Design and Analysis of Aspect Oriented Metric Tool (DAAOMT)

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

Abstract: Design and Analysis of Aspect Oriented Metric Tool (DAAOMT) is used for measuring different software metrics and characteristics of AspectJ programs by accessing class file or byte code and gives in-depth detail of the project in the form of its number of packages, classes, their methods, childrens, objects and fields. The metrics used in DAAOMT are used to predict various characteristics at the earlier stages of software lifecycle by just compiling the completed modules and getting its detail by using the proposed tool in mathematical form. The evaluated metrics results are presented in a graphical user interface that gives exhaustive detail in a very clear and simple way. It establishes quality benchmarks to identify potential design problems at early stages of software lifecycle rather than putting it later ends. The overall aim of DAAOMT is to calculate various software metrics and design attributes for java projects so as to reduce the complexity of maintenance and moving it towards design and coding phase.

Keywords: Aspect Oriented Tool, Metrics, Cognitive, and Coupling.

I. Introduction

Aspect-oriented paradigm has become a prominent software development technology in these days. The quality of AO software is measured using several AO metrics proposed by many researchers in the past. The existing metrics do not reflect the real complexity of AO systems because they did not consider the cognitive complexity while computing such metrics. Hence, it led to the development of cognitive complexity metrics in AO system. Many researchers have proposed various cognitive complexity metrics in AO system. To use these metrics, a new tool has to be proposed for data collection, data analysis, and metrics validation [Her 98]. Manual implementation of software metrics data collection process is a time consuming and laborious task for software engineers. Hence, to overcome the above problem, a new tool is proposed namely, Design and Analysis of Aspect Oriented Metric Tool (DAAOMT), to measure various cognitive complexity metrics of AO software.

II. Existing Tools

Development of software begins from data collection, data analysis, design and implementation. Quality of software can be measured using metrics. These metrics can be computed through various metric tools. This section provides a set of AO metric tools that are already available.

JavaNCSC [Met 03]

It is a simple command line utility that measures two standard source code metrics for the Java programming language [Met 03], namely,

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

These metrics are collected globally for each class and function. JavaNCSC can optionally present its output with a little graphical user interface [Met 03].

CCMAT - Cognitive Complexity Metrics Analysis Tool [Alo 14]

CCMAT is an automatic computational tool to determine the quality of software. Firstly, the tool parses the given project file to collect the metric data. Secondly, the collected metrics data are stored in the general repository. Thirdly, the Object-Oriented and cognitive complexity metrics are calculated in a builtin framework. Finally, the user interface is used to view the results graphically [Alo 14].

CCMATAOP - Cognitive Complexity Metrics Analysis Tool Aspect Oriented Programming [Aro 18]

The tool is designed for AspectJ programs for measuring various aspect oriented metrics. As it is designed only for AspectJ programs, the input given to this tool is .aspectj files [10]. AspectJ programs are first compiled so that the source files (.java file) get converted into the byte code (.class files). These object files (.class files) are feed as an input to the tool and the use of the .class file over .java file is to provide an efficient result as .java files are just text files that can easily be modified.

II. PROPOSED TOOL: Design and Analysis of Aspect Oriented Metric Tool (DAAOMT)

Proposed Metrics

Cognitive Weighted Coupling on Method Call (CWCMC), counts the modules or interfaces declaring different return type methods that are invoked by a module and multiplied by the number of parameters.

Cognitive Weighted Coupling on Field Access (CWCFA), considers the cognitive complexity of the different data types of character, integer, float, long and double.

Coupling Between Objects (CBO) for a class is a count of the number of other classes to which it is coupled. But it does not consider several types of coupling for class & aspect. So the new metric for AOP, Cognitive Weighted Coupling Between Objects (CWCBO) metric is considered various types of coupling for class and aspect and calculated by adding the coupling complexity of classes ($CWCBO_C$) and aspects ($CWCBO_A$).

