Measurement of Dynamic Cohesion using Aspect Oriented Approach

Manju[1], *Pradeep kumar Bhatia*[2]

[1]*Research Scholar,*[2] *Professor*

[1]*Computer Science and Engineering*
[1]*Guru Jambheshwar University of Science and Technology, Hisar, India*

Abstract: There is lots of cohesion metrics exist in literature to find the cohesion of a class at compile time. Many researchers has given cohesion metrics but most of them are static i.e. can be calculated from design patterns. Only few metrics are available to find cohesion of a class at runtime. In this paper, we introduced a new metric to find cohesion of a class at runtime and value of purposed metric is calculated by using a purposed tool named DynaDlcom. The purposed tool is implemented in AspectJ using aspect oriented programming. After that, comparison of static LCOM and purposed metric named DLCOM values is done. An experimental study is done on 15 java classes and it is concluded that purposed metric DLCOM plays a significant role to find cohesion, hence quality of a software system.

Keywords : **Dynamic, cohesion ,quality , static**

I. Introduction

Metrics at any given time during the project execution indicate the quality of the product being built. Metric analysis for similar projects execution gives the capability levels of processes being used over a period of time. Metric are simple to defined, easy to understand, robust to use and able to improve [3]. Henderson et. al.[9] has introduced cohesion metric to find lack of cohesion of methods at compile time. LCOM (96a)[8] is basically based on number of attributes invoked by all the methods in a class at compile time. If all the attributes are used by all the methods in the class then the value of LCOM is 0 and class is declared as highly cohesive class.

$$LCOM96a = \frac{(\frac{1}{a}\sum_{j=1}^{a}\mu(Aj)) - m}{1 - m}$$

Where a and m are number of attributes and methods respectively, and $\mu(A_j)$ is the number of methods accessing attribute A_j. In this paper, we extract the value of LCOM96a metric by using CodeMR tool. CodeMR is mainly a Eclipse plugin to find the quality of software in terms of 3 main quality measures i.e. complexity, cohesion and coupling for java, C++ and scala languages .It also provides Graph and Dependency views of the metrics calculated by the tool. For dynamic code analysis we purposed a new tool DynaDlcom implemented in AspectJ, using aspect oriented programming. There are some advantages of Aspect oriented approach over the other approaches to trace the events at runtime.

1. The process of building a tracing or profiling frame work using aspect oriented approach is relatively simpler than any other approach.
2. This approach does not generate a large amount of data.
3. This approach produces results which are valid among all programme execution environment.
4. This approach is relatively inexpensive and much more practical in nature.

2. Proposed Metric

DLCOM (Dynamic Lack of Cohesion of Methods) : This metric is used to find the cohesion of a class at runtime. It is mainly combination of three metrics RLCOM, RAAR, RMMC. Detailed definition of these metrics is defined as following:

RLCOM (Runtime Lack of cohesion of methods): It is defined as number of methods access their class attributes at runtime. If all the attributes are accessed by all the methods at runtime then value of this metric comes out to be 0 i.e. highly cohesive class.

$$RLCOM = \frac{(\frac{1}{a}\sum_{i=1}^{m}\sum_{j=1}^{a}a_{ij}) - m}{1 - m} \qquad (1)$$

Where a is the total number of attributes accessed at runtime , m is the total number of times methods of a class executed at runtime and a_{ij} is the number of times attribute j is accessed by method i at runtime.

RAAR (Runtime Attribute Access Rate): It is defined as number of read and write operations are done on class attributes at runtime by all the methods of a class.

$$RAAR = \frac{a}{m\sum_{i=1}^{m}\sum_{j=1}^{a}r_{ij} + m\sum_{i=1}^{m}\sum_{j=1}^{a}w_{ij}} \qquad (2)$$

Where a is total number of attributes in class, m is the total number of time methods of a class executed at runtime ,r_{ij} is the number of times attribute j is read by method i at runtime and w_{ij} is the number of times attribute j is write by method i at runtime.

