Comparison between Inheritance & Interface UML Design through the Coupling Metrics

Maya Yadav, Pradeep Baniya, Ganesh Wayal

Abstract: - In this paper we have applied various coupling metrics for measuring the comparison between object oriented class inheritance and interface. In this we have applied metrics on class design diagram and evaluate the metrics values. The coupling metrics presented identifies complexity between inheritance and interface programming In this paper we want to show which concept is good to use and beneficial for software developer.

Keywords: - measurement of metrics, object oriented technology, CBO, NOC, DIT, C K metrics.

I. INTRODUCTION

A key factor for any engineering discipline is measurement. Without measurement or metrics it is impossible to detect problems, defects and quality early. To improve software products and process, measurements are essential. So measurement is becoming very important in managing the software projects. Measurement is done by using metrics. The main objective of this paper is to measure the difference in class inheritance and interfaces in object oriented programming by finding the values of CK-metrics.

Object oriented programming provides generalized solutions for many problems, programming applications and also provides benefits like reusability and decomposition of problems into small easily understandable objects. Object oriented programming helps to perform modifications in future and to do functional extensions in already developed systems [2]. Many object oriented metrics in the literature lack in theoretical proof and some have not been validated. The metrics that evaluate object oriented metrics are: classes, methods, inheritance, coupling and cohesion. Very few metrics are presented for object oriented interfaces. In this paper a measurement analysis has been done for measuring the values for object oriented classes and interfaces by calculating the values of CK-Metrics. Any metrics must be defined as a complete and well designed quality improvement paradigm (QIP). According to QIP CK-metrics are used to measure the betterment in performance in using inheritance and interfaces in object oriented programming [3] [4] [5].

A. Traditional Metrics

Traditional metrics are important to measure Non-objectoriented programming. Some metrics are very simple and some are more complicated A large number of object oriented metrics have been developed and also large number of tools exist to collect metrics from programs [6]

Table -1

Sr. NO	Metrics	Definition
1	*Source Lines of Code (SLOC)*	The SLOC metric measures the number of physical lines of active code, that is, no blank or commented lines code [7]. Counting the SLOC is one of the earliest and easiest approaches to measuring complexity. It is also the most criticized approach [8]. In general the higher the SLOC in a module the less understandable and maintainable the module is.
2	*McCabe Cyclomatic Complexity (CC)*	Cyclomatic complexity is a measure of a module control flow complexity based on graph theory [8]. Cyclomatic complexity of a module uses control structures to create a control flow matrix, which in turn is used to generate a connected graph. The graph represents the control paths through the module. The complexity of the graph is the complexity of the module [9],8].
3	Comment percentage(CP)	The CP metric is defined as the number of commented lines of code divided by the number of non-blank lines of code. Usually 20% indicates adequate commenting for C++ [10]. A high CP value facilitates in maintaining a system.
4	defect	Number of fault in specification, Design or implementation.
5	Number of procedure	Number of statement groups which can be compiled independently in the program

B. B. CK METRICS[Chidamber91; Chidamber94]

Chidamber and Kemerer proposed a first version of these metrics and later the definition of some of them was improved. Only three of the seven CK metrics are available for a UML class diagram (see Table 2).

Manuscript Received on May 23, 2012.

Maya Yadav, Department of Computer Science RKDF IST, Bhopal.
Pradeep Baniya, Department of Computer Science MITM, Indore.
Prof. Ganesh Wayal, Department Of Computer Science RKDF IST, Bhopal.

