A Comparative Study of Static Object Oriented Metrics

***Manik Sharma**[1], **Gurdev Singh**[2], **Anish Arora**[2] **And Parneet Kaur**[2]*
1Department of Computer Science & Applications, Sewa Devi S.D. College Tarn Taran, Punjab, India
manik_sharma25@yahoo.com
2Department of Computer Science & Engineering, Adesh Institute of Engineering and Technology Faridkot, Punjab, India
Corresponding Author Email: singh.gndu@gmail.com

Abstract

Software metrics is one consistent topic of research in software engineering. The role of software metrics is to find significant estimates for software products and directs us in intriguing managerial and technical decisions. Software metrics have become an integral part of software development and are used during every phase of the software development life cycle. Research in the area of software metrics tends to focus predominantly on static metrics that are obtained by static analysis of the software artifact. The goal of this study is to perform the comparative analysis on static metrics for object oriented programming. This study is done to analyze the different object oriented techniques like class, constructor, and inheritance. The various metrics under study are AHF, AIF, DIT, MHF, MIF, AVPATHS, and SEIMI etc. The study of such metrics can become a useful tool for reverse software engineering.

Keywords: *Object oriented programming, Metrics, Measures.*

1. Introduction

The focus of Software engineering is to develop and maintain the software for business use. The objective of software engineering is to develop cost effective and quality oriented software products for small, medium and large scale. Software engineering provides a range of facility to realize the software completely for its maintenance or for its reusability. Software metric is an important component of software engineering. There are number of software metrics developed by different researchers, but some of them are helpful to software developer. Software metrics acts as an indicator for software attribute. The name software metric [1][2] is associated with varied measurements of computer software and its development. Software metrics provides the relationship between collected measures. With the help of software metric anybody can judge of measure the performance of various features of the software. Computer science researchers are putting all their efforts in measuring quantitative information from software component. Software metric [3] plays a major role in civilizing the quality of software, planning its budget, its cost estimation etc. We apply some software logical of mathematical technique to software process or product to supply or improve engineering and management information [4]. Some of the software metric's objective [5][6] are perception, software inspection, planning, optimization and quality improvement. Numbers of metrics are developed for different purposes; some of them are very beneficial for the programmer or developer. The complexity in the calculation of software metrics provides hindrance in the research and usability of software metrics. Practically the metrics can be categorized [7] into different categories like metrics for analysis model, metrics for design model, metrics for source code, metrics for testing, metrics for quality assurances etc.

Further according to Roger S Pressman a metric should possess features like simple & computable, consistent, platform autonomous, objective, empirically and intuitively credible.

Today a project is of no sense if it does not have any object oriented features. The perception of object oriented analysis and design in software engineering has many rewards that help the programmer to realize and develop the program efficiently. Object oriented metrics helps a lot to a programmer or developer to understand and unravel the object oriented problem easily and precisely. Object oriented metrics helps in analyzing the usefulness of object oriented technologies or in simple terms Object-oriented metrics depict characteristics of object-oriented programming.

1.1 Objectives: The objectives of this paper are:

- To understand the concept of static metric.
- Measuring the attributes of static metrics for object oriented programming techniques.
- Comparing the various static metrics for different Object Oriented techniques.
- To find the mathematical relation of different object oriented technique with different object oriented static metrics under study.
- Finding an optimized object oriented technique in terms of complexity by using the concept of measured metrics.

2. STATIC METRICS

Software metrics plays an important role in project coordination and project management. With the help of software metrics different attributes of a project can be measured. Software metrics also helpful in the area that is prone to an error. There are different types of metrics like size metrics, quality metrics, satiability metrics, object oriented metrics etc. The credit of introducing the concept of software metrics goes to Wolver ton who performs a research on production ratio of the programmer by using the concept of LOC i.e. line of code. Software metrics explore the attributes of software to measure the different characteristics of software.. Metrics helps in to measure the various attributes like cost of software development, its complexity, number of operands, operators and statement, hiding factor, coupling factor etc. Predictive metric are normally associated with software product. According to Somerville the metric can be classified into two categories i.e. control metric and predictive metric. Predictive software metric [5] plays a major role in determining both static as well as dynamic characteristics of the software. In this paper the focus is given on static predictive metrics of object oriented programming.