RMMC (Runtime method method call): It is defined as number of times a method calls another method at runtime within a class.

$$RMMC = \frac{\frac{\sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m} n_{ij}}}{a*m} \tag{3}$$

Where a and m are total number of times attributes and methods of a class executed at runtime, c_{ij} is a count of a method i calling another method j at runtime and n_{ij} is the count of number of times a method i call another method j at runtime.

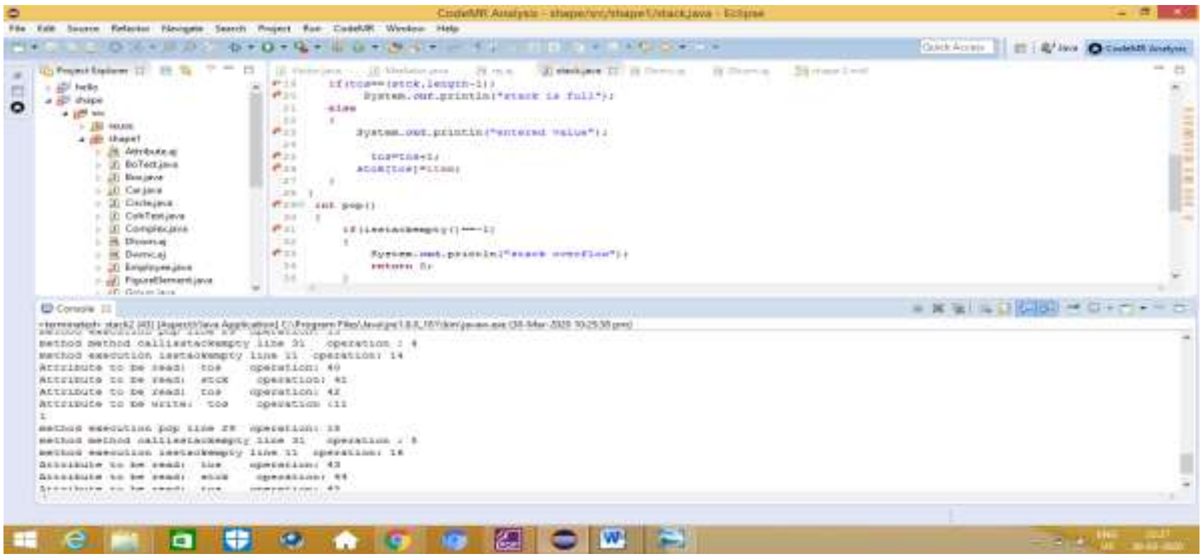
DLCOM (Dynamic Lack of Cohesion of Methods): It is defined as combination of RLCOM (Runtime Lack of cohesion of methods), RAAR (Runtime Attribute Access Rate) and RMMC (Runtime method method call).From equation (1),(2) and (3),

$$\text{DLCOM} = \text{RLCOM} + \text{RAAR} + \text{RMMC}$$

Where RLCOM is Runtime Lack of cohesion of methods, RAAR is Runtime Attribute Access Rate and RMMC is Runtime method method call .The value of this metric is mainly ranges between 0-1.5. If the value of metrics comes out 0 i.e. class is highly cohesive and maintainability of class is high. If the value of metric is around and above 1 then class has to be redesign and maintainability of class is very low hence decrease the quality of the system.

