Identification, Analysis & Empirical Validation (IAV) of Object Oriented Design (OO) Metrics as Quality Indicators

Dr. Brajesh Kochar [1]
Shailendra Singh Gaur [2]
Dinesh Kumar Bhardwaj[3]
[1]Department of C.S.E.,Bhagwan Parshuram Institute of Technology, G.G.S.I.P.U.,Delhi, India
[2] Department of I.T.,Bhagwan Parshuram Institute of Technology, G.G.S.I.P.U.,Delhi, India
[3] Department of C.S.E.,Bhagwan Parshuram Institute of Technology, G.G.S.I.P.U.,Delhi, India

Abstract : Metrics and Measure are closely inter-related to each other. Measure is defined as way of defining amount, dimension, capacity or size of some attribute of a product in quantitative manner while Metric is unit used for measuring attribute. Software quality is one of the major concerns that need to be addressed and measured. Object oriented (OO) systems require effective metrics to assess quality of software. The paper is designed to identify attributes and measures that can help in determining and affecting quality attributes.

The paper conducts empirical study by taking public dataset KC1 from NASA project database. It is validated by applying statistical techniques like correlation analysis and regression analysis. After analysis of data, it is found that metrics SLOC, RFC, WMC and CBO are significant and treated as quality indicators while metrics DIT and NOC are not significant. The results produced from them throws significant impact on improving software quality.

Keywords:- *CK object oriented (OO) design metrics , Software quality, C++, Internal and External attributes.*

*****

1. INTRODUCTION

The metrics used in software engineering are being used by every industry to enhance, develop and maintain given software. Software metrics are magnanimous in number and are dependent on type of software attributes that user wants to estimate. There are mainly two categories of software metrics viz process metrics and product metrics. Process metrics deals with how much effort is required (Person-Months), quantity of resources and methodology involved in it. Product metrics deals with specifications of project like requirements, complexity, cohesion, coupling, reliability and maintenance. Software product involves finding number of lines of code (LOC), complexity of code and test cases generated during testing process.
It is not feasible to design full fault prone system due to changing business requirements and complexity of software. A magnanimous amount of research is being done by researchers to improve software quality by innovating novel techniques. Object oriented (OO) metrics are commonly used for quality estimation. As the name suggests, OO metrics are related to measurement of design characteristics like encapsulation, inheritance, information hiding and message passing. Software quality is measured in terms of metrics. It is measured value that is assigned to product for measuring its quality. So, it can be said that it is possible to predict software processes by relying on software metrics. Software metrics acts as crucial source of information for decision making **[22]**. Testing and validation of all parts of software is time consuming and cost effective. So, it is essential to identify attributes that can predict fault proneness and acts as measurable quality attributes. It requires proper selection of product design metrics.

The objective of study defines exploring various metrics and validating them by taking public KC1 dataset under NASA project database. The remainder of paper is organized as follows: Section 2 makes readers aware with the methodology involved in study. Section 3 deals with some terminologies and background of conducted studies by various researchers. Section 4 conducts empirical study by collecting, processing and analyzing data using various statistical techniques. Section 5 provides answers to research questions. Section 6 concludes the given paper.

2. METHODOLOGY

It is inevitable that research elements are not limited to review literature on the analysis of metrics and measures of quality factors being studied under Kitchenham and Charters [**1**] unified approach.
Methodology covers following points:

- To explore various studies conducted by researchers in context of OO metrics and software quality.
- To make users aware of software related terminologies by providing suitable examples.
- To identify OO metrics that can act as quality indicators by predicting fault prone classes.
- To validate metrics by taking some dataset and evaluate them through statistical techniques.

2.1 Research Questions that needs to be answered at the end of study.

RQ1. How to estimate effort using Lines of Code (LOC) measure?

RQ2: How to measure % error in length of program and estimated program level if operands and operators are given? Does error in length affects quality of software?

3. STATE OF ART & TERMINOLOGY

3.1 State of Art

A continuous efforts and studies have been conducted in finding and measuring metrics for Object oriented design (OO) systems. The external metrics deal with features like portability, reliability, usability etc while internal factors deal with factors that lead to internal movement of software modules. They are cohesion and coupling.

Anna et. Al proposed framework for measuring reusability by refining CK metrics **[4,5].** Zhao and Xu **[6, 7]** defined cohesion and coupling metrics that works on dependency graphs between software modules and dependencies. Coupling is implemented with the help of Aspect J. The coupling measures are identified by dependencies between aspects and classes only. Kumar et. al **[8]** designed framework for measuring complexity of software but model failed to measure interaction among various components of software.