The proposed metric Cognitive Weighted Depth of Inheritance (CWDIT) is sum of longest path from class and longest path from aspect. In this metric consider both class and aspect in a known module.

Cognitive Weighted Number of Children (CWNOC) is by adding the weighting factor of different types of inheritance is multiplied by aspect and java's number of children in the hierarchy root.

Architecture of DAAOMT

DAAOMT is designed for java programs for measuring various object oriented metrics like, methods, classes, fields, encapsulation and inheritance. As it is designed for java programs only, the input given to this tool is .java files [10]. Java programs are first compiled so that the source files (.java file) get converted into the byte code (.class files). These object files (.class files) are feed as an input to the tool and the use of the .class file over .java file is to provide an efficient result as .java files are just text files that can easily be modified. Also .java files are not readable by Java Virtual Machine (JVM) so it becomes important to convert it in to its byte code. DAAOMT works in three different phases by creating classes dynamically at run time only. The structure is shown in figure 1.

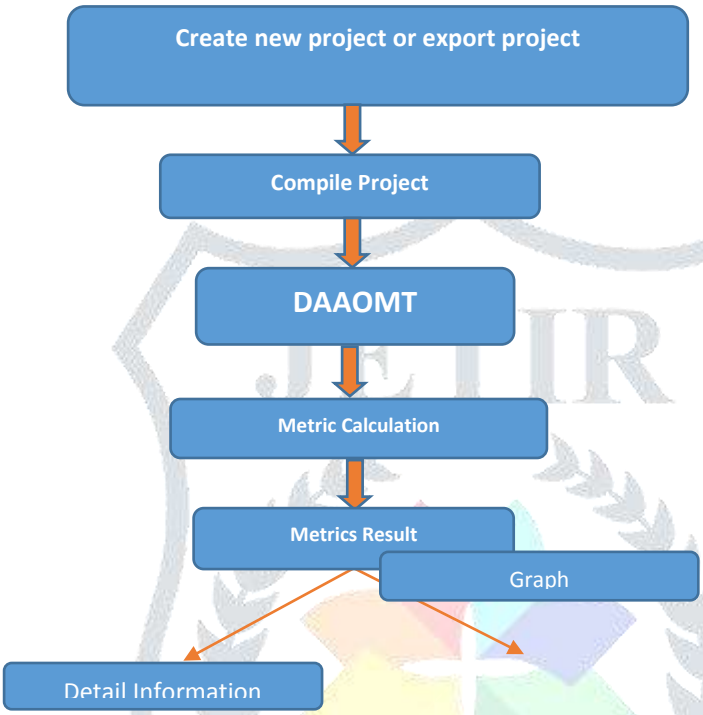

Figure 1 Architecture of DAAOMT

Phase 1: Compiling Java Programs: Phase 1 of DAAOMT is to import various java projects into the Netbeans IDE or Eclipse IDE or any other. Here the role of IDE is just to compile the java programs so that all source files get converted into byte code (*.class* files). Another way of compiling the java programs is through Command Prompt. One way of getting the programs is by importing it from the external sources and another way is to create a new program and then compile it. Compilation of programs completes the phase 1.

Phase 2: Metrics Calculator: This component calculates the AOP metrics. It takes the converted *.class* files from Phase 1 and calculates various Aspect oriented metrics and individual project's attributes based on the logic and formulas defined.

Phase 3: Metrics Results Viewer: This component displays the metrics result in a graphical user interface. This GUI provides different options for detailed information. At the end, it gives the total count option which displays all result at one place.

DAAOMT Implementation

The proposed DAAOMT is implemented using Eclipse Luna. The Java parser is developed to parse the given Java project file to collect the metrics information. A new IDE is developed to integrate and deploy our built-in framework.

