ISSN: 2278-5213

RESEARCH ARTICLE

A hybrid approach for the prediction of fault proneness in object oriented design using fuzzy logic

Rajinder Vir[1*] and P.S. Mann[2]
[1]Dept. of Computer Science, CT Institute of Engineering and Management Technology, Jalandhar
[2]Dept. of Information Technology, DAV Institute of Engineering and Technology, Jalandhar, India
jas4281@gmail.com[*]; psmaan@hotmail.com; +91 8427700837, 9888395367

Abstract

Empirical studies conducted by the researchers on object-oriented design metrics are useful for forecasting the fault-proneness of classes in object-oriented design. In this study, we propose an integrated hybrid model to empirically investigate the fault-proneness of object-oriented design. We will use a subset of the Chidamber and Kemerer suite and all of the MOOD metrics to predict fault-proneness of object oriented design. Moreover with the increasing demand for quality software there is an increase in metrics which can measure OO attributes such as coupling, cohesion and inheritance. Therefore, there is a need for quality models that investigate the association between these properties and quality attributes such as fault proneness, maintainability, extendibility, effectiveness or productivity, to be able to use the metrics effectively and efficiently. The aim of this study is to empirically investigate the association between object oriented design metrics and fault proneness of object oriented systems.

Keywords: Object-oriented design metrics, fault-proneness, MOOD metrics, maintainability, extendibility.

Introduction

In today's software advancement, environment object oriented design and development is gaining a lot of popularity among the researchers as it improves the software productivity, reusability and flexibility of the software. In object-oriented design, there are five key structures that should be measured: classes, messages, cohesion, coupling and inheritance. Measuring software quality in the early stages of software development is very necessary. Thus, it is the key to develop high quality software when we measure the software quality in the early stages of software development. A large number of software metrics have been proposed in software engineering to measure the quality attributes of the software in early stages. Although various researchers have proposed many metric suites to evaluate the OOD quality, the best out of them are the CK metric suite (Chidamber and Kemerer, 1994) and the MOOD metric suite (Abreu and Carapua, 1994). Any of the metric suites cannot alone reflect the quality of design.

Therefore, there is a need of an integrated hybrid mechanism to combine them into a single output. Metrics offer a mechanism for attaining more accurate estimations of project milestones, and developing a software system that contains minimal faults (Bellin *et al.,* 1994). With the help of metrics software engineers can measure and predict software processes, necessary resources for a project and products relevant for a software development effort. Various kinds of object oriented metrics are available and quite helpful in obtaining information about the software quality and fault proneness of the object oriented design.

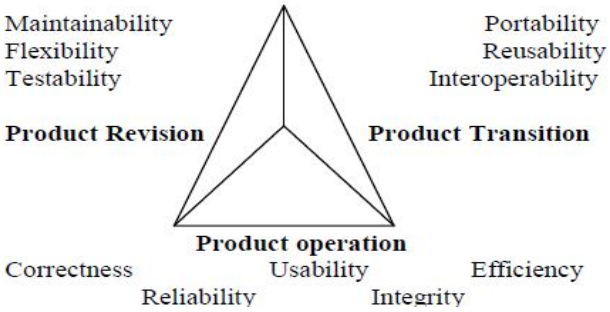
In this study, we describe how we calculated the defect index from CK metrics namely WMC, DIT, NOC and MOOD metrics viz., MHF, AHF, AIF, MIF, CF and PF. The two metric suites have been validated by Basili *et al*. (1996). A large number of metrics have been proposed in the past for so many years to confine the OO design, code and constructs. These metrics provide ways to assess the quality of software and their use in early phases of software development which can help software companies in evaluating large software development quickly and at a reasonable cost (Aggarwal *et al.,* 2005). There have been large number empirical studies evaluating the impact of OO metrics on faulty classes. Saxena and Saini (2011) provided a review of all those empirical studies from 1995 to 2010 to predict software fault-proneness with a specific focus on techniques used.

