International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 3, March - 2014

Survey on Object Oriented Matrices

Ms. Soniya S. Dadhania
M.E Computer Science and Engineering
Lecturer Computer Engineering
R C Technical Institute, Ahmedabad

Ms. Avni S Galathiya
MTech Computer Engineering
Lecturer Computer Engineering
R C Technical Institute, Ahmedabad

***Abstract*: Software quality estimation in terms of performance and reliability can be made by using software matrices. Software matrices are used to calculate various types of complexities, dependability, coupling, reusability, cohesion etc., This measures are useful to estimate the quality of the software because more reusable the software is less time and cost required to develop new software by use of exiting one. In this paper we have studied CK matrices suit which contains Weighted methods per class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling Between Object Classes (CBO), Response for a Class (RFC), Lack of Cohesion in Methods (LCOM), and other matrices IFANIN, NIM, NIV. In this paper, Comparison between two projects is made, one project consist of large number of classes and another contains comparatively less number classes.**

I. INTRODUCTION

A. *CK Matrices Suit:*

Depth of the Inheritance tree (DIT)

The maximum length from the node to the root of the tree.
As DIT grows, it becomes difficult to predict behavior of a class because of the high degree of inheritance [5].

Number of children (NOC)

Count of the subclasses immediately subordinate to a class
As NOC grows, reuse increases.As NOC grows, abstraction can become diluted. Increase in NOC means the amount of testing will increase [5].

Coupling between object classes (CBO)

The number of collaborations listed for a class
As CBO increases, reusability of the class decreases
High CBO values complicate modifications
In general, CBO values for each class should be kept as low as possible [5].

Response for a class (RFC)

The number of methods that can potentially be executed in response to a message received by an object
As RFC increases, testing effort increases because the test sequence grows.
As RFC increases, the overall complexity of the class increases [5].

Lack of cohesion in methods (LCOM)

Measure of the number of methods within a class that access the same instance variables
If no methods access the same attributes, LCOM = 0.
As LCOM increases, coupling between methods (via attributes) increases, and thus class complexity increases [5].

II. EVALUATION

In this report, we have analyzed some metrics by using Understand 2.6(build 581) tool. In our analysis I have use two java projects to measure object oriented metrics.
In this paper, we focused mainly LCOM (Percent Lack of Cohesion), DIT (Max Inheritance Tree), IFANIN (Count of Base Classes), CBO (Count of Coupled Classes), NOC (Count of Derived Classes), RFC (Count of All Methods), NIM (Count of Instance Methods), NIV (Count of Instance Variables), WMC (Count of Methods), as this tool support those metrics.
Project -1 details

Classes:	85
Files:	33
Library Units:	513
Lines:	22271
Lines Blank:	997
Lines Code:	18983
Lines Comment:	2987
Lines Inactive:	0
Executable Statements:	10074
Declarative Statements::	2784
Ratio Comment/Code:	0.16

Project -2 details

Classes:	19
Files:	12
Library Units:	49
Lines:	2589
Lines Blank:	113
Lines Code:	2224
Lines Comment:	614
Lines Inactive:	0
Executable Statements:	946
DeclarativeStatements: :	548
Ratio Comment/Code:	0.28

Table A and Table B represent project1 and project2 metrics respectively.

