given Java project file.

IV. COMPARISONS OF TOOLS

Most of the existing tools support C++ software measurement, while the offered tool ropes AspecJ software measurement. Unlike the accessible AOP-CCMAT, existing tools do not maintenance any cognitive complexity metrics. As an alternative they upkeep simple AOP metrics, such as Weighted Method per Class (WMC), which are also maintained by the accessible CCMAT. The presented tool divides the depiction of the system from the design procedure.

V. COMPARATIVE STUDY

The AOP-CCMAT is established and used to extent all the metrics of three different programs. The consequences are shown below in the Table 1 and the graphical demonstration is shown in Figure 3.

The AOP-CCMAT tool is used to bring out assessment between AOP metric suite and AOP Cognitive Complexity Metrics. AOP metrics are used since it has been extensively recognised by the researchers as typical metrics for aspect oriented system. This amendment is used to demonstrate the efficiency of the proposed Cognitive Complexity Metric Suite.

Table 1. Complexity values for various classes for their relevant metric

Proposed Metrics	PRG1	PRG2	PRG3
WMC	1	2	6
WPA	3	3	7
CAE	1	1	4
CoAR	2	2	7
CWMC	2.33	4	10
CWPA	4.53	8	14
CWCAE	2.8	3	6
CWCoAR	5.2	5	9

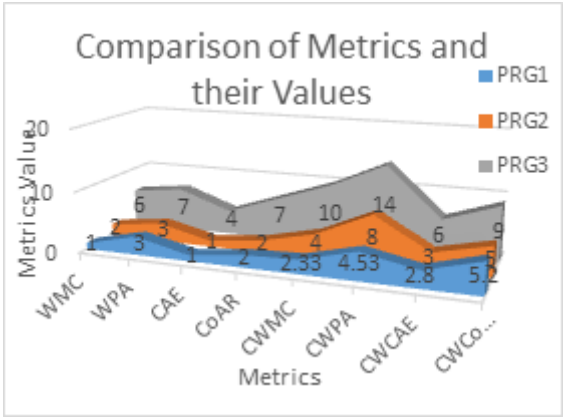

Figure 3. Comparison of metrics and their values

VI. IMPLEMENTATION

A. Aspect Oriented Programming - Cognitive Complexity Metric Analysis Tool (AOP-CCMAT)

In this paper the proposed tool AOP-CCMAT is used to implement the metrics such as CWMC, CWCAE, CWPA and CWCoAR. There are many metric tools available to automatically compute the traditional aspect oriented metrics. But, the proposed tool AOP-

CCMAT is used to compute various cognitive complexity metrics of aspect oriented design.

The AOP-CCMAT collects various cognitive complexity metrics for aspect oriented program. The aspect cognitive complexity metrics data that can be collected using the tool are Cognitive Weighted Method per Class (CWMC), Cognitive Weighted Coupling on Advice Execution (CWCAE), Cognitive Weighted Pointcut Designator (CWPA), and Cognitive Weighted Coupling on Attribute Reference (CWCoAR).

B. Eclipse and Cloud

Eclipse is an integrated development environment (IDE). It contains a vile working area and an extensible plug-in system for adapting the environment. Eclipse is engraved mostly in Java and its primary use is for mounting Java applications, but it may also be used to progress applications in other programming languages through the use of plugins, including: Ada, ABAP, AspectJ, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Ssscheme, and Erlang. It can also be used to progress packages for the software Mathematica. Expansion environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

Nowadays Cloud Computing has become an inevitable technology. The organizations are quickly migrating from existing technology to Cloud services. As the cloud solutions are economical and having complexity, the proposed tool will be highly useful for measuring the software complexity on the deployed solutions.

C. Simulation and Experimentation

In this section, the proposed tool will be described with inputs and outputs. After running the tool, the screen is appeared in the following portion:

When the file is chosen from the file chooser, the path of the file will be displayed in the input panel of the AOP-CCMAT. Figure 4 shows by clicking the ok button the file will be displayed in the input panel.

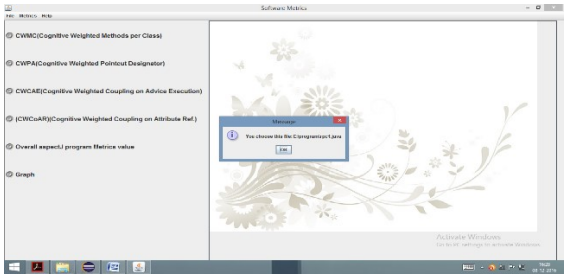

Figure 4. Screenshot for path choosing AspectJ program

The Text Area mentions as to the number of rows and columns, respectively, that the text area should show the AspectJ program in figure 5.

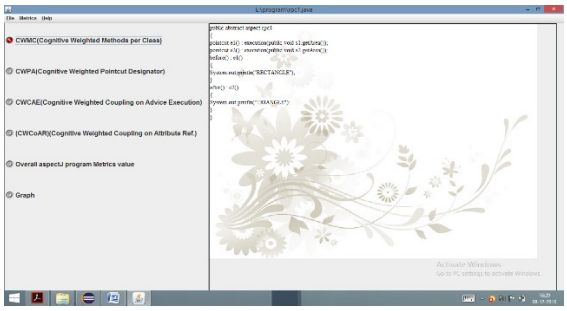

Figure 5. Screenshot for showing AspectJ program

The metric CWMC is selected. The required file wil be displayed in the input panel. The output of the metric CWMC will be displayed in the output panel. First the existing metric WMC will be calculated and the value of the WMC metric is 2. Figure 6 shows that the proposed metric CWMC is calculated and the value of the CWMC metric is 2.33.

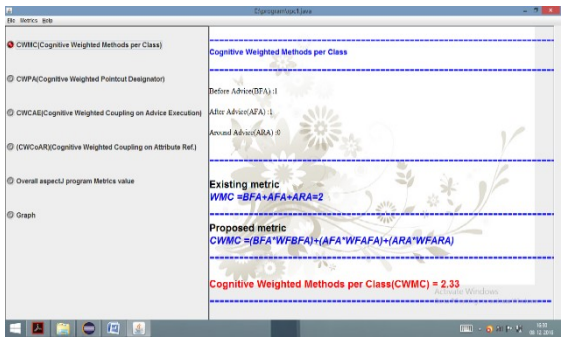

Figure 6. Screenshot for showing CWMC metric

Figure 7 shows that the metric CWPA is selected. The required file wil be displayed in the input panel.

The output of the metric CWPA will be displayed in the output panel. First the existing metric WPA will be calculated and the value of the WPA metric is 2.5. Then the proposed metric CWPA is calculated and the value of the CWMC metric is 13.03. Here the CWPA is calculated by adding CWPA and CWMC.

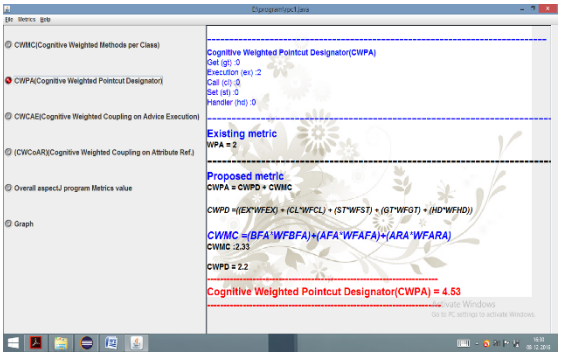

Figure 7. Screenshot for showing CWPA metric

Figure 8 shows that the metric CWCAE is selected. The required file wil be displayed in the input panel. The output of the metric CWCAE will be displayed in the output panel. First the existing metric CAE will be calculated and the value of the CAE metric is 1. Then the proposed metric CWMC is calculated and the value of the CWMC metric is 1.4.

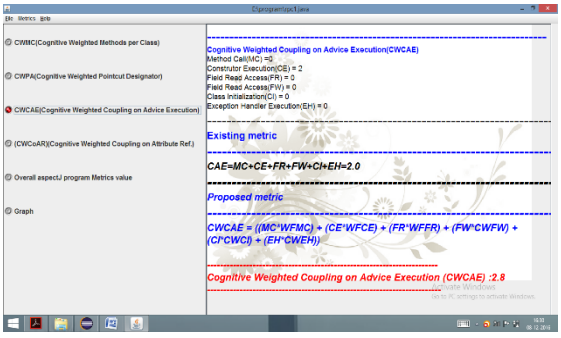

Figure 8. Screenshot for showing CWCAE metric

Figure shows that the metric CWCoAR is selected. The required file wil be displayed in the input panel. The output of the metric CWCoAR will be displayed in the output panel. First the existing metric CoAR will be calculated and the value of the CoAR metric is 1. Then the proposed metric CWCoAR is calculated and the value of the CWCoAR metric is 1.4.

