ISSN 2229-5518

Analytical Review of Fault-proneness for Object Oriented Systems

Akhilendra Singh Chauhan, Sanjay Kumar Dubey

Abstract— There are a number of paradigms that are used to design the software system but now a days object oriented paradigm are very much used. We all always want high quality software that is possible only when we are able to measure the various aspects of the system like coupling, cohesion, size, inheritance etc. To measure these aspects of the software system there are a number of metrics through which we can quantify these aspects. These metrics can also be used to predict the fault proneness of the system. Early prediction of the fault proneness leads us to test only those classes which are found to be fault prone. Thus these metrics will help us to design high quality software. The aim of this paper is to review the previous research papers which are related to object oriented metrics and fault proneness. There are a large number of metrics used in the object oriented system to predict the fault proneness and each study uses a different combination of metrics on different data sets to predict the fault proneness. In each study different approaches are used to predict fault proneness like logistic regression, machine learning, Bayesian method etc. As this paper is all about review of previous papers published in various journals and conferences, so this paper laid down the year of publishing, Authors, metric used, methods used, data set used and availability of data sets.

Index Terms— Software Quality, Object Oriented, Fault, Metrics, Fault-proneness, Testing Methodology, Logistic regression

1 INTRODUCTION

Over the past years, software quality has become one of the most important requirements in the development of systems. Fault-proneness estimation could play a key role in quality control of software products [3]. To maintain this quality and develop fault free software, almost in every organisation that is involved in the software development there are various activities that have to be performed. Such an activity is Testing. Software testing is a critical and essential part of software development that consumes maximum resources and effort.

Software testing almost takes at least half of the resources and still doesn't assure the 100% correctness of the system. Also every part of the software seems impossible to test. That's why everyone wants to focus or test only those parts of the software those have high probability to be fault-prone. The aim of various researches is to find such parts of system. Such parts includes classes, methods, inheritance paths etc. To find such parts of the system there are various methods such as logistic regression, linear regression, machine learning methods, Bayesian methods, neural networks etc. To find fault prone parts a number of metrics are used that includes CK metric suite, Briand metrics, MOOD, QMOOD, L&K etc.

A Fault is a bug or anomaly in a system that may cause the system to behave incorrectly.

Fault proneness is defined as the probability of fault detection in a class of Object Oriented System [2].

Metrics are the measures that quantify the degree to which a system, sub-system, or process possesses a given attributes.

In this paper, we attempt to study various models and metrics used to estimates the fault-proneness in different object oriented systems. In this paper we have provided the review of related paper from 2000 to 2012 that are published in different journals and conferences such as Software quality journal, IEEE transactions on software Engineering, Journal of computer science and technology etc. In this study we laid down Name of Authors, Year of publishing, metrics used, data sets used and method used to predict the fault proneness.
We organised this paper as:

- Section 2 provides the Research Methodology.
- Section 3 provides study of Evaluation Technique.
- Section 4 provides Analysis.
- Section 5 provides the Futute scope.
- Conclusion is presented in Section 6.
- Appendix-A provides the summary of analysis.
- Abbreviations of various terms is presented in Appendix-B
- Title of reviewed papers is given in Appendix-C.

2 RESEARCH METHODOLOGY

There are various methodologies that can be used to make a research such as

- By making questionnaires
- By using online tools
- By mail surveys
- By interviews
- By journals and research papers etc.

In this study we used the journals and research papers methodology. We have searched and reviewed a lot of previous

- *Akhilendra Singh Chauhan is currently pursuing master's degree program in Computer Science and Engineering in Amity University, Noida, India, PH-09650160023. E-mail:chauhan.akhilendra23@gmail.com*
- *Sanjay Kumar Dubey is currently working as Assistant Professor in CSE Department in Amity University, Noida, India, PH-08826035766. E-mail: skdubey1@amity.edu*

papers in the related topic published in various journals and conferences. Then we choose the most appropriate papers as suggested by the senior Assistant Professor of Amity University and mainly focused on methods used to predict the fault proneness, data set used to analyse the fault proneness, metrics used and analysis method.