Several OO metrics have been proposed in order to predict software quality with the help of classification and prediction models. It is important to know the metrics that needs to be measured like effort, productivity, fault tolerance etc. Use of OO methodology is widespread in all applications that need development of software. Previous research studies are being referenced that tells about empirical validation of various product metrics like NOC, LCOM, McCabe metric **[23] [24], [25**]. Li and Henry **[12**] have proposed prediction model consisting of ten OO metrics using statistical analysis technique in order to derive relationship between maintenance and metrics. Khoshgaftar et .al **[13]** used NN to estimate software quality. They compared parametric model and ANN model to estimate accuracy. Giovani **[14]** maintains relationship between static metrics and software fault proneness by computing static metrics (Cyclomatic complexity) and dynamic metrics (dataflow coverage). Emam et. al **[15**] devised model to predict faulty classes in java application. K, Rakesh., Kaur, Gurvinder 2011 [**16**] studied on Comparing Complexity in Accordance with Object Oriented Metrics. The study highlighted the object-oriented software metrics proposed in 90s' by Chidamber, Kemerer and several studies were conducted to validate the metrics and discovered several deficiencies. Gyimothy, T., Ferenc, R., & Siket, I. 2005 [**17**] conducted a study on Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. The study is based on source code of the well-known open source Web and e-mail suite called Mozilla. The study also used these modified metrics and added one more object-oriented metric i.e. Lack of cohesion on methods (LCOM) and the well-known lines of code metric (LOC). The study used logistic regression and machine learning methods to predict the fault proneness of the code.

Few software organizations like TRW **[26]** and SEI **[27]** have designed their own product metrics to build cost and fault prone models.

3.2 Terminology

Before going to literature study, it is mandatory to make users aware of some software related terminologies.

Software Complexity: - According to Basili **[2**], complexity is defined as measure of resources needed by system while performing given task. Complexity has no units as it is measured as function of time and space. It is measured on basis of number of inputs. There is quite difference between terms complicated and complex. Complicated means the solution is not known presently but it has possibility to get solved later. Complex specifies interactions between software modules are difficult to understand.

Entity and Attributes:-

Measurement is done on entities and attributed. Entity is defined as an object, event or action over specific time. An attribute is characteristic of an entity like size of program. Both entities and attributes are dependent on each other i.e. saying only measure a program is vague sentence because its attribute is not specified. "Measure size of program" is valid sentence holding both entity and attribute.

Table 1: Types of Attributes

Internal Attributes	**External Attributes**
1. These attributes depend only on entity.	1. They depend both on entity as well as context of entity.
2. It is easy to measure. Examples: Size, cost, effort etc.	2. It is difficult to measure because it requires several other factors and parameters to be tested under consideration. Examples: Maintenance, reliability etc.

Table 2: Definition of External Attributes

Attributes	Definition
1. Effectiveness & Extendibility	Allows adaptive changes in system design as per user requirements. It covers *inheritance* metric.
2. Functionality & Reusability	Allows redefining existing component in software system and reusing it in order to achieve higher outcome. It relates to *complexity* metric.
3. Understandability	Measures the degree of complexity and level of programmer to understand design of software. It relates to *coupling and cohesion* metric.

Types of Measures

Measures depend on size of program, structure and nature of modules. Few measures that are commonly used in software industry:

(a) Lines of Code (LOC):- It is simplest and well understood measure that is used for prediction of effort and fault proneness. LOC helps in finding number of executable lines in given code.

(*b) Halstead Software*:- It is used to identify basic elements of program and measures them to predict their attributes. It states that program is combination of operators and operands.

(c) Cyclomatic Complexity:- It measures complexity of given code from control flow graph. It counts base paths that start from one point and end till the program is finished. This approach is used to find number of independent paths through a program. An independent path is any path through the program that contains at least one new condition. The calculation of paths is given by Mc Cabe's Cyclomatic Metric **[3]** by using formula:

V (g) = e-n+2p (Equation 1)

Where v= vertices of graph, e = edges, n = number of nodes in graph and p= connected components.

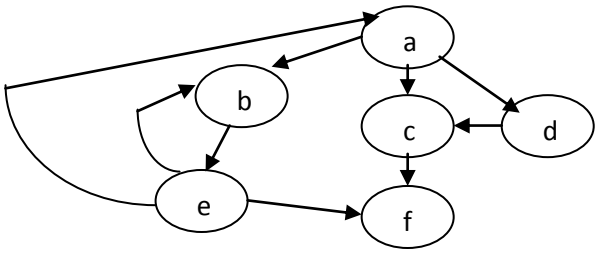

Fig 1: Flow Graph

Each node in a graph represent block of code in a program and arcs represent branches in program. In flow graph, the flow is sequential and it is based on assumption that each node can be reached by starting node and ending node.