TABLE-A

Class	LCOM	DIT	IFANIN	CBO	NOC	RFC	NIM	NIV	WMC
Class1	100	5	1	5	0	452	2	1	2
Class2	80	5	2	6	0	457	7	3	7
Class3	0	2	1	6	0	19	2	0	2
Class4	0	1	1	2	0	12	0	9	0
Class5	87	5	2	6	0	457	7	5	7
Class6	0	2	1	4	0	20	3	0	3
Class7	0	1	1	2	0	12	0	9	0
Class8	83	5	2	6	0	457	7	5	7
Class9	0	2	1	4	0	20	3	0	3
Class10	0	1	1	2	0	12	0	9	0
Class11	83	5	2	6	0	457	7	5	7
Class12	0	2	1	5	0	21	4	0	4
Class13	0	1	1	2	0	12	0	10	0
Class14	83	10	2	13	0	914	7	5	7
Class15	0	4	1	10	0	44	5	0	5
Class16	0	2	1	4	0	24	0	10	0
Class17	83	5	2	6	0	457	7	5	7
Class18	0	2	1	5	0	22	5	0	5
Class19	0	1	1	2	0	12	0	10	0
Class20	83	5	2	6	0	457	7	5	7
Class21	0	2	1	5	0	22	5	0	5
Class22	0	1	1	2	0	12	0	10	0
Class23	83	5	2	8	0	457	7	5	7
Class24	0	2	1	7	0	24	7	0	7
Class25	0	1	1	2	0	12	0	10	0
Class26	0	5	1	5	0	452	2	1	2
Class27	83	5	2	6	0	457	7	5	7
Class28	0	2	1	7	0	25	8	0	8
Class29	0	1	1	2	0	12	0	10	0
Class30	83	5	2	6	0	457	7	5	7
Class31	0	2	1	7	0	27	10	0	10
Class32	0	1	1	2	0	12	0	10	0
Class33	84	5	2	6	0	457	7	6	7
Class34	0	2	1	8	0	29	12	0	12
Class35	0	1	1	2	0	12	0	10	0
Class36	84	5	2	6	0	457	7	6	7
Class37	0	2	1	8	0	29	12	0	12
Class38	0	1	1	2	0	12	0	10	0
Class39	84	5	2	6	0	457	7	6	7
Class40	0	2	1	9	0	30	13	0	13
Class41	0	1	1	2	0	12	0	10	0
Class42	84	5	2	6	0	457	7	6	7
Class43	76	2	1	9	0	30	13	2	13
Class44	0	1	1	2	0	12	0	10	0
Class45	84	5	2	6	0	457	7	6	7
Class46	78	2	1	9	0	31	14	2	14
Class47	0	1	1	2	0	12	0	10	0
Class48	87	5	2	6	0	457	7	8	7
Class49	0	2	1	9	0	33	16	0	16
Class50	0	1	1	2	0	12	0	10	0
Class51	85	5	2	6	0	457	7	9	7
Class52	0	2	1	9	0	36	19	0	19
Class53	0	1	1	2	0	12	0	10	0
Class54	60	5	2	7	0	454	4	5	4
Class55	85	5	2	6	0	457	7	8	7
Class56	0	2	1	9	0	37	20	0	20
Class57	0	1	1	2	0	12	0	10	0
Class58	85	5	2	6	0	457	7	8	7
Class59	0	2	1	9	0	37	20	0	20
Class60	0	1	1	2	0	12	0	10	0
Class61	85	5	2	6	0	457	7	9	7
Class62	0	2	1	10	0	42	25	0	25
Class63	0	1	1	2	0	12	0	10	0
Class64	85	5	2	6	0	457	7	10	7

Class65	0	2	1	10	0	42	25	0	25
Class66	0	1	1	2	0	12	0	10	0
Class67	85	5	2	6	0	457	7	9	7
Class68	0	2	1	9	0	42	25	0	25
Class69	0	1	1	2	0	12	0	10	0
Class70	85	5	2	6	0	457	7	10	7
Class71	0	2	1	9	0	42	25	0	25
Class72	0	1	1	2	0	12	0	10	0
Class73	52	5	2	7	0	454	4	17	4
Class74	55	5	2	7	0	454	4	17	4
Class75	0	2	1	4	0	19	2	0	2
Class76	56	5	2	7	0	454	4	18	4
Class77	0	2	1	5	0	19	2	0	2
Class78	56	5	2	7	0	454	4	18	4
Class79	0	2	1	5	0	19	2	0	2
Class80	96	5	2	8	0	456	6	18	6
Class81	0	2	1	5	0	19	2	0	2
Class82	0	1	1	2	0	12	0	8	0
Class83	80	5	2	8	0	457	7	3	7
Class84	0	2	1	5	0	19	2	0	2
Class85	0	1	1	2	0	12	0	8	0

TABLE B

Class	LCOM	DIT	IFANIN	CBO	NOC	RFC	NIM	NIV	WMC
Class1	85	7	1	26	0	109	5	22	5
Class2	0	1	1	9	0	14	0	0	2
Class3	0	1	1	9	0	14	0	0	2
Class4	0	1	1	9	0	14	0	0	2
Class5	0	1	1	9	0	14	0	0	2
Class6	0	1	1	9	0	14	0	0	2
Class7	0	1	1	7	0	14	0	0	2
Class8	0	1	1	7	0	14	0	0	2
Class9	0	1	1	7	0	14	0	0	2
Class10	0	1	1	7	0	14	0	0	2
Class11	0	1	1	7	0	14	0	0	2
Class12	0	1	1	8	0	14	0	0	2
Class13	0	1	1	8	0	14	0	0	2

LCOM (Lack of Cohesion in Methods):

Research: Chidamber & Kemerer - Lack of Cohesion in Methods (LCOM/LOCM)

Description: 100% minus average cohesion for class data members. Calculates what percentage of class methods use a given class instance variable. To calculate, average percentages for all of that class'es instance variables and subtract from 100%. A lower percentage means higher cohesion between class data and methods.[1] Maximum LCOM in Table A and Table B is 100 and 85 respectively.

