Chapter - 10

Conclusion and Future Work
10.1 Introduction

Aspect-oriented paradigm has become a conspicuous software development technology in these days. The quality of Aspect-Oriented (AO) software is measured using several Aspect Oriented Programming (AOP) 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.

But in the proposed Cognitive Complexity Metrics Suite (CCMS) for AOP it takes the advantage by giving cognitive weighing factors.

The objective of the research work is to propose a Cognitive Complexity Metrics Suite (CCMS) for Aspect Oriented Programming. The sub-objectives are;

- To propose a CCMS based on cognitive complexity
- To develop a tool to collect and analyze the proposed CCMS
- To validate the proposed CCMS theoretically and empirically
- To compare the cognitive complexity metrics with existing metrics.
- To propose a software maintenance effort prediction model using cognitive complexity metrics

10.2 Findings and Contributions

This section elucidates various features of the proposed metrics viz., CWMC, CWCAE, CWPA, CWCoAR. These metrics are collected using a tool called
CCMAT-AOP. The empirical validation of CCMS through Maintenance Effort Prediction Model is also discussed in this section.

10.2.1 CWMC

The cognitive weights for the proposed CWMC metric are derived from the effort needed to understand its types. WMC counts number of methods or advices in a given module proposed by Ceccato et al. [Cec 04] and Kotrappa Sirbi et al. [Kot 10]. Advice complexity can be measured in terms of the different types of advices, namely, Before Advice (BFA), After Advice (AFA), Around Advice (ARA). A set of cognition based experiments were conducted to assess the effort needed to understand the types of advice. The indication behind assigning CWs to different types of advice in the relative cognitive load needed to understand them. The proposed CWMC metric has been validated theoretically using Fenton properties. The proposed metric has been proved to be a better measure of complexity of Class with advices, using an experimental case study.

A comparative study has been done with accepted metric [Par 14], with existing complexity metric, namely, WMC using ten AspectJ programs taken from standard websites. It is found that none of the metrics provides the total complexity of the class. This discriminates the proposed metric from other metrics. Mishra et al. [San, 06] suggested that one can calculate the complexity of the class by using cognitive weights of the methods and attributes. In WMC, the weight of each aspect is computed by summing up the Advice Complexity. The complexity of each advice is calculated using cognitive weights suggested by Mishra et al. [San, 06]. The researcher found that the resulting value of CWMC is higher than WMC. This is due
to the fact that the weight of each advice is assigned a value of 1 in WMC. However, including cognitive weights for calculation of the CWMC is more realistic because it provides a real complexity of the internal architecture of advice. This makes CWMC a better indicator of complexity than WMC. AOP-CCMAT is used to collect WMC and CWMC of various AspectJ programs. From the analysis, it is observed that CWMC value is larger than WMC value when various types of advices are used. Hence, it is concluded that CWMC is a better indicator of complexity of the aspect with advice because of the contribution of WMC.

10.2.2 CWCAE

The complexity of the class/aspect includes the internal complexity of the class/aspect and the response set complexity. The CAE metric is proposed by Ceccato et al. [Cec, 04] and Kottrappa Sirbi et al. [Kot, 10]. The metric CAE Ananthi et al. [Ana, 15] counts the number of aspects containing advices possibly triggered by the execution of methods, advices or method intertype declarations, attribute and attribute intertype declarations in a given class or aspect.

The assigned cognitive weight of the CAE is validated using a test of comprehension of AspectJ programs. It is found that the cognitive load to understand the different types of jointpoints such as object initialization joinpoint, exception handler joinpoint, call joinpoint and advice execution joinpoint. The cognitive load was also changed for various joinpoints. In CWCAE, the cognitive weights are assigned based on the effort needed to understand the type of joinpoint. All these types of joinpoint do not have the same amount of effort needed to comprehend them. The relative cognitive load needed to comprehend the different types of joinpoint was
the philosophy for the assignment of cognitive weights to them. Various cognition based experiments were conducted to assign the cognitive weight for the various types of joinpoint. The CWCAE for a class/aspect is defined as the sum of the product of the total number of joinpoint to other class/aspect and their corresponding cognitive weights. The total CWCAE of a package is the summation of the individual CWCAEs in that package. The proposed metric, CWCAE has been validated theoretically using Fenton properties. Then, this metric has been validated by means of a comparative study with the existing metric CAE using ten AspectJ programs taken from a standard websites. The CAE and CWCAE values are compared and found that CWCAE metric value is larger. From the analysis, it is observed that CWCAE value is higher than CAE value because it includes the cognitive complexity of various joinpoint. This concludes that CWCAE is a better indicator of complexity of the classes/aspects with joinpoints.