From above graph, the value of Cyclomatic complexity is given as:

V (G) = 9 – 6 + 2 = 5 where e=9, n=6 and p=1

There are 5 independent paths in flow graph as:

Path 1 : acf

Path 2: abef

Path 3: adcf

Path 4: abeacf

Path 5: abebef

There are other two methods for calculating complexity as follows:

(1) It is equal to number of predicate (decision) nodes plus one V (G) = P + 1 where P = predicate nodes in graph.

(2) It is equal to number of regions of flow graph.

Object Oriented (OO) Metrics

(a) CK Metrics

The word CK stands for Chidamber and Kemerer **[21].** They are of six types:

(1) Depth of Inheritance Tree (DIT):- This metric is defined as length of root to the deepest leaf node in tree. In this metric, the class becomes more complex on going from top to bottom. Depth of tree is also called as height of tree.

(2) Number of Children (NOC):- This metric calculates number of classes in inheritance tree down from class. It relates to reusability attribute that gets affected. If NOC increases, then amount of effort required in testing also increases.

(3) Coupling between objects (CBO):- It relates the number of other modules that are coupled to the current module either as a client or supplier. Increase in CBO will decrease the usability. It is used to measure complexity, reusability and quality.

(4) Response for a class (RFC):- It defines number of methods that are executed in context of messages received by objects of class and that methods are called as local methods. Greater number of methods more will be complexity of class.

(5) Lack of Cohesion on Methods (LCOM):- LCOM is the difference between the number of methods whose similarity is zero and the methods whose similarity is not zero. It is not a good metric of quality.
(6) Weighted Methods per Class (WMC): - WMC is the number of all member functions and operators defined in each class. It is used to measure the understandability, reusability, maintainability and complexity and quality.

(b) MOOD Metrics

Srinivasan et.al **[9]** defined object oriented design metrics commonly known as MOOD metrics. MOOD deals with structural phases like polymorphism, inheritance, cohesion and coupling. They are defined as below:
(1) Method Inheritance Factor (MIF):- This metric is defined as ratio of number of inherited methods (NIM) to the sum of number of inherited methods and number of defined methods (NDM) in the class.
Mathematically,
MIF = NIM / NIM + NDM (Eq. 2)

(2) Attribute Inheritance Factor (AIF):- This metric is similar to MIF except the methods are replaced by attributes. It is defined as ratio of number of inherited attributes (NIA) to the sum of number of inherited attributes and number of defined attributes (NDA) in the class.
Mathematically,
AIF = NIA / NIA + NDA (Eq. 3)
(3) Polymorphism Factor (PF):- This metric is defined as ratio of number of overriding methods to total possible number of overridden methods in given class. Its value ranges from (0 – 100) %.
(4) Coupling factor (CF):- Coupling means inter-relatedness among modules. This metric is defined as ratio of number of possible couplings (NPC) of given class with other classes to the number of actual classes (NAC) minus 1.
Mathematically,
CF = NPC / (NAC -1)

Table 3: Quality factors belonging to various metrics and measures

S.No.	Quality factors	Related to	Type of metrics and measures	
			MOOD	CK
1	Effectiveness & Extendibility	Inheritance	MIF, AIF	DIT
2	Functionality & Reusability	Complexity	CF	WMC, NOC, CBO, RFC
3	Understandability	Cohesion and Coupling	CF	CBO

4. EMPIRICAL STUDY

4.1. Constituents of empirical study

The first step is to select dependent and independent variables that needs to be measures and validated using OO metrics. Prediction of fault prone classes in system can be treated as good indicator to tell about quality of software. So, it is treated as dependent variable of our study. The relationship between CK OO design metrics and this dependent variable is also being presented in paper.
Now, its turn to measure size of classes involved in project. It is known that various metrics like LOC, FPA are language dependent **[28, 29].** So, we have used concepts of C++ programming language among these metrics. They are defined as follows:

Table 4: Definitions of OO metrics in context of C++

CK OO design metrics	Definition in context of C++
WMC (Weighted methods per class)	Number of member functions and operators defined in each class. It is invalid if we take member functions and operators inherited from super class.
DIT (Depth of Inheritance Tree)	Classes are organized into directed acyclic graphs (DAG) instead of trees because C++ allows multiple inheritances. It is inheritance related OO metric.
NOC (Number of children)	Number of direct children of each class. It is inheritance related OO metric.
CBO (Coupling between objects)	If a class uses member functions of other class, it is said to be coupled.
RFC (Response for Class)	Number of C++ functions directly called by member functions or operators of given class.
LCOM (Lack of Cohesion on Methods)	It is given as member functions (x,y) without variables – member functions (x,y) with variables.

4.2. Hypothesis

It is performed in order to validate and maintain relationship with OO metrics and fault proneness dependent variable. The results later on are used to describe above metrics as quality indicators. For each metric, hypothesis is given

below that is validated by taking public data set in further section.

HWMC- A class with more member functions is more complex and thus more fault prone.

HDIT- The class located in deepest level hierarchy is treated as more fault prone.

HCBO- Highly coupled classes are more fault prone because they depend more on methods and objects of other class.

HNOC- Classes with large number of children are more faults prone and requires more testing.

HRFC- Classes having larger response sets in response to member functions are likely to more fault prone.

HLCOM- Classes that binds member functions and variables are more likely to fault prone i.e. low cohesion.

4.3. Data Collection

The following components that need to be collected are:

- LOC of C++ programs at end of implementation phase related to developed project
- Data about C++ programs
- List of erroneous data found in testing phase
- Replaced source code of C++ programs at maintenance phase.

Source of data:

The study involves usage of public dataset KC1 from NASA IV and V Facility Metrics Data Program Repository (MDP) **[30].** The dataset consists of 43 KSLOC of C++ code with 145 classes and 2107 instances.

Data processing:

Empirical analysis of OOCK metrics and code metric (SLOC) has been done in order to predict number of faults associated with different severity levels by making use of public data set KC1 (NASA program metric data program database). The dataset contains faulty data at method level (faulty classes according to severity level) while metric information is at class level.

Different severity levels represent impacts of faults on performance of system. In KC1 NASA dataset, severity of faults decreases from 1st to 5th level. But for simplicity, faults that have similar impacts on system can be treated as single fault. According to **[31],** fault data is categorized into three levels-high (severity 1), medium (severity 2) and low (severity 3,4,5).

Table 5: Distribution of classes among three severity levels **[31]**

Level	No. of faulty classes	% of faulty classes	No. of faults	% of faults
High	24	16.55	49	7.63
Medium	58	40	449	69.94
Low	38	26.20	144	22.43

The intention of this study is to compare and validate effect of OO metrics on quality of software.

4.4. Data Analysis

This section presents statistical analysis of six OO metrics and 1 code metric (SLOC) using descriptive method, correlation analysis and regression analysis.

(a) Descriptive Analysis

It includes different categories of values for all seven metrics namely Minimum (Min), Maximum (Max), Mean, Mode, Standard Deviation (SD) and Median.

Below table shows statistics of 145 classes from KC1 dataset.

Table 6: Descriptive statistics of 145 studied C++ classes of KC1 **[32]**

Metrics	Min	Max	Mean	Mode	S.D.	Median
WMC	0	110	17.42	8	17.45	12
RFC	0	222	34.38	7	36.20	28
CBO	0	24	8.32	0	6.34	8
DIT	1	7	2	1	1.26	2
NOC	0	5	0.20	0	0.70	0
LCOM	0	100	68.71	1	36.89	84
SLOC	0	2313	211.1	0	345.6	108

In above table, high mean value of LCOM indicates classes are less cohesive.