3 STUDY OF EVALUTION TECHNIQUES

There are various techniques that can be used to evaluate the fault proneness in systems. Some of them are briefly explained here-

- **Corelation Analysis:** Corelation analysis is used to correlate or find out the dependency among variables. Many Authors used correlation analysis to find the fault proneness. Considered metrics are taken as independent variables and fault proneness is taken as dependent variable. Spearman's rank corelation analysis is commonly used. For a sample size n, the n scores Xi and Yi are first converted to ranks xi and yi and then rank co-effiecient is computed as

  $$\rho=1- 6\sum di^2 / n(n^2-1)$$

  Where di=xi-yi

- **Logistic Regression:** Logistic Regression is the most widely used statistical technique to predict the fault proneness. Logistic Regression can be univariate and multivariate. It is used to find the effects of independent variables (metrics are taken as independent variables) and dependent variables (fault proneness is taken as dependent variable). Univariate Logistic Regression analysis is used to find the individual effect of each independent variable on the dependent variable.

  $$P(X_1, X_2, \ldots, X_n) = e^{(A0+A1X)} / 1+e^{(A0+A1X)}$$

  Where P is the probability that a fault was found in a class during testing and Ai's are the regression coefficients corresponding Xi's (metrics).

  Multivariate Logistic Regression analysis is used to find out the combined effect of independent variables on the dependent variable.

  $$P(X_1, X_2,\ldots, X_n) = e^{(A0+A1X1+\ldots.+AnXn)} / 1+e^{(A0+A1X1+\ldots.+AnXn)}$$

- **Machine Learning Techniques:** Machine Learning Techniques are the scientific techniques that take empirical data as inputs and produces patterns or predictions based on the given data. They can also recognise complex patterns and make intelligent decisions based on the input data. There are various machine learning techniques that are used to predict fault proneness. Some of them are titled below-
  - Decision Tree Learning
  - Artificial Neural Networks
  - Genetic Programming
  - Clustering
  - Support Vector Machines
  - Bayesian Networks

4 ANALYSIS

We have analysed and reviewed many papers published in different journals, conferences etc related to fault proneness in object oriented system. We focused on variables used, method used to analyse the fault proneness and dataset used and availability of these datasets.

There are various methods like statistical analysis, logistic regression and neural networks etc which are used by authors to analyse and predict the fault proneness. Such models are developed by using past data and can be applied for identifying faulty classes in future applications or future releases [4]. We analyse that most of the authors use logistic regression.

Authors used various object oriented metrics to predict faults or defects in various parts of object oriented system. After reviewing the various papers it has been seen that CK metric suite is very much used in the study. To predict the faults or defects Authors used dataset from various domains like dataset from public domain, dataset from open source or from commercial domain. It also has been seen from this review that dataset from open source systems is mostly used. The reason is that it is available easily to all without pay anything.

Summarily:

- Logistic Regression is most widely used technique among various techniques.
- CK metrics suite is mostly used that includes WMC, DIT, NOC, CBO, and RFC.
- Dataset from open source is commonly used.

5 FUTURE SCOPE

This paper only has the summary of review of previous papers on object oriented metrics & fault proneness. As the future scope we will extend this review to a research by applying a method to a dataset by considering a set of metrics to predict or detect the fault proneness in a dataset.

6 CONCLUSION

After this review it has been concluded that there are various methods, metrics and datasets used by authors in their studies but most commonly used method is logistic regression, mainly used metric suite is CK metric and some authors have define its own metric to predict fault proneness, most common dataset used is from open source software. It is also concluded that early prediction of faults or defects in the systems help us to test only faulty parts that save a lot of resources & efforts. This increase the productivity and chances of success of a system.

APPENDIX-A: SUMMARY OF REVIEW

S.N	Journal Name	Year	Authors	Metrics Used	Methods	Dataset Used to Predict Fault Proneness	Availability of Dataset
1.	National Research Council Canada, Council National de researches Canada (NRC-CNRC)	2000	Daniela Glasberg, Khaled El Emam, Walcelio Melo, Nazim Madhavji	WMC, DIT, ACAIC, OCAIC, DCAFC, OCAFC, ACMIC, OCMIC, DCMEC, OCMFC, WMC	Logistic Regression	A java application that implements an XML editor and contain 145 classes	Commercial
2.	IEEE Transaction on Software Engineering	2000	Michelle Cartwright, Martin Shepperd	ATTRIB, STATES, EVNT, READS, WRITES, DELS, RWD, DIT, NOC, LOC, LOC-B, LOC-H, DFCT	Shlaer mellor method	Subsystem of a telecommunications product written in C++ having 133000 lines	Commercial
3.	European Conference on Software	2001	F. Fioravanti, P. Nesi	Coupling Metrics, Cohesion Metrics, Inheritance	Logistic Regression	A Small system	Developed by Students
4.	ICSE ACM	2002	Giovanni Denaro, Mauro Pezze		Logistic Regression	Apache 1.3 & 2.0	Open source
5.	IEEE Transaction on Software Engineering	2003	R. Subramanayam, M. Krishanan	Subset of CK metric suite: WMC, CBO, DIT, LOC	Pearson & Weighted Least Sqaure	405 C++ classes and 301 Java classes	Commercial
6.	IEEE Transaction on Software Engineering	2005	T. Ostrand, E. Weyuker, R. Bell	Size, Prior faults, Age, file status, program type	Negative Binomial	Two Object Oriented system	Commercial
7.	IEEE Transaction on Software Engineering	2005	Tibor Gyimothy, Rudolf Ferenc, Istvan Siket	WMC, DIT, RFC, NOC, CDO, LCOM, LComN, LOC	Logistic Regression, linear machine learning	Mozilla(version 1.0-1.6)	Open source
8.	IEEE Transaction on Software Engineering	2006	Yuming Zhou, Hareton Leung	WMC, DIT, RFC, NOC, CBO, LCOM, SLOC	Logistic Regression, machine learning methods	Dataset, KCI from NASA in C++ contains 145 classes and 2017 methods	Public domain
9.	Journal of Object Technology	2006	K.K. Aggrawal, Yogesh, Arvinder Kaur, Ruchika Melhotra	RFC, NOA, NOM, WMC, CBO, DAC, MPC, CF, LCOM, TCC, LCC, ICH, MHF, AHF, NOC, DIT, MIF, AIF, NMO, PF, Reuse ratio, specialisation ratio	Mean, Median, Variance, PCA	3 java projects	Commercial
10.	ACM	2006	Erik Arisholn, Lionel C. Briand	Cc, LCOM, EXT, HIER, LOC,INST, MOD, INTR, PACK, RFC, MPC, FIN, FOUT, NSUP, NSUB	Logistic Regression, PCA	A legacy java system contains 1700 classes	Commercial

11.	Journal of Computer Science and Technology	2007	Piotr Tomaszewski, Lars Lundberg, Hakan Grahn	Coup, NoC, Base, WMC, RFC, DIT, LCOM, Stmt, StmtExe, StmtDecl, Comment, Maxcyc, ChgSize, CtC	Linear Regression, Co-relation analysis	Two large systems from Telecommunication Industry contain 800 and 1000 classes.	Commercial
12.	IEEE Transaction of Software Engineering	2007	Hector M.Olague, Letha H. Etzkorn	WMC, DIT, NOC, CBO, RFC, LCOM, AHF, MHF, AIF, MIF, Q MOOD-ANA, QMOOD-CAM, QMOOD-CIS, QMOOD-DAM, QMOOD-DCC, QMOOD-MOA, QMOOD-MFA, QMOOD-NOM	Logistic Regression, Co-rrelation analysis	Rhino software(six version)in java	Open source
13.	IEEE Transactions on Software Engineering	2007	Ganesh J. Pai, Joanne Bechta Dugan	WMC,DIT, RFC, NOC, CBO, LCOM, SLOC	Bayesian method	Dataset KCI from an OO software in C++, having 2107 methods & 145 classes	Open source
14.	Journal of Zhejiang University	2008	Peng Huang, Jie ZHU	CBO, CSAO, CSA, CSI, CSO, DIT, LOC, LOCM, NAAC, NAIC,NAOC, NOIC, NPavgC, NSUB, OSavg, PA, PPPC, RFC,SLOC, TLOC, WMC	Layered kernel, support vector machine	Artificial dataset, A modern communication system by Alcatel-lucent	Commercial
15.	ACM	2009	Michael English Lero, Chris Exton, Irene Rigon Lero, Brendan Cleary	NMC, NOC, DIT, CBO, RFC, LOC	Logistic Regression	CVS repository & Bugzilla database	Open source
16.	Proceeding of the World Congress on Engineering	2009	Yogesh Singh, Arvinder Kaur, Ruchika Melhotra	CBO, LCOM, NOC, DIT, WMC, RFC, SLOC	Support vector machine, ROC analysis	KCI NASA data set having 145 classes developed using C++	Public domain
17.	IEEE Transactions of Software Engineering	2009	Ana Erika Camargo Cruz, Koichiro Ochimizu	CBO, RFC, WMC	Logistic Regression	E-commerce system, A banking system written in java	Open source
18.	Software Qual J.	2010	Yogesh Singh, Arvinder Kaur, Ruchika Melhotra	CBO, RFC, LCOM, NOC, DIT, WMC, SLOC	Logistic Regression & machine learning methods	Dataset, KCI from NASA in C++ having 145 classes, 2017 methods	Public domain