If LCOM is high, methods may be coupled to one another via attributes and then class design will be complex. So, designers should keep cohesion high, that is, keep LCOM low [2].
Therefore, Project 2 is having better LCOM then Project 1.
Graphical representation of LCOM from table A and table B.

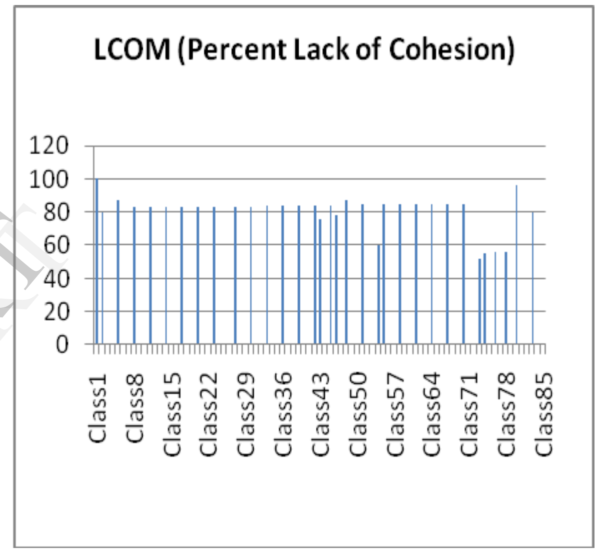

TABLE A

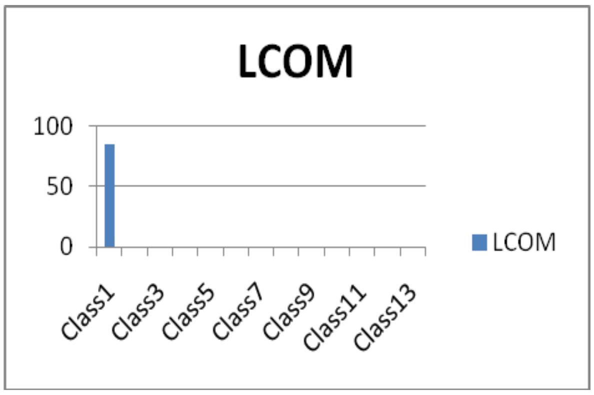

TABLE B

DIT(Depth of Inheritance Tree)

Research: Chidamber & Kemerer - Depth of Inheritance Tree(DIT)
Description: The depth of a class within the inheritance

hierarchy is the maximum number of nodes from the class node to the root of the inheritance tree. The root node has a DIT of 0. The deeper within the hierarchy, the more methods the class can inherit, increasing its complexity [1].

Maximum DIT in Table A and Table B is 10 and 7 respectively.

From the above result we can conclude that Project 2 is less complex and project 1 is more complex.

Graphical representation of DIT from table A and table B.