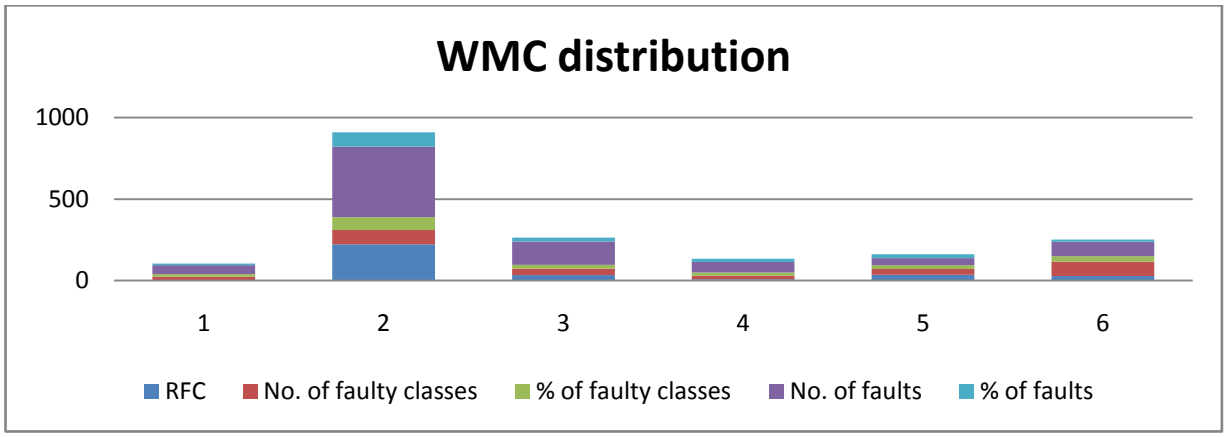

Fig 2: WMC metric distribution among faulty classes

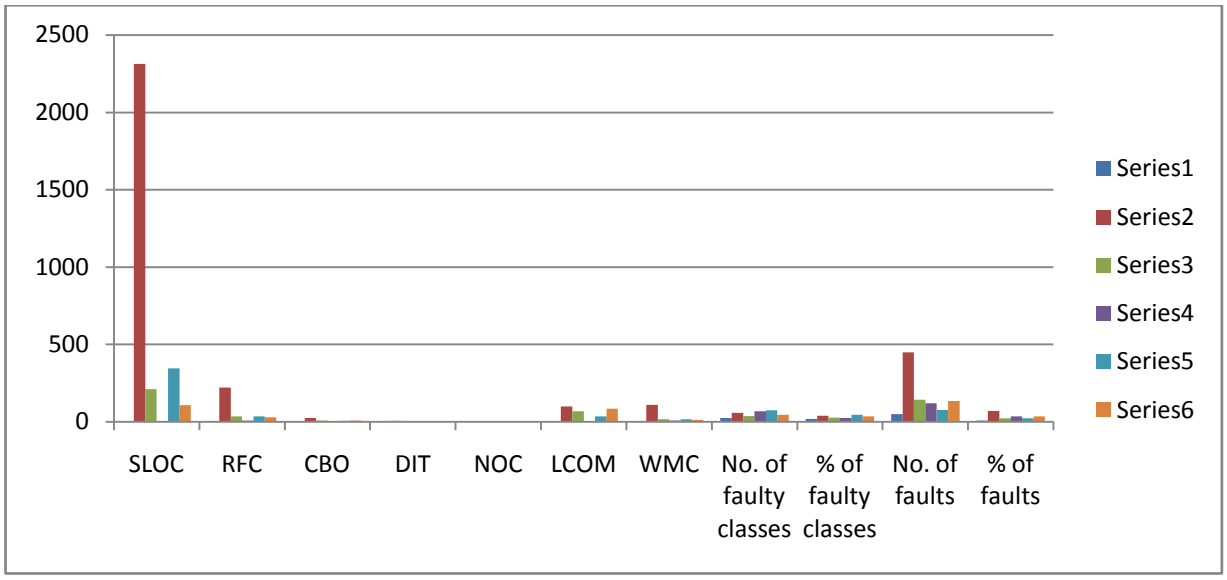

Fig 3: Combined distribution results of all metrics

From above graph, DIT, NIC metrics states that inheritance is not well properly suited to KC1 project. SLOC shows classes are larger in size. Values of DIT, NOC are very low so it is not observable in various classes of KC1 dataset. Rest of features like abstraction, encapsulation, and complexity are observable.

(b) Correlation Analysis

This technique is used to find dependency among dependent and independent variables. Metrics are taken as independent while fault prone is dependent. The formula is given by Spearman's rank function:

$\rho = 1 - 6\sum di^2 / n(n^2-1)$ **[33]**

Where di=xi-yi, n is sample size of project

Table 7: Correlation results after Spearman's rank

Metric	WMC	RFC	CBO	DIT	NOC	LCOM	SLOC
WMC	1						
RFC	0.528	1					
CBO	0.234	0.379	1				
DIT	0.134	0.654	0.460	1			
NOC	0.026	0.015	--0.001	--0.0.32	1		
LCOM	0.218	0.30	0.217	0.217	--0.28	1	
SLOC	0.625	0.509	0.572	0.345	--0.034	0.217	1

Categorization of correlation values according to Hopkins **[34]** is shown in table

Table 8: Category wise distribution of values

Range of values	category
<0.1	Trivial
0.1-0.3	minor
0.3-0.5	moderate
0.5-0.7	large
0.7-0.9	very large
0.9-1.0	perfect

From table 7, it is seen that SLOC, CBO, WMC and RFC are strongly connected with each other. So, these metrics are not independent and they lead to redundancy.