10.2.3 CWPA

The proposed metric called CWPA, which adding cognitive weight of the pointcut designator (CWPD) and cognitive weight of the joinpoint signature used in an aspect. Pointcut designator describes when the advices are woven into the joinpoints. The join point signature describes the functions that are related to the respective definitions.

Various cognition based experiments were conducted to assign the cognitive weight for the various types of pointcut designator. The relative cognitive load needed to comprehend the different types of pointcuts and joinpoints was the basic for the assignment of cognitive weights to them. The CWPA for a class is defined as the sum
of the product of the total number of pointcuts and joinpoints to other class/aspect and their corresponding cognitive weights. The proposed metric CWPA has been validated theoretically using Fenton properties. Then, this metric has been validated by a comparative study with the existing metric WPA using ten Java programs taken from a standard website. The result obtained from this study show strong correlation between WPA and the effort required to understand the program. These suggests that the proposed CWPA could be used to predict external software quality factor such as understandability, maintainability, etc. The results from the comparative study and the test of comprehension reveal that this metric CWPA is a better indicator of software complexity.

10.2.4 CWCoAR

The cognitive weights for the proposed CWCoAR metric are derived from the effort needed to understand its types. CoAR counts the number of operations from one component that references an attribute of another component [Kul, 06]. The attribute allusion can range from statically, by legacy, or with dynamism. The scope of this article is AspectJ only and in this the range of attribute allusion is implemented statically (Static Reference Variable – SRV), by inheritance (Inheritance Reference Variable – IRV), or dynamically (dynamic Reference Variable – DRV). A set of cognition based experiments were conducted to assess the effort needed to understand the types of attribute. The indication behind assigning CWs to different types of attribute in the relative cognitive load needed to understand them. The proposed CWCoAR metric has been validated theoretically using Fenton properties. The proposed metric has been proved to be a better measure of complexity of Class with advices, using an experimental case study.
A comparative study has been done with accepted metric [Par 14], with existing complexity metric, namely, CoAR using ten AspectJ programs taken from standard websites. It is found that none of the metrics provides the total complexity of the class. This discriminates the proposed metric from other metrics. Mishra et al. [San, 06] suggested that one can calculate the complexity of the class by using cognitive weights of the methods and attributes. In CoAR, the weight of each aspect is computed by summing up the Attribute Complexity. The complexity of each attribute is calculated using cognitive weights suggested by Mishra et al. [San, 06]. The researcher found that the resulting value of CWCoAR is higher than CoAR. This is due to the fact that the weight of each attribute is assigned a value of 1 in CoAR. However, including cognitive weights for calculation of the CWCoAR is more realistic because it provides a real complexity of the internal architecture of attribute. This makes CWCoAR a better indicator of complexity than CoAR. AOP-CCMAT is used to collect CoAR and CWCoAR of various AspectJ programs. From the analysis, it is observed that CWCoAR value is larger than CoAR value when various types of attributes are used. Hence, it is concluded that CWCoAR is a better indicator of complexity of the aspect with attribute because of the contribution of CoAR.

10.2.5 CCMAT-AOP

There are many metric tools available to automatically compute the traditional AOP metrics. But, there is no tool to compute various cognitive complexity metrics of AOP Design. Hence, the CCMAT has been developed to collect and analyze various cognitive complexity metrics. The CCMAT collects various cognitive complexity metrics for AOP program. The cognitive complexity metrics data that can be collected
using the tool are CWMC, CWCAE, CWPA and CWCoAR. In addition to these metrics, the following metrics data, namely, Weighted Methods per Class (WMC), Coupling on Advice Execution (CAE), Weighted Pointcut per Aspect (WPA) and Coupling on Attribute Reference (CoAR) are also collected. This comprehensive tool can be used to collect the proposed cognitive complexity metrics, AOP metrics and some other complexity metrics for comparative study. The CCMAT can be used to analyse the quality of the AspectJ application.

