International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

Exploring the Quality Improvement in Small-Scale Project Using Aspect-Oriented Design

Sufia Nadeem Chishti, S. K. Singh,

Abstract*: Aspect-oriented software development (AOSD) seeks to renovate the software quality by use of alteration to the design in object-oriented based systems. Each system component has been divided into numbers of modules and arranges a design in a way that the module has not crosscut to each other. A module is a collection of block of codes, whose role is to restrict and conceal the design definition. This type of behavior is known as the Separation of Concerns (SoC). AOP helps programmers for separating crosscutting concerns. It can be helpful at better modularizing system, the problem with complex system is that the component require different, crosscutting descriptions at different levels and need maintenance too. In this paper, a study was conducted to analyses quality in aspect-oriented software design. This paper also explores varieties of existing metrics related to AOP also and will discuss in detail with comparison of OOPs using small scale projects that how the new programming paradigm design effects on quality of the software.

***Index Terms*: AOP, AOSD, Aspect, AOP Design, Design Pattern**

I. INTRODUCTION

Software design methodologies are noteworthy for the development of existing software; it is centralized to achieve high quality in several programming buildings, including design stability, reuse, evolution capabilities, configuration options, support capacity, performance and expansion capabilities. The current daily program of software engineering tools uses software design with the amount of tons needed. They can often control a predefined set of projects that cannot be customized and expanded, or even allow developers to code a new project in low-level locations, even if personalization or expansion is allowed. The evolution of software engineering often leads to the degradation of software design. Accordingly, In Object-Oriented (OO) systems, this often leads results in the packages that are hard to recognize and difficult to maintain, as they group together in heterogeneous classes with discrete responsibilities. AOP is a concept and as such, it is not bound to a certain programming language. It can help to the shortcomings of all languages that use single, hierarchical decomposition. AOP is still a long way from being an entrenched standard since its history just scopes very nearly 20 years. There are a couple of issues that need to be look forward as a new challenge in research, issues to be handled.

In a decisive manner the key distinction in the middle of OOP and AOP is that the center of OOP is to separate the programming undertaking into items, which embody information and techniques, while the center of AOP is to separate the project into crosscutting concerns. AOP is an effective technique for modularizing crosscutting concerns, but effective design principles are applied to create appearance systems that support long-term maintenance and evolution. Several designs patterns have been described in the literature, and because they are widely used in industry, software designers have proven to be powerful tools to improve software quality. The biggest challenge in designing an AOP is to provide a clear and clear definition of where and when instructions will be implemented [11]. The use of programming indicators can assess the various qualitative characteristics of targeted programming, such as design stability, performance, reuse, ability to improve, configuration options, durability and expansion, size. For example, dimension criteria can support the definition of modularization problems: large modules can be divided into smaller modules with fewer requirements or their accents can be transformed into different modules. An AOP based metrics is another approach to creating high-quality applications. Complexity, coupling, cohesion and defect, etc. Such AO-based application factors are mainly used to control and study the nature of programming. When designing, many product engineers lack cost, reliability etc. Analyzing such factors as imagining whether the needs are too modular, exchange ideas about the product, regardless of whether the line is high quality or not. Table 1 shows an example of a software metric for the project design. Each software measurement is an attempt to measure or estimate the internal or external properties of a product, process or resource. The table contains an array of software criteria for the space design and illuminates the recognized features.

Table 1: software metrics for design

Entity	Attributes	
	Internal	External
Design	Size	Quality
	Reuse	Complexity
	Modularity	Maintainability

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
Sufia Nadeem Chishti*, School of Computing Science and Engineering, Galgotias University, Greater Noida (U. P.) India.
S. K. Singh, Department of Computer Science & Engineering, GCET, Greater Noida (U. P.) India.

	Coupling	
	Cohesion	
	Inheritance	
	Functionality	

The reminder of this paper is organized as follows: section 2 discusses the related works. Section 3 explains about the small-scale project (i.e. JHotDraw) and section 4 describes the discussion about the calculated values of metrics, which explore the quality, when the design is altered. Finally, section 5 concludes the paper.