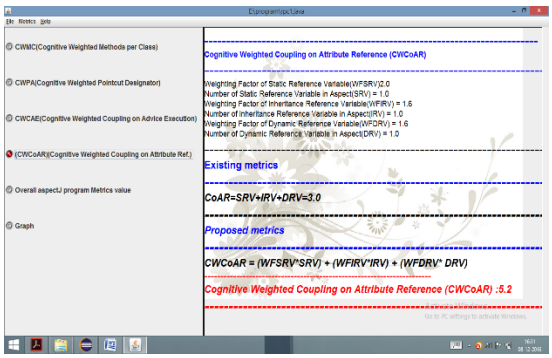

Figure 9. Screenshot for showing CWCoAR metric

VII. CONCLUSION AND FUTURE WORK

This paper defines about AOP-CCMAT, a tool for determining cognitive complexity in aspect-oriented systems. This uses cognitive complexity as its essential feature which is helpful in measuring the Software Quality. It is very suitable to get the AOP metrics and cognitive metrics like CWMC, CWPA, CWCAE, and CWCoAR. During the testing period, this tool is experienced with variety of aspectj programs. The results are confirmed with manual consequences and by the professionals. Improving or adapting this tool is not a critical one and the tool is also not limited for further upgrade because this tool is developed by java program. The organisation of the obtainable tool makes it general and these in turn streamline the support of other aspect-oriented languages and metrics. This paper has also presented precipitate of other existing tools. In future, if there is a need to analyse some other metrics, the CCMAT tool can be prolonged.

Quality of Service (QoS) metrics are used to measure the quality in AOP. QoS metrics are alsp playing a vital role in selecting cloud service providers and efficient utilization of resources. Hence there is a necessity for proposing metrics in cloud environment.

VIII. REFERENCES

[1]. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. M.,& Irwin, J. (1997). Aspect-oriented programming (pp. 220-242). Springer Berlin Heidelberg.

[2]. The AspectJ Team. The AspectJ Programming Guide. 2003.

[3]. Ceccato M, and Tonella P,"Measuring the Effects of Software Aspectization", WARE, 2004.

[4]. KotrappaSirbi and Prakash JayanthKulkarni, "Metrics for Aspect Oriented Programming-An Empirical Study", IJCA, 2010, pp 17-23.

[5]. Parthipan, SenthilVelan and ChitraBabu, "Design Level Metrics to Measure the Complexity Across Versions of AO Software", IEEE, 2014.

[6]. N.Fenton, S.P.fleeger, "Software Metrics: A Rigorous and Practical Approach". PWS Publishing Company, 1997.

[7]. Sant'Anna, C., Garcia, A., Chavez, C., Lucena, C., & Von Staa, A. (2003, October). On the reuse and maintenance of aspect-oriented software: An assessment framework.

[8]. In Proceedings of Brazilian Symposium on Software Engineering (pp. 19-34).

[9]. Chidamber S.R., Kemerer, C.F., "A metrics suit for object oriented design", IEEE, Trans. Software Engineering, 1994, vol 20, pp 476-498.

[10]. Roberto E. Lopez-Herrejon and Sven Apel, Measuring and Characterizing Crosscutting in Aspect-Based Programs: Basic Metrics and Case Studies.

[11]. Alemneh, E. (2014). Current States of Aspect-Oriented Programming Metrics.

[12]. Bartsch, M., Harrison, R, "An Evaluation of Coupling Measures for AspectJ", LATE Workshop AOSD, 2006.

[13]. Kitchenham, B., Pfleeger, S. L., & Fenton, N. (1995). Towards a framework for Software measurement validation. Software Engineering, IEEE Transactions On, 21(12), 929-944.

[14]. Meneely, A., Smith, B., & Williams, L. (2012). Validating software metrics: A Spectrum of philosophies. ACM Transactions on Software Engineering and Methodology (TOSEM), 21(4), 24.