3. Proposed Tool

DynaDlcom is a tool to calculate the purposed metric. Tool is implemented in AspectJ[2]. AspectJ is mainly an implementation of Aspect Oriented programming in java. Fig. 1. shows the running state of aspect in AspectJ which can be run with any java class to find the value of purposed metric. DynaDlcom computes the total number of read operations on class attributes, total number of write operations on class attributes, total number of method executed at run time, how many times methods of a class execute at runtime and number of methods calling another method at runtime.



```
        if(tos==(stck.length-1))
            System.out.println("stack is full");
        else
        {
            System.out.println("entered value");

            tos=tos+1;
            stck[tos]=item;
        }
    }
    int pop()
    {
        if(isstackempty()==-1)
        {
            System.out.println("stack overflow");
            return 0;
        }
```

```
<terminated> stack2 [AspectJ/Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (30-Mar-2020 10:25:38 pm)
method method callisstackempty line 31   operation : 4
method execution isstackempty line 11  operation: 14
Attribute to be read:  tos    operation: 40
Attribute to be read:  stck    operation: 41
Attribute to be read:  tos    operation: 42
Attribute to be write:  tos    operation :11
1
method execution pop line 29  operation: 15
method method callisstackempty line 31   operation : 5
method execution isstackempty line 11  operation: 16
Attribute to be read:  tos    operation: 43
Attribute to be read:  stck    operation: 44
Attribute to be read:  tos    operation: 45
```

Fig 1. DynaDlcom tool in AspectJ

4. Case Study

Consider a program written in java shown in Fig. 2. In this program, stack class contains 2 attributes and 5 methods named push, pop, isstackempty, topofstack and stack class constructor.

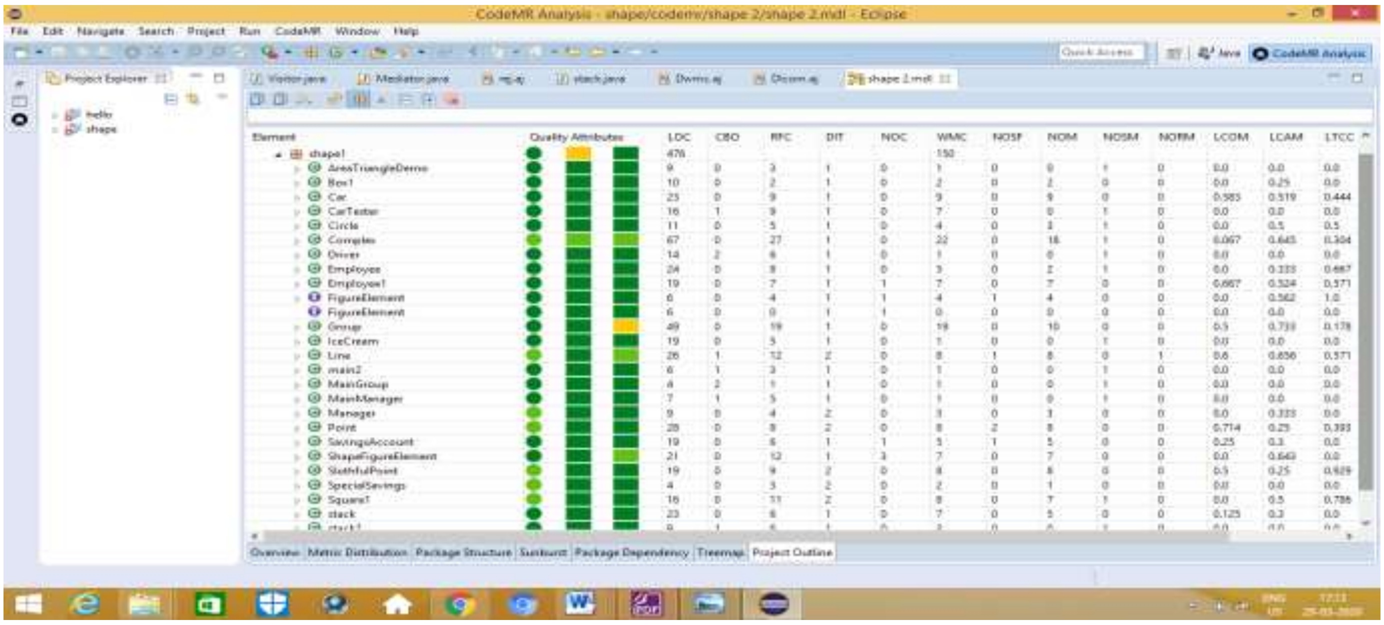
```
public class stack {
    int stck[];
    int tos;
    stack(int size)
    {
            stck=new int[size];
            tos=-1;
    }
    int isstackempty()
    { return tos;}
    int topofstack()
    {
           return stck[tos-1];
    }
void push(int item)
{
  if(tos==(stck.length-1))
                System.out.println("stack is full");
  else
  {
               System.out.println("entered value");

        tos=tos+1;
               stck[tos]=item;
  }
}
 int pop()
 {
            if(isstackempty()==-1)
            {
                        System.out.println("stack overflow");
                        return 0;
            }
            else
                        return stck[tos--];
 }
}
class stack2
{
public static void main(String str[])
{
              stack s1=new stack(5);
              System.out.println("enter values in stack");
              for(int i=0;i<5;i++)
                          s1.push(i);
              System.out.println("popped values");
              for(int i=0;i<5;i++)
                          System.out.println(s1.pop());
}
}
```

