International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-5, June 2019

Interpretation of Inheritance with Object Oriented Metrics in context of Software Complexity

Mohit Kumar Sharma, Amardeep Gupta

Abstract: ***Reusability is an important characteristic in Object Oriented Software Development, that involves requirement specification, analysis, design, coding, testing and implementation of existing software to rebuilt in same procedure to create new updated software application with reliable cost and on-time. It is systematic procedure of creation of software applications from existing to enable more powerful features to get productivity, quality and effective performance based on present need of hour. Inheritance is an important concept of reusability that is used in Object Oriented programming for creating new classes from existing classes. Object Oriented Software applications are benefited with inheritance based features for real world models. It has a capacity to solve complexity and make it reliable with inheritance metrics. Inheritance metrics are part of Object Oriented metrics given by different researchers from time to time for measurement. Inheritance metrics can be applied to interpret source code as a quality indicator. In software complexity, design or implementation is difficult to execute, understand and test. Interpretation has been done with different inheritances on Object Oriented Inheritance metrics that evaluated different computation values to explore, solve and reduce software complexity.***

Index Terms: C-K metrics, Complexity, Inheritance hierarchy, L-K metrics, MOOD

I. INTRODUCTION

Reusability is systematic process of development of Object Oriented software applications from existing code to enable more powerful features to get productivity, quality and effective performance based on present need of hour [1]. Software developers can include additional characteristics in the existing software and develop new enhanced version of software [2]. It helps to save time, development cost, reduce complexity and improves reliability of software in market for the users [3]. Inheritance is a powerful characteristic of Object Oriented programming that makes software reusable. Inheritance allows an object to take on the characteristics and functions of different objects [4]. Software developers connect objects by combining them together in different classes and by combining classes into hierarchies. It provides additional features to existing software applications as need of user requirement [5]. Inheritance avoids writing the same code again and again. It is procedure to create a new class from existing class. The existing class is a base class and new created class is a derived class. The derive class inherits some or all the features from the base class and add new features in the software. Object Oriented software applications are benefited with inheritance based feature for real world models [6]. It has a capacity to solve complexity and make it reliable with the help of inheritance metrics. Inheritance metrics are part of Object Oriented metrics given by different researchers from time to time for measuring performance [7]. Object Oriented Complexity is explained as a software has a design or implementation that is difficult to run, understand and check [8]. Complexity makes software processing speed slow and requires more space for execution [9]. Object Oriented programming divides whole software in different modules, each module has its own specific target to run and solve a specific problem. Due to competition, IT companies are producing software applications with minimum cost, with maximum features, timely availability and by increasing software quality based on inheritance concepts to reach these targets as-

A. Software Quality

Inheritance ensures enhanced features in existing software to enrich software quality for the users. The quality level of developed software is benefited by the usage of inheritance, because the code used have already been re-tested and ensures accuracy with reliability [10].

B. Productivity

Productivity is increased and a time to develop software reduced, which helps easily availability in market for users in competitive market.

C. Risk Reduction

Risk management is done by software developers to secure market value of software for guaranteed run and access.

D.Availability in Market

Software user wants software applications available as soon as possible for their use. Reusability ensures with minimum development cost and within time availability for the users.

E. Effective Performance

Inheritance adds advanced features to existing software and reliable performance by effective use of resources as fast speed, minimum space to execute and resource allocation [11]. It allows an improved functionality and effective performance of real world models for the users. It has a capacity to solve complexity.

Manuscript published on 30 June 2019.
* Correspondence Author (s)
Mohit Kumar Sharma*, Research Scholar, Faculty of Computational Science, GNA University, Phagwara, Punjab, and Assistant Professor, Computer Science, J.C. D.A.V. College, Dasuya, Punjab, India
Dr. Amardeep Gupta, Principal, J.C. D.A.V. College, Dasuya, Punjab, India

