IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-5 E-ISSN: 2347-2693

Analysis of Aspect Oriented Systems: Refactorings using AspectJ

Geeta Bagade (Mete) [1*], Shashank Joshi[2]

[1*]Department of Computer Science,Yashwantrao Mohite College, Bharati Vidyapeeth University, Pune,India
[2] Department of Computer Engineering, Engineering College, Bharati Vidyapeeth University, Pune, India

Available online at: www.ijcseonline.org

Received: Apr/26/2016 Revised: May/07/2016 Accepted: May/19/2016 Published: May/31/2016

***Abstract*—Refactoring is one of the most important activity in software development. It is done to improve the design of the software, to make the software easier and better to understand and to help us in writing programs faster. After the software is refactored, it is important to note the behaviour of that software. In this paper, we propose refactorings that we can apply of Aspect Oriented Programs. In the last paper some of the refactorings were introduced. Here we are introducing the results of the refactorings introduced and the systems considered for Aspect Oriented Programming using Aspect. This research paper is in continuation with the previous one. Initially we introduce the refactorings identified, then the Systems that are used for applying these refactoring are mentioned. Then the tool is discussed and finally the analysis of the system is presented.**

Keywords- Refactoring, Aspect Oriented Programming, AOP, Pointcut, Joinpoint, Refactoring Advice Aspect Oriented Programming, Aspect Oriented Concerns, AspectJ, Concerns, Aspect, Aspect Mining

I. INTRODUCTION

In the previous paper, we had listed the refactorings that we have identified. The refactorings identified were

1. Make the aspect unprivileged
2. Replace the pointcut name with its designator
3. Introduce the get and set pointcut , introduce before and after advice
4. Remove the word abstract for the aspect

The above refactorings were applied on a sample code and their results were analyzed. Here we are applying the above refactorings on 10 systems that make use of AspectJ. Here we first discuss about the systems that we have used. Later we discuss about the tool that we have used for analyzing the systems. And in the last section we present the analysis and the conclusion.

II SYSTEMS USED

In this section we discuss the systems that have been selected. We have selected 10 systems that use AspectJ. These systems have been used because they have been used extensively by other researchers for the various purposes in the domain of AOP. Here we discuss in brief about each system that has been considered

A. **Banking Application (BA)[1]:** This is a simple system that provides the basic functionality of a banking system like depositing money and withdrawing money. An aspect is used to check whether the requested operations like depositing money and withdrawing money should be allowed or not.

B. **TracingAspect(TA)[2]:** This system is provided by Xerox Corporation. It contains classes like

1. Circle
2. Square
3. TwoDShape (Abstract class)
4. Trace

It contains one Aspect named TraceMyClasses which is connecting the functions in the Trace class with the constructors and methods in the Application class.It also contains one class ExampleMain which contains the entry point for the program execution i.e the main() method.The code has been slightly modified by changing the access specifier which was originally protected in each class (Circle, Square and TwoDShape) to private to test the effect of the refactoring mentioned above

C. **Figures_Shapes(FS)[3]:** This project contains an interface FigureElement which has two methods moveBy and draw. The method moveBy moves a line or a point by x and y co-ordinates. It has a class Point which implements the interface FigureElement i.e. it provides the code for the methods moveBy() and draw().It also has a class Line which draws a line from Point p1 to Point p2. It also moves a line using the method moveBy(). There is the class Display. Aspects present in this project are BoundPoint which sets the minimum and maximum x and y co-ordinates.The aspect BoundPointEnforcement extends the aspect BoundPoint and enforces the x and y coordinate bounds. The aspect BoundPointPreCondition which has the advice "before" that checks the value of x co-ordinate and y co-ordinate.The aspect BoundPointPostCondition which has the advice "after" that again checks the value of x

co-ordinate and y co-ordinat.Other aspects are DisplayUpdating, FactoryCheck, PointCaching and SetterCheck. To test the refactoring, slight changes have been made to the original code.