19.	International Journal of Computer Application	2010	Dr. M.P. Thapaliyal, Garima Verma	WMC, CBO	Linear Regression	50 java classes of different projects	Simple projects developed by students
20.	World Academy of Science, Engineering and Technology	2010	Parvinder S.Sandhu, Jagdeep Singh, Vikas Gupta, Mandeep Kaur, Sonia Manhas, Ramandeep Sidhu	CBO, LCOM, NOC, DOI, WMC, RFC, NPM, COC	K-means based clustering approach	JEdit in Java	Open source
21.	Software Engineering & International Journal	2011	Ruchika Melhotra, Yogesh Singh	WMC, DIT, NOC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, MFA, CAM, IC, CBM, AMC	Logistic Regression, machine learning	Data set Arc, contains 234 classes	Open source
22.	European Journal of Scientific Research	2011	P.M.Shanthi, K.Duraiswamy	WMC, DIT, NOC, CBO, RFC, LCOM, Ce, Ca, I, A, Dn, LOC, MLOC,SLOC,NBD	Spearman's rank correlation	Rhino software	Open source
23.	International Journal of Computer Application	2011	Pradeep Singh, K.D. Chaudhary, Shrish Verma	WMC, DIT, RFC, NOC, CBO, RFC, LCOM, LOC	Spearman's rank correlation	Two open source, java projects of emulators, Jac64,JMShell Atari2600	Open source
24.	International Journal Computer Technology & Application	2012	Amjan Shaik, M.Sudhir Kumar, Md. Riyazuddin, S.V. Achuta Rao, Mohd. Zainuddin	TC,CBO,RFC,WMC,CA,DIT, NOC, NPM, LCOM	PCA(principal component analysis)	Five java projects (1) BLACK DUCK KODERS (2) STRAR um2 (3) Open Office Draw 3.0 (4) Infra Recorder 0.50 (5) Gimp shop 2.2.11	Open source
25.	Proceeding of the World Congress on Engineering	2012	Ruchika Melhotra	CBO, LCOM, NOC, DIT, WMC, RFC, NPM, CS, LCOM3, DAM, MOA, MFA, CAM, IC, CBM, AMC, CC, LOC	Logistic Regression	Apache Ant 1.7 software written in java, 745 classes	Open source

APPENDIX-B: ABBREVIATIONS

NBD	Nested Block Depth
AHF	Attribute Hiding Factor
MHF	Method Hiding Factor
AIF	Attribute Inheritance Factor
MIF	Method Inheritance Factor
QMOOD-ANA	Quality Model for OO Design Average Number Ancestors
QMOOD-CAM	Quality Model for OO Design Cohesion Among Methods
QMOOD-CIS	Quality Model for OO Design Class interface Size
QMOOD-DAM	Quality Model for OO Design Data Access Metric
QMOOD-DCC	Quality Model for OO Design Direct Class Coupling
QMOOD-MOA	Quality Model for OO Design Measure Of Aggregation
QMOOD-MFA	Quality Model for OO Design Measure of Functional Abstraction
QMOOD-NOM	Quality Model for OO Design Number Of Methods
DOI	Depth Of Inheritance
NPM	Number of Public Methods
ATTRIB	Count of Attributes per class
STATES	Count of States per class
EVNT	Count of Events per class
READS	Count of all Reads access by class
WRITES	Count of all Writes access by class
DELS	Count of all Deletes access by class
RWD	Count of synchronous accesses sum of READS,WRITES, DELS per class
LOC-B	Line Of Code (Body)
LOC-H	Line Of Code (Header)
NSUB	Number Of Subclass
OSAVG	Average Operation Size
PA	Private Attribute Usage
PPPC	Percentage of Public Protected Members
SLOC	Source Line Of Code
TLOC	Total Line Of Code
EXT	Number of External methods called
HIER	Number of method called in class Hierarchy
INST	Number of Instance
MOD	Number of Modifiers
PACK	Number of Package imported
MPC	Message Passing Coupling
FIN	The sum of the number of unique methods that call the methods in the class
FOUT	Number of distinct non inheritance related class on which a class depends
NSUB	Number of SUBclass
NSUP	Number of SUPerclass
LCOMN	Lack of Cohesion on Methods Allowing Negative value
Ce	Efferent Coupling
Ca	Afferent Coupling
I	Instability
A	Abstractness
Coup	Coupling
Base	Number of Base Classes
Stmt	Number of Statements
StmtExe	Number of Executable Statements
StmtDecl	Number of Declarative Statements
Comment	Number of Comment Lines
MaxCyc	Maximum Cyclomatic Complexity

