# Comparison of Class Inheritance and Interface Usage in Object Oriented Programming through Complexity Measures

V. Krishnapriya [1] and Dr. K. Ramar [2]

[1] Head, Dept of Computer Science, Sri Ramakrishna College of Arts & Science for Women, Coimbatore, Tamilnadu. .
[2] Principal, Sri Vidya College of Engineering and Technology, Virudhunagar, Tamilnadu.

## ABSTRACT

*It is widely acknowledged that in software engineering, the usage of metrics at the initial phases of the object oriented software can help designers to make better decisions. The quality of class diagrams could be a major determinant for the quality of the software product that is finally delivered. Quantitative measurements are useful to assess class diagram quality. Following this innovative thinking, two UML class diagrams are taken to measure the complexity and size. A set of metrics of complexity measures are used to measure the class diagrams. Seven known complexity measures are evaluated and compared for inheritance and interface usage in object oriented programming. Two UML class diagrams are introduced with possible interfaces and measured the complexity metrics and a comparison has been made between the class inheritance and class interface usage through complexity measurements.*

## KEYWORDS

*Class diagrams, Interface diagrams, Object oriented metrics, UML, Complexity.*

## 1. INTRODUCTION

Software engineering metrics are important measurements for project planning and project measurements. The increasing importance of software measurement and metrics led to the development of new software measures and metrics. Many metrics have been proposed for traditional programming and object oriented programming.

"Software quality is the degree to which software possesses a desired combination of attributes such as maintainability, testability, reusability, complexity, reliability, interoperability etc" – IEEE 1992.

The increased demand for the software quality has resulted in higher quality software and nowadays quality is the main differentiator between the software products. Due to this reason the software designers and developers need valid measures for the evaluation, improvement and validation of product quality from initial stages. The early focus on class diagrams quality helps the software engineers and developers to build better software without doing unnecessary revisions at later stages of development. Revisions or changes at later stages will lead to increase in expenditure and be more complex to perform. Nowadays software measurement plays an important role for measuring quality and complexity of software. The early availability of software metrics for UML diagrams were used for quality and complexity evaluation. [8][10][11]

### 1.1. Measurement and Metrics

Nowadays, software engineering plays a most important technology in the world. As computer software has grown, the software developers have continually attempted to develop new technologies. In these newly developed technologies some of them focused on object oriented technologies [13]. In this paper object oriented class inheritances are differentiated with object oriented interface class diagrams through complexity measures.

"If you cannot measure it's not Engineering Community" is often said by the engineering community. [6]

The key factor for any engineering discipline is measurement. Without measurement or metrics it is impossible to measure quality and complexity to detect problems before it is released. So measurement is very important in managing the software projects. [2][12][14][15]

Metrics are used as a powerful tool in software research, maintenance and development for estimating cost, effort, complexity, quality, maintenance and to control etc[5]. Metrics serves as an early warning tool for potential problems happening in software development [14]. Any metrics must be defined as a complete and well designed quality improvement paradigm (QIP) [4].

## 2. BACKGROUND

The concept of an interface in object oriented programming is quite old. Software engineering has been using interfaces for more than 25 years. Software measurement activities were not addressed to most of their requirements for providing information and to support for managerial decision making [12]. Many metrics are available to measure class, method, inheritance, polymorphism and system level. There is no significant work on the design of human computer interfaces. In literature, relatively little information has been published on interface metrics. Those metrics provide only little information about the quality and usability of the interfaces.

Finding difference in classes makes it more effective for object oriented programming. The difference in using an inheritance and interfaces in class diagrams are measured. These measures are done by using structural complexity metrics.

### 2.1 RELATED WORK

The concept of interfaces has been measured in java programming by Fried Stiemann and Co [7]. He represented that the usage of interfaces compared to classes are 4:1.

Ken Pugh [10] stated that finding commonality among classes makes it more effective for object oriented programming and he also explored the commonality in using inheritance and using interfaces in object oriented programming.

The novel idea in this paper is finding the difference in using class inheritance and interface through structural complexity metric measures.

Measuring complexity of software products was and still is a widely scattered research project.