(c) Logistic Regression Analysis

It is most widely statistical technique for predicting faulty classes in system. It is of two types- univariate and multivariate.

Univariate logistic regression- It is used to analyze individual effect of each independent variables and dependent variables. It is given by equation:

$P(X1, X2, \ldots, Xn) = e^{(A0+A1X)} / 1+e^{(A0+A1X)}$ **[33]**

Where P is probability that fault was found in system and series of A are regression coefficients.

Multivariate logistic regression- It is used to find combined effect of both variables.

$P(X1,X2,\ldots, Xn) = e^{(A0+A1X1+\ldots+AnXn)} / 1+e^{(A0+A1X1+\ldots+AnXn)}$ **[33]**

The detail of regression model includes coefficient (coeff), constant, pvalue(statistical effect), R^2 value (coefficient of determination) and std err for estimation.

Table 9: Univariate regression analysis for severity fault as high

	WMC	RFC	CBO	DIT	NOC	LCOM	SLOC
Coeff	0.12	0.0004	0.016	0.010	--0.39	0.002	0.001
Constant	0.085	0.118	--0.13	0.301	0.371	0.105	0.101
R^2	0.61	0.051	0.117	0.0	0.02	0.012	0.131
p value	0.001	0.002	0.0	0.694	0.117	0.121	0
std err	0.87	0.98	0.95	1.01	1	1.004	0.93

From above table, four metrics SLOC, CBO, RFC and WMC have less p values thus they are significant. SLOC has maximum R^2 value and NOC has negative coefficient which means large number of children leads to less faults. DIT, NOC, LCOM is non significant metrics.

Table 10: Univariate regression analysis for severity fault as medium

	WMC	RFC	CBO	DIT	NOC	LCOM	SLOC
Coeff	0.200	0.051	0.412	0.116	--1.11	0.013	0.16
Constant	--0.500	1.140	--0.290	2.71	2.91	1.190	0.17
R^2	0.116	0.071	0.112	0.0	0.09	0.012	0.391
p value	0	0.02	0.0	0.82	0.193	0.14	0
std err	5.91	7.62	7.12	6.91	7.81	7.82	06.87

Table 11: Univariate regression analysis for severity fault as low

	WMC	RFC	CBO	DIT	NOC	LCOM	SLOC
Coeff	0.031	0.032	0.13	0.16	--0.17	0.005	0.004
Constant	0.210	0.510	--0.18	0.65	1.1	0.512	0.100
R^2	0.110	0.031	0.147	0.007	0.002	0.008	0.421
p value	0	0.012	0.0	0.2	0.5	0.34	0
std err	2.14	2.18	1.98	2.27	2.17	2.27	1.95

The results show that overall four metrics (SLOC, RFC, CBO and WMC) are significant.

Table 12: Multivariate regression analysis for severity fault as high

	WMC	RFC	CBO	DIT	SLOC
Coeff	--0.007	0.001	0.021	--0.32	0.001
std err	0.007	0.04	0.12	0.07	0.0
p value	0.201	0.16	0.12	0.001	0.07

From above table, WMC, DIT are negative here that are positive in univariate analysis. It occurs due to interaction between various metrics included in regression model.

Table 13: Multivariate regression analysis for severity fault as medium

	NOC	DIT	SLOC
Coeff	--0.9	--1.40	0.12
std err	0.59	0.41	0.01
p value	0.15	0.41	0.0

NOC is still negative but DIT is also negative here that leads them as non quality indicators.

5. ANSWERING RESEARCH QUESTIONS

RQ1. How to estimate effort using Lines of Code (LOC) measure?

Boehm devised a formula for estimating maintenance costs and uses a quantity called as Annual Change Traffic (ACT) which is related to number of times the requests are handled to perform changes in software.

It is given as:

$$\mathbf{ACT = KLOC_{add} + KLOC_{del} / KLOC_{tot}}$$

Then Annual Maintenance Effort is being computed on basis of ACT as:

AME (person –months) = ACT * SDE

where SDE is software development effort (person-months)

Justification of above formula:

It is given that ACT in software system is 25% per year. The initial development cost was Rs 20 lacs. Total lifetime for the software is 10 years. What is total cost of software system?

Total cost= software development cost + 0.10 * (software development cost * 0.25)

= 20 + 0.10 (20 * 0.25)

= 20.5 lacs

RQ2. How to measure % error in length of program and estimated program level if operands and operators are given? Does error in length affects quality of software?