II. RELATED WORKS

Measurement is a reinforcement to help regulate product procedure, project control and control. Good software engineer, quality analysis and design model, source code, experiments, etc. Use the means to assess. A significant number of object-oriented and aspect-oriented metrics have been developed in the literature. For example, the metrics proposed by Abreu [1], metrics of C.K [2], the metrics Li and Henry [3], the metrics of MOOD [4], the records of Lorenz and Kidd [5], Eddy et al. [6], Burrows et al. [7], Kumar and Singh [8], Ceccato and Tonella [9] etc. C. K. metrics, the most popular among them (used). The next complete set of measurements are MOOD measurements. A pack of AOP-based metrics proposed by Ceccato and Tonella, tested by the well-known Metric Chidamber and Kemerer team [2] for the object-oriented paradigm [9]. Some metrics have been extended to make Chidamber, the Kemerer package in its current form, and others have applied to the more aspect-oriented software. In the Ceccato and Tonella packages, the module is used as a common term for classes and angles. In the same way, methods, different types of recommendations and various types of presentations, etc. This is shown in the process conditions.

Definitions of Various metrics are as follows:

A. *Crosscutting Degree of an Aspect (CDA)*

It is a measure of a module in a specific aspect module that is influenced by different types of point segments and different types of input data. CDA measures all modules that the dimension can affect. This gives an idea of the effect of functions on other modules. In addition, the difference between CDA and CIM gives the number of modules affected in terms of class / address independence, without explicit indication in a direction that may indicate the degree of generality of the address. In general, high CDA values and low CIM values are desirable [6].

B. *Concern Diffusion over Components (CDC)*

Its main goal is to measure classes and aspects and other classes and directions that reach them, contributing to the problem [6].

C. *Concern Diffusion over Operations (CDO)*

Its main purpose is to contribute to the implementation of the problem, measures of processes or methods and other processes or methods that achieve them [6].

D. *Lack of Cohesion in Operations (LCO)*

It is a measure of the degree of class or method alignment and the size of a recommendation that does not achieve the same class fields or variables [8]. In negative results, LCO becomes zero. When all methods fail to reach any field or variable, LCO = 0. In object-oriented systems, the lack of compliance of methods (LCOM) is similar to [2]. If all transactions in the class or address have a common data structure that is manipulated or accessed, the LCO will be low.

E. *Depth of Inheritance Tree (DIT)*

It is a measure of the longest sequence from a single module to the same root sequence or class as the object system. Properties are measured using DIT. A deeper class or view in the classroom increases the number of operations that can be achieved, making knowledge and change more complex. Because directions can change the inheritance relationship through a stable cross-section, the effects of this size should be taken into account when calculating these criteria [2].

F. *Line of Concern Code (LOCC)*

LOCC is also a dimensional matrix that measures the degree of precision of the line in terms of the program used to weave in the AJDT compiler.

G. *Number of Children (NOC)*

It is a measure of the degree of subclasses or sub-dimensions of each module. Like DIT, it also measures the range of the characteristics, but in the opposite direction than DIT. NOC of the module means the proportion of modules that depend on properties inherited from the given [8].

H. Coupling between Modules (CBM)

This is a measure of the degree of interdependence between a module or interface that describes the methods of areas available for a specific module shown in Figure 1. This can be achieved by controlling the amount of parameters that pass between the modules, or by preventing unwanted data from being created during the connection. It is also tantamount to assigning between objects in the metrics CK Metrics Suite (CBO) and in Field Access Match Assignment (CFA) and Method Call Match Metrics (CMC) [8].

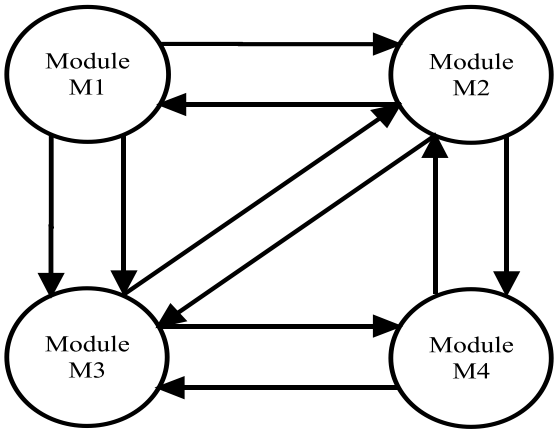

Figure 1: Coupling between modules

I. Coupling of Method Call (CMC)

This is a measure of the degree of module or interface reported by another module. This matrix is based on matching objects between CBM equivalent objects in the oriented direction (CBO).

When determining possible methods, the perception of dimension should be taken into account. Working with different methods from different modules indicates that the methods of the given modules cannot be easily separated from others. Higher feedback means greater dependence on functions in other modules [8].

III. SMALL-SCALE PROJECT – JHOTDRAW

JHotDraw is an application based on Java. Both cases (i.e., Java and Aspect) are considered case studies. A graphical editor based on an open source graphical interface for special and useful images enables working in various file formats. There are many classes and tips inside. There are many features, methods and creativity in the classroom or size. The original paths of many threads. Developed by Erich Gamma and Thomas Agneschler. The main purpose of the JHotDraw application is [10]:

- Creating new applications based on JHotDraw.
- To increase and update existing code.
- Identification and reorganization of new design patterns.
- To check the availability of the new Java API in JHotDraw

There are various applications on the market in both forms (i.e. OOP and AOP). JHotDraw is the most famous of them. In this way, JHotDraw is used for quality control.

IV. DISCUSSION

This section gives the calculated value of the metric, as in the table presented in Table 2. The devices calculate the values of some matrices, and some are manual. Tools are developed from Java. Table 2 analyzes the quality of the application based on the value of the matrix and shows how the quality is improved to change the design of the application. The transfer of AOP to OOP technology depends largely on the ability to systematically understand positive and negative effects through design changes. Each application requires less coordination and more coordination; this is a key element of effective and sustainable implementation. In addition, we want to maintain the separation of SoC, eliminating these applications. The design between different modules, forms, subsystems or related / indirect systems translates into our goal of implementing SoC with less linkage and high coordination at this point. AOP is used to protect SoC. Multiple use and maintenance depends on the design model. If it is reusable and maintenance is improved, the quality of the application increases. The calculations (in Table 2) have shown that this applies to the majority of AOPs and gives more lines of code in AOP performance. In light of the calculated value, the AOP strongly affects the OOP configuration project and improves the quality of the application (shown in Figure 2).

V. CONCLUSION

This research paper examines how quality improves if the design of the application has been changed. The measuring set is used to control the quality shown in Figure 2, defined by the design presented in Table 2. Our research was used to compare the quality of applications with metric values in OOP and AOP. In AOP, the program code becomes smaller and smaller. Essentially, AOP takes into account the better structure of the project, and therefore improves modularity and reuse. The assessment shows that AOP is used as an OOP extension, with significant priorities (increased priorities, modularity, reuse, maintenance and subsequent). If it is manufactured using AOP, it provides the most adaptable design for the application without easy maintenance, reuse and optimization.

REFERENCES

1. Abreu, Fernando Britoe and Rogério Carapuça. “Candidate metrics for object-oriented software within a taxonomy framework.” Journal of Systems and Software 26 (1994): 87-96.
2. S. R. Chidamber and C. F. Kemerer, “A Metrics Suite for Object Oriented Design,” IEEE Transactions on Software Engineering, Volume: 20 No.: 6, Pages: 476-493, June 1994.
3. Li,Wei , Henry, Salley.: "Maintenance Metrics for the Object Oriented Paradigm", First International Software Metrics Symposium. Baltimore,Maryland, May 21-22, 1993. Los Alamitos, California: IEEE Computer Society Press, 1993.
4. Abreu, Fernando B: "The MOOD Metrics Set," Proc. ECOOP'95, Workshop on Metrics, 1995.
5. Lorenz, Mark & Kidd Jeff: “Object-Oriented Software Metrics”, Prentice Hall, 1994.
6. M. Eaddy, T. Zimmerman, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho, “Do Crosscutting Concerns Cause Defects?,” IEEE Transactions on Software Engineering, Volume: 34, Issue: 4, Pages: 497-515, 2008.
7. R. Burrows, et al., “An Empirical Evaluation of Coupling metrics on Aspect-Oriented Programs,” ACM, 2010.
8. P. Kumar and S. K. Singh, “A Systematic Assessment of Aspect-Oriented Software Development (AOSD) using JHotDraw Application,” in 2nd IEEE International Conference on Computing, Communication and Automation (ICCCA-2016), Greater Noida (Uttar Pradesh), INDIA, Page No.: 1 - 6, April 2016.
9. M. Ceccato and P. Tonella, “Measuring the Effects of Software Aspectization” 1st workshop on Aspect Reverse Engineering (WARE), 2004.
10. JHotDraw Site: http://www.jhotdraw.org/ [last visited: 19/02/2019].
11. P. Kumar, and S. K. Singh, “Architectural Development of E-Learning Application using Aspect-Oriented Programming (AOP) Principles” International Journal of Computer Applications (IJCA), Foundation of computer science, New York, USA, Volume: 180, Issue: 2, Page No.: 21-25, December 2017.
12. S.K.Singh, P.Kumar and S.N.Chishti. “A Comprehensive Investigation of Quality of AOP-Based Small Scale Projects Using Aspect-Oriented Software Quality (AOSQ)Model”, IEEE International Conference on Advances in Computing Communication Control and Networking.