Fig 2. Stack Class implemented in java

Value of the LCOM96a metric for stack class is 0.125 as shown in Fig. 3. Value of LCOM96a is calculated using CodeMR tool shown in Fig. 3. To calculate the value of DLCOM, DynaDlcom tool is used. DLCOM.aj is an aspect created in AspectJ and run with stack.java class without interrupting the functioning of Stack class and the value of DLCOM metric is calculated from runtime log as shown in Fig. 3.Value of DLCOM metric is 0.275 as RLCOM=0.107, RAAR=0.002 and RMMC=0.166.Stack class is highly cohesive at compile time but less cohesive at runtime that effects the maintainability of the class hence quality of software system.

5. Experimental Study

An experimental study is conducted for the purposed metrics using 15 java classes. Classes are taken from web. CodeMR tool is used to calculate value of LCOM96a metric for these classes as shown in Fig.3 and DynaDlcom tool is used to calculate the values of purposed metric using AspectJ shown in Fig. 1. AspectJ first calculate total number of read operations on attributes within class and total number of write operations on attributes within class. Total number of method calls are also calculated i.e. number of methods calling another method within class. Total number of methods executed at runtime is also calculated to find value of DLCOM.



Element	LOC	CBO	RFC	DIT	NOC	WMC	NOSF	NOM	NOSM	NORM	LCOM	LCAM	LTCC
shape1	478					150							
AreaTriangleDemo	9	0	3	1	0	1	0	0	1	0	0.0	0.0	0.0
Box1	10	0	2	1	0	2	0	2	0	0	0.0	0.25	0.0
Car	25	0	9	1	0	9	0	9	0	0	0.583	0.519	0.444
CarTester	16	1	9	1	0	7	0	0	1	0	0.0	0.0	0.0
Circle	11	0	5	1	0	4	0	3	1	0	0.0	0.5	0.5
Complex	67	0	27	1	0	22	0	18	1	0	0.067	0.645	0.304
Driver	14	2	6	1	0	1	0	0	1	0	0.0	0.0	0.0
Employee	24	0	8	1	0	3	0	2	1	0	0.0	0.333	0.667
Employee1	19	0	7	1	1	7	0	7	0	0	0.667	0.524	0.571
FigureElement	6	0	4	1	1	4	1	4	0	0	0.0	0.562	1.0
FigureElement	6	0	0	1	1	0	0	0	0	0	0.0	0.0	0.0
Group	49	0	19	1	0	19	0	10	0	0	0.5	0.733	0.178
IceCream	19	0	5	1	0	1	0	0	1	0	0.0	0.0	0.0
Line	26	1	12	2	0	8	1	8	0	1	0.6	0.656	0.571
main2	6	1	3	1	0	1	0	0	1	0	0.0	0.0	0.0
MainGroup	4	2	1	1	0	1	0	0	1	0	0.0	0.0	0.0
MainManager	7	1	5	1	0	1	0	0	1	0	0.0	0.0	0.0
Manager	9	0	4	2	0	3	0	3	0	0	0.0	0.333	0.0
Point	28	0	8	2	0	8	2	8	0	0	0.714	0.25	0.393
SavingsAccount	19	0	6	1	1	5	1	5	0	0	0.25	0.3	0.0
ShapeFigureElement	21	0	12	1	3	7	0	7	0	0	0.0	0.643	0.0
SlothfulPoint	19	0	9	2	0	8	0	8	0	0	0.5	0.25	0.929
SpecialSavings	4	0	3	2	0	2	0	1	0	0	0.0	0.0	0.0
Square1	16	0	11	2	0	8	0	7	1	0	0.0	0.5	0.786
stack	23	0	6	1	0	7	0	5	0	0	0.125	0.3	0.0