"A lower software structural complexity could lead to a greater software reliability" – Fenton and Pfleeger, 1997.

The structural complexity measure is the most important measurement to evaluate the quality of UML class diagrams. [3]

It is well known that software structural complexity metrics are very useful to evaluate the different characteristics that affect the quality of object oriented software. In literature there are several measures of complexity. With the above said idea in mind a set of seven different

metrics are taken to measure the structural complexity of object oriented UML diagrams to find the difference in using class inheritance and interface concepts in object oriented programming.

## 2.2 METRICS USED

No single metric is available to measure the complexity of software [7]. The metrics discussed below are used to measure the complexity of UML diagrams [9].

### 2.2.1 Number of Aggregation - NAgg

The number of aggregation metric is defined as the total number of aggregation relationships within the class diagram.

### 2.2.2 Number of Dependencies – NDep

It is defined as total number of dependency relationships with in the class diagram. Dependency is a weaker form of relationship which indicates that one class depends on another class because it uses it at some point of time [10][11].

#### 2.2.3 Number of Generalisations – NGen

The number of generalisation metrics is defined as the total number of generalisation relationships with in a class diagram. Generalisation is a relationship between two classes [11].

General/super class
Special/subclass

#### 2.2.4 Number of Generalisation Hierarchies – NGenH

The number of generalisation hierarchy metric is defined as the total number of generalisation hierarchies with in the class diagram. A generalisation hierarchy is a structural grouping of entities that share common attributes. Each instance of super type entity must appear in at least one subtype. An instance of the subtype must appear in subtype [11].

#### 2.2.5 Number of Aggregation Hierarchies – NAggH

The number of aggregation hierarchy metric is defined as the total number of aggregation hierarchies with in a class diagram.

### 2.2.6 Maximum Depth of Inheritance Tree – MaxDIT

Depth of a class with in the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree or the length of the longest path from the class to the root of the hierarchy. This is measured by the number of ancestor classes.

### 2.2.7 Maximum Hierarchy Aggregation – MaxHAgg

The maximum hierarchy aggregation metric is defined as the maximum between the hierarchy aggregation value for each class of the class diagram. The hierarchy aggregation value for a class with in the aggregation hierarchy is the length of the longest path from the class to the leaves.

# 3. GOAL AND RESEARCH HYPOTHESES

Two Examples and two hypotheses are used to achieve the goal.

Goal: Exploring the difference in using Class inheritance and interface measures using complexity measures.

Hypotheses 1: A set of complexity metrics are taken to measure the complexity of two concepts.

Hypothesis 2: length is considered as complexity measure and is measured for both examples.

### 3.1 METHODOLOGY

Software complexity is measured in two ways.

(1). Software complexity is calculated by measuring the above said metrics

i. Two Class inheritance diagrams are taken and are measured using the above said seven structural complexity measures.
ii. The Two class diagrams are introduced with maximum number of possible interfaces and the complexity measurement metrics are measured.
iii. The results are compared for class inheritance and class interface diagrams.
iv. Length is defined as the number of lines of code. [1]

(2). Complexity is calculated by using length also.

$$e(p) = l(p) * c(p) \quad \text{---------- I}$$

Where e (p) is the total complexity, l (p) is the length of the software and c (p) is the average complexity.

## 4. APPLYING METRICS TO UML DIAGRAMS

Two UML class inheritance diagrams are taken and all the above said metrics are applied to measure complexity. The two diagrams are introduced with maximum possibility of interfaces and metrics which are used to measure the complexity. Both inheritance and interface diagrams complexity measures are compared. First UML class diagram has been taken as vehicle classification.

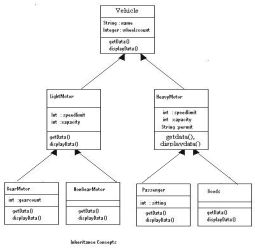



Figure 1: Vehicle Classification with Class Inheritance

The above figure 1 vehicle classification diagram is introduced with maximum possible interfaces and is shown in figure 2.