Benlarbi *et al.* (1999) surveyed that the basic premise behind the development of object oriented metrics is that they can serve as early predictors of classes that contain faults or that are closely maintain. They have shown that size can have an important confounding effect on the validity of object-oriented metrics. Khalsa (2009) proposed an algorithm using fuzzy logic to measure fault proneness and defect density of the software development process and hence can be used to minimize rework. Kamiya *et al*. (1999) proposed a new method to estimate the fault-proneness of an object class in the early phase, using several complexity metrics for object-oriented software. Four checkpoints were introduced in to the analysis/design/implementation phase and estimates were done on the fault-prone classes using applicable metrics at each checkpoint.

Menzies *et al.* (2003) compared Decision Trees, Naïve Bayes, and 1-rule classifier on the NASA software defect data. A clear trend was not observed and different predictors scored better on different data sets. Malhotra *et al*. (2010) built a Support vector machine (SVM) model to find the relationship between object-oriented metrics given by Chidamber and Kemerer and fault-proneness, at different severity levels. Malhotra (2012) founded the relation between object oriented metrics and fault-proneness using logistic regression method. The results were analyzed using open source software. Further, the performance of the predicted models was evaluated using receiver operating characteristic (ROC) analysis.

Evaluating object oriented design quality means identifying those design entities that are relevant for the analysis of their properties and relationships that exist between them. Measuring quality has been a major challenge for the software development but there is a lack of standards for measuring the quality. With the increasing popularity of object oriented software development, we need to investigate the object oriented design metrics with respect to the software quality. Since measuring the software quality early in the development phases is the key to develop high quality software system. According to McCall, following factors have an effect on software quality (Fig. 1):
a. Direct factors (such as defect proneness)
b. Indirect factors (such as extendibility, effectiveness).

Fig. 1. McCall's quality factors (Pressman, 2011).



Metrics provide useful indicators to measure different factors related to software quality and software development process. Object oriented metrics are an essential part of software development since they permit the designers to software quality early in the process, make appropriate changes that will reduce the complexity, number of defects and improve the continuing capability of the object oriented design (Abreu and Carapua, 1994). Over the past years, a significant number of object oriented metrics have been proposed by Chidamber and Kemerer (1994), MOOD metrics proposed by Abreu and Carapua (1994), Lorenz and Kidd Metric (1994), QMOOD metrics by Bansiya and Davis (2002) (Table 1-4). Out of these the CK metrics are the most popular followed by MOOD metrics.

Table 1. CK Metric Suite (Chidamber and Kemerer, 1994).

Metric	Description
Weighted Methods per Class (WMC)	It defines the number of methods in a certain class.
Depth of Inheritance Tree (DIT)	It is a measure of how many ancestor classes can potentially affect a given class.
Number of Children (NOC)	Number of direct subclasses that a certain class contains.
Lack of Cohesion among Methods (LCOM)	Number of disjunctive method pairs of a certain class.
Coupling Between Objects (CBO)	Number of coupling between a certain class and all other classes.
Response For Class (RFC)	Number of methods that can be performed by a certain class in response to a received message.

Table 2. MOOD Metric Suite (Abreu and Carapua, 1994).

Metric	Description
Attribute Inheritance factor(AIF)	It is defined as the ratio of the sum of inherited attributes in all classes of the system.
Method Inheritance Factor(MIF)	The MIF metric states the sum of inherited methods in all classes of the system under consideration.
Attribute Hiding Factor(AHF)	Measure how well attributes and properties are encapsulated.
Method Hiding Factor(MHF)	Measure how well methods and variables are Encapsulated.
Polymorphism factor(POF)	The POF represents the actual number of possible different polymorphic situation
Coupling Factor(COF)	It is defined as the ratio of the maximum possible number of couplings in the system to the actual number of coupling is not imputable to inheritance

Table 3. Relationship between CK Metrics and software quality factors (Rosenberg and Hyatt, 1997).

Metric	Software Quality Factor
AIF	Functionality, Effectiveness, Extendibility, Defect Proneness
MIF	Functionality, Effectiveness, Extendibility, Defect Proneness
AHF	Understandability, Complexity, Extendibility
MHF	Understandability, Complexity, Extendibility
POF	Complexity
COF	Complexity, Reusability

Table 4. Relationship between MOOD metrics and software quality factors (Chandra and Linda, 2010).