III. Design and Analysis of Aspect Oriented Metric Tool (DAAOMT)

In this chapter, the proposed tool DAAOMT is used to implement the metrics such as CWCMC, CWCFA, CWCBO, CWDIT, and CWNOC. There are many metric tools available to automatically compute the traditional Aspect-Oriented metrics. But, the proposed tool DAAOMT is used to compute various cognitive complexity metrics of Aspect-Oriented design. The DAAOMT collects various cognitive complexity metrics for the Aspect-Oriented program. The aspect cognitive complexity metrics data that can be collected using the tool are Cognitive Weighted Coupling on Method Call (CWCMC), Cognitive Weighted Coupling on Field Access (CWCFA), Coupling Between Objects (CBO), Cognitive Weighted Depth of Inheritance (CWDIT), and Cognitive Weighted Number of Children (CWNOC).

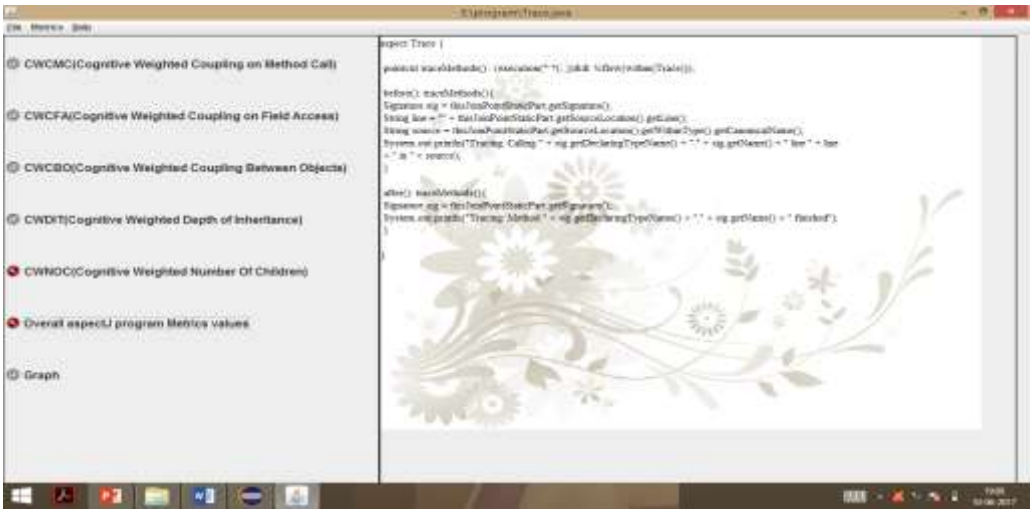

Figure 2 Menu Page

The figure 2 shows that menu page consists of three items namely, file, metrics and help. And also it consists of radio buttons, input panel and output panel. This Radio buttons are groups of buttons in which, by convention, only one button at a time can be selected. The Swing release supports radio buttons with the JRadioButton and ButtonGroup classes. The TextArea provides a component that displays the AspectJ program and optionally allows the user to edit the text. If the text area to be displays, its text using multiple fonts or other styles, one should use an editor pane or text pane. If the displayed text has a limited length and is never edited by the user, use a label.

Simulation and Experimentation

In this section, the proposed tool will be described with inputs and outputs. After running the tool, the screen appears. The home page is the first page a visitor navigating to a project from a software metrics and may also serve as a landing page to attract the attention of visitors. The home page is used to facilitate navigation to other pages on the site, by providing links to important and recent articles and pages, and possibly a search box.

The menu page consists of three items, namely, file, metrics and help, and also it consists of radio buttons, input panel and output panel. These Radio buttons are the groups of buttons in which, by convention, only one button at a time can be selected. The Swing release supports radio buttons with the JRadioButton and ButtonGroup classes. The TextArea provides a component that displays the AspectJ program and optionally allows the user to edit the text. If the text area is to be displayed, its text using multiple fonts or other styles, one should use an editor pane or text pane. If the displayed text has a limited length and is never edited by the user, a label is used. The file menu of DAAOMT consists of three items, namely, input, clear output and exit. The input option is used to select the input; the clear output option is used to clear the output; the exit option is used to exit from the application. When the input option is selected, the open dialog box will appear. The file choosers provide a GUI for navigating the file system, and then either choosing an AspectJ file from a list or entering the name of a file or directory. The file chooser consists of various documents and files from which the required file can be chosen. The file chooser is shown in Figure 3.