D. **Point_Introduction(PI)[4]:** This system is made available by Xerox Corporation.This project has a class Point class with data members as x , y coordinates, theta, rho and boolean variables to indicate whether the Point is polar or rectangular. The Point class has appropriate setter and getter methods. Here there are three aspects CloneablePoint, ComparablePoint and HashablePoint which are used to clone the point, compare it and hash it.

E. **Subject Observer Protocol(SOP)[5]:** This system is also made available by Xerox Corporation. It has two interfaces Subject and Observer. It has three classes i.e. Button, ColorLabel, Display and one main class Demo. It also has two aspects SubjectObserverProtocol and SubjectObserverProtocolImpl. The system consists of a colored label. The color of the label changes from the given set of colours. It also displays a number that tells the number of cycles it has gone through. There is a button that is serving as an Action Item which records when it is being clicked. So using these two objects, a Subject /Observer relationship is designed and implemented where the labels are the observers and the buttons are the subjects

F. **Game of Tetris(GT)[6]:** This system has been done by Gustav Evertsson It was developed to test and observe how to use AspectJ by refactoring the existing Tetris program. This system is basically a game, where a block is dropped and the player is supposed to make a line out of the dropped blocks. The game comes to an end when the block reaches the top of the game board. The game system is divided into three sections. The first section deals with the rules of the game. The second section deals with the logic for all data manipulation and the third section deals with the GUI of the game. The code has been modified to illustrate the refactoring. The author has created the following aspects

1. Design checker: to check that the layer
2. Game Info– A Panel for showing info about the game
3. Menu– Adds a menu
4. Counter– Counts the number of deleted lines
5. Level– It makes use of counter to check how many lines have been deleted
6. New Blocks– Used to add two new types of blocks to the game.
7. Next Block–Shows what the next block is when the current block is dropped

G. **RacerAJ(RJ)[7]:** This system is developed by Eric Bodden. It containes a set of aspects which are used for detecting errors that arise during concurrency control in Java applications and AspectJ applications. This system implements the Racer algorithm. It is used for detecting data races in Java and AspectJ programs. This project has implemented 3 novel pointcuts which are provided as an extension to the AspectBench Compiler. The pointcuts are

1. lock() : matches whenever one enters a synchronized block or synchronized method
2. unlock() : matches whenever one exits a synchronized block or synchronized method
3. maybeShared() : is a "semantic pointcut," matches any field access (set or get)

H. **Telecom Application (TAp)[8]:** This system simulates the telephone system. In this the customer makes, accepts, merges or hangs up on a local call or a long distance call. The architecture of this system is divided into three parts. The first part provides the basic functionality of the call and connection. It also provides the functionality to simulate a customer. The second part simulates the timing feature. Here a timer is used with each connection by using aspects. Aspects also manage the total time per customer. The third part of the system simulates the bill given to the customer. The billing depends on the timing. So the billing aspect is built upon the timing aspect. The system has been simulated with three different configurations. They are BasicSimulation, TimingSimulation and BillingSimulation.

I. **Spacewar Game[9]:** This system implements the video game SpaceWar. This comes along with AJDT. This games helps in understanding how the features of AspectJ should be used. This game was developed in 1962 and it is one of the first video games created. It has two spaceships "the needle" and "the wedge" which are engaged in fighting like a dog. Both the ships are controlled by a player which is a human. Also each ship has limited amount of fuel and a limited number of torpedoes. Ships get destroyed when they bump over a torpedo or star.

J. **ToyExample(TE)[10]:** This system consists of three classes BaseClass and Main_LayeredAspects and Driver class. It consists of three aspects: Quote, QuoteDynamic and QuoteStatic. Quote aspect is abstract whereas QuoteStatic and QuoteDynamic are abstract and are extended from the Quote aspect.The Driver class loads a .properties file and checks whether the property is activated or not. The BaseClass is used for validating the text.