2.1 Static metric: First static metric [8] (LOC/KLOC) was used to measure the productivity of a program. The most commonly used complexity metric before 1990 was cyclomatic [9] complexity that was measured by McCabe. He uses the flow graph and some mathematical equations to compute software complexity. This metric was used in code development risk analysis [10], change risk analysis in maintenance and in test planning. In 1976 McCabe [11] defined the cyclomatic complexity number metric. The metric measures the number of independent paths through a software module. Although cyclomatic complexity is widely used, critique on it exists claimed that it's based on poor theoretical foundations and an inadequate model of software development. The cyclometic complexity has been selected to be a part of the benchmarks.

3. ANALYSIS OF OO METRICS

To perform the comparative analysis of the object oriented techniques like simple class, inheritance, constructor and parametric constructor, the code is implemented in one of the famous object oriented programming language, an invention of Bjarne Stroustrup i.e. C++. With the help Object-oriented one can verify the victory or malfunction of a process or person, and to quantify improvements throughout the software process. Object oriented metrics can be used to support a examine quality oriented and reliable code from different programming techniques. With the development of object oriented paradigm [11] the design of software become better, reliable and easier to access as compare to traditional structured programming techniques. We have same program by using different above said technique and calculated various attributes of object oriented programming in the form of object oriented static metrics [12] [14]as given below in the table 1.

Table 1: Static Metrics versus OO Techniques

Parameters	Simple Class	Inheritance	Constructor	Parametric Constructor
AHF	1	1	1	1
AIF	0	0.67	0	0
AVPATHS	0	0	0	0
ACLOC	18	12.50	18	18
AMLOC	6.0	6.67	5.67	5.67
PDIT	0	1	0	0
LOC	28	36	27	27
MHF	0	0	0	0
MIF	0	0	0	0
NCLASS	1	2	1	1
SEIMI	117.98	112.49	119.33	118.95
SLOC	28	36.0	27	27

Description of Parameters

AHF	This metric is used to measure the invisibilities of attributes in classes. The attributed invisibility is defined as the percentage of the total classes from which the attribute is not visible.
AIF	Attribute inheritance factor
AVPATHS	Average Depth of Paths is calculated by counting the number and size of all paths from all methods, and then dividing that number by the number of methods which had other method calls. In other words, the average depth of paths from methods that have path at all.
ACLOC	Average lines per class: This metric gives the average Class size in terms of LOC.

AMLOC	Average lines per method: This metric gives the average Method size in terms of LOC.
PDIT	Depth of Inheritance tree: The Depth of Inheritance Tree for a Project is the deepest or maximum of all inheritance trees within the project.
LOC	Lines of code: Number of Lines in the project, including source, whitespace and comments.
MHF	Method Hiding Factor is one of the important metrics of object oriented programming that is calculated by summing the visibility of each method in respect to the other classes in the project. It is used to measures the invisibilities of methods in classes. The invisibility of a method is the percentage of the total classes from which the method is not visible.
MIF	Method inheritance factor [15] gives the information about the impact of inheritance in your file or program. It is calculate as ratio of inherited methods to the total number of methods.
NCLASS	It is another static metrics that count the number of classes in a program.
SEIMI	SEI Maintainability Index is one of the important measures of maintenance. SEIMI is a measure of the maintainability of the project, as described by the Software Engineering Institute.
SLOC	Source lines of Code [10] are an important measure of source line of code. Counting lines is used for estimating the amount of upholding or maintenance required and it can be used to normalize other software metrics.

The following diagrams(1(a),1(b),1(c),1(d),1(e), 1(f) & 1(g)) how simple class, inheritance, constructor and parametric constructor affect the various object oriented metrics like ACLOC, AMLOC, LOC, SEIMI.