AUTHORS PROFILE

Sufia Nadeem Chishti is currently working as research scholar, at School of Computing Science and Engineering, Galgotias University, Greater Noida (U. P.) India.

Dr. S. K. Singh is currently working as Professor, at Department of Computer Science & Engineering, Galgotias college of Engineering & Technology, Greater Noida (U. P.) India.

Table 2: Description of JHotDraw with different metrics values

Metrics	Pure Java Code			Java with AspectJ Code			Java Code (%)	AspectJ Code (%)
	OOP	AOP	Total	OOP	AOP	Total		
CDA	NA	NA	0	10	91	101	0.00	221.78
CDC	NA	NA	0	0	28	28	0.00	196.43
CDO	NA	NA	0	0	136	136	0.00	130.88
LCO	29864	NA	29864	28122	444	28566	95.65	104.54
DIT	148	NA	148	149	0	149	100.68	99.33
LOCC	72890	NA	72890	54130	19398	73528	100.88	99.13
NOC	78	NA	78	77	2	79	101.28	98.73
CBM	1420	NA	1420	1355	131	1486	104.65	95.56
CMC	1350	NA	1350	1311	119	1430	105.93	94.41
CFA	57	NA	57	57	6	63	110.53	90.48

*NA-Not Applicable

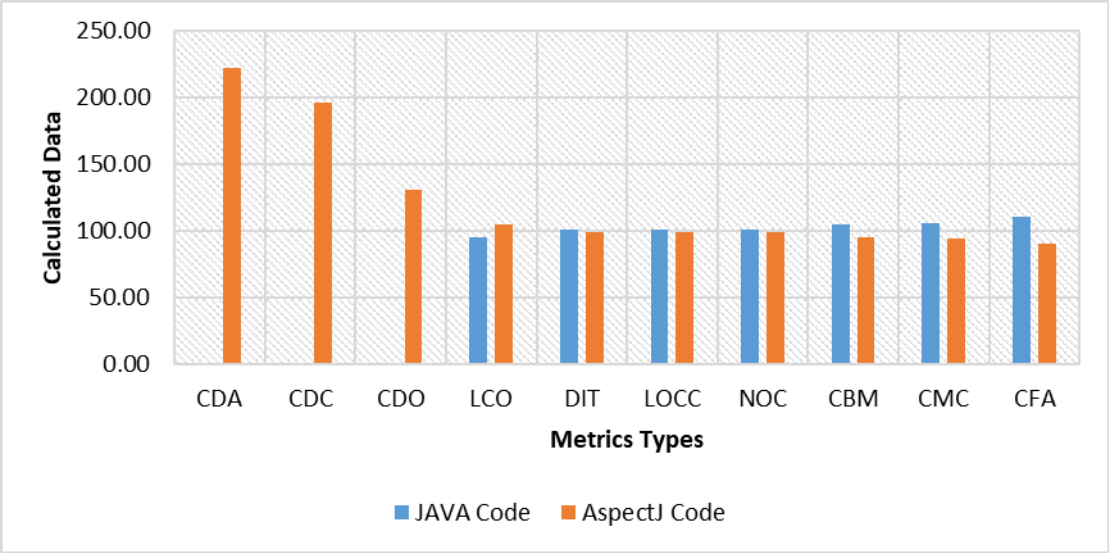

Figure 2: Java Code vs. AOP Code