International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

Analysing The Quality Attributes of AOP using CYVIS Tool

hinam Garg
Mtech Student
CSE
RIMT-IET
Mandi Gobindgarh, India
E-mail: shinamgarg@yahoo.co.in

Mohit Garg
Assistant Professor
CSE
RIMT-IET
Mandi Gobindgarh, India
E-mail: mohit_nabha@yahoo.com

Abstract— The aim of thesis is to provide the difference between AOP and OOP in AspectJ by using CYVIS tool. It gives a way to separate the code from essential crosscutting concerns, such as logging and security, from Java programs core application logic cleanly by making code more readable, less error-prone, and easier to maintain. It also solves the problem of code scattering and code tangling by providing the aspect to crosscutting concerns (concerns which are repeated again and again). In this paper, aspect oriented programming is shown better than OOP by considering cyclomatic complexity and overcoming all those problems which were occurring in OOP. Two examples such as banking system and conversion of int and float values to hexadecimal are illustrated and shows the difference of quality attributes in AOP than OOP.

Keywords- *Aspect Oriented Programming(AOP), Object Oriented Programming(OOP), Join points, Point cut descriptors,Aspect J.*

INTRODUCTION

Aspect Oriented Programming is a new programming paradigm which increases the modularization of the program by defining cross cutting concerns in a separate module. It provides way to separate the code for essential crosscutting concerns, such as logging and security, from your Java programs core application logic cleanly. It can make user code more readable, less error-prone, and easier to maintain. When software starts to develop then its problem can be divided in two types:-

- common concern
- Cross cutting concern.

(a) Common concern: It refers to main goals for which program are going to develop.

(b) Crosscutting Concern: It is a problem that is scattered over the multiple modules in a program and makes the code tangled, difficult to change and redundant. These are the problems that are not important from user point of view but user has to deal with them during development process and they cut across multiple modules of the program.

The rest of the paper is organized as follows. Section II introduces the concept of crosscutting concerns. Section III gives the overview of different heterogeneous composition of AOP languages. In section IV, types of metrics in AOP are presented. Section V presents the proposed work. Experimental results are described in section VI. Finally the conclusion is provided in section VII.

CROSSCUTTING CONCERNS

Example of Banking System to explain CCN's:- In banking system our main goal is money transaction, balance enquiry etc. But there are other concerns we have to keep in mind like security, authentication, performance etc. Suppose to make a balance enquiry we have an enquiry module, but user has to authenticate himself before making an enquiry. In same way before withdrawing or transferring money user has to authenticate. In the program there are separate modules for every task but authentication is a mandatory process in all these modules. Authentication code is repeated again and again in our modules. So authentication is a cross cutting concern. To remove the problem of crosscutting concern AOP introduces a new modular unit called aspect which encapsulates the functionality of crosscutting behaviour.

DIFFERENT HETEROGENEOUS COMPOSITION OF AOP LANGUAGES

Three different heterogeneous compositions of AOP languages are AspectJ, CaesarJ and Hyper-J.

(a) **Aspect J**: - AspectJ is a general purpose aspect oriented extension for java. It is an open –source project initiated by PARC and now led by IBM. It can be downloaded freely. It was built to provide support of AOP in java programs. It is an extension of java with several complementary mechanisms, namely Join Points, Pointcut Descriptors (PCDs), advices, introductions and aspects.

- Join Points: JPs represent well-defined points in a program's execution. Typical join points in AspectJ include method calls, access to class members and the execution of exception handler blocks.
- PCD is a language construct that picks out a set of join points based on defined criteria.
- Advice is code that executes before, after or around a join point.
- Introduction allows aspects to modify the static structure of a program
- Aspects: advice, pointcuts, ordinary data members and methods are grouped into class-like modules called aspects.

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

AOP languages and frameworks provide a very similar composition model to the aspect such as Springs AOP framework and JBoss AOP.

(b) **Caesar J**: - It represents a different family of AOP languages, which doesn't have aspects. It supports additional concepts such as virtual classes, mixin composition, aspectual polymorphism and bindings.

(c) Hyper J: - A developer provides three different inputs which are hyperspace file, concern mapping file and hypermodule file

- Hyperspace File: describes the Java class files
- Concern Mapping File: describes the pieces of java source map to concerns
- Hyper module File: describes which dimensions of concern should be integrated.