1. https://github.com/101companies/101repo/tree/master/languages/AspectJ/aspectJSamples/accounts
2. https://github.com/101companies/101repo/tree/master/languages/AspectJ/aspectJSamples/tracing
3. https://github.com/101companies/101repo/tree/master/languages/AspectJ/aspectJSamples/figures
4. https://github.com/101companies/101repo/tree/master/languages/AspectJ/aspectJSamples/introduction
5. https://github.com/101companies/101repo/tree/master/languages/AspectJ/aspectJSamples/observer
6. http://www.guzzzt.com/coding/aspecttetris.shtml
7. https://github.com/ericbodden/cocoaj/tree/master/RacerAJ/src/ca/mcgill/sable/racer
8. https://eclipse.org/aspectj/doc/released/progguide/printable.html#a-simple-telecom-simulation
9. https://github.com/101companies/101repo/tree/master/languages/AspectJ/aspectJSamples/spacewar
10. https://github.com/rcaa/ToyExample-

III TOOL USED

In this section we discuss the tool that has been used for analyzing the systems before and after refactoring is done. Here we use the tool called AJATO[11] which is an assessment tool. It provides the quantitative analysis of software artifacts. It helps in computing the AO metrics and also supports the use of heuristics

The tool provides following metrics

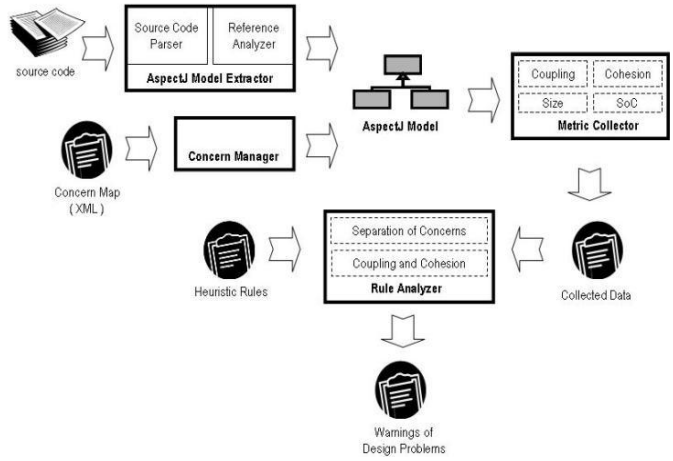
1. Separation of Concerns
2. Coupling
3. Size

In case of Separation of Concerns it provides Number of Attributes per Concern (NOAconcern), Concern Diffusion over Operations, Number of Operations per Concern (NOOconcern) and Concern Diffusion over Components (CDC).

With respect to coupling it provides Number of Children (NC), Depth of Inheritance Tree (DIT).

With respect to Size it provides Number of Statements (NS), Number of Operations (NO), Number of Attributes (NA), Vocabulary Size (VS), Weighted Operations per Component (WOC) and Lines of Code (LC).

The Architecture Model of AJATO is as shown



Source: http://homepages.dcc.ufmg.br/~figueiredo/ajato/

Figure1

The current systems have been analyzed in terms of Coupling and Size. Below, we discuss the meaning of the terms and its impact on the code.

Coupling Metrics

1. Depth of Inheritance Tree (DIT): Inheritance is also referred as generalization. It is an important concept and should be used very carefully. A class or an aspect that is too deep in the inheritance tree is very complex to develop, maintain as well as test. Therefore it is very important to control this depth. So this metric provides the location of the class or the aspect in the inheritance tree. It normal range should be between 0 and 4. A value which is greater than 4 will bargain encapsulation and will increase the complexity of the system
2. Number of Children (NC): It indicates the number of sub classes derived from the super classes. It is used to measure the scope i.e breadth of the class hierarchy. DIT measures depth. Depth is much better than breadth because depth is used for promoting the reuse of the methods. So NOC and DIT are very much related with each other. A high value of NOC indicates too much use of the base class. So base class requires extensive testing. It also indicates irregular or improper abstraction of the super class. High NOC indicates high reuse. A class or aspect with high NOC indicates that there is complexity at the top of the class hierarchy. This is an indication of poor design and so redesign is suggested. The normal range[12] for NOC is between 1 and 4

Size Metrics

1. Vocabulary Size (VS): It is the number if aspects and class declarations in a system. So each class and each aspect is counted. The class can be of different types like inner class, sub class, super class as well as static and abstract class. Also aspects can be of various types like default aspect, inner aspect and abstract aspect.