Metric	Software Quality Factor
WMC	Complexity, Usability, Reusability
DIT	Reusability, Understandability, Testability
NOC	Design
LCOM	Design, Reusability
CBO	Design, Reusability
RFC	Design, Usability, Testability

Research has shown that the CK Metric Suite does not account for the complexity that occurs from the object oriented design factors such as encapsulation and polymorphism but the metrics proposed by Abreu are able to measure the object oriented design aspects properly (Subramanyam and Krishnan, 2003). The metrics AHF and MHF measure the information hiding aspects of the class, PF metrics measure the polymorphism. Hence, we have used an integrated hybrid model to measure the defect Proneness of the object oriented systems. The work described in this study focuses on the use of Object Oriented (OO) metrics in predicting defect prone classes.

Software quality is controlled by many types of uncertainties that occur during software development process which makes it difficult for the designer to evaluate the software quality. Various software quality modes have been proposed over the recent years but none of them proved to be simple, practical and widely accepted. Since evaluating the quality of object oriented design is a fuzzy evaluation process. Therefore to get an accurate objective and empirical evaluation of the software quality based on defect proneness, we will use rule based fuzzy logic system proposed by Zadeh (1965) to evaluate the defect proneness of the object oriented software systems.

Fuzzy Logic is a technique used for modeling complex systems (Zadeh, 1965). Since the real world is full of vagueness fuzzy logic has proved to be very successful in many areas such as decision support and expert systems. Moreover, human reasons in fuzziness. Fuzzy logic can be constructed either without any data or little data which makes fuzzy logic superior over other data driven approaches such as neural networks, regression analysis and case based reasoning (Zadeh, 1965). Researchers have successfully applied fuzzy logic in software engineering disciplines such as effort estimation, project management. For e.g. Gray and Mac Donell developed a tool called Fulsome (Handa and Wayal, 2012) (Fuzzy Logic for software metrics), Ryder applied fuzzy logic to COCOMO and Function Point models for making effort estimation (Bhatnagar *et al.,* 2010).

Materials and methods

Proposed model: The proposed model for calculating the fault proneness uses a subset of CK metrics and MOOD metrics (Fig. 2-6).

1. A fuzzy logic model FCD-CK (Fuzzy Controller for defect – CK) is used to predict the Defect Index from three of the CK metric (WMC, DIT and NOC) as these three metrics of the CK metric suite are shown to be very good predictors of defect proneness of object oriented design and proved empirically (Subramanyam and Krishnan, 2003). To get the defect index, Mamdani fuzzy inference model is used. The three inputs are fed into the fuzzy systems. Depending upon the input values of the metric, some rules out of the total 27 rules from the knowledge base gets fired. The Mamdani inference engine is used to determine the degree of membership of firing. The technique used for defuzzification is Centroid method.

Fig. 2. FCD-CK Fuzzy controller.

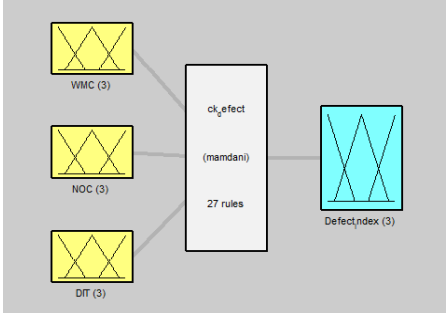
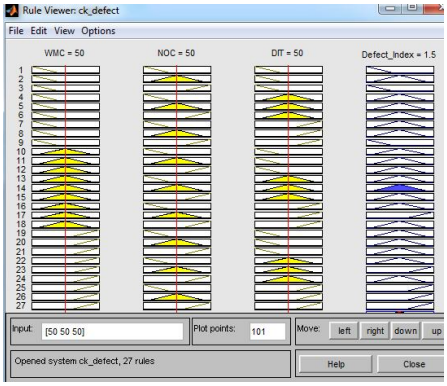


Fig. 3. Rule viewer for FCD-CK.