Fig. 3. Value of LCOM96a metric of java classes using CodeMR tool.

The values of LCOM and purposed metric of 15 java classes is shown in Table 1. From Table 1, we can say that there is huge difference between compile time and runtime lack of cohesion of methods. Behaviour of every java class is different. Some classes like 11,12 shows same behaviour at both compile time and run time that means all the attributes of class are accessed by all the methods of the class at runtime and there is no method calling another methods within the class. Classes like 4,6,10,13,15 shows huge difference between compile and run time that depicts class attributes are not accessed by methods of class fully and there exist method methods call within the class that makes quality of class very low and there is a need to redesign that class. Therefore, DLCOM makes a count as class 4 at compile time it is highly cohesive according to Henderson et al. metric LCOM96a but at runtime it is highly in danger zone and needs to be redesign. So the purposed metric helps the developer to find classes that are not cohesive and affects the quality. Hence the purposed metrics plays a vital role to find quality of software.

Table 1. Values of LCOM and DLCOM metric of 15 java classes

Sr. No.	Class Name	LCOM	DLCOM
1	Box	0	0.25
2	Stack	0.125	0.275
3	Car	0.583	0.653
4	Circle	0	1.27
5	Cohtest	0	0.25
6	Complex	0.067	0.46
7	Employee	0	0.314
8	Ice Cream	0	0.27
9	Employee 1	0.66	0.88
10	Manager	0	0.5
11	Saving Account	0.25	0.22
12	Special Saving	0	0
13	Area Triangle	0	0.75
14	2D Shape	0.33	0.5
15	Square	0	1.13

6. Conclusion

The purposed metric DLCOM, Dynamic Lack of Cohesion plays a significant role to find maintainability of software. Value of purposed metric is calculated by DynaDlcom tool .An experimental Study is done on 15 java classes and it is found that there is huge difference between values of compile time class cohesion and run time class cohesion. Hence, to measure the quality of software, purposed metric plays a significant role by measuring the runtime cohesion of software. Analysis result of these classes is shown in Fig4. It Contains 15 java classes and compared values of LCOM and purposed DLCOM metric.