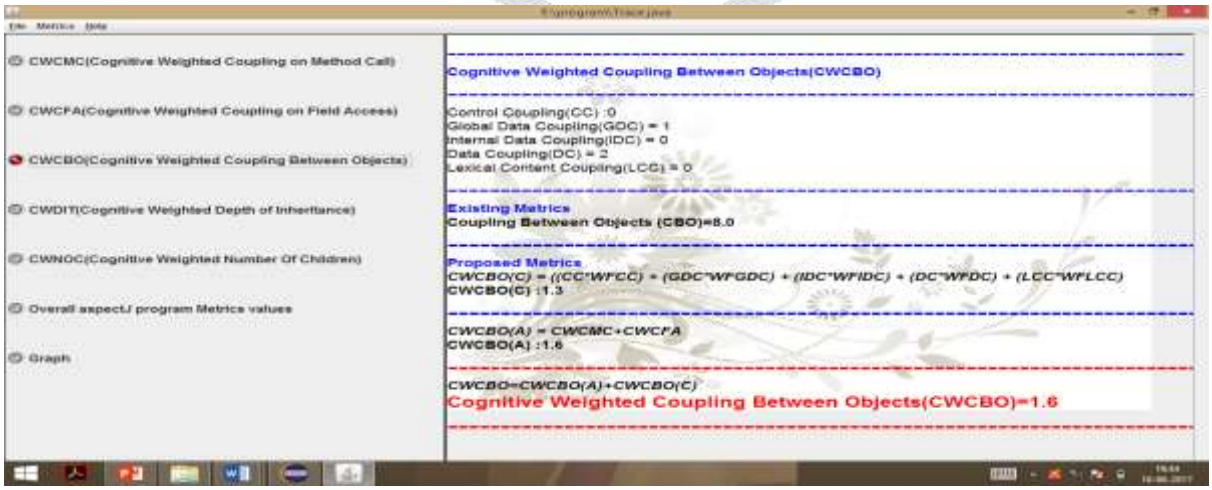

Figure 3 File Chooser Page

The required file is opened in the text area of the GUI and shown in Figure 4.

