Cognitive Complexity
Metric Analysis Tool -
Aspect Oriented
Programming
CHAPTER - 8

COGNITIVE COMPLEXITY METRIC ANALYSIS TOOL - ASPECT ORIENTED PROGRAMMING

8.1 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 [Boo, 86]. 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, Cognitive Complexity Metrics Analysis Tool - Aspect Oriented Programming (CCMAT-AOP), to measure various the cognitive complexity metrics of AO software.

8.2 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.
8.2.1 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)

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

8.2.2 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].

8.3 PROPOSED TOOL: Cognitive Complexity Metric Analysis Tool - Aspect
Oriented Programming (CCMAT-AOP)

8.3.1 Proposed Metrics

Cognitive Weighted Methods per Class (CWMC), which considers the cognitive
complexity of the different types of advices, such as before, after, and around. This
CWMC metric is discussed in Chapter 4.

Cognitive Weighted Coupling on Advice Execution (CWCAE), which
considers the cognitive complexity of the different types of joinpoints. This CWCAE
metric is discussed in Chapter 5.
Cognitive Weighted Pointcut per Aspect (CWPA), adds cognitive weight of the pointcut designator (CWPD) and cognitive weight of the joinpoint signature used in an aspect. This CWPA is discussed in Chapter 6.

Cognitive Weighted Coupling on Attribute Reference (CWCoAR) metric can measure the attribute factor complexity based on cognitive perspective and consider the different types of attribute allusion. This CWCoAR is discussed in Chapter 7.

### 8.3.2 Architecture of CCMAT-AOP

The architecture of the proposed tool consists of three important components, namely java parser, built-in framework and user interface as shown in Figure 8.1. The three components are briefly explained in the following sub sections.

**Java Parser**

In software engineering, various metrics are used to evaluate the quality of software. Data collection is an important phase to calculate any software metrics. The parsing engine is used to extract the information, and it helps to compute 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 collect the metrics data for any Java project file.
The built-in framework is used to calculate the aspect-oriented metrics, existing cognitive complexity metrics and the proposed cognitive complexity metrics. This framework contains two modules, namely, existing AOP metrics, and proposed AOP-Cognitive Complexity metrics which are depicted in Figure 8.2. The Metric tool contains 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 (CWCAE)
- Cognitive Weighted Pointcut per Aspect (CWPA)
- Cognitive Weighted Coupling on Attribute Reference (CWCoAR)

![Figure 8.2 Metrics Built-in Framework](image)

**User Interface**

The graphical user interface is used to display the computed metric results in the graphical form. It collects all the metrics value from the built-in framework. This interface helps the user to know the complexity of the given Java project file.
8.3.3 CCMAT-AOP Implementation

The proposed CCMAT 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.

8.4 Comparisons of Tools

Most of the existing tools support C++ software measurement, while the presented tool supports Java software measurement. Unlike the presented CCMAT-AOP, existing tools do not support any cognitive complexity metrics. Instead, they support simple AOP metrics, such as Weighted Method per Class (WMC), which are also supported by the presented CCMAT. The presented tool separates the representation of the system from the calculation procedure.

8.5 Comparative Study

The CCMAT-AOP is developed and used to measure all the metrics. The results are shown below in Table 8.1, and the graphical representation is shown in Figure 8.3.

The CCMAT-AOP tool is used to carry out a comparison between AOP metrics and AOP Cognitive Complexity Metrics. AOP metrics are used since it has been widely accepted by the researchers as standard metrics for Aspect-Oriented system. This tool is used to prove the effectiveness of the proposed Cognitive Complexity Metric Suite.
Table 8.1 Complexity values for different classes for their respective metric

<table>
<thead>
<tr>
<th>Existing &amp; Proposed Metrics</th>
<th>PRG1</th>
<th>PRG2</th>
<th>PRG3</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>WPA</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>CAE</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>CoAR</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>CWMC</td>
<td>2.33</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>CWPA</td>
<td>4.53</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>CWCAE</td>
<td>2.8</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>CWCoAR</td>
<td>5.2</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 8.3 Comparison of metrics and their values
8.6 Experimental Setup

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

In this chapter, the proposed tool CCMAT-AOP 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 CCMAT-AOP is used to compute various cognitive complexity metrics of Aspect-Oriented design. The CCMAT-AOP 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 Method per Class (CWMC), Cognitive Weighted Coupling on Advice Execution (CWCAE), Cognitive Weighted Pointcut Designator (CWPA), and Cognitive Weighted Coupling on Attribute Reference (CWCoAR).

Figure 8.4 Menu Page
The figure 8.4 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.

**Eclipse**

Eclipse is an Integrated Development Environment (IDE), which contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java, and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins, including Ada, ABAP, C, C++, COBOL, FORTRAN, Haskell, JavaScript, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java Development Tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The Eclipse Software Development Kit (SDK) includes the Java development tools meant for Java developers. Users can extend its abilities by installing plug-ins
written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their plug-in modules. Eclipse was inspired by the Smalltalk-based VisualAge family of Integrated Development Environment (IDE) products. Eclipse supports development for Tomcat, GlassFish and many other servers and is often capable of installing the required server (for development) directly from the IDE. It supports remote debugging, allowing the user to watch variables and step through the code of an application that is running on the attached server.

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 screen is shown in Appendix D. 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 CCMAT-AOP 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 8.5.

![File Chooser Page](image)

**Figure 8.5  File Chooser Page**

The required file is opened in the text area of the GUI and shown in Figure 8.6.
The metrics consist of CWMC, CWCAE, CWPA and CWCoAR. According to the selection of the user, the corresponding calculated metric value of the given program is shown in the text area.
The above Figure 8.7 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 CWMC, CWCAE, CWPA, and CWCoAR are displayed by using the tool, and the screen shots are given in Appendix D.

8.7 Conclusion

This chapter describes CCMAT-AOP, 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 CWMC, CWPA, CWCAE, and CWCoAR. 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 CCMAT tool 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.