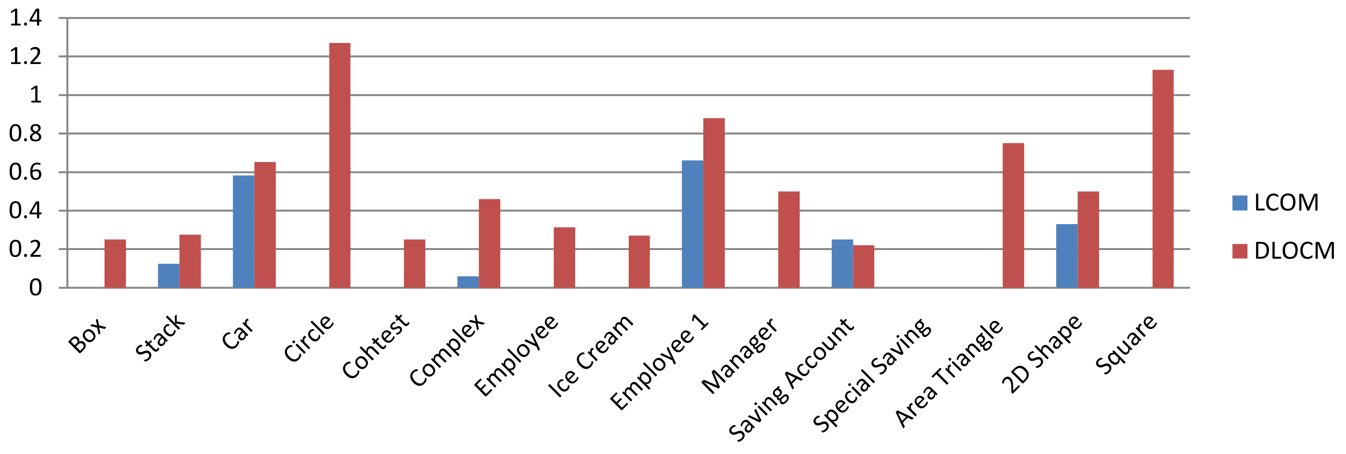


Fig. 4. Comparison of LCOM(96a) and DLCOM metric of 15 java classes

Statistical results of 15 java classes are shown in Table 2.In Table 2. The Karl Pearson Product –Moment Correlation method is used to find correlation between LCOM96a and purposed metric DLCOM. The computed value of correlation coefficient(r) is 0.18 at a significance level of 0.05 that shows there is huge difference between compile time and runtime values of cohesion metric i.e. DLCOM has its own existence and not similar to existing LCOM96a.Statistical Values are calculated by using MATLAB tool.

Table 2. Statistical results between LCOM96a and DLCOM

Sr.No.	Statistic Applied	LCOM96a	Purposed Metric(DLCOM)
1.	Mean	0.133	0.51
2.	Median	0	0.46
3.	Min	0	0
4.	Max	0.66	1.27
5.	Standard Deviation	0.22	0.34

References

[1] J. Wang, Y. Zhou, L. Wen, Y. Chen, H. Lu, and B. Xu, “DMC: A More Precise Cohesion Measure for Classes”, *Information and Software Technology*, vol. 47, no. 3, pp. 167-180, 2005.
[2] AspectJ < http://www.eclipse.org/aspectj>
[3] L.C. Briand, J.W. Daly, and J. Wust, “A Unified Framework for Cohesion Measurement in Object-Oriented Systems”, *Empirical Software Engineering: An International Journal*, vol. 3, no. 1, pp. 65–117, 1998.
[4] H.S. Chae and Y.R. Kwon. “A Cohesion Measure for Classes in Object-Oriented Systems”, *Proc. Fifth International Software Metrics Symposium*, IEEE Computer Society Press, Bethesda, MD, USA, pp. 158-166, 1998.
[5] H.S. Chae, Y.R. Kwon, and D.H. Bae. “A Cohesion Measure for Object-Oriented Classes”, *Software Practice & Experience*, vol. 30, no. 12, pp. 1405-1431, 2000.
[6] S.R. Chidamber and C.F. Kemerer, “Towards a Metrics Suite for Object Oriented Design”, *Proc. 6th ACM Conf. Object-Oriented Programming: Systems, Languages and Applications (OOPSLA)*, Phoenix, AZ, pp. 197-211, Oct. 1991.
[7] S.R. Chidamber and C. F. Kemerer, “A Metrics Suite for Object-Oriented Design”,*IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.
[8] tusharma.in/technical/revisiting-lcom/
[9] B. Henderson-Sellers, Software Metrics, Prentice Hall, Hemel Hempstaed, U.K., 1996.
[10] Henderson-Sellers, Brian, Larry L. Constantine and Ian M. Graham. “Coupling and cohesion (towards a valid metrics suite for object-oriented analysis and design).” Object Oriented Systems 3 (1996): 143-158.