2. Weighted Operations per Component (WOC): It is the count of the number of arguments of the advices and methods in a system.
3. Number of Attributes (NA): It is the count of the number of fields in the class and aspect.
4. Number of Operations (NO): It is the count of the number of methods declared in the classes as well as aspects present in the system. It does not count abstract methods or constructors present in the aspect or class.
5. Number of Statements (NS): It counts the number of statements in a method. A statement always ends with a "semicolon". It counts all types of statements like constructors, for, return, if, switch etc.
6. Lines of Code (LC): It counts the number of lines in the source code of the class or aspect. It does not count single line, multiple line comments or blank lines and java docs statements.

Other Metrics

1. Time taken for Execution (TE): It is the time taken to execute the system. It is calculated as the difference between the start time and the end time. An average of 6 runs of the systems is considered as the execution time.

11. http://support.objecteering.com/objecteering6.1/help/us/metrics/metrics_in_detail/number_of_children.html
12. Eduardo Magno Lages Figueiredo [emagno @ inf.puc-rio.br], Claudio Nogueira Sant'Anna [claudio @ les.inf.puc-rio.br], Alessandro Fabricio Garcia [garciaa@comp.lancs.ac.uk], Carlos José Pereira de Lucena [lucena@inf.puc-rio.br]

IV ANALYSIS AND RESULTS

1. Make the aspect unprivileged

No.	Para	BA		TA		FS		PI		SOP		GT	
		Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
1.	VS	3	3	6	6	14	14	5	5	7	7	16	16
2.	NA	1	1	8	8	11	11	6	6	6	6	79	78
3.	NO	5	6	32	35	31	31	16	17	19	20	69	69
4.	WOC	12	13	57	60	67	67	26	28	29	30	127	127
5.	NS	13	15	61	63	53	53	90	97	32	33	659	611
6.	LC	36	40	134	142	199	199	137	146	102	105	1034	985
7.	DIT	3	3	8	8	15	15	5	5	10	10	18	18
8.	NC	0	0	0	0	3	3	0	0	1	1	0	0
9.	TE	167.5	147	230.5	227	40.33	34.2	171.7	149.7	329	317	3268	2733

Figure2: Refactoring 1 Comparison

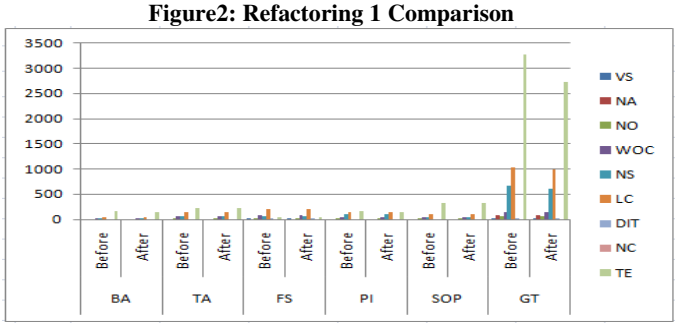


Figure3: Refactoring 1 Comparison Chart

2. Replace the pointcut name with its designator

No.	Para	BA		TA		FS		SOP		GT		RJ		SW		Tap		TE	
		Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
1.	VS	3	3	6	6	14	14	7	7	16	16	16	16	21	21	13	13	6	6
2.	NA	1	1	8	8	11	11	6	6	79	80	25	25	50	50	21	21	1	1
3.	NO	6	6	32	32	31	31	19	19	69	69	84	84	108	108	52	52	22	22
4.	WOC	13	13	61	61	67	67	29	29	127	127	184	184	196	196	94	94	42	42
5.	NS	15	15	61	61	53	50	32	36	659	663	306	306	460	460	128	128	51	46
6.	LC	36	38	134	131	199	192	102	101	1034	1025	574	573	1472	1466	265	263	119	114
7.	DIT	3	3	8	8	15	15	10	10	18	18	25	25	29	29	18	18	5	5
8.	NC	0	0	2	2	3	3	1	1	0	0	9	9	4	4	5	5	2	2
9.	TE	158.2	149	185.8	178	40.3	36	329	324	3268	2929	4	2	23932	23204	697m	601m	288	268