10.3 Maintenance Effort Prediction Model

The purposes of this study are:

- To empirically validate the proposed cognitive complexity metrics suite
- To measure their ability to predict maintenance effort
- To propose a maintenance effort prediction model derived from the results.

The research design of this study suggests that design complexity has positive impact on the maintenance efforts. This research work is conducted to find the existence of relationship between cognitive complexity metrics and maintenance efforts. Moreover, this research work is used to predict the maintenance effort using cognitive complexity metric suite. The metrics, CWMC, CWCAE, CWPA and CWCoAR are empirically validated with an experiment that was conducted to investigate the effect of design complexity (as measured by the four metrics) on maintenance time. The results of regression analysis confirmed that each of the metrics CWMC, CWCAE, CWPA and CWCoAR was found to have a statistically significant positive relationship with maintenance time. Each of the four cognitive complexity metrics by itself was found to be useful in measuring the design complexity. From the multiple
regression model, it is concluded that combined metric suite of CWMC, CWCAE, CWPA and CWCoAR can be used to predict maintenance effort better than the individual metrics. From the experiments, it is found that CWCAE, CWPA, and CWCoAR explained more of variance than CWCoAR with maintenance time. It is found that the result of Regression analysis between cognitive complexity metrics and maintenance time show variance of 22% for CWMC (Predicted Maintenance Time (PMT = 77.1975 + 2.964761412 * CWMC), 30% for CWCAE (PMT = 77.1213 + 2.876116274 * CWCAE), 32% for CWPA (PMT = 66.3855 + 2.245081749 * CWPA) and 27% for CWCoAR (PMT = 62.6567 + 2.753505918 * CWCoAR).

Multiple regression analysis with all three metrics together was then performed to determine the combined explanatory power of these metrics. The regression analysis confirms with a variance 35%, which is greater than the percentage of variance of any other single metric. Hence, it is concluded that it would be more appropriate to use combined cognitive metrics suite to predict maintenance effort

**10.4 Summary of the Findings and Interpretations**

A good metric is one that considers not only the number of methods, classes, subclasses and relations between them but also the internal structure of the methods [San, 12]. It is clear from the examples that each of the proposed complexity metrics of AOP-CCMS fulfils the requirements of a good metric. It also considers the internal architecture of the member function (method) by considering the cognitive complexity of Advice, Joinpoint, Pointcut and Attribute Reference.
The major findings of this empirical validation are presented below:

- Each of the four cognitive complexity metrics by itself is found to be useful in measuring the design complexity.
- The combined metrics suite is found to be more accurate in measuring the design complexity.
- The results are compared with existing AOP metrics and found that the proposed CCMS-AOP proved to be more accurate protection than the widely used CK metrics suite.

10.5 Limitations

The psychological experiments conducted to calibrate various sets of cognitive weights employed the students undergoing Bachelor and Master Degree in Computer Science course. This is the main limitation of the research work. Though, as mentioned earlier, many have done their empirical study in this way, the use of students as subjects in the experiment is a question of concern. However, it is believed that, even with these issues, the study can provide useful information to software engineering practitioners and researchers. The second concern is the use of particular Aspect Oriented Programming language, namely AspectJ language, for all the empirical data collection. Though, Abreu et al. [ Abr, 01 ], after experimenting with a specific programing language, have extended to other languages with their bindings. As far as the proposed metrics are concerned, it is still to be verified empirically. Other concern is the complex nature of software maintenance task. Many issues may be likely to prop up, besides the complexity of the design that is measured by the
proposed metrics. These concerns and other unknown and unforeseeable limitations only indicate that the research work is successful within its scope.

10.6 Further Research

The experimental study can be extended in the following manner:

• The metrics suite can be validated for other quality indicators, namely, fault process, reliability, reusability, etc.

• The metrics suite can be implemented in Cloud Environment.

• Cognitive complexity metrics can be proposed for measuring the complexity of Component-Based System and other Aspect-Oriented Systems like CeasarJ, HyperJ, Ada, etc.

• An effort prediction model can also be implemented in any learning methodology in the social aspect.