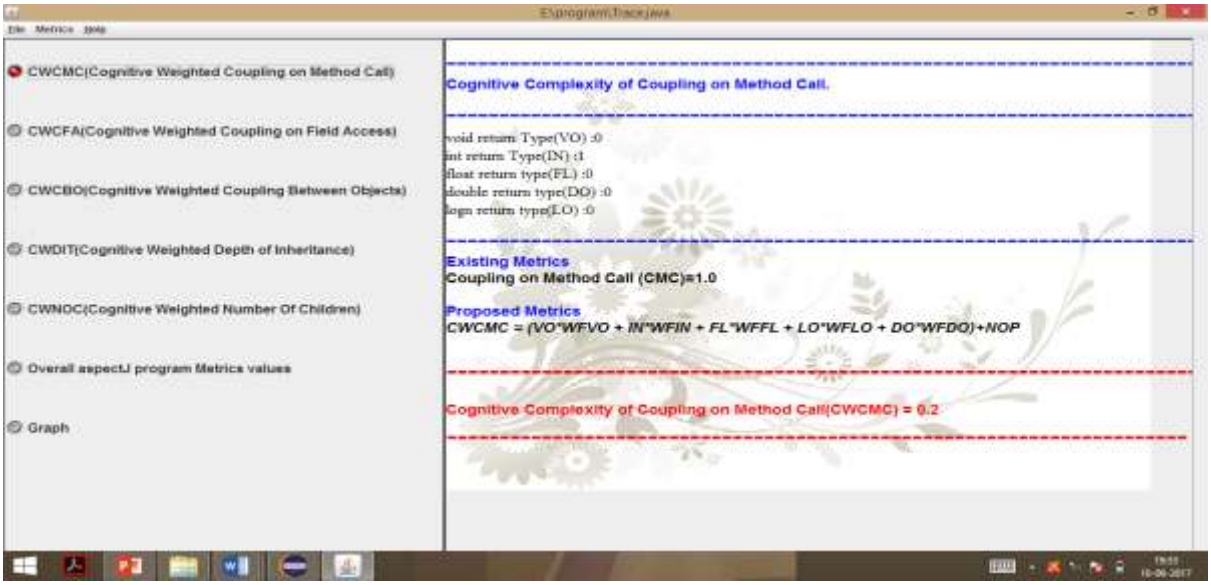

Figure 4 Text Area Page

The metrics consist of CWCMC, CWCFA, CWCBO, CWDIT, and CWNOC. According to the selection of the user, the corresponding calculated metric value of the given program is shown in the text area.

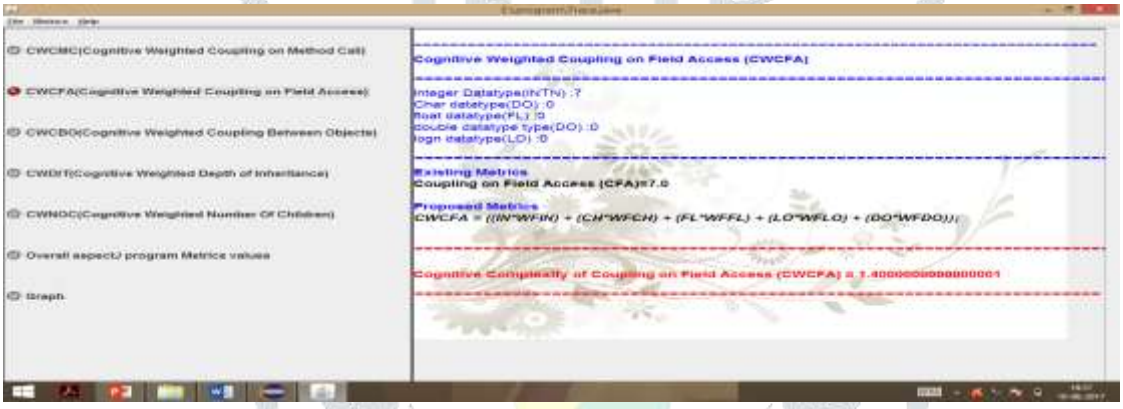

Figure 5 Output of CWCMC Metric

The above Figure 5 shown the calculated metric value of CWMC. Each proposed metrics and their existing metrics are given on the Home page as radio buttons. The entire metric suite values are also calculated and displayed on the screen when the Overall AspectJ Program metrics value is selected on the Home page. The equivalent graph for the metrics CWCMC, CWCFA, CWCBO, CWDIT, and CWNOC are displayed by using the tool.

IV. Conclusion

This chapter describes DAAOMT, a tool for measuring cognitive complexity in aspect-oriented systems. This uses cognitive complexity as its important feature which is helpful in measuring the Software Quality. It is very useful to get the AOP metrics and cognitive metrics like CWCMC, CWCFA, CWCBO, CWDIT, and CWNOC. During the testing period, this tool is tested with a variety of AspectJ programs. The results are verified with manual results and by the experts. Enhancing or modifying this tool is not a critical one, and the tool is also not restricted for further upgradation because this tool is developed in Java. The structure of the presented tool makes it general and these in turn simplify the support of other aspect-oriented languages and metrics. This chapter has also presented summary of other existing tools. In future, the DAAOMT can be extended to calculate the other metrics. The following chapter enumerates the empirical validation of the proposed CCMS using maintenance effort prediction model and the comparative study with other AOP metrics.

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/progguide/ startingaspectj.html