TYPES OF METRICS IN AOP

1) Weighted Operations in Module (WOM)

Weighted operations in module counts number of operations in a given module. Operations include advice, introductions, methods etc. The number of operations and the complexity of operations involved is a predictor of how much time and effort required to develop and maintain the module. Modules with large number of operations limit the possibility of reuse.

2) Depth of inheritance(DIT)

DIT of a class is its depth in the inheritance tree, if multiple inheritances are involved. Maximum path from the node representing the class to the root. The deeper a module is in the hierarchy, the greater the number of operations it is likely to inherit, making it more complex to predict its behavior. So complexity of design will increase if more operations and modules are involved.

3) Number of children(NOC)

It is a number of immediate subclasses or sub-aspects of a given module. Greater the number of children then greater the reuse due to inheritance, improper abstractions of parent module, misuse of sub classing and also it requires more testing of operations in the module.

4) Cyclomatic Complexity

Complexity is a software metric used to indicate the complexity of a Program. It gives the number of paths that may be taken when a program is executed. Methods with a high Cyclomatic complexity tend to be more difficult to understand and maintain. Some of the tokens (in java) responsible for the program taking different paths during execution are:

- While & do while statements.
- If statements.
- For statements.
- Ternary Operators & Logical Operators.
- Switch case statements.
- Return, throw, throws, catch statement.

Example of Cyclomatic Complexity Calculation:

```
public static int getGreatest(int a, int b)

{
   return a>b?a:b;
   return a;

}
```

The cyclomatic complexity for this method would be 3. Since the method has a ternary operator, it acts as one if and else, which in turn adds 2 to the cyclomatic complexity. We then add one to the cyclomatic complexity (raising it to 3) to denote the default path (in this case both if and else failing and the second return statement being executed).

The cyclomatic complexity should be low. A low cyclomatic complexity is one factor to improve readability, maintainability and testability of code. The cyclomatic complexity is the number of edges minus the number of nodes plus 2.

CC = No. Of edges - No. Of nodes + 2

5) *Instruction Count*

It represents the no of instructions in a particular code.

PROPOSED WORK

In this section calculation of cyclomatic complexity and instruction count in AOP and OOP by using Banking system and by using the program of converting the integer and float values to hexadecimal is mentioned in detail.

- In the example of banking system using OOP:- When program starts then there are 4 options that system asks. Balance Enquiry (), Money Withdrawl(), Submit Money()/Deposit Money(), Get Account No() and then exit. There is a class i.e.Banking usingOOP in which it is having fields name, password, accountno,, amount etc.create user account by giving name, amount, account no and password. In 1st Module i.e. Balance Enquiry module.enter the name and then password. if name and password matches then "balance is"i.e. system will tell the balance else authentication will be failed. In 2nd module i.e. Money WithDrawl() .system will ask the

name and password if it matches then it will ask the amount of money user wants to withdraw. if amountWithdrawl>=amount present then system will msg that you can't withdraw that amount of money and if amountWithdrawl<amount present then money will be withdraw and transaction completes successfully. Then the total remaining balance will be displayed. In 3rd module i.e. Submit money/Deposit money. System will ask the name and password. Then system will ask the amount of money user want to submit then that amount will be add to previous amount.

In 4th module i.e. Get Accountno .System will ask the name and password of user.if user and password matches then system will gives the account no . 5th module is exit button .

- Banking Example using AOP:- When user start program in banking system then 4 options that system will ask to user. Balance Enquiry (), Money Withdrawal (), Submit Money()/Deposit Money(), Get Account No() and then exit. There is a class i.e. Banking usingOOP in which it is having fields name, password, account no,, amount etc. Create user account by giving name, amount, account no and password. In 1st Module i.e. Balance Enquiry module. Enter the name and then password. If name and password matches then "balance is" else authentication will be failed. In 2nd module i.e. Money Withdrawal () .system will ask the name and password if it matches then it will ask the amount of money user wants to withdraw. if amountWithdrawl>=amount present then system will msg that you can't withdraw that amount of money and if amountWithdrawl<amount present then money will be withdraw and transaction completes successfully. Then the current balance will be shown

  In 3rd module i.e. Submit money/Deposit money. System will ask the name and password. Then system will ask the amount of money user want to submit then that amount will be added to previous amount.

  In 4th module i.e. Get Accountno.system will ask the name and password of user. if user and password matches then system will gives you the account no .

  Whereas 5th module is exit button.