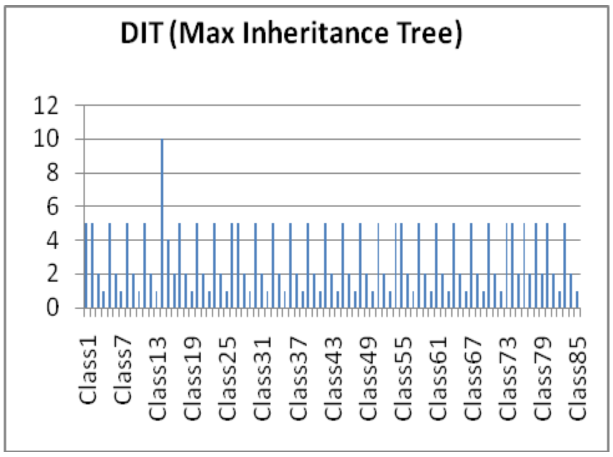

TABLE A

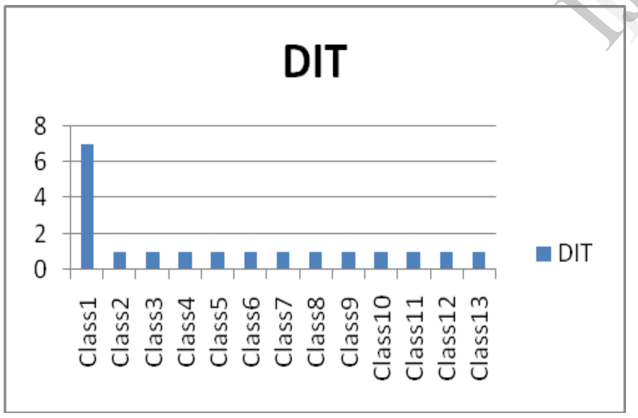

TABLE B

IFANIN (Count of Base Classes)

*Research:*IFANIN
Description: Number of immediate base classes.[1]
Maximum IFANIN in Table A and Table B is 2 and 1 respectively.
Graphical representation of IFANIN from table A and table B.

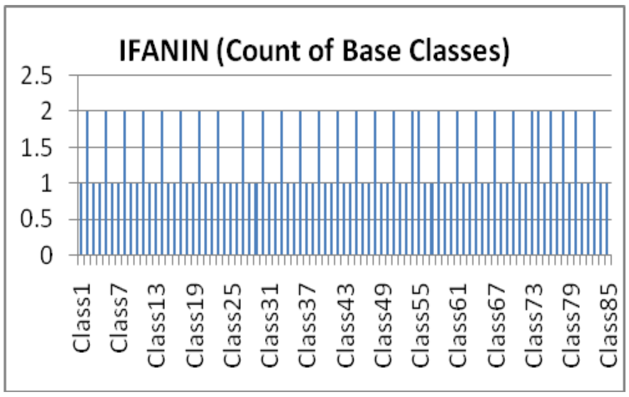

TABLE A

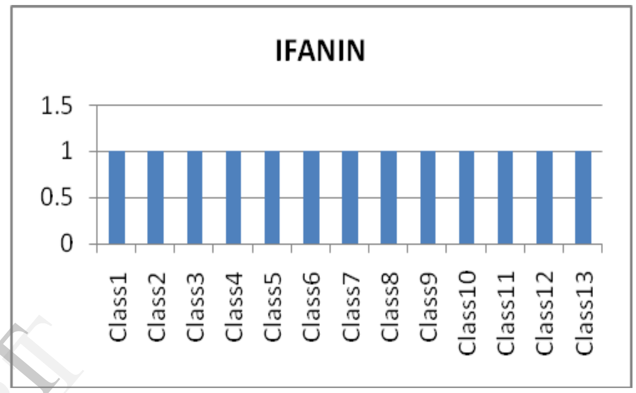

TABLE B

CBO (Count of Coupled Classes)

Research: Chidamber & Kemerer - Coupling Between Objects(CBO)
Description: The Coupling Between Object Classes (CBO) measure for a class is a count of the number of other classes to which it is coupled. Class A is coupled to class B if class A uses a type, data, or member from class B. This metric is also referred to as Efferent Coupling (Ce). Any number of couplings to a given class counts as 1 towards the metric total [1].

Chidamber & Kemerer suggest that:
1) Excessive coupling between object classes is detrimental to modular design and prevents reuse.
2) Inter-object class couples should be kept to a minimum.
3) The higher the inter-object class coupling, the more rigorous testing needs to be[1].

Maximum COB in Table A and Table B is 13 and 26 respectively.
Therefore, we can conclude that the reusability of project 1 is more than that of project 2.
Even though Project 2 is having less no of classes than project 1, reusablity of project 2 is less then project 1. Thus we can conclude that, the reusability does not depend on the number of class and size of code.

Graphical representation of CBO from table A and table B.