II. LITERATURE REVIEW

Object Oriented Metrics concentrate on measurement that can be applied to class and a design characteristic as Localization, Encapsulation, Inheritance, Information hiding, Polymorphism, Messaging and Object abstraction [12]. Chidamber, S.R., et al. suggested Object Oriented Metrics are the intellectual research for Object Oriented software. Depth of Inheritance Tree and Number of Children are object oriented inheritance metrics also referred as C-K metrics. Depth of Inheritance Tree metric is computed as the largest length from the node to the root of the tree in class. In project's, when Depth of Inheritance is maximum, then probability of failures and mistakes occurrence will be maximum. Depth of Inheritance Tree metric satisfies the Weyuker's properties 1,2,3,4,5 and property 6 is not fulfilled by this metric. Number of Children metric is measured as total of immediate inherited classes subordinated to a class [13]. This metric is a computation of number of derived classes that are going to inherit the methods of the super class. The maximum number of children, then reusability factor is maximum. Metrics for Object Oriented Design (MOOD) proposed by F. Brito e Abreu and explored a basic structural procedure of the object oriented attributes as encapsulation, inheritance, polymorphisms, message passing. Method Inheritance Factor and Attribute Inheritance Factor are inheritance metrics. Method Inheritance Factor is the ratio of the total of the inherited functions or methods in all classes of the software under consideration to the total number of available all classes functions [14]. Attribute Inheritance Factor is the ratio of the total addition of all class inherited attributes of the software under consideration to the total numbers of available all classes attribute [15]. Lorenz, M., et al. proposed object oriented inheritance metrics to find assessment and measurement of object oriented software quality. Attribute Count software metric is computation of total number of class attributes including both inherited class attributes and the attributes defined in the class are computed. Method Count software metric is computation of total number of class methods including both inherited class methods and methods defined in the class are computed [16]. Number of Operation Overridden by subclass software metric is computation of total number of subclass operations overridden. Maximum values of this metric states that there is a fault in software design and outcomes in unique new method names. Gulia, P. in thesis, contributes to a better understanding of object oriented system, its measurement and testing. Firstly, the proposed work provides a theoretical framework to the measurement and testing of object oriented system. Secondly, this research presents existing and proposed metrics for complexity measurement as well as the effect of complexity on other factors [17]. Kajla, P. in thesis, recommended a new metric as Nested Class Complexity Metric to find the complexity of nested classes and outcome is compared with present available metrics. Nested classes are the general requirement in the programming like Java [18].

III. RESEARCH OBJECTIVES

A. To interpret the influence of different inheritances on Object Oriented Metrics in context of complexity.
B. To analyze simulation results to reduce complexity and improves software quality.

IV. PROBLEM FORMULATION

Inheritance is an important characteristic of Object Oriented programming for developing well engineered software having effective quality with minimum development cost and on-time availability. Inheritance have mainly five types as – single, multiple, hierarchical, multilevel and hybrid inheritance. One Project has been analyzed without inheritance and Five Different Object Oriented Inheritance based software projects namely Student Management Information System developed in C++ language have been analyzed with metric tools to understand the relation between different inheritances and Object Oriented inheritance metrics in context of complexity level.

Table I. Object Oriented Inheritance based Projects

Project	Category	Description
I	without Inheritance	Student Management Information System is a design of student related information and fee record etc. in object oriented paradigm. It describes class student with data and methods for student related information without inheritance.
II	Single Inheritance	Project II explores the concept of single inheritance that have a derive class fee_record inherits from base class student. Derive class fee_record have all the features of base class student with its own data and methods.
III	Multiple Inheritance	Project III describes multiple inheritance that have a derive class meritorious inherits from two base classes student and fee_record. Derive class meritorious have all the features of multiple base classes student and fee_record with its own data and methods.
IV	Hierarchical Inheritance	Project IV is a hierarchical inheritance that have two derive classes meritorious and fee_record inherits from base class student. Derive classes have all the features of base class student with its own data and methods.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

V	Multilevel Inheritance	Project V explores the concept of multilevel inheritance that have a derive class meritorious inherits from a class fee_record that itself inherits from base class student. Derive class meritorious have all the features of a class fee_record and fee_record have all data and methods of base class student.
VI	Hybrid Inheritance	Project VI describes hybrid inheritance have two derive classes meritorious and fee_record inherits from base class student. Derive class placement inherits from two base classes meritorious and fee_record. Derive classes have all the features of base class student with its own data and methods. Hybrid inheritance is combination of hierarchical and multiple inheritance to inherit more classes.