The table consisting of operands and operators are given below:

Operators	Occurrences	Operands	occurrences
main ()	1	-	-
;	1	extern variable	1
for	2	main function	3
==	3	found	2
!=	4	lim	3
getchar	1	-	
()	1	-	
&&	3	e	4
return	1	t	2
++	4	i	1
printf	6	-	
if	1	k	3
getline	1	0	4
while	3	MAXLINE	2
Total (n1 = 14)	Total (N1= 32)	Total (n2=10)	total (N2= 25)

Program vocabulary is given by n (n1 +n2) = 24

Program length N = N1+N2 = 57

Estimated Length = NL = n1 log n1 + n2 log n2

= 14 log 14 + 10 log 10

= 25.4

% error = (57 -25.4) / 57 = 0.55 * 100 = 55%

Yes error in length affects the quality of software as less error means there is less improvement of making changes in software design.

6. CONCLUSIONS & FUTURE WORKS

Software developers and researchers have acknowledged that software quality prediction plays vital role in Object oriented design metrics. Metrics play an important role in software engineering. There are mainly two categories of software metrics viz process metrics and product metrics. Process metrics deals with how much effort is required (Person-Months), quantity of resources and methodology involved in it. Product metrics deals with specifications of project like requirements, complexity, cohesion, coupling, reliability and maintenance. Categorization of faults on basis of their severity can be helpful in performing validation of OO design metrics to predict number of faults in system. The paper validates the CK OO design metrics by employing statistical methods like correlation, regression (univariate, multivariate) under study from KC1 dataset by NASA MDP.

After applying statistical techniques, it is observed that metrics SLOC, WMC, RFC and CBO are significant and related to each other in low, medium and high severity levels. DIT and NOC are non significant metrics due to their inheritance concept and low p values. The SLOC metric is most significant that is termed as quality attribute. The study also concludes that metrics (SLOC, CBO, WMC, RFC) measuring class size, complexity, cohesion, coupling are linked more tightly rather than inheritance metrics (DIT, NOC).

Future Works

As future work, this study can be replicated by taking industrial project database in C++, Ada95 and Java. We can extend this empirical investigation to some soft computing methods like applying machine learning techniques (neuro fuzzy, neural networks) in order to produce more refined results. It will lead to better understanding of prediction capabilities of suite of CK OO metrics. The best subsets of classes in multivariate regression model can be used to produce more significant values of metrics. It is called introducing Mallow CP in multivariate analysis.

REFERENCES

[1] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering "(version 2.3),Keele University, UK, Tech. Rep. *EBSE Technical Report EBSE-2007-01*, 2007

[2] Basili, V., Briand, L., & Melo, W. A Validation of Object-Oriented Design Metrics as Quality Indicators. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 22, NO. 10, OCTOBER 1996

[3] Puja Sexsena, Monika Saini ," Empirical studies to predict fault proneness: a Review, Interntional journal of computer application (0975-8887) Volume 22- No 8 may 2011

[4] Sant'Anna, C., Garcia, A., Chavez, C., Lucena, C., and Staa, A.V., On the Reuse and Maintenance of Aspect-Oriented software: An Assessment Framework, Proceedings of Brazilian Symposium On Software Engineering (SEES'03), pp. 19-34, 2003.

[5] Chidamber, S. R. and Kemerer, C. F., A Metrics Suite for Object-Oriented Design, IEEE Transactions on Software Engineering, vol. 20 no. 6, pp. 476–493, 1994

[6] Zhao, J. and Xu, B., Measuring aspect cohesion, Proceedings of 7th International Conference on Fundamental Approaches to Software Engineering (FASE'04), Lecture Notes in Computer Science, Volume 2984, Springer-Verlag, pp. 54–68, 2004.

[7] Zhao, J., Measuring coupling in aspect-oriented systems", Technical report, Information Processing Society of Japan (IPSJ), 2004.

[8] Kumar, R., Grover, P.S., and Kumar, A., A Fuzzy Logic Approach to Measure Complexity of Generic Aspect-Oriented Systems, Journal of Object Technology, Vol. 9, No. 3, 2010.

[9] K.P. Srinivasan, Dr. T.Devi, "A Complete and Comprehensive Metrics Suite for Object-Oriented Design Quality Assessment", *International Journal of Software Engineering and Its Applications 8(2)*, 2014, 173-188.