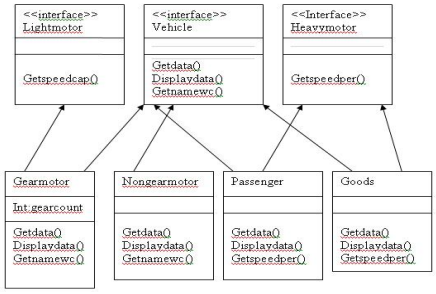


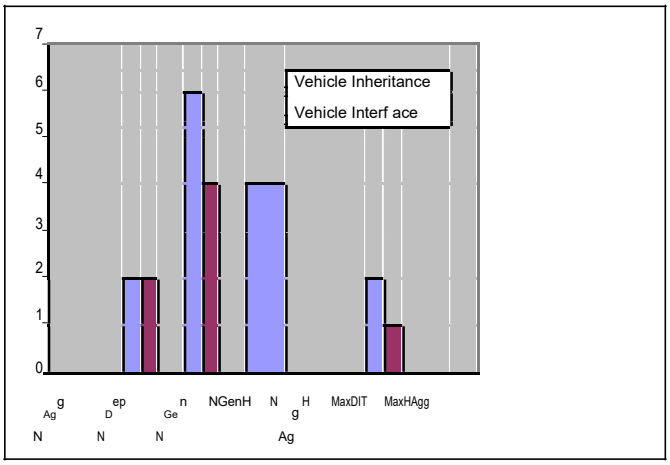

Figure 2: Vehicle Classification with Interface Diagram

The above said measurement metrics are applied for class inheritance and class interface diagrams. The table 1 shows the measurement values for the above said metrics.

Table1: Complexity Measurement for Vehicle Classification

| Diagram/ Metric | Length | NAgg | NDep | NGen | NGenH | NAggH | Max DIT | Max HAgg | Avg. Complexity | Total complexity |
|---|---|---|---|---|---|---|---|---|---|---|
| Vehicle Inheritance | 125 | 0 | 2 | 6 | 4 | 0 | 2 | 0 | 2.0 | 250 |
| Vehicle Interface | 93 | 0 | 2 | 4 | 2 | 0 | 1 | 0 | 1.29 | 119.97 |

The average complexity is calculated by finding the mean for complexity metrics.




Graph1. Comparison of Metrics for Vehicle Classification

The second diagram referred is types of shapes which are shown in figure 3.

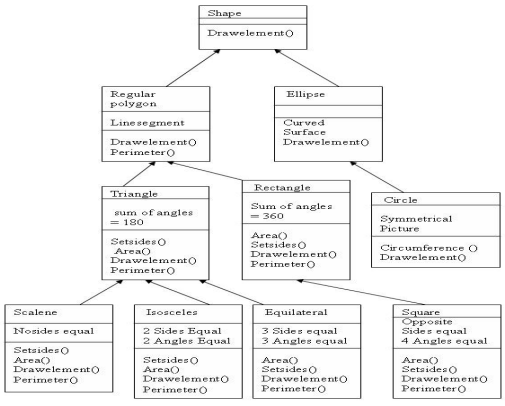



Figure 3: Types of Shapes Using Class Inheritance The above diagram is introduced with possible number of interfaces and the diagram is shown in figure 4.

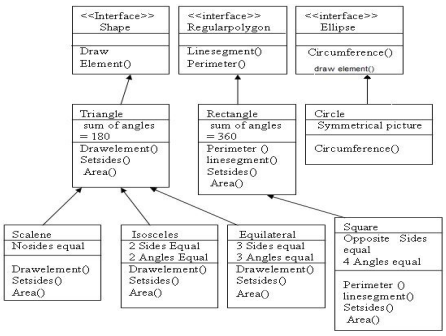



Figure 4: Types of Shapes with Interfaces

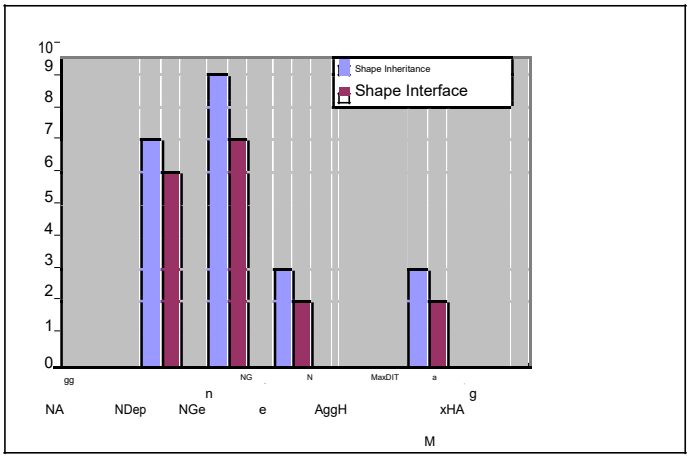
For the above said two figures 3 and 4 the complexity are measured through the above said seven metrics. The resulted values are tabulated in table 2.

Table 2: Complexity Measurement for Shapes Classification

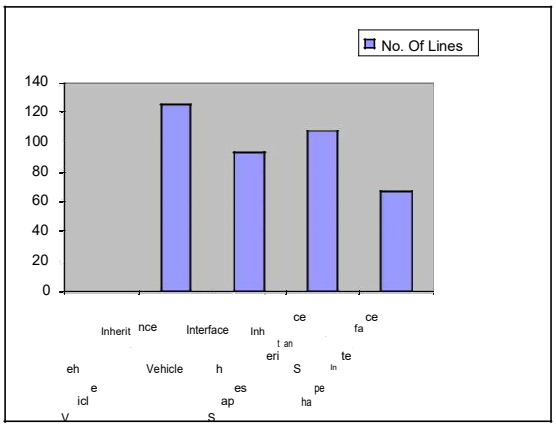
| Diagram/ Metric | Length | NAgg | NDep | NGen | NGenH | NAggH | Max DIT | Max HAgg | Avg. Complexity | Total complexity |
|---|---|---|---|---|---|---|---|---|---|---|
| Shape Inheritance | 108 | 0 | 7 | 9 | 3 | 0 | 3 | 0 | 3.14 | 339.12 |
| Shape Interface | 67 | 0 | 6 | 7 | 2 | 0 | 2 | 0 | 2.43 | 162.81 |

Total complexity represented in two tables is calculated by using the above said formula I. For table 1 and 2 graphs have been drawn to show the improvement in using interface concepts. Graph 2 depicts the difference in improvement in structural metrics for the second example.

Graph 3 shows the difference in concepts using length for two examples.

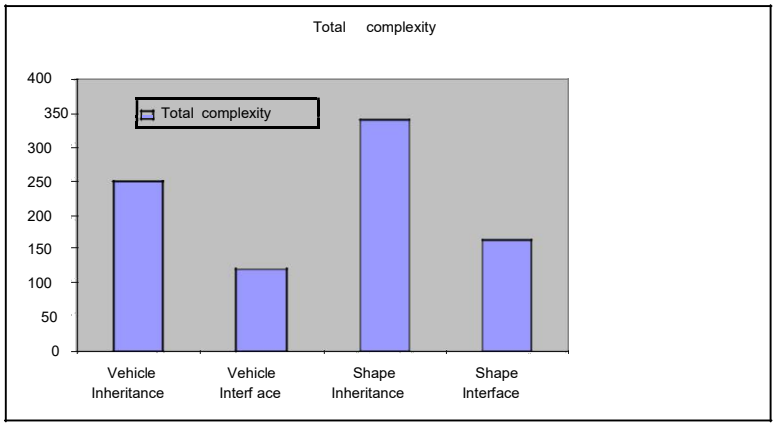



Graph 2: Comparison of Metrics for Shapes Hierarchy




Graph 3: Inheritance Vs Interface Concepts Using length

The number of lines is measured for the above said two examples. The length is reduced for the concept of interfaces compared to inheritance concept. Introduction of interfaces in object oriented programming in possible places is better for producing good quality and high reliable software.