CtC	Ratio Comment to Code

Dn	Normalized Distance from main sequence
MLOC	Method line Of Code
SLOC	Source Line Of Code
TC	Transitive Coupling
CBO	Coupling Between Object
RFC	Response For Class
LCOM	Lack Of Cohesion
NOC	Number Of Children
DIT	Depth of Inheritance
WMC	Weight of Methods per Class
NPM	Number of Public Methods
LCOM3	Lack Of Cohesion in Methods
DAM	Data Access Metric
MOA	Measure Of Aggregation
MFA	Measure of Functional Abstraction
CAM	Cohesion Among Methods of class
IC	Inheritance Coupling
CBM	Coupling Between Methods
AMC	Average Method Complexity
CC	Cyclometic Complexity
LOC	Lines Of Code
CSAO	Class Size (Attribute & Operation)
CSA	Class Size (Attribute)
CSI	Class Specialisation Index
CSO	Class Size (Operation)
NAAC	Number of Attributes Added in the Class
NAIC	Number of Attributes Inherited in the Class
NAOC	Number of Attributes Operation added in the Class
NAIC	Number of Attributes Inherited in the Class
NPavgC	Avg number of method Parameters in the class

APPENDIX-C: TITLES OF REVIEWED PAPERS

S. N.	Title of Papers
1.	Validating Object-oriented Design Metrics on a Commercial Java Application
2.	An Empirical Investigation of an Object-Oriented Software System
3.	A Study on Fault-proneness Detection of objects Oriented Systems
4.	An Empirical Evalution of Fault Proneness Models
5.	Empirical Analysis of CK Metrics for Object Oriented Design Complexity: Implications for Software Defects
6.	Predicting the Location and number of Faults in Large Software Systems
7.	Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction
8.	Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults
9.	Improving Fault Detection in Modified Code- A Study from the Telecommunication Industry
10.	Empirical Study of Object-Oriented Metrics
11.	Predicting Fault-prone Components in a Java Legacy System
12.	Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes
13.	Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods
14.	Predicting the fault-proneness of class hierarchy in object-oriented software using a layered kernel
15.	Fault Detection and Prediction in an Open-Source Software Project
16.	Software Fault Proneness Prediction Using Support Vector Machines
17.	Towards Logistic Regression Models for Predicting Fault-prone Code across Software Projects
18.	Empirical validation of object-oriented metrics for predicting fault proneness models
19.	Software Defects and Object Oriented Metrics – An Empirical Analysis
20.	A K-Means Based Clustering Approach for Finding Faulty Modules in Open Source Software Systems
21.	On the Applicability of Machine Learning Techniques for Object Oriented Software Fault Prediction
22.	An Empirical Validation of Software Quality Metric Suites on Open Source Software for Fault-Proneness prediction in Object Oriented Systems
23.	An Investigation of the Relationships between Software Metrics and Defetcts
24.	Transitive Coupling (TC) and Fault Proneness (FP) in Object Oriented Systems: A New Methodology
25.	A Defect Prediction Model for Open Source Software

7 REFERENCES

[1] Software Fault prediction for Object Oriented Systems: A literature Review by Ruchika Malhotra and Ankita Jain. ACM Software Engineering Notes Sept-2011 Volume 36 Number 5

[2] Huang, P. and Zhu, j. 2008. "Predicting the Fault-proneness of Class Hierarchy in Object-Oriented Software using a Layered Kernel". Journal of Zhejiang University SCIENCE A, 9(10):1390-1397.

[3] ICECCS '05 Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems.

[4] "The Prediction of Faulty Classes Using Object-Oriented Design Metrics" by Khaled El Emam and Walcelio Melo November 1999

[5] C. Catal and B. Diri, "A Systematic Review of Software Fault Prediction Studies" Expert Systems with Applications, vol. 36, no. 4, pp. 7346–7354, 2009.