  For each module in AOP, AOP Authenticates the username and Password for which user make the aspect Authenticate which is like a constructor.everytime when the we enter the username and password then control will 1st move to the aspect of authentication and after the point cut control will go back to that particular module. In AOP we just write once that code which is repeated again and again in aspect.

- Conversion of integer and float values to hexadecimal in OOP: - In this program two classes have taken namely: - Handler (i.e. main class) & Converter. Converter class has two methods that will convert int and float numbers in hexa-decimal strings respectively.

  In handler class, a Converter type instance field and there are two methods that are calling methods of Converter class. Before making a call to Converter method user will check if Converter fields hold an object reference and if converter is null an exception is thrown. This procedure is repeated in both the methods. This condition checks is a cross cutting concern and it can be defined in a separate module to make our program better.

  - Conversion in AOP program:- AOP program Handle.java is an extension of java program in AspectJ. The only difference is that a new module aspect Check is added to the program. In this program , any kind of condition within the methods of Handle class before calling the methods of Converter class willn't be checked. Instead of that user will defined a condition in aspect Check when Handle class methods execution will start control will be transferred to the aspect Check, it will check the condition if converter field is null exception will be thrown else method of converter will be called.

In the actual implementation, a program is made and then writes that program in AOP language as well as OOP language using AspectJ. Then jar file of AOP and OOP will be created and then executed on Cyvis tool. For this run the cyvis tool. Select the new project and then select the project directory where jar files are stored. then add that jar files .click on the save button and then finish. The project view will be opened. It shows all the packages that are in project. It provides all classes and method in GUI way. Red color represents class with high complexity, green color represents the class with low complexity and yellow represents the moderate complexity. Grey color represents the interfaces and white represents there are no interface between components. This will result in the complexity values difference which is shown in this tool. Html and text reports can also be generated in cyvis tool.

RESULTS AND DISCUSSIONS

CyVis Tool collects data from java class or jar files. Once the raw data is collected, certain metrics like number of lines, statements, methods, classes and packages are obtained. Other metrics like cyclomatic complexity, instruction count are also be deducted. Cyclomatic Complexity is a software metric used to indicate the complexity of a program. The cyclomatic complexity should be low. Table I shows the cyclomatic complexity and instruction count (which represents the no of lines in a byte –code instruction) in Cyvis tool using AspectJ. . A low cyclomatic complexity is one factor to improve readability, maintainability and testability of code. This

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

table represents the Cyclomatic complexity and instruction count for each particular method and also for default methods which were created by aspect. Table II shows the cyclomatic complexity and instruction count for each particular method in OOP.This table shows the Program of banking system in OOP having high complexity than OOP just because the problem of scattering and tangling. Table III shows the cyclomatic complexity and instruction count for converting the integer values and float values to hexadecimal values in AOP. The value of cyclomatic complexity is low for this table which is good.1, 2, 7, 8 represents the all those methods which were created by java by itself and <init>method represents the constructor and float to hex and int to hex are user defined methods. Table IV shows the 5 methods i.e. float to hex, int to hex and main functions are user defined methods.

ATTRIBUTE VALUES FOR AOP IN BANKING SYSTEM

Table I

S.NO.	Method Name	Cyclomatic Complexity	Instruction count
1.	Main	9	110
2.	MoneyWithdrawl	3	72
3.	getAccountnumber_aroundBody7$advice	3	56
4.	sendMoney_aroundBody5$advice	3	56
5.	moneyWithdrawl_aroundBody3$advice	3	56
6.	balanceEnquiry_aroundBody1$advice	3	56
7.	<init>	1	68
8.	AccountConfirmation	1	65
9.	SendMoney	1	30
10.	BalanceEnquiry	1	18
11.	GetAccountNumber	1	18
12.	CreateAccount	1	13
13.	getAccountNumber_aroundBody6	1	4
14.	sendMoney_aroundBody4	1	4
15.	moneyWithdrawal_aroundBody2	1	4
16.	balanceEnquiry_aroundBody0	1	4

ATTRIBUTE VALUES FOR OOP IN BANKING SYSTEM

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

Table II

S.NO.	Method Name	Cyclomatic Complexity	Instruction count
1.	Main	9	90
2.	MoneyWithdrawl	6	117
3.	SendMoney	3	66
4.	BalanceEnquiry	3	51
5.	GetAccountNumber	3	51
6.	accountConfirmation	1	69
7.	<init>	1	68
8.	CreateAccount	1	13