Figure4: Refactoring 2 Comparison

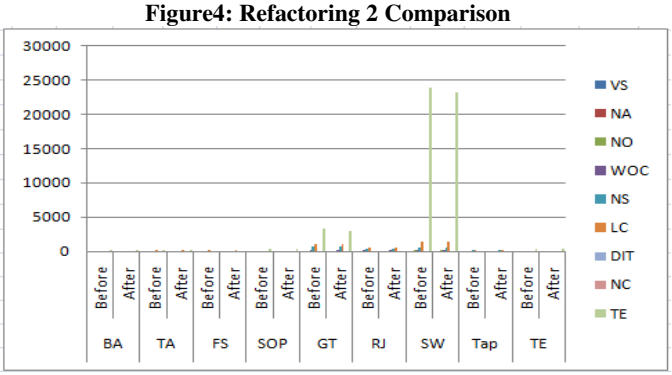


Figure5: Refactoring 2 Comparison Chart

3. Introduce the get and set pointcut , introduce before and after advice

No.	Para	BA		TA		FS		SOP		TE	
		Before	After	Before	After	Before	After	Before	After	Before	After
1.	VS	1	3	6	6	14	14	7	7	6	6
2.	NA	1	1	8	8	11	11	6	6	1	1
3.	NO	2	6	32	44	31	43	19	23	22	22
4.	WOC	4	13	57	74	67	77	29	35	42	42
5.	NS	8	15	61	79	53	76	32	40	51	68
6.	LC	23	42	134	190	199	247	102	124	119	173
7.	DIT	1	3	8	8	15	15	10	10	5	5
8.	NC	0	0	2	2	3	3	1	1	2	2
9.	TE	166.6	103	190	186	40.33	37	329.33	287	288	258

Figure6: Refactoring 3 Comparison

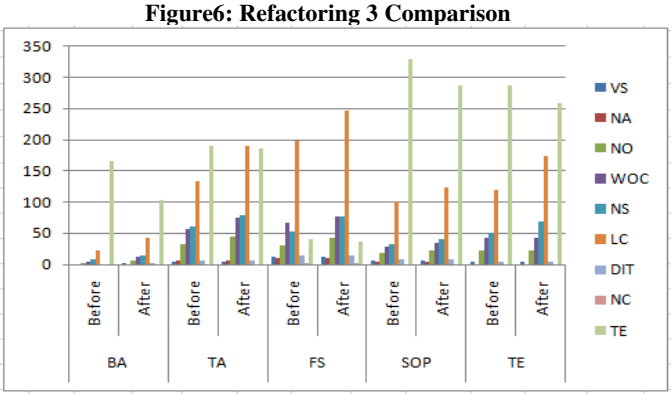


Figure7: Refactoring 3 Comparison Chart

4. Remove the word abstract for the aspect

No.	Para	SW		SOP		TE	
		Before	After	Before	After	Before	After
1.	VS	21	21	7	7	6	6
2.	NA	50	52	6	6	1	1
3.	NO	108	108	19	20	22	22
4.	WOC	196	196	29	30	42	42
5.	NS	460	460	32	33	51	51
6.	LC	1472	2474	102	107	119	119
7.	DIT	29	27	10	9	5	5
8.	NC	4	4	1	0	2	2
9.	TE	23932	22731	329.33	327	288	231