V. RESULTS AND FINDINGS

Different categories of inheritance have been interpreted by Object Oriented metrics as-

Table II. C-K Metrics Values Computation

Project	Class	DIT	NOC
I Without Inheritance	Student	0	0
II Single Inheritance	Student	0	1
	Fee_record	1	0
III Multiple Inheritance	Student	0	1
	Fee_record	0	1
	Meritorious	1	0
IV Hierarchical Inheritance	Student	0	2
	Meritorious	1	0
	Fee_record	1	0
V Multilevel Inheritance	Student	0	1
	Fee_record	1	1
	Meritorious	2	0
VI Hybrid Inheritance	Student	0	2
	Meritorious	1	1
	Fee_record	1	1
	Placement	2	0

Project I design is without inheritance having large Student class, it should be in modular design having different modules. Project II design have two classes Student and Fee_record is derived class; it should be more break up to reduce complexity. Project III design have two base classes Student and Fee_record, Meritorious is derived class; Student class should be break up into more classes. Project IV design have base class Student with NOC as 2 and Fee_record, Meritorious are derived classes with DIT as 1. Project V design have base class Student with NOC as 1 and Fee_record with NOC as 1 and DIT as 1, Meritorious is derived class with DIT as 2. Project VI is a combination of hierarchical and multiple inheritance having modular approach.

Table III. MOOD Values Computation: AIF

Project	Class	AIF	
		Attributes Inherited ($A_I(C_I)$)	**Attributes Available ($A_A(C_I)$)**
I Without Inheritance	Student	0	19
II Single Inheritance	Student	0	12
	Fee_record	12	19
III Multiple Inheritance	Student	0	11
	Fee_record	0	7
	Meritorious	18	23
IV Hierarchical Inheritance	Student	0	11
	Meritorious	11	16
	Fee_record	11	18
V Multilevel Inheritance	Student	0	10
	Fee_record	10	17
	Meritorious	17	21
VI Hybrid Inheritance	Student	0	10
	Meritorious	10	15
	Fee_record	10	17
	Placement	22	27

Table IV. MOOD Values Computation: MIF

Project	Class	MIF	
		Methods Inherited ($M_I(C_I)$)	**Methods Available ($M_A(C_I)$)**
I Without Inheritance	Student	0	17
II Single Inheritance	Student	0	12
	Fee_record	12	18
III Multiple Inheritance	Student	0	12
	Fee_record	0	6
	Meritorious	18	27
IV Hierarchical Inheritance	Student	0	12
	Meritorious	12	21
	Fee_record	12	18
V Multilevel Inheritance	Student	0	6
	Fee_record	6	10
	Meritorious	10	15
VI Hybrid Inheritance	Student	0	6
	Meritorious	6	15
	Fee_record	6	12
	Placement	21	23

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

Attributes Inherited ($A_I(C_1)$) is total number of inherited attributes from base class. Attributes Available ($A_A(C_1)$) is total number of base and inherited attributes of a class. Methods Inherited ($M_I(C_1)$) is total number of inherited methods from base class. Methods Available ($M_A(C_1)$) is total number of base and inherited methods of a class. MOOD value computation helps to find inheritance strength level of a specific project. Project I is complex poor design and Project VI have more attributes and inherited methods with modular approach and can be further decomposed to reduce complexity level.

Table V. MOOD - AIF and MIF Values Computation

Project	Class	AIF	MIF
I Without Inheritance	Student	0	0
II Single Inheritance	Student	0.38	0.40
	Fee_record		
III Multiple Inheritance	Student	0.43	0.40
	Fee_record		
	Meritorious		
IV Hierarchical Inheritance	Student	0.49	0.47
	Meritorious		
	Fee_record		
V Multilevel Inheritance	Student	0.56	0.51
	Fee_record		
	Meritorious		
VI Hybrid Inheritance	Student	0.60	0.58
	Meritorious		
	Fee_record		
	Placement		

Attribute Inheritance Factor (AIF) is calculated as =

$$\sum A_i(C_i) / \sum A_a(C_i)$$

where i is from 1 to total number of classes and acceptable range of AIF is from 0 to 48%

Method Inheritance Factor (MIF) is calculated as =

$$\sum M_i(C_i) / \sum M_a(C_i)$$

where i is from 1 to total number of classes and acceptable range of MIF is within 20% to 80% [19].

Different MOOD values are calculated to find complexity level in these projects. Attribute Inheritance Factor (AIF) is calculated for Project I, II & III as no complex design within range of 0 to 48%. But Project IV, V & VI are showing more complexity due to out of range. So, Solution for these designs is to minimize unused attributes and use related attributes with modular approach. Method Inheritance Factor (MIF) is calculated for Project I, II, III & IV as no complex design within range of 20 to 80%. Projects V and VI have more complexity level as compare to other designs, but within range.