Graph 4: Inheritance Vs Interface using Total Complexity

## 5. CONCLUSION

The structural complexity is measured between the usage of class inheritance and interfaces in object oriented programming. In this paper, a set of seven structural metrics are used to measure UML class diagram structural complexity with respect to the usage of UML relationships such as aggregations, associations, dependencies and generalisations. The average and the total complexity values are reduced for both examples of object oriented interfaces compared to object oriented class inheritance concepts. Interface concept has shown better performance compared to inheritance concept in object oriented programming. Software reliability will increase with lower software complexity.

## REFERENCES

[1.] Adrain Costea," On Measuring Software Complexity", Journal of Applied Quantitative Methods, vol.2,no.1 , Spring 2007.

[2.] Agarwal K.K.,Yogesh Singh, Arvinder Kaur, Ruchika Malhotra,"Emprical Study of Object-Oriented Metrics", Journal of Object Technology, Vol. 5, Nov-Dec 2006.

[3.] Baowen Xu, Dazhou Kang and Jianjiang,"A Structural Complexity Measure for UML Class Diagrams", vol.3036, P.No:421-424, May 2004.

[4.] Carlo Ghezzi, Mehdi Jazayeri, Dino Manddrioli,"Fundamentals of software Engineering, P.No: 366, 2nd Edition, Prentice Hall India, 2003.

[5.] El Hachemi Alikacem, Houari A. Sahraoui, "Generic Metric Extraction Framework",IWSM/Metrickon, Software Measurement Conference 2006.

[6.] Ivar Jacobson, Magnus Christerson, Patrick Johnson, Gunnar OverGarrd,"Object Oriented Software Engineering-A Use Case Driven Approach", P.NO:468, Pearson Education @ 2001.

[7.] Jorge Cardoso,"Control-flow Complexity Measurement of Process and Weyuker's Properties", World Academy of Science Engineering and Technology, Aug 2005.

[8.] Manso M., Genero M. and Piattini M.,"No-Redundant Metrics for UML Class Diagram Structural Complexity", Advanced System Engineering, LNCS 2681, P.No: 127-142, Springer 2003.

[9.] Marcela Genero,Mario Piattini "Empirical validation of measures for class diagram structural complexity through controlled experiments", Proceedings of the 2002 International Symposium on Empirical Software Engineering.

[10.] Marcela Genero, Mario Piattini and Coral Calero, "Empirical Validation of Class Diagram Metrics", Proceedings of the 2002 International Symposium on Emprical Software Engineering (ISESE'02) @ 2002 IEEE.

[11.] Marcela Genero, Mario Piattini and Coral Calero, "A Survey of Metrics for UML Diagrams", Journal of Object Technology, P.No: 55-92, Vol. 4, No. 9, Nov-Dec 2005.

[12.] Norman E. Fenton, Shari Lawrence Pfleeger,"Software Metrics – A Rigorous & Practical Approach", 2nd Edition.

[13.] Roger S. Pressman,"Software Engineering a Practitioner's Approach", 6th Edition.

[14.] Stephen R. Schach, "Object Oriented and Classical Engineering", 5th Edition,Tata McGraw Hill,2002.

[15.] Watts S. Humphery,"A discipline for Software Engineering, SEI Series in Software Engineering, P.No:209-210, Pearson Education Asia, 2001.

## Authors

1. V. Krishnapriya M.C.A., M.Phil.,

She is currently Head, Department of computer Science at Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamilnadu and pursuing her Ph.D Mother Teresa Women's University, Kodaikanal. She has 13 years of teaching experience and presented more than 9 papers in National and International Conferences and produced 3 M.Phils so far.

2. K.Ramar B.E, M.E, Ph.D

Received the Ph.D Degree in Computer Science from Manonmaniam Sundaranar university, Tirunelveli and prior degrees from PSG College of Technology, Coimbatore and Govt College of Engineering, Tirunelveli. He is currently Principal, Sri Vidya College of Engineering and Technology, Virudhunagar**,** Tamilnadu. He is life member in the CSI-Mumbai, ISTE-NewDelhi, SSI-Trivandrum and Fellow in Institution of Engineers, Kolkatta. He has published 10 articles in National and International journals and presented papers in more than 40 conferences. He has produced 5 Ph.D and 15 M.Phils so far.