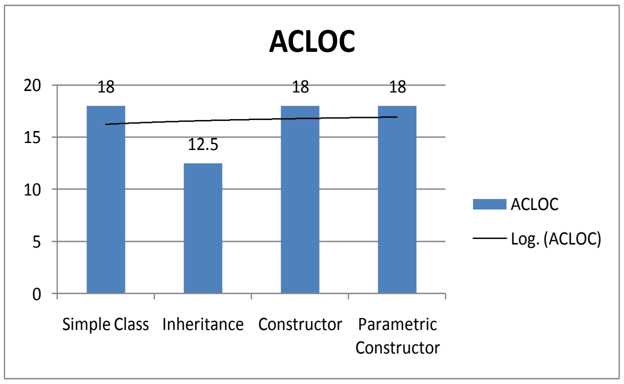

Figure 1(a): AMLOC versus Object Oriented Techniques

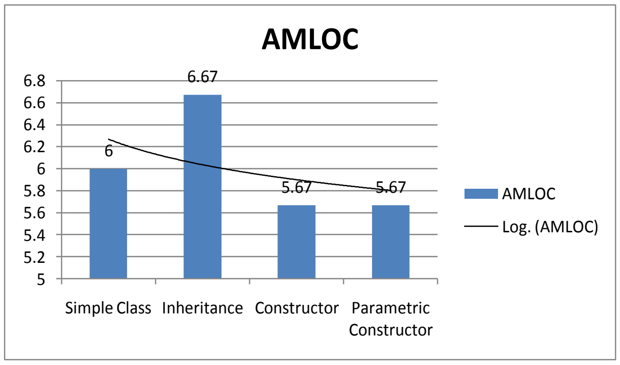

Figure 1(b): AMLOC versus Object Oriented Techniques

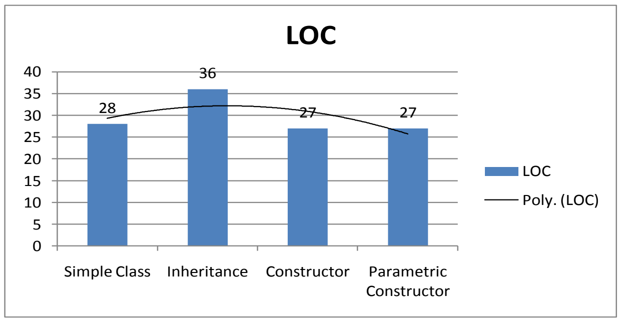

Figure 1(c): AMLOC versus Object Oriented Techniques

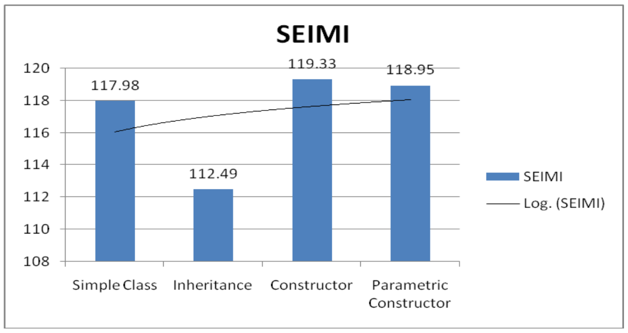

Figure1 (d): SEIMI versus Object Oriented Techniques

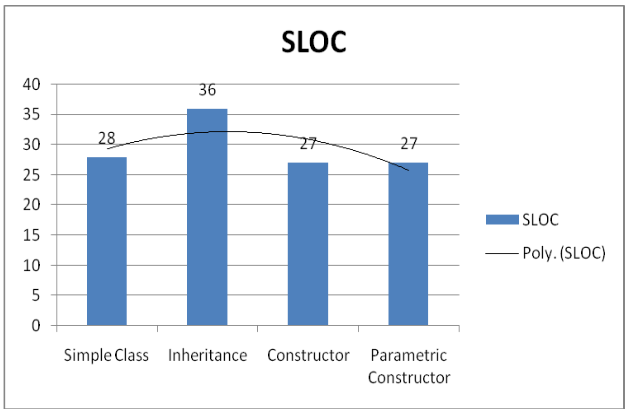

Figure1 (e): SLOC versus Object Oriented Techniques

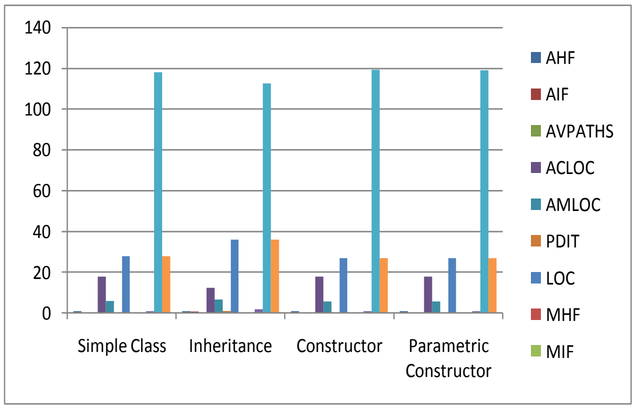

Figure1(f): Analysis of metrics versus OO techniques

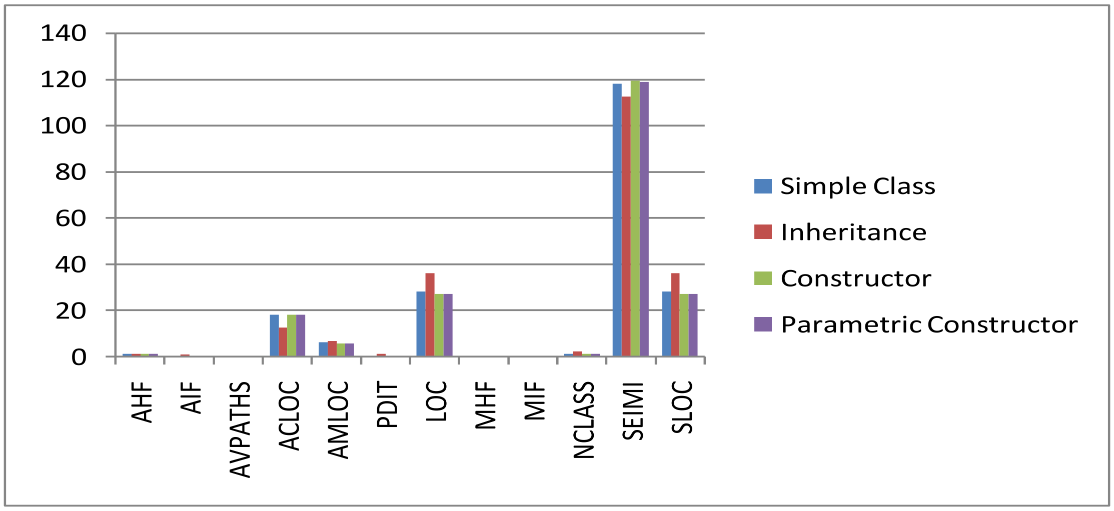

Figure1 (g): Analysis of metrics versus OO techniques

From the above analysis it can be concluded that mathematical variation of different metrics like ACLOC, AMLOC, LOC with above said object oriented technique are in logarithmic or in polynomial form is as shown below in the following table. By using the following set of equation one can easily calculate the following object oriented metric for simple class.

Table 2: Mathematical Equation versus Metrics

Metric	Curve	Equation
ACLOC	Logarithmic	Y=.514ln(x)+16.21
AMLOC	Logarithmic	Y=-0.33ln(x)+6.268
LOC	Polynomial	y = -2x2 + 8.8x + 22.5
SEIMI	Logarithmic	y = 1.421ln(x) + 116.0
SLOC	Polynomial	$y = -2x^2 + 8.8x + 22.5$
NCLASS	Polynomial	$y = -0.25x^2 + 1.15x + 0.25$

Halstead has brought the revolution in the field of metric by collaborating information science and psychology. By using the concept of Halstead metrics an analyst is able to compute the size, complexity, volume, length, difficulty of a project. The basic attributes of Halstead metrics are n1,n2, N1 & N2.