[15]. J. Shao and Y. Wang, "A new measure of software complexity based on cognitive weights", CJECE, 2003.

[16]. A.Aloysius, G.Arockia Sahaya Sheela, "A Review on Aspect Oriented Programming Metrics" 2015.

[17]. Aloysius. A, "Coupling Complexity Metric: A Cognitive Approach", MECS, 2012, vol 9, pp 29-35.

[18]. UshaChhillar, ShuchitaBhasin, "A New Weighted Composite Complexity Measure for Object-Oriented Systems", IJICT, 2011,Vol 1 No 3.

[19]. Puneet Jai Kaur, Sarita Rani, "Analysis of Maintainability Metrics for Aspect Oriented Software & Object Oriented Software", IJETCAS, 2014.

[20]. IvicaBoticki, MarijaKatic, Sergio Martin, "Exploring the Educational Benefits of Introducing Aspect-Oriented Programming Into a Programming Course", IEEE Transactions on Education, 2013, Vol 56, No 2.

[21]. Amit Kumar Jakhar, Kumar Rajnish, "Measuring Complexity, Development Time and Understandability of a Program: A Cognitive Approach", MECS, 2014, Vol 12, pp53-60.

[22]. Joseph D.Gradecki, Nicholas Lesiecki, "Mastering AspectJ – Aspect-Oriented Programming in Java",Wiley Publishing, Inc., 2003.

[23]. Sivanandam, S.N., Sumathi, S., Deepa, S.N., -Introduction to Fuzzy Logic using MATLAB, Springer, Heidel-berg, 2007.

[24]. MacDonell S.G., Gray A.R., Calvert J.M., -Fuzzy Logic for Software Metric Practitioners and Researchers‖, Published in Proceedings of the 6th International Conference on Neural Information Processing ICONIP, Perth, pp. 308-313, 1999.s

[25]. Ryder J., -Fuzzy Modeling of Software Effort Prediction‖, Published in Proceedings of IEEE Information Technology Conference, pp. 53-56, Syracuse, New York, 1998.

[26]. http://www.mathworks.in/help/fuzzy/building-systems-with-fuzzy-logic-toolboxsoftware. html (last accessed on 12/02/14).
[27]. RamnivasLaddad, "AspectJ in Action: Practical Aspect-Oriented Programming", Manning Publications co., 2003.
[28]. P.K. Singh, O. P. Sangwan, A. Pratap, A. P. Singh, An Analysis on Software Testability and Security in Context of Object and Aspect Oriented Software Development, International Journal of Security and Cybercrime, Romania, Vol. 3, Issue 1, pp. 17-28, 2014.
[29]. G.Arockia Sahaya Sheela, and A.Aloysius, "Design and Analysis of Aspect Oriented Metric CWCAE using Cognitive Approach", International Journal of Engineering Research & Technology(IJERT), ISSN: 2278 – 0181, Vol. 5 Issue 04, April, 2016. (Scopus Indexed)
[30]. Dr.Herbert Raj, Dr.A.Aloysius, Dr.L.Arockiam, "Cognitive Complexity Metrics Analysis Tool", IJETCCT, Vo1.1, No.1, Feb. 2014.
[31]. G. Arockia Sahaya Sheela and A. Aloysius, "Analysis of Measuring the complexity of Advice using a Cognitive Approach", "International Journal of Applied Engineering Research (IJAER)",Vol. 10, No.82, 2015.
[32]. Kumar, Avadhesh, Rajesh, and P. S. Grover. "Generalized coupling measure for aspect-oriented systems." ACM SIGSOFT Software Engineering Notes, pp. 1-6, 2009.
[33]. G.Arockia Sahaya Sheela, (2016). Statistical Analysis of Pointcut Complexity Metric using Cognitive Approach, International Journal of Control Theory and Applications (IJCTA), ISSN: 0974-5572, 9(27), 2016, pp. 29-34.
[34]. G.Arockia Sahaya Sheela, (2017). Design and Analysis of Aspect Oriented Metric CWCOAR using Cognitive Approach, IEEE, ISBN: 978-1-5090-5573-9, pp. 195-197.
[35]. https://eclipse.org/aspectj/doc/next/progguide/ semanticsadvice.html
[36]. https://eclipse.org/aspectj/doc/released/proggui de/ startingaspectj.html