2. Another fuzzy model called FCD-MOOD is created which uses six inputs of MOOD metrics. Similarly as the above model is mentioned, rules from the knowledge base are fired depending upon the input values of the metrics. For MOOD metrics the Sugeno inference engine is used to determine the degree of membership of firing. We are proposing a hybrid model for calculating the defect index for fault proneness. The defuzzification technique used is wtaver.

Fig. 4. FCD-MOOD Fuzzy controller.

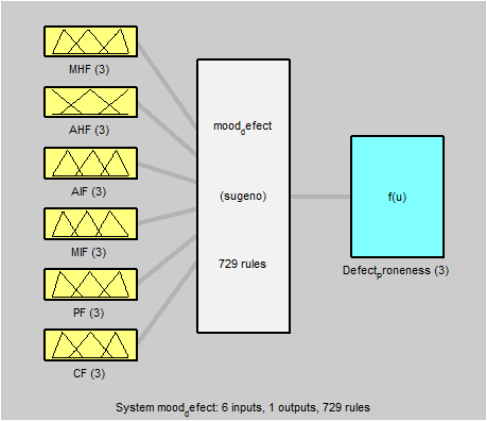


Fig. 5. Rule viewer for FCD-MOOD.

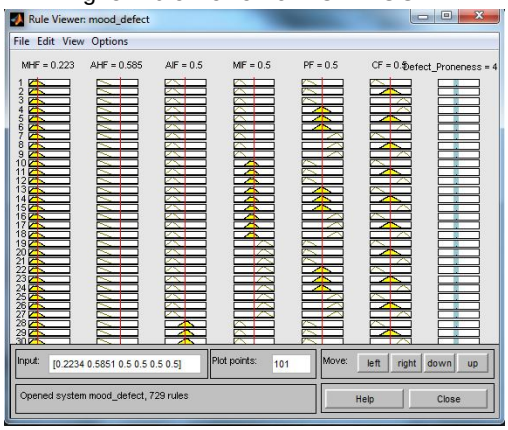
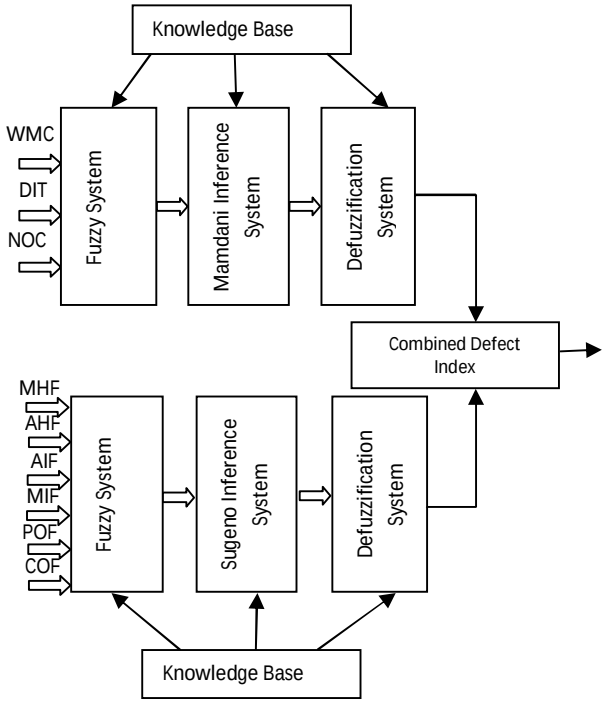


Fig. 6. Hybrid model.



Results

To perform the empirical investigation, we have used project data set named KC1 available publicly from the PROMISE data repository for validating the fuzzy model-FCD-CK. To validate the second fuzzy model using MOOD metrics we used various open source software and project analyzer tool. After creating the rule base to depict the true picture the results were obtained as shown in the form of graphs in Figures 7 to 14. The input data for the calculation of defect index is derived from various open source software's and the PROMISE data repository. The values of all the metric is computed with the help of PROJECT ANALYZER tool. The FUZZY controller called FCD-CK and FCD-MOOD were developed. The output value of the FCD_CK is called DEFECT INDEX and that of FCD_MOOD is called DEFECT PRONENESS.