ATTRIBUTES VALUE FOR CONVERTING INT AND FLOAT VALUES TO HEXADECIMAL IN AOP

Table III

S.NO.	Method Name	Cyclomatic Complexity	Instruction count
1.	floatToHex_aroundBody3$advice	3	24
2.	intToHex_aroundBody1$advice	3	24
3.	Main	1	35
4.	FloatToHex	1	11
5.	IntToHex	1	11
6.	<init>	1	10
7.	floatToHex_aroundBody2	1	8
8.	intToHex_aroundBody0	1	8
9.	<init>	1	6

ATTRIBUTES VALUE FOR CONVERTING INT AND FLOAT VALUES TO HEXADECIMAL IN OOP

Table IV

S.NO.	Method Name	Cyclomatic Complexity	Instruction count
1.	FloatToHex	3	20
2.	IntToHex	3	20
3.	Main	1	34
4.	<init>	1	9
5.	<init>	1	5

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

Chart result for instruction count

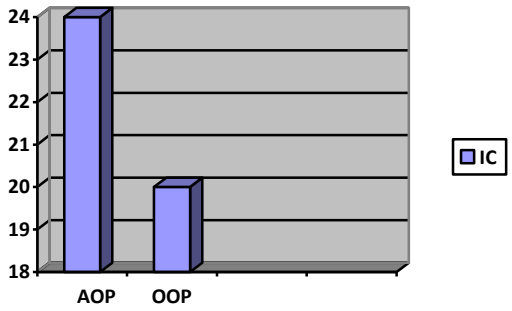
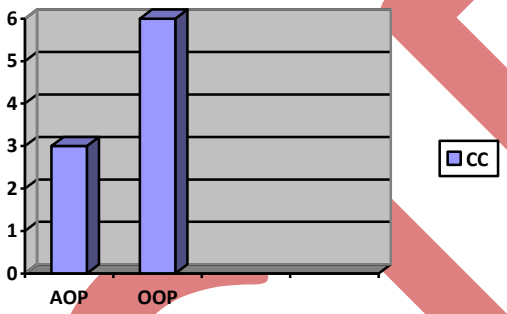


Chart result for cyclomatic complexity



CONCLUSION

Understandability: Complexity in the base code was decreased by the AO approach slightly. So understandability increases.

Maintainability Almost 20% of the base code was modularized into the aspects removing the scattered code and making modifications to the application easier.

Manageability Instability is an indicator of manageability.

AOP is 3 times more stable than the OO

In this thesis work, a framework for better cyclomatic complexity and instruction code in AspectJ by using Cyvis tool has been proposed. This considered two different programming languages i.e. Aspect oriented and Object Oriented Programming in AspectJ on the same program by using Cyvis tool. Cyvis tool shows how AOP provides better cyclomatic complexity than OOP. In Cyvis tool it is also generating Cyvis report at backend.

In future, the all other metrics for AOP and OOP can be validated. With the help of other metrics , which programming language is providing better results can be proven.

REFERENCES

[1] Avadesh Kumar,Rajesh Kumar and P.S.Grover May 2009. "Generalising Coupling Measures for Aspect Oriented Systems", International Conference on Advances in Recent Technologies in Communication and Computing, IEEE.

[2] Rupali Ahuja and Anita Goel 2008. "A study of Applications of Aspect Oriented Programming", National conference on Challenges & Opportunities in Information Technology.

[3]G.kiczales, J.Lamping, A.Mendhekar, C.maeda, C.Lopes, J.M.LoingTier and J.Irwin 1997. "Aspect Oriented Programming" 11th European Conference on Object Oriented Programming.

[4] Jay Gattani 2004, "An Analysis of Aspect Oriented Programming with AspectJ" Technical report, University of Houstan.

[5] Nicholas Leisiecki 2005 "AOP @work:Enhance design patterns with AspectJ".

[6] .www.cs.umu.se

[7] Kotrappa sirbi and Prakash Jayanth Kulkarni "Design Patterns v/s AOP – A quantitative and Qualitative Assessment"

[8] L.Berger , A.M.Dery and M.Fornarino "Interactions between objects:An aspect of Object – oriented Languages"

[9] Sven Apel and Don Batory 2008 "How AspectJ is used- An analysis of eleven AspectJ programs"

[10] www.enwikipedia.org

[11] http://cyvis.sourceforge.net/inde