Table 3: Analysis of Main Method With different object oriented techniques

	n1	n2	N1	N2	V(G)	V'(G)	D	E	V	LOC
Simple Class	1	6	2	8	1.0	4.0	0.67	18.72	28.07	8.0
Inheritance	1	9	2	10	1	4.0	.56	22.15	39.86	9.0
Constructor	1	5	1	6	1.0	3.0	.60	10.86	18.10	7.0
Parametric Constructor	1	7	1	8	1	3.0	.57	15.43	27.0	7.0

n1,n2, N1, N2	Halstead [13] has proposed different metrics for measuring the size of a program he uses different variables n1,n2, N1,N2 the number of distinct operators, number of distinct operands, number of operators and number of operands respectively.
V (G)	Cyclometic Complexity: it is one of the important measures of programming code. Normally the introduction of conditional and looping statement adds some complexity in the program. The concept of cyclometic complexity is given by McCabe. Mathematically it is calculated as V(G)=e-n+p
V' G):	Extended Cyclometic Complexity
D:	Halstead program difficulty
E:	Halstead Program Effort metrics helps in determining the programming effort required to develop project.
V	Halstead Program Volume [16] is one the important metrics that instruct the analyst to consider the programming language while calculating the length of the program. In technical terms it is minimum number of bits that are used to encode the program.
LOC:	It is one the basic static metric that is used to measure the size of code segment. It helps in measuring the cost of project in an effective way.

The following figures 2(a), 2(b), 2(c), 2(d), 2(e) , 2(f) and 2(g) shows the variation of various factor like cyclometic complexity, extended cyclometic complexity when compared with different object oriented techniques i.e. simple class, inheritance, constructor and parametric constructor.