Fig. 7. Graph for Defect Index vs. WMC.

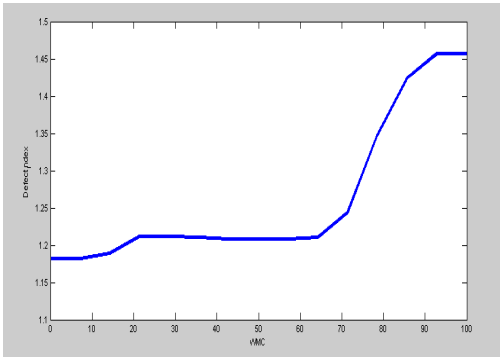


Fig. 8. Graph for Defect Index vs. NOC.

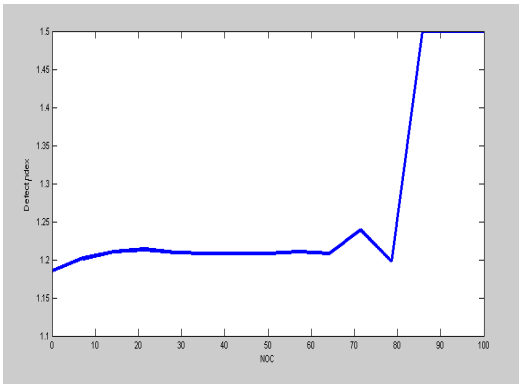


Fig. 9. Graph for Defect Index vs. DIT.

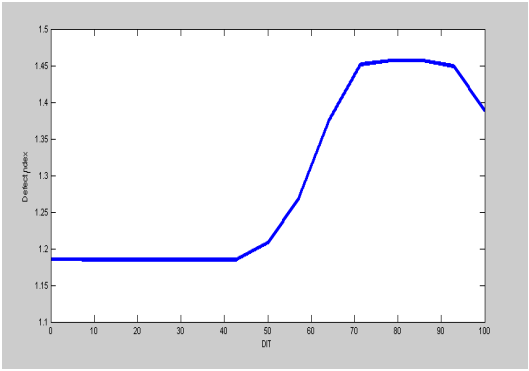


Fig. 10. Graph for Defect Proneness vs. MHF.

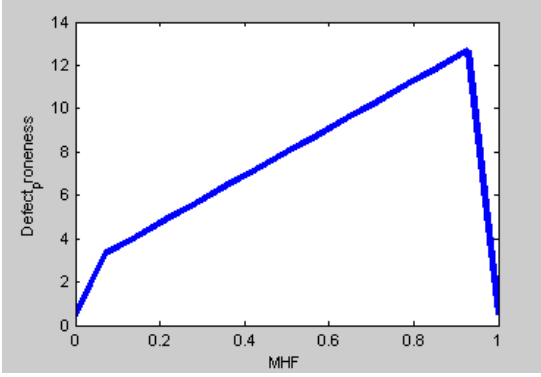


Table 5. Values of the output Metric (Defect Index Computed from the Hybrid model).

Metrics	Source Code 1	Source Code 2	Source Code 3	Source Code 4	Source Code 5	Source Code 6	Source Code 7
WMC	20	31	35	10	5.5	17	28
NOC	2	3	2	4	0	6	5
DIT	2	2	4	3	0	7	6
MHF	0.305	0.897	0.834	0	0.55	0	0
AHF	0.375	0.667	0.444	0.16	1	0.94	0.86
AIF	0.676	1	1	0.3	0	0.5	0.4
MIF	0.491	1	1	0	0	0.4	0.13
PF	0	0.8	1	0	0	0.8	0.4
CF	0.78	0.25	0.29	0.2	0	0.5	0.3
Defect_CK	1.0630	1.3739	1.4405	0.4116	0.3977	0.9253	1.3106
Defect_MOOD	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
Hybrid_Defect	1.5630	1.8739	1.9405	0.9116	0.8977	1.4253	1.8106

Fig. 11. Graph for Defect Proneness vs. AIF.