[6] Cartwright, M. and Shepperd, M. 2000. "An Empirical Investigation of an Object-Oriented Software System". IEEE Transactions on Software Engineering, 26(8):786-796.

[7] Briand, L.C., Wiist, J., Daly, J.W. and Porter, D.V. "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented systems".

[8] Glasberg, d., Emam, K. E., Melo, W. and Madhavji, N. "Validating Object-Oriented Design Metrics on a Commercial Java Application. Technical report: NRC 44146.

[9] Emam, K.E., Melo, W. and Machado, J.C. 2001. "The Prediction of Faulty Classes using Object Oriented Design Metrics". The journal of system and software, 56: 63-75H.

[10] Briand, L. C., Wiist J. and Lounis, H. 2001. "Replicated Case Studies for Investigation Quality Factors in Object-Oriented Design". Empirical Software Engineering International Journal (Toronto, Ont.), 6(1):11-58.

[11] Gyimo thy, T., Ference, R. and Siket, I.2005. "Empirical Validation of Object-Oriented Metrics on Open Software for Fault prediction. IEEE Transaction on Software Engineering, 31(10):897-901.

[12] Zhou, Y. and Leung, H. 2006. "Empirical Analysis of Object Oriented Design Metrics or Predicting High and Low Severity Faults". IEEE transactions software engineering, 32 (10): 771-789

[13] Arisholm, E. and Briand, L.C. 2006. "Predicting Fault prone Components in A Java Legacy System". ISESEJ

[14] Pai, G.J and Dugan, J.B. 2007. "Empirical Analysis of Software Fault Content and Fault Proneness using Bayesian Methods". IEEE Transaction on software engineering, 33(10):675-686.

[15] Tomaszewski, P., Hakanson, J., Lundberg, I., Grahn, H. 2007. "Statistical Models vs. Expert Estimation for Fault Prediction in Modified Code-an Industrial case Study." The Journal of Systems and Software. 81: 1868-1882.

[16] Shatnawi, R. and Li, W.2008. "The Effectiveness of Software Metrics in Identifying Error-prone Classes in Post-release Software Evolution process". The journal of systems and software, 81: 1868-1882.

[17] Lero, M.E., Exton, C., and Lero, I.R., 2009. "Fault Detection and Prediction in Open source software projects." Proceeding: PROMISE '09 proceeding of the 5th International Conference on Predictor Models in Software Engineering.

[18] Singh, Y Kaur, A. and Malhotra, R. 2009. "Software Fault proneness Prediction using Support Vector Machines." Proceeding of the World Congress on Engineering 2009, Vol 1, WCE 2009, July 1-3, London.

[19] Zhou, Y., Xu. B. amnd Leung, h. 2010. "On the Ability of complexity Metrics to predict fault-prone Classes in Object Oriented Systems.

[20] Singh, Y., Kaur, A. and Malhotra, R. 2010. "Empirical Validation of Object Oriented Metrics for predicting fault proneness Models".

[21] Singh, Y., Malhotra, R. 2011. "On the Applicability of Machine Learning Techniques for Object Oriented Software Fault Prediction." Software Engineering: An International Journal (SEIJ) VOL 1, No 1, SEPTEMBER 2011.

[22] Malhotra, R. "A Defect Prediction Model for Open Source Software." Proceedings of World Congress on Engineering 2012 Vol II

[23] "Transitive Couling (TC) and Fault-Proneness (FP) in Object Oriented System: A New Methodology." Aman Shaik et al, Int. J. Computer Technology and Application, Vol 3(2), 639-649.

[24] Shanthi, P.M., Duraiswamy, K., "An Empirical Validation of Software Quality Metric Suites on Open Source Software for Fault Proneness Prediction in Object Oriented Systems." European Journal of Scientific Research. ISSN 1450-26X Vol 51 No. 2(2011).

[25] Singh, P., Chaudhary, K.D., Verma, S. "An Investigation of Relationships between Software Metrics and Defects." International Journal of Computer Application (0975-8887) Vol. 28, No. 8 August 2011.

[26] F. Fioravanti and P. Nesi, "A Study on Fault Proneness Detection of Object Oriented systems." In Proceedings of the 5th European Conference on Software Maintenance and Re-engineering, pages 121-130, 2011.

[27] T. Ostrand, E. Weyuker, and R. Bell, "Predicting the Location and number of Faults in large Software System." IEEE Transactions of Software Engineering, 31(4):340-355, April 2005.

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