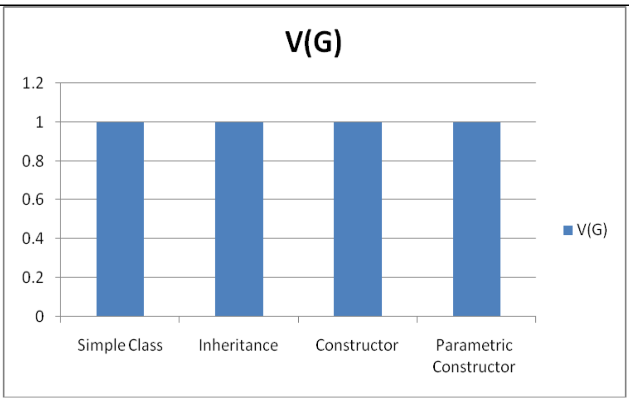

Figure 2(a) : Cyclometic complexity versus various OO Techniques

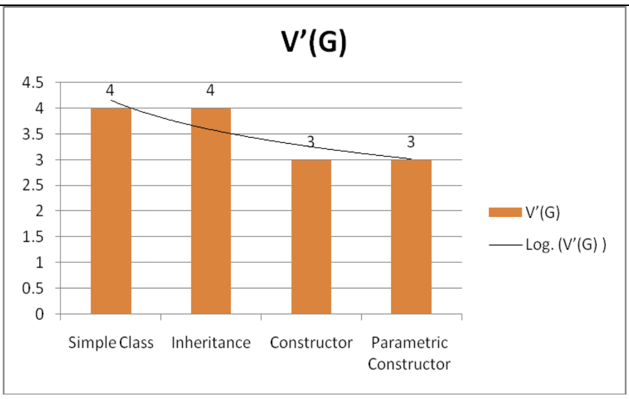

Figure 2(b) : Cyclometic complexity versus various OO Techniques

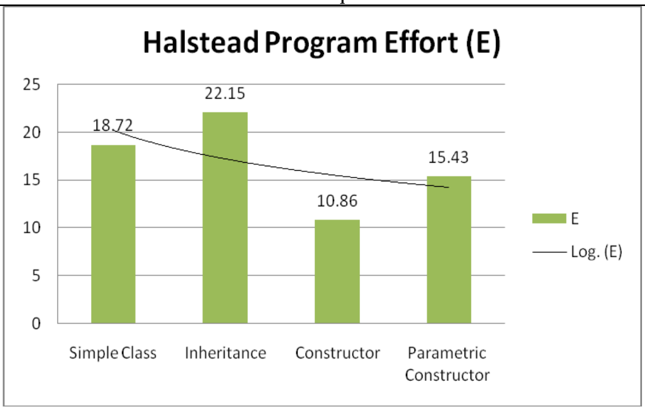

Figure 2(c): Programming Effort versus OO Techniques

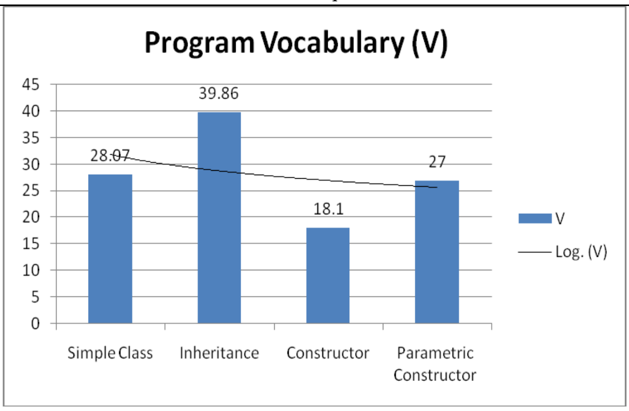

Figure 2(d) : Programming Vocabulary versus OO Techniques

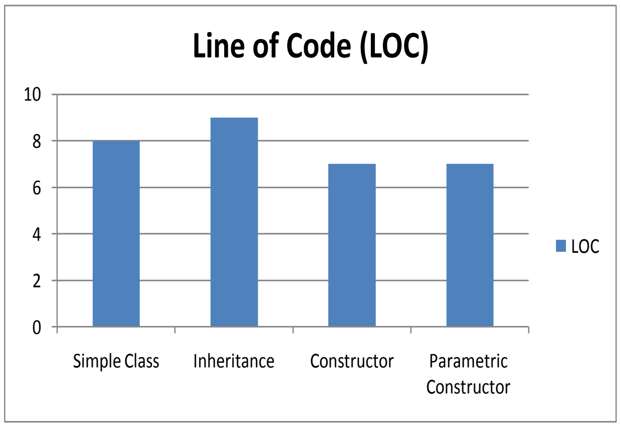

Figure 2(e): Lines of Code versus OO Techniques

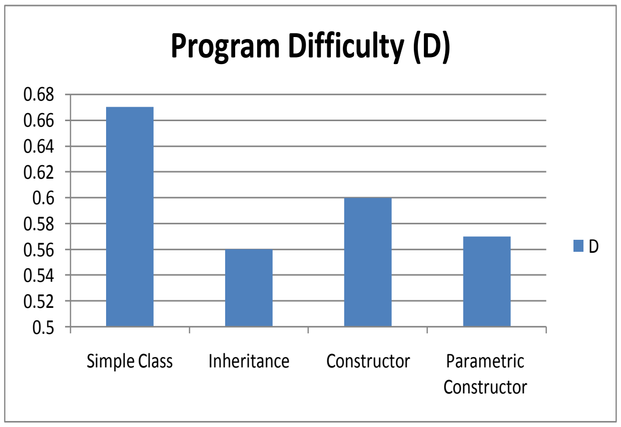

Figure 2(f): Lines of Code versus OO Techniques

By using the graphical analysis of various metrics one can conclude that mathematical relationship and nature of curve formed by above said metrics when compared with different object oriented technologies like class, inheritance, constructor and parametric constructor.

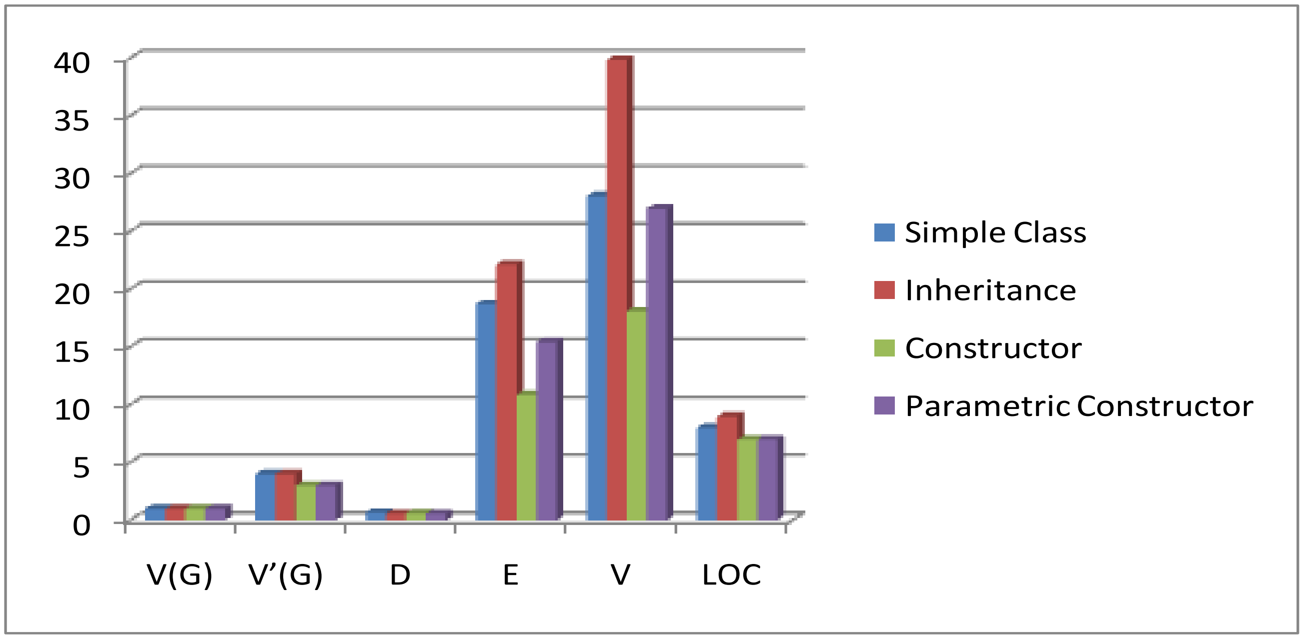

Figure 2(g): Variable Halstead size metrics versus OO Techniques

The analysis is shown in the form of table 4 and figure 3 as shown below:

Table 4: Mathematical Equation versus Metrics

Metric	Curve	Equation
V(G)	Logarithmic	Y=1
V'(G)	Logarithmic	$y = -0.82\ln(x) + 4.156$
E	Logarithmic	$y = -4.32\ln(x) + 20.22$
V	Polynomial	$y = -2.497x + 34.5$
LOC	Polynomial	$y = -0.5x + 9$
D	Polynomial	$y = -0.026x + 0.665$

4. Conclusions

In object oriented analysis we came to conclusion that constructor is the best choice among the above said technique due to its low ACLOC, AMLOC, LOC and higher value of SEIMI. In method comparison it is concluded that all the above said techniques like simple class, inheritance, constructor and parametric constructor has same value for cyclometic complexity, on the other hand constructor has low value of extended cyclometic complexity as compare to simple class and inheritance. In regard to Halstead program difficulty it is found that inheritance has lowest value of difficulty and simple class has maximum value of program difficulty. In regard to program effort it is found the constructor is the best choice among above said techniques. Further constructor also minimizes the line of code (LOC). We have also concluded the different polynomial or logarithmic equation for different object oriented metrics under study.

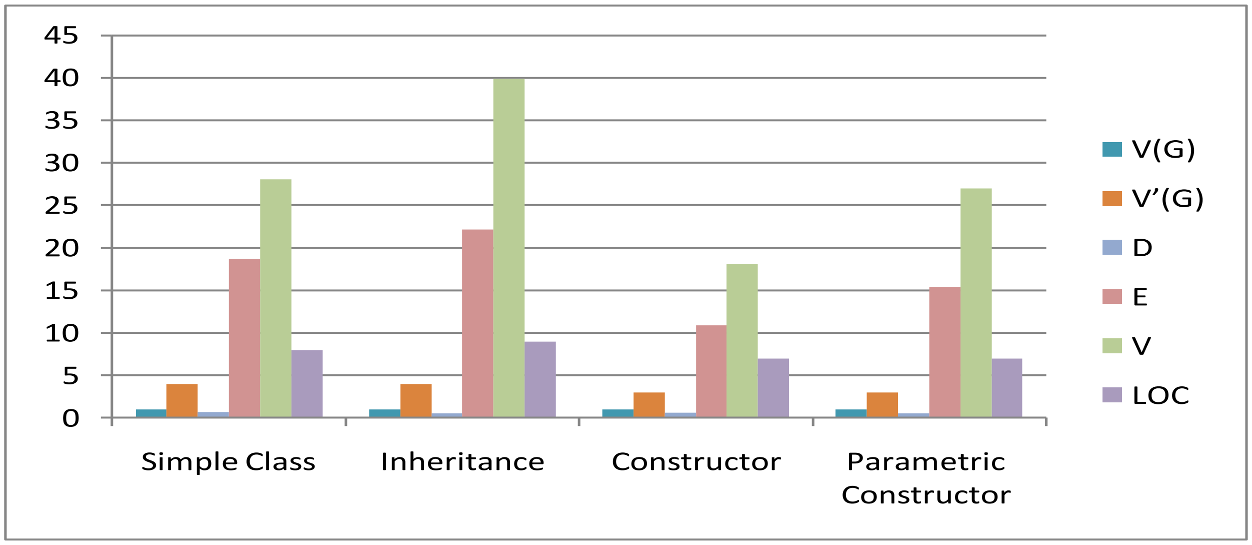

Figure 3: Analysis of Complexity Metrics with different OO Techniques

Acknowledgments

Authors are highly thankful to Dr. Gurvinder Singh, Associate Professor, Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar, India for his precious guidance from time to time.

References

[1] H F Li, W K Cheung "An Empirical Study of Software Metrics" Software Engineering IEEE Transactions on (1987) Volume: SE-13, Issue: 6, Pages: 697-708
[2] N E Fenton "Software Metrics" Conference Proceedings of on the future of Software engineering ICSE 00(2000) Volume: 8, Issue: 2, Publisher: ACM Press
[3] Kuljit Kaur Chahal, Hardeep Singh "Metrics to study symptoms of bad software designs" ACM SIGSOFT Software Engineering Notes (2009) Volume: 34, Issue: 1, Pages: 1
[4] 12 Steps to Useful Software Metrics by Linda Westfall, [online] *www.westfallteam.com/Papers/12_steps_paper.pdf*
[5] Manik Sharma , Gurdev Singh "Static and Dynamic metrics- A Comparative Analysis", Emerging Trends in Computing and Information Technology 2011.
[6] Tu Honglei , Sun wei, Zhang Yanan, "The Research of Software metric and software complexity metrics" International Forum on Computer Science Technology and Applications(2009) Publisher: IEEE, Pages: 131-136
[7] Roger S. Pressman "Software Engineering-A Practitioner's Approach" 6th Edition, McGraw Hill International Edition pp 466-472
[8] Li, H.F., Cheung, W.K. "An Experimental investigation of software metric and their relationship to software development effort", IEEE Transaction on software engineering 649-653, Piscataway, NJ, USA.
[9] Thomas J McCabe, "A Complexity Measure", IEEE Transaction on Software Engineering, Vol. SE-2 No. 4 [308-320]
[10] Manik Sharma, Dr. Gurdev Singh, "Analysis of static and Dynamic Metrics for Productivity and Time Complexity", IJCA, Volume 30– No.1, September 2011.
[11] Satwinder Singh, K.S. Kahlon, "Static Analysis to Model & Measure OO Paradigms", SAC, ACM.
[12] F.B. Abrew, R. Carapuca, "Candidate metrics for object oriented software within a taxonomy framework", Journal of Systems and Software, 26(1).
[13] M. Halstead, "Elements of Software Science", Elsevier North-Holland, New York 1977

[14] S.R Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 20 No. 6

[15] KP Srinavan, Dr. T Devi, "Design and Development of procedure for new object oriented design metrics', IJCA, Vol. 24, No. 8, Jun 2011

[16] Rajib Mall, "Fundamentals of Software Engineering", 3rd Edition, PHI, pp 78.