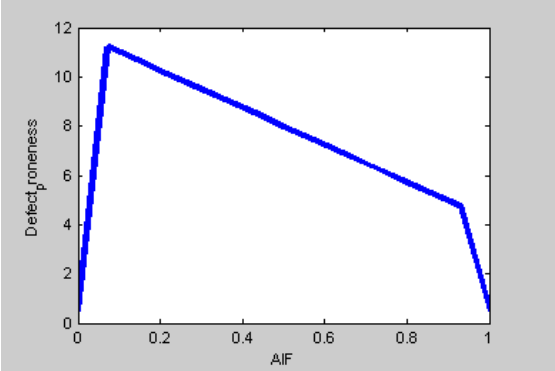


Fig. 12. Graph for Defect Proneness vs.MIF.

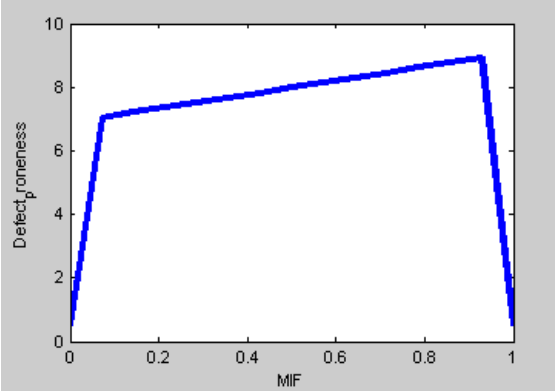


Fig. 13. Graph for Defect Proneness vs. POF.

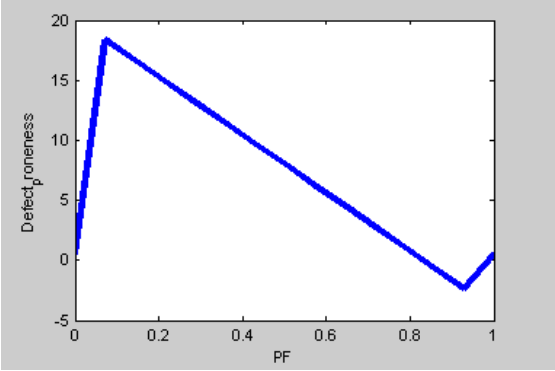
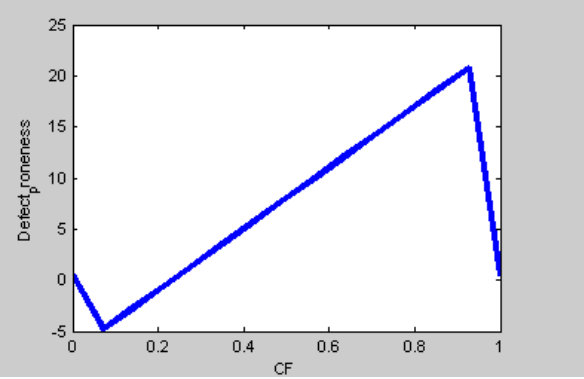


Fig. 14. Graph for Defect Proneness vs. COF.



Now, the combined defect index obtained from MAMDANI and SUGENO based FUZZY controllers is combined together to achieve the defect index as shown in Table 5. From the values, it is clear that classes with lesser value of defect index are less prone to faults as compared to classes with higher value of defect index and hence, they need to be reconsidered.

Conclusion

We have analyzed the performance of proposed model using the fuzzy logic approach. The proposed model includes the metrics given by Chidamber and Abreu (1994). The model can be effectively used for predicting the faulty classes in the early phases of SDLC which in result minimize the effort of the software developers. Hence, the model can help in improving the quality and reducing faulty classes in the OOD early. The study can be extended to deal with object oriented design specifications. More combinations of the different available metrics can be integrated depending upon the requirements of the user. We used 3 metrics of CK and 6 metrics of MOOD metric suite, correlation of other metrics can also be examined and they can also be used to estimate the prediction of fault proneness. We used fuzzy logic approach another approaches like neural networks, case based systems can also be used to make the system more effective. We can also find the solution to other inconsistencies to which the solution has not been proposed yet.