MOOD – AIF and MIF calculated values are graphically represented to show complexity levels as -

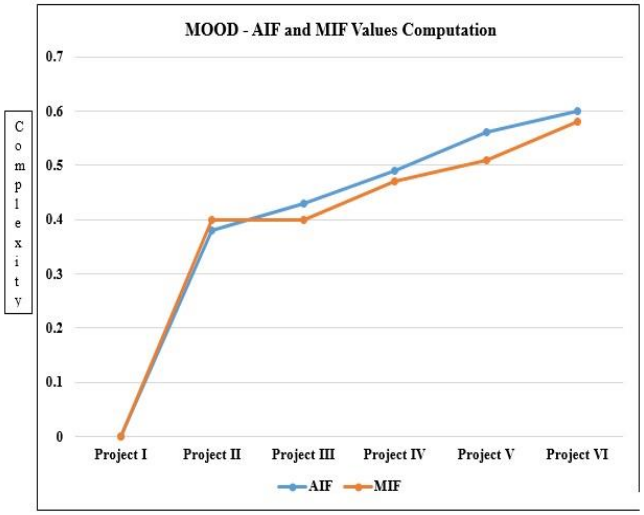

Fig. 1. MOOD – AIF and MIF Complexity Level

Lorenz and Kidd have given metrics as Attribute Count is calculated by total attributes available in design, either inherited or in base class and Method Count is calculated by total methods available in design, either inherited or in base class. Overridden is another important concept in Object Oriented design as one name used in different ways for method operation. Same name of methods can be used for different purposes having different value passing.

Table VI. L-K Metrics Values Computation

Project	Class	AC	MC	NOO
I without Inheritance	Student	19	17	0
II Single Inheritance	Student	19	18	1
	Fee_record			
III Multiple Inheritance	Student	23	27	0.66
	Fee_record			
	Meritorious			
IV Hierarchical Inheritance	Student	23	27	0.33
	Meritorious			
	Fee_record			
V Multilevel Inheritance	Student	21	15	0.66
	Fee_record			
	Meritorious			
VI Hybrid Inheritance	Student	27	23	0.50
	Meritorious			
	Fee_record			
	Placement			

Calculation for Number of Methods Overridden by a subclass is calculated as = Total no of overridden methods/Total no of classes

In above table, L & K metrics values are calculated for different project designs. Project I is poor complex design, whereas Project II, III, IV, V and VI having overridden methods to explore reusability and overloading concept.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

L-K values calculated are graphically represented to show overridden levels as -

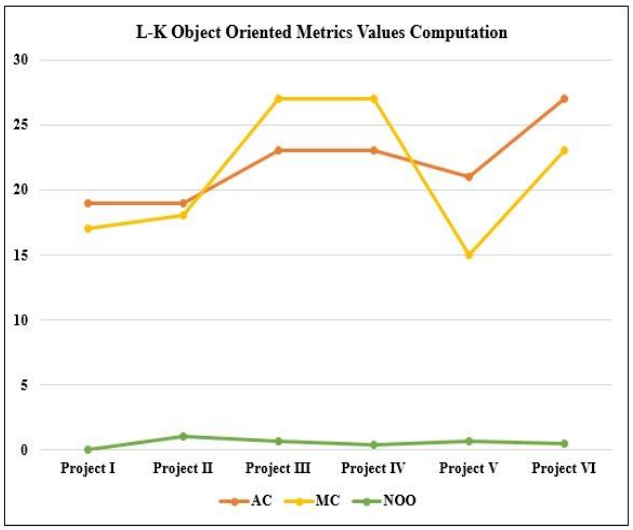

Fig. 2. L-K Metrics Complexity Level

VI. CONCLUSION

Inheritance is important concept of reusability that is used in Object Oriented programming for creating new classes from existing classes. Various inheritance metrics given by researchers for metrics computation and interpretation has been done to find the significance level of Inheritance on different Object Oriented metrics. Different project designs have been used as problem formulation and complexity level have been studied on inheritance metrics. Results showing the concept of inheritance is valuable and productivity for quality product having minimum complexity. If results are not within level of complexity showing more complexity. Software developers can include additional characteristics in the existing software and develop new enhanced version of software. It helps to reduce development time and complexity in programming. With use of inheritance as quality level of software and complexity is in controlled level.