[10] Vanitha N, "A Report on the Analysis of Metrics and Measures on Software Quality Factors – A Literature Study", IJCSIT, Vol. 5, 2014

[11] Mukesh Bansal, Dr. C.P. Agarwal, Dr. P. Sasikala, 2012, "Predict Software Fault Proneness Using Object Oriented Metrics", international journal of computing, intelligent an communication technology, ISSN 2319-748X

[12] W. Li and S. Henry, ―Object-Oriented Metrics that Predict Maintainability , Journal of Systems and Software, vol 23, no.2, pp.111-122, 1993.

[13] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl and S.J. Aud "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 902--909, 1997.

[14] Giovanni (2000), Estimating Software Fault-Proneness for Tuning Testing Activities Proceedings of the 22nd International Conference on Software Engineering (ICSE2000), Limerick, Ireland, Jun.2000

[15] E.L. Emam, W. Melo and C.M. Javam ―The Prediction of Faulty Classes Using Object-Oriented Design Metrics , Journal of Systems and Software, Elsevier Science, pp. 63-75, 2001

[16] Dr. Rakesh Kumar and Gurvinder Kaur,"Comparing Complexity in Accordance with Object Oriented Metrics. International Journal of Computer Applications", Published by Foundation of Computer Science. BibTeX 15(8):42–45, February 2011

[17] Gyimothy, T., Ferenc, R., & Siket, I.Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 10, OCTOBER 2005

[18] Rohitt Sharma, Paramjit Singh, Sumit Sharma, "An Approach Oriented Towards Enhancing a Metric Performance", *International Journal On Computer Science And Engineering (IJCSE)*, 4(5), 2012, 743-748

[19] Sonia Montagud, Silvia Abrahao, Emilio Insfran, **"**A systematic review of quality attributes and measures for software product line**"**, *Software Quality Journal, 20(3-4),* 2012, 425-486.

[20] Sharma Aman Kumar , Kalia Arvind , Singh Hardeep," Metrics Identification for Measuring Object Oriented Software Quality", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-5, November 2012.

[21] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design." Software Engineering, IEEE Transactions on 20.6 (1994): 476-49

[22] W. Harrison, "Software Measurement: A Decision-Process Approach," Advances in Computers, vol. 39, pp. 51-105,1994

[23] V Basili and D Hutchens, "Analyzing a Syntactic Family of Complexity Metrics," IEEE Trans. Software Eng, vol 9, no. 6, pp. 664-673, June 1982

[24] V. Basili, R. Selby, and T.-Y. Phdips, "Metric Analysis and Data Validation Across Fortran Projects," IEEE Trans Software Eng, vol.9, no. 6, pp 652-663, June 1993

[25] S.D. Conte, **H.E.** Dunsmore, and V.Y. Shen, *Software Eng. Metrics and Models*, Benjamin/Cummings, 1989.

[26] B. Boehm, *Softzkre Eng. Economics*, Prentice-Hall, 1981

[27] F. McGarry, R. Pajersk, G. Page, S. Waligora, V. Basili, and M. Zelkowitz, *Software Process Improvement in the NASA Software Eng. Laboratory*. Camegie Mellon Univ., Software Eng. Inst., Technical Report CMU/SEI-95-TR-22, Dec. 1994

[28] N.I. Churcher and M.J. Shepperd, "Comments on 'A Metrics Suite for Object-Oriented Design,"' *IEEE Trans. Software Eng.*, vol. 21, no. 3, pp. 263-265, Mar. 1995

[29] Chidamber, S.R. and C.F. Kemerer, 1994. A metrics suite for object-oriented design. IEEE Trans. Software Eng., 20: 476-493. DOI: 10.1109/32.295895.

[30] Sayyad Shirabad, J. and Menzies, T.J. (2005) The PROMISE Repository of Software Engineering Databases., School of Information Technology and Engineering, University of Ottawa, Canada. Available: http://promise.site.uottawa.ca/SERepository

[31] Yogesh Singh, Arvinder Kaur & Malhotra, Ruchika (2010). Empirical validation of object-oriented metrics for predicting fault proneness models. Software Quality Journal, 18(1), 3-35.

[32] http://promise.site.uottawa.ca/SERepository/datasets/kc1.arff

[33] Akhilendra Singh Chouhan, Sanjay Kumar Dubey. Analytical Review of Fault Proneness for Object Oriented Systems. IJSER, ISSN 2229-5518, vol.3, Issue 12, Dec 2012

[34] Hopkin, W.G. (2006). A new view of statistics, Sport science, MDP. Available online http://sarpresults.nasagov/viewresearch

[35] R. Subramanayam and M. Krishanan, "Empirical Analysis of CK Metrics for Object Oriented Design Complexity: Implications for Software Defects." IEEE Transactions on Software Engineering, 29(4):297-310, 2003.