Table – 2

Sr. No	Metric	Definition
1	WMC	The Weighted Methods per Class is defined as follows: $WMC=\sum_{i=1} c_i$; Where $c_1, ..., c_n$ be the complexity of the methods of a class with methods M1, ...,Mn. If all method complexities are considered to be unity, the WMC = n, the number of methods7.
2	DIT	The Depth of Inheritance of a class is the DIT metric for a class. In cases involving multiple inheritances, the DIT will be the maximum length from the node to the root of the tree.
3	NOC	The Number of Children is the number of immediate subclasses subordinated to a class in the class hierarchy.
4	CBO	Classes are coupled if methods or instance variables in one class are used by the other. CBO for a class is number of other classes coupled with it.
5	RFC	Count of all methods in the class plus all methods called in other classes.
6	Number of Dependencies IN ***(NDepIN)***	The Number of Dependencies IN metric is defined as the number of classes that depend on a given class. When the dependencies are reduced the class can function more independently. This metrics was introduced by Brian.
7	Number of Dependencies Out ***(NDepOut)***	The Number of Dependencies Out metric is defined as the number of classes on which a given class depends. When the metric value is less the class can function independently. This metric was introduced by Brian.

II. OBJECTORIENTED INHERITENCE INTERFACES

Inheritance is one of the initial features of object oriented programming. Through inheritance a derived class receives the attributes and methods of the base class. The relationship between derived and base class is referred as "is-a/is-a-kind-of". Inheritance feature creates a class hierarchy. [17]

Nowadays interfaces are heavily used in all disciplines especially in object oriented programming [11]. With interface construct, object oriented programming features a good concept with high potential code reusability. Interfaces are used to organize code and provide a solid boundary between the different levels of abstraction [12].

It is good to use interfaces in large type of applications because interfaces make the software/program easier to extend, modify and integrate new features.

An interface is a prototype for class. With the construct of an interface java allows a concept of high potential for producing a reusable code. Interfaces in object oriented programming just contain names and signatures of methods and attributes, but no method implementations. Interfaces are implemented by classes. The inheritance hierarchy of interfaces is independent than that of class inheritance tree. Therefore object oriented languages like java gives higher potential to produce reusable code than abstract classes [13] [16]. Interfaces are prototype for a class. Interfaces can be used like classes in declarations and signatures. With interface construct, object oriented programming features a good concept with high potential code reusability. Today interfaces are heavily used in all disciplines. Interfaces are advisable to be used in large type of applications because the interfaces make the application easier to extend, modify and integrate new features [29][30]. Interfaces in object oriented programming just contain names and signatures of methods and attributes, but no method implementations [13]. Interfaces are used to organize code and provide a solid boundary between the different levels of abstraction [12] [18].

III. RELATED WORK

The concept of interfaces has been measured in java programming by Fried Stiemann and Co [16] whorepresented the usage of interfaces compared to classes as 4:1. Ken Pugh [17] said that finding commonality among classes makes more it effective for object oriented programming. He explored the commonality in using inheritance and interfaces in object oriented programming. In this paper, the usage of interfaces is increased and the benefits of using interfaces are shown by coupling measures.

IV. PROPOSED APPROACH

Goal: Comparing the inheritance and interface concepts in object oriented programming through metrics.

Hypothesis: Six object oriented metrics are used for coupling measures in object oriented inheritance class diagram and interface diagram.

1. Two object oriented class diagrams are used with inheritance concept in this paper.
2. These diagrams are introduced with maximum possible interfaces.
3. All the four metrics are applied to both inheritance and interface diagrams.
4. The results are compared between inheritance and interface coupling measures.

The metrics discussed above are applied for both inheritance and interface UML diagrams. The results are discussed.

The first study considered is a vehicle classification class inheritance diagram which is represented in Diagram1

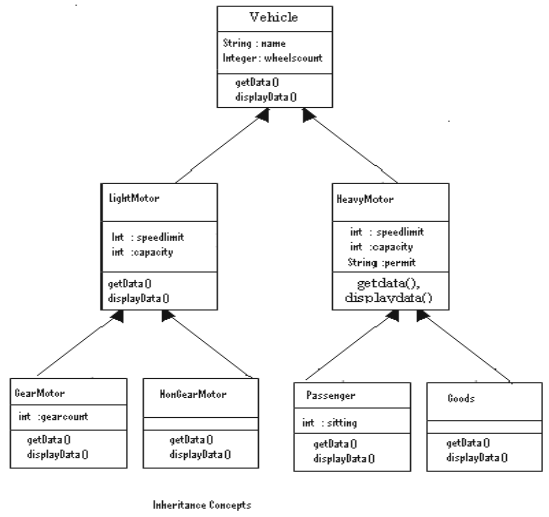


Diagram 1: Vehicle Classification using Class Inheritance - ADOPTED FROM

The above said class inheritance Diagram 1 is introduced with possible number of interfaces and is represented in Diagram 2.