REFERENCES

1. B. Hughes, M. Cotterell and R. Mall, *Software Project Management*, India, McGraw Hill Education, Sixth Edition, 2018
2. R.S. Chhillar, *Software Engineering: Metrics, Testing and Faults*, New Delhi, Excel Books, 2003
3. I. Alvertis, S. Koussouris, D. Papaspyros, E. Arvanitakis, S. Mouzakitis, S. Franken, S. Kolvenbach and W. Prinz, "User Involvement in Software Development Processes", Cloud Futures: From Distributed to Complete Computing, Madrid, *Procedia Computer Science* 97, pp. 73 – 83, 2016
4. Coad, P. and Yourdon, E. 1991. *Object-Oriented Analysis*, Prentice Hall, Englewood Cliffs, N.J. ASDS
5. I. Jacobson, M. Christerson, P. Jonsson and G. Overgaard, *Object-Oriented Software Engineering: A Use-Case Driven Approach*, Addison-Wesley, 1992
6. B. Henderson and Sellers, *Object-Oriented Metric*, New Jersey: Prentice Hall, 1996
7. S. Pasupathy and R. Bhavani, "Object Oriented Metrics Evaluation", *International Journal of Computer Applications*, Volume 78 – No.1, September 2013
8. J. K. Chhabraa, K.K. Aggarwal and Y. Singh, "Measurement of Object-Oriented Software Spatial Complexity", *Information and Software Technology* 46, pp. 689–699, 2004
9. K. F. Gerould, C. Theory, and A. Einstein, *Measuring Complexity*, John Wiley and sons Inc., 2006, pp.54-65
10. A. Basu, *Software Quality Assurance and Testing*, Prentice Hall of India, 2015
11. Y. Suresh, J. Pati and S.K. Rath, "Effectiveness of software metrics for object-oriented system", *2nd International Conference on Communication, Computing & Security*, *Procedia Technology* 6, 2012, pp. 420 – 427
12. Norman Fenton and James Bieman, *Software Metrics in Software Engg.*, CRC Press, Chapman Hall Book, 2014
13. Shyam R. Chidamber and Chris F. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Transactions for Software Engineering*, Vol. 20 No. 6, pp. 476-493, June 1994
14. K.P. Srinivasan and T. Devi, "A Comprehensive Review and Analysis on Object-Oriented Software Metrics in Software Measurement", *International Journal on Computer Science and Engineering*, ISSN: 0975-3397 Vol. 6 No.07 Jul 2014
15. Basili, V.R., Briand, L.C., and Melo, W.L., "A Validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, October 1996, pp. 751-761
16. Mark Lorenz and Jeff Kidd, *Object-Oriented Software Metrics*, Pearson Education, 1994
17. Preeti Gulia, "Analysis and Design of Object Oriented Complexity Metrics and Test Cases", *Ph.D. Thesis*, Maharshi Dayanand University Rohtak, 2012
18. Parveen Kajla, "Study and Design of Object-Oriented and Component-Based Metrics", *Ph.D. Thesis*, Maharshi Dayanand University, Rohtak, Haryana, February, 2014
19. E Da-wei "Analysis and Implementation of Software Metric for Object-Oriented ", IEEE International Conference, pp.1-4, 2009

AUTHORS PROFILE

Mohit Kumar Sharma is a Research Scholar in Computer Science with Faculty of Computational Science, GNA University, Phagwara, India. He is working as Assistant Professor, Computer Science, J.C. D.A.V. College, Dasuya, Punjab, India. He received his M.Phil. (Computer Sc.) degree from Madurai Kamaraj University, India. He has 11 years' teaching and 2 years' research experience. He has 7 research paper publications in international reputed journals, 15 paper presentations in International /National conferences/ seminars and 3 book publications. His research interest is in Software Engineering and Object Oriented Design and Analysis.

Dr. Amardeep Gupta is the Principal of J.C. D.A.V. College, Dasuya, Punjab, India. He received his Ph.D. (Computer Sc. & Engg) degree from I. K. Gujral Punjab Technical University, India. He was former Head of the Department, Post Graduate Department of Computer Science, DAV College, Amritsar, Punjab. He has 21 years' teaching and 6 years' research experience. He has more than 35 research paper publications in international journals, conferences and 12 book publications. His research interest is in Parallel Processing and Object Oriented Programming.