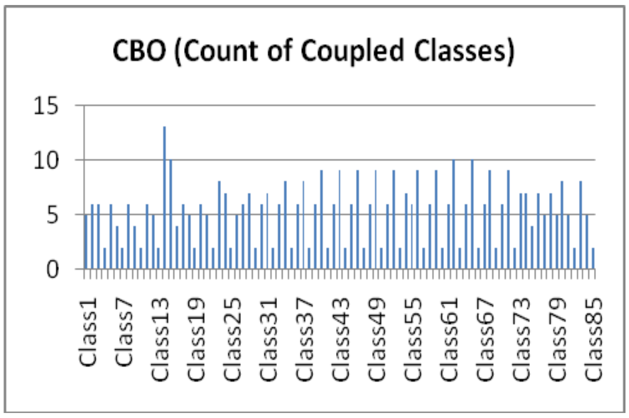

TABLE A

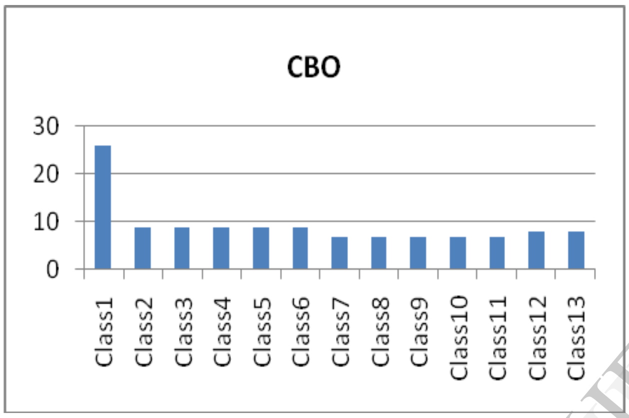

TABLE B

NOC (Count of Derived Classes)

Research: Chidamber & Kemerer - Number of Children (NOC)
Description: Number of immediate subclasses. (i.e. the number of classes one level down the inheritance tree from this class)[1].

NOC metric measures the number of direct subclass of a class. Since more children in a class have more responsibility, thus it is harder to modify the class and requires more testing. So NOC with less value is better and more NOC may indicate a misuse of subclassing[2].

In our analysis, both project1 and project2 have 0 NOC.

If NOC grows it means reuse increases. On the other hand, as NOC increases, the amount of testing will also increase because more children in a class indicate more responsibility. So, NOC represents the effort required to test the class and reuse [2].

Project1 and project2 both requires fewer efforts for testing and have less reusability.

RFC (Count of All Methods)

Research: Chidamber & Kemerer - Response For a Class (RFC), Lorenz & Kidd - Number of Methods (NM)
Description: Number of methods, including inherited ones [1].
RFC is the number of methods that can be invoked in response to a message in a class.
Pressman [3] States, since RFC increases, the effort required for testing also increases because the test sequence grows. If RFC increases, the overall design complexity of the class increases and becomes hard to understand. On the other hand lower values indicate greater polymorphism. The value of RFC can be from 0 to 50 for a class[4].

Maximum RFC in Table A and Table B is 914 and 109 respectively.
In our analysis both the projects are having very high RFC , so we should reduce the RFC for better maintainability. Higher the number of RFC more difficult to test and maintain the class. Thus, complexity of both the project is high and it is hard to maintain. Project 2 is comparatively less complex than that of project 1.

Graphical representation of RFC from table A and table B.

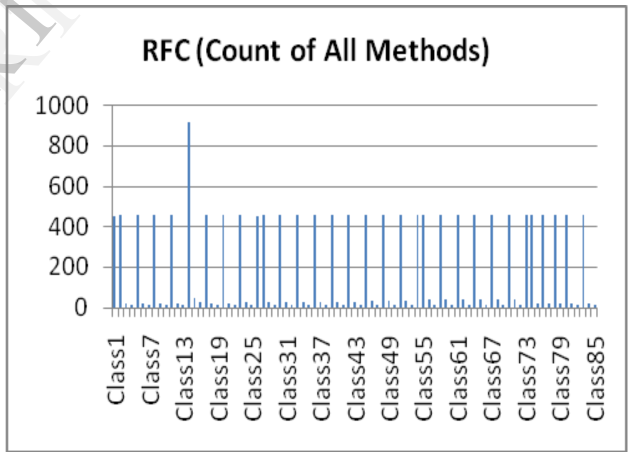

TABLE A

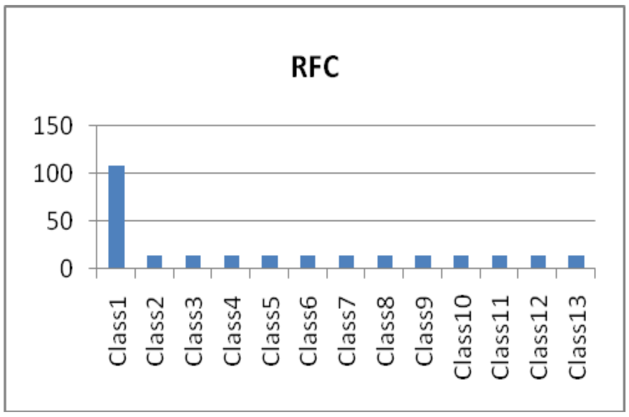

TABLE B

NIM (Count of Instance Methods)

*Research:*NIM
Description: Number of instance methods[1].

Maximum number of NIM in project 1 and project 2 is 25 and 5 respectively.

NIV (Count of Instance Variables)
*Research:*NIV
Description: Number of instance variables.

Maximum number of NIV for project 1 and project 2 is 18 and 22.

WMC (Count of Methods)
Research: Chidamber & Kemerer - Weighted Methods per Class (WMC)
Description: Number of local (not inherited) methods[1].

Low WMC indicates greater polymorphism in a class and high WMC indicates more complexity in the class [2].

In our analysis, maximum WNC in project 1 and project 2 is respectively 25 and 5. In project 2 all the classes excluding one has WNC 2 and in 25 classes has WNC 7,24 classes have WNC 0,and between 25 to 10 WNC is found in 10 classes.
Thus, we can conclude that project 2 has greater polymorphism then project 1. And project 2 is less complex then project 1. So, it is hard to maintain project 1 then project 2.

Graphical representation of WNC from table A and table B.

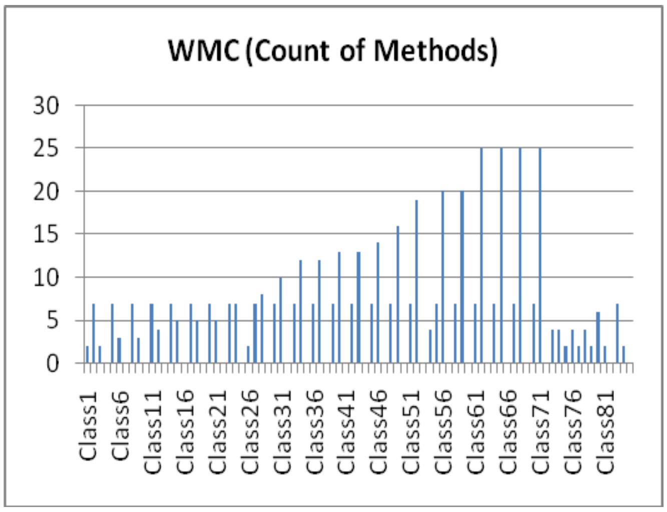

TABLE A

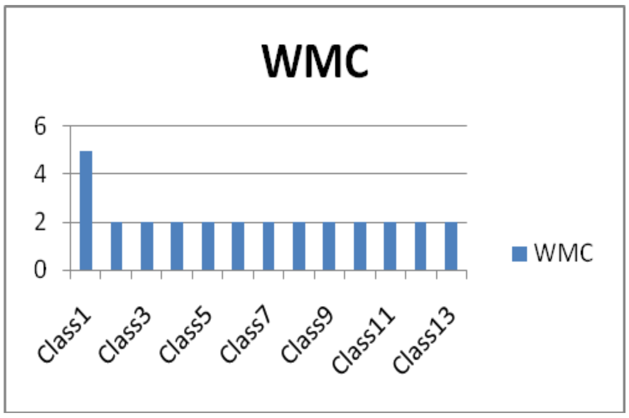

TABLE B

III. CONCLUSION

We have applied the set of matrices defined by CK and other matrices. By applying these matrices important issues of complexity, cohesion, dependability, polymorphism and reusability of any object oriented system can be judged. The reusability, complexity and maintainability of any OO system does not depend on the only the one of classes. A system having less number of classes can be less reusable and more complex than that of system containing more classes.

REFERENCES

1. www.scitools.com
2. An overview of Object Oriented Design Metrics, Department of Computer Science, Umeå University, Sweden
3. Software Engginering by **Roger Pressman**
4. http://www.refactorit.com/
5. www.adammikeal.org

AUTHORS

S. S. Dadhania has received her B.E. from Atmiya Institute of Technology and Science, Rajkot, Gujarat in 2009. And She has completed M.E. in Computer Science and Engg from Gujarat Technological University, Gujarat in 2012. She is working as a Lecturer at R C Technical Institute, Ahmedabad, Gujarat. Her current research interest includes Data Mining and Software Engineering.

A. S. Galathiya has received her B.E and MTech degrees in 2007 and 2012 in Computer Engineering from Dharmsinh Desai University, Nadiad, Gujarat, India. Her general research includes Data Mining and Software Engineering. She is Lecturer at R. C. Technical Institute of Computer department, Ahmedabad, Gujarat, India.