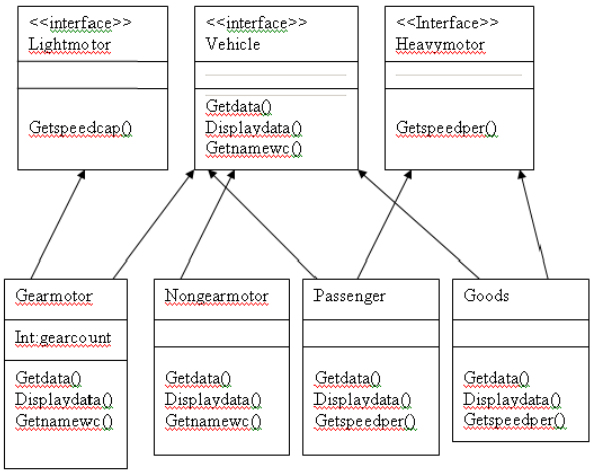


Diagram2:Vehicle classification using interfaces

For the above said two diagrams the coupling metricsare measured and tabulated in Table 3. By comparing the table values for both the diagrams the interface values are reduced for almost all metrics For the above said two diagrams the coupling metrics are measured and tabulated in Table 3. By comparing the table values for both the diagrams the interface values are reduced for almost all metrics.

Table 3: Coupling Measures for Diagrams 1 & 2

	Metrics/Class	CBO	NASSocC	NDepIN	NDepOut	NOC	DIT
Diagram 1	**Vehicle**	2	2	2	0	2	0
	Light Motor	3	3	2	1	2	1
	HeaveyMotor	3	3	2	1	2	1
	Gearmoter	1	1	0	1	0	2
	Nongearmoter	1	1	0	1	0	2
	passenger	1	1	0	1	0	2
	Goods	1	1	0	1	0	2
Diagram 2	**Vehicle**	0	0	0	0	0	0
	Light Motor	0	0	0	0	0	0
	HeaveyMotor	0	0	0	0	0	0
	Gearmoter	0	0	0	0	0	0
	Nongearmoter	0	0	0	0	0	0
	passenger	0	0	0	0	0	0
	Goods	0	0	0	0	0	0

The second diagram chosen is shapes hierarchy and is given in Diagram 3

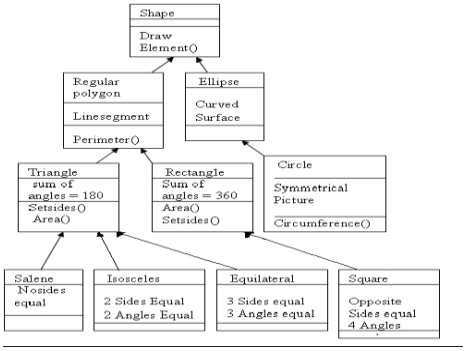


Diagram 3: Class Inheritance for shapes

The above class inheritance diagram is converted into interface concept diagram and is represented as Diagram 4.

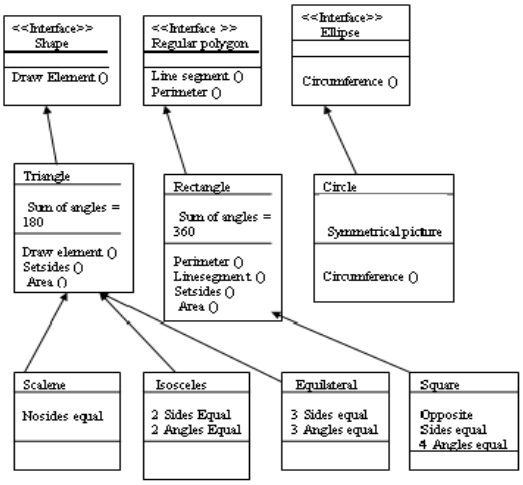


Diagram 4: Diagram for shapes using Interfaces

For Diagram 3 and Diagram 4 the above said metric values are measured and tabulated in Table 4.

Table- 4

	Metrics/ Class	CBO	NASSocC	NDepIN	NDepOUT	NOC	DIT
Diagram 3	**Shape**	2	2	2	0	2	0
	Regular Polygon	3	3	2	1	2	1
	Ellipse	2	2	1	1	1	1
	Triangle	4	4	3	1	3	2
	Rectangle	2	2	1	1	1	2
	Circle	1	1	1	0	0	2
	Scalene	1	1	1	0	0	3
	Isosceles	1	1	1	0	0	3
	Equilateral	1	1	1	0	0	3
	Square	1	1	1	0	0	3
Diagram 4	**Shape**	0	0	0	0	0	0
	Regular Polygon	0	0	0	0	0	0
	Ellipse	0	0	0	0	0	0
	Triangle	3	3	3	0	3	0
	Rectangle	1	1	1	0	1	0
	Circle	0	0	0	0	0	0
	Scalene	1	1	0	1	0	1

	Isosceles	**1**	**1**	**0**	**1**	**0**	**1**
	Equilateral	**1**	**1**	**0**	**1**	**0**	**1**

In table 4 , the values for most of the metrics are reduced for class diagram using interfaces compared to class inheritance diagram. When the interface usage is increased the dependencies of classes are reduced and classes can function more independently. When inheritance concepts are more, classes will become more dependent and coupling between objects are more.

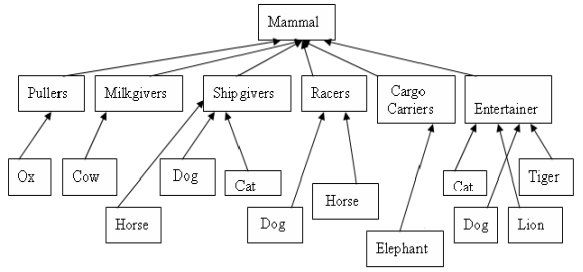


Diagram 5: Class Inheritance for animals

The third example chosen is animal hierarchy and is shown in diagram 5. The above diagram 5 is introduced
with maximum possible interfaces and is represented in diagram 6

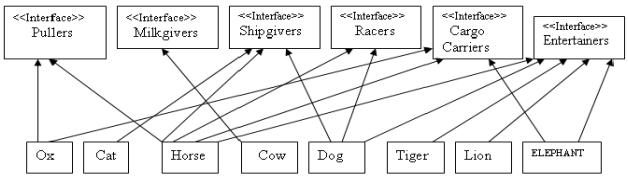


Diagram 6: Animal hierarchy using interfaces

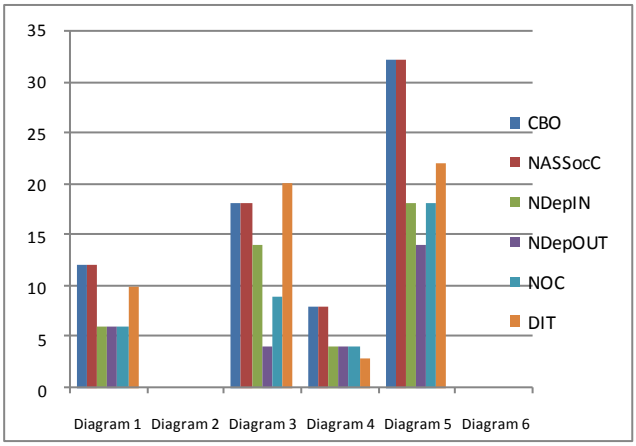
Table 5: Coupling Measures for diagrams 4&5.

	Metrics/ Class	CBO	NASSocC	NDepIN	NDepOUT	NOC	DIT
Diagram 5	Mammal	6	6	6	0	6	0
	Pullers	2	2	1	1	1	1
	Milkgiver	2	2	1	1	1	1
	Shipgiver	4	4	3	1	3	1
	Racers	3	3	2	1	2	1
	Cargocarriers	2	2	1	1	1	1
	Entertainer	5	5	4	1	4	1
	Ox	1	1	0	1	0	2
	Cat	1	1	0	1	0	2
	Horse	1	1	0	1	0	2
	Cow	1	1	0	1	0	2
	Dog	1	1	0	1	0	2
	Tiger	1	1	0	1	0	2
	Lion	1	1	0	1	0	2
	Elephant	1	1	0	1	0	2
Diagram 6	Pullers	0	0	0	0	0	0
	Milkgiver	0	0	0	0	0	0
	Shipgiver	0	0	0	0	0	0
	Racers	0	0	0	0	0	0
	Cargocarriers	0	0	0	0	0	0
	Entertainer	0	0	0	0	0	0
	Ox	0	0	0	0	0	0
	Cat	0	0	0	0	0	0
	Horse	0	0	0	0	0	0
	Cow	0	0	0	0	0	0
	Dog	0	0	0	0	0	0
	Tiger	0	0	0	0	0	0
	Lion	0	0	0	0	0	0
	Elephant	0	0	0	0	0	0

In table 5, the values for all metrics are reduced for the diagram using interfaces compared to class inheritance diagram. In table 2, 3 and 4 the results are specified by measuring the CBO,NOC,DIT and other coupling metrics by using the coupling measures discussed above. All coupling measures are reduced for interface diagrams in all the three examples compared to class inheritance approach. Due to the reduction of CBO faults can be reduced. All other coupling metrics are also reduced and is used to measure the stability of the diagram. Excessive coupling indicates weakness in class encapsulation and it will restrict reusability [19].By comparing the table values from table 2, 3 and 4 for the above diagrams the total coupling measures for each metric for all class diagrams are tabulated in table 6 .

Table 6: Coupling measures for total Inheritance and Interface diagrams

Metrics/ Diagram Name	CBO	NASSocC	NDepIN	NDepOUT	NOC	DIT
Diagram 1	12	12	6	6	6	10
Diagram 2	0	0	0	0	0	0
Diagram 3	18	18	14	4	9	20
Diagram 4	8	8	4	4	4	3
Diagram 5	32	32	18	14	18	22
Diagram 6	0	0	0	0	0	0



Graph 1: Coupling Measures Comparison

The graph 1 shows the difference in reduction of coupling values.

V.CONCLUSION

This paper presents an idea on how to reduce coupling in object oriented programming. Due to the reduction in coupling, developers can produce quality programs. When Coupling is reduced reusability will be increased. High coupling will support low encapsulation and produce more faults. Due to the reduction in values of coupling metrics the stability of the structure will be good. When the coupling measures are reduced, the classes can function more independently. The more independent a class it is easier to be reused by another application. To improve modularity

and encapsulation the inter object class coupling measures should be minimum. As the number of couples becomes

larger the maintenance is more difficult. By using more interfaces compared to inheritance the coupling measures

are reduced.

REFERENCES

1. Krishnapriya, V., Dr. Ramar, K.: Exploring the Difference between Object Oriented Class Inheritance and Interfaces Using Coupling Measures. International Conference on Advances in Computer Engineering, 978-0-7695-4058-0/10, IEEE (2010)
2. Rene Santaolaya Salgado, Olivia G. Fragosco Diaz, Manuel A. Valdes Marrero, Issac M. Vaseuqz Mendz and Shiela L. delfin Lara,"Object Oriented Metric to Measure the Degree of Dependency Due to Unused Interfaces", ICCSA 2004, LNCS 3046, P.No:808-817,2004 @ Springer, Verlag Berlin HeidelBerg.
3. El Hachemi Alikacem, Houari A. Sahraoui, "Generic Metric Extraction Framework",IWSM/Metrickon, Software Measurement Conference 2006.
4. Stephen R. Schach, "Object Oriented and Classical Engineering", 5th Edition,Tata McGraw Hill,2002.
5. Carlo Ghezzi, Mehdi Jazayeri, Dino Manddrioli,"Fundamentals of software Engineering, P.No: 366, 2nd Edition, Prentice Hall India, 2003
6. Christopher L. Brooks, Chrislopher G. Buell, "A Tool for Automatically Gathering Object-Oriented Metrics", IEEE, 1994.
7. Lorenz, Mark & Kidd Jeff, Object-Oriented Software Metrics, Prentice Hall, 1994.
8. Tegarden, D., Sheetz, S., Monarchi, D., "Effectiveness of Traditional Software Metrics for Object-Oriented Systems", Proceedings: 25th Hawaii International Conference on System Sciences, January, 1992, pp. 359-368.
9. McCabe and Associates, Using McCabe QA 7.0, 1999, 9861 Broken Land Parkway 4th Floor Columbia, MD 21046.
10. Rosenberg, L., and Hyatt, L., "Software Quality Metrics for Object-Oriented System Environments", Software Assurance Technology Center, Technical Report SATC-TR-95-1001, NASA Goddard Space Flight Center, Greenbelt, Maryland 20771.
11. FriedRich Steimann, Philip Mayer, Andreas MeiBner,"Decoupling Classes with Inferred Interfaces ", Proceedings of 2006 ACM, Symposium on Applied Computing, Pg.No:1404-1408.
12. Matthew Cochran,"Coding Better: Using Classes Vs. Interfaces", January 18th, 2009.
13. Markus Mohenen, "Interfaces with Default Implementations in Java", Aachen University of Technology.
14. Khan R.A., K.Mustafa And S.A.Ahson, "Software
15. Quality - Concepts And Practices", P.No:140.
16. Fried Stiemann, Wolf Siberski and Thomas Kuhne, " Towards the Systematic Use of Interfaces in Java Programming", 2nd Int. Conf. on The Principles and practice of Programming in Java PPJ 2003, P.No 13-17
17. Ken Pugh," Interface Oriented Design", Chapter 5, 2005
18. Dirk Riehle and Erica Dubach,"Working With Java Interfaces and Classes-How to Separate Interfaces from Implementations", P.No:35-46, Published in Java Report 4, 1999.
19. Pradeep Kumar Bhatia, Rajbeer Mann, " An Approach to Measure Software Reusability of OO Design", Proceedings of 2nd International Conference on Challenges & Opportunities in Information Technology(COIT-2008),RIMT-IET,Mandi Gobindgarh, March 29,2008
20. Rajib Mall,"Fundamentals of Software Engineering", Chapter 1, Pg.No:1-18,2nd Edition, April 2004.
21. Rene Santaolaya Salgado, Olivia G. Fragosco Diaz, Manuel A. Valdes Marrero, Issac M. Vaseuqz Mendez and Shiela L. Delfin Lara, "Object Oriented Metric to Measure the Degree of Dependency Due to Unused Interfaces", ICCSA 2004, LNCS 3046, P.No: 808-817,2004 @ Springer, Verlag Berlin Heidelberg.
22. Rudiger Lincke, Jonas Lundberg and Welf Lowe,"Comparing Software Metrics Tools", ISSTA'08, July 20-24, 2008, ACM 978-1-59593-904-3/07.
    Santonu Sarkar, Member, IEEE, Avinash C. Kak, and Girish Maskeri Rama," Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software, IEEE Transactions on Software Engineering, Vol. 34, No. 5, Sep-Oct 2008.
23. Shyam R. Chidamber, Chris F. Kemrer,"A Metrics Suite for Object Oriented Design. M.I.T., Sloan School Of Management, 1993.
    Terry .C. and Dikel .D.,"Reuse Library Standards Aid Users in Setting up Organizational Reuse Programs",Embedded System Programming Product News,1996.