References

1. Abreu, F.B. and Carapua, R. 1994. Candidate metric for OOS within taxonomy framework. *J. Syst. Software*. 26: 1.
2. Aggarwal, K.K., Singh, Y., Kaur, A. and Malhotra, R. 2005. Software reuse metrics for object-oriented system. Proc. of 3rd ACIS Int. Conf. on software engineering research, management and applications (SERA'05). *IEEE Computer Soc*. pp.48-55.
3. Bansiya, J. and Davis, C.G. 2002. A hierarchical model for object-oriented design quality assessment. *IEEE Trans. Software Engg.* 28: 1.
4. Basili, V.R., Briand, L.C. and Melo, W.L. 1996. A validation of object oriented design metrics as quality indicators. IEEE Trans. on software engineering. 22(10): 751-761.
5. Bellin, D., Tyagi, M. and Tyler, M. 1994. Object-Oriented metrics: An overview. Proc. of Conf. on advanced studies on collaborative research (CASCON '94). p.4.
6. Benlarbi, S., Emam, K.E. and Geol, N. 1999. Issues in validating object-oriented metrics for early risk prediction. Proc. of 10th Int. Symp. on software reliability engineering (ISSRE'99). Boca**.**
7. Bhatnagar, R., Bhattacharje, V. and Ghose, M.K. 2010. A proposed novel framework for early effort estimation using fuzzy logic techniques. *Global J. Comp. Sci. Technol.* 10: 14.
8. Chandra, E. and Linda, P.E. 2010. Assessment of software quality through object oriented metrics. *CIIT Int. J. Software Engg*. 2: 2.
9. Chidamber, S.R. and Kemerer, C.F. 1994. A metrics suite for object oriented design. IEEE Trans. on software engineering. 20(6): 476-493.
10. Dubey, S.K. and Rana, A. 2010. A comprehensive assessment of object-oriented software systems using metrics approach. *IJCSE*. 2: 2726-2730.
11. Handa, A. and Wayal, G. 2012. Software quality enhancement using Fuzzy logic with object oriented metrics in design. *Int. J. Comp. Engg. Technol.* (IJCET). 3(1): 169-179.
12. Kamiya, T., Kusumoto, S. and Inoue, K. 1999. Prediction of fault-proneness at early phase in object-oriented development. Proc. of 2nd IEEE Int. Symp. on object oriented real-time distributed computing (ISORC '99). pp.253-258.
13. Khalsa, S.K. 2009. A Fuzzified approach for the prediction of fault proneness and defect density. Proc. of the World Congress on Engineering. Vol. I. WCE 2009. July 1-3, London, U.K.
14. Malhotra, R. 2012. A defect prediction model for open source software. Proc. of the World Congress on Engineering. Vol. II. July 4-6. London (UK).
15. Malhotra, R., Kaur, A. and Singh, Y. 2010. Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines. *Int. J. Syst. Assurance Engg. Management*. 1(3): 269-281.
16. Menzies, T., Ammar, K., Nikora, A. and Stefano, S. 2003. How simple is software defect prediction? *J. Empirical Software Engg*. October.
17. Pressman, R.S. 2001. Software engineering-A practitioner's approach. McGraw-Hill international edition. 5th edition.
18. Rosenberg, L.H. and Hyatt, L. 1997. Software quality metrics for object oriented environments. Crosstalk J.
19. Saxena, P. and Saini, M. 2011. Empirical studies to predict fault proneness: A review. *Int. J. Computer Appl*. 22(8): 41-45.
20. Subramanyam, R. and Krishnan, M.S. 2003. Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Trans. Software Engg*. 29(4): 297-310.
21. Zadeh, L.A. 1965. Fuzzy sets, information and control. 8: 338-353.