Figure8: Refactoring 4 Comparison

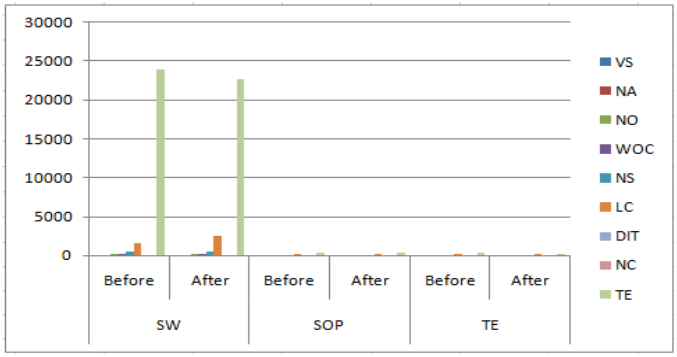


Figure9: Refactoring 4 Comparison Chart

IV CONCLUSIONS

As seen in the result, for the first refactoring Vocabulary Size, Number of Attributes remains same. But there is a change in the Number of Operations, Weighted Operations per component, Number of Statements, Lines of Code and Execution time. So this refactoring reduces the time taken for execution even though the number of lines of code increases. We can see the same pattern in the other three refactorings. Therefore we can conclude that if the above refactorings are applied to the code, the system will execute faster.

REFERENCES

[1] A. Rani and H. Kaur, "Refactoring Methods and Tools", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 12, pp. 256-260, 2012.

[2] Puneet Jai Kaur, Sarita Rani, "Impact of Aspect Oriented Programming on Software Maintainability - A Descriptive Study, University Institute of Engineering and Technology, Panjab University, Sector 25, Chandigarh, International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), IJETCAS 14-340; 2014

[3] Pradeep Kumar Singh, Om Prakash Sangwan, Amar Pal Singh Amrendra Pratap, "An Assessment of Software Testability using Fuzzy Logic Technique for Aspect-Oriented Software", I.J. Information Technology and Computer Science, 2015, 03

[4] Freddy Munoz,Benoit Baudry, Romain Delamare, Yves Le Traon "Inquiring the Usageof Aspect-Oriented Programming: An Empirical Study"

[5] Tom Mens, Tom Tourw´e "A Survey of Software Refactoring", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. XX, NO. Y, MONTH 2004

[6] Eduardo Figueiredo,Alessandro Garcia, Carlos Lucena, AJATO: an AspectJ Assessment Tool

[7] Muhammad Sarmad Alia, Muhammad Ali Babar,, Lianping Chen, Klaas-Jan Stol, Information and Software Technology, 52 , 871–887(2010)

[8] Terry Hon, A Simple, Modern AspectJ Compiler

[9] Sven Apel, and Don Batory," How AspectJ is Used:" An Analysis of Eleven AspectJ Programs", Technical Report, Number MIP-0801, Department of Informatics and Mathematics,University of Passau, Germany,April 2008

[10] Khine Zar Ne Winn,"Quantifying and Validation of Changeability and Extensibility for Aspect-Oriented Software", International Conference on Advances in Engineering and Technology (ICAET'2014) March 29-30, 2014 Singapore

[11] Piyush Chandi," A Survey : Code Optimization using Refactoring", International Journal on Computer Science and Engineering (IJCSE), Vol. 5 No. 05, May 2013

Authors Profile

Geeta Bagade(Mete), a Master in computer Science from the University of Pune and currently pursuing her Ph.D in Computer Science from Bharati Vidyapeeth,Pune has more than 12 years of experience in IT training. She possesses good technical skills with respect to programming languages as well as databases.

Dr. Shashank Joshi, is a B.E. in Electronics and Telecommunication from Govt. College of Engineering, Pune in 1988. He also completed the M.E. and Ph. D. Degree in Computer Engineering from Bharati Vidyapeeth Deemed University Pune. He is currently working as the Professor in Computer Engineering Department, Bharati Vidyapeeth Deemed University, College of Engineering, Pune. His research interests include software engineering. Presently he is engaged in SDLC and secure software development methodologies. He is a passionate professor with overal experience of more than 20 yrs.