Empirical Research in Object Oriented Software Metrics Threshold Values At Acceptable Risk Level

Sarabjit Kaur
Assistant Professor
Dept. of CSE
CTITR, Maqsudan.Jalandhar(Punjab)

Kamaljeet Singh
Assistant Professor
Dept. of FCS
GNA University, Phagwara(Punjab)

Abstract

For the software engineers, Object-Oriented metrics are more beneficial. Object-oriented design and development is becoming very popular in software development environment. Object oriented development requires not only a different approach to design and implementation, it requires a different approach to software metrics. In the current paper, we use logistic regression to investigate the threshold values against the bad smell for the Chidamber and Kemerer(CK) metrics at five different levels. Two versions of jfreechart were used as a dataset to validate the study. Only the significantly associated metrics were considered for finding the threshold values. In this study, accuracy of both the versions 4 gives more than 70% results for the selected metrics: LOC, WMC, RFC, CBO and DIT.

Keywords

Object-oriented metrics, Threshold values, Risk levels, open source software.

I. INTRODUCTION

Software quality has been a major challenge in various software projects since years. The quality of software can be evaluated using different types of software metrics. Although it has always been the major concern in software development but still lacks the outline of standards which can measure it [2]. As quality is effected by maintainability, in order to

achieve it one needs to know what characteristics of a product actually affects it. In their work, has emphasized the factors that decrease maintenance effort. These are use of structured techniques, Use of modern software, Use of automated tools, Use of data-base techniques, Good data administration, and experienced maintainers. Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment. Basically, this means that any activity that modifies software product after its release is software maintenance. Adaptive maintenance is the modification of a software product performed after delivery to keep a computer program usable in a changed or changing environment. Corrective maintenance is the reactive modification of a software product performed after delivery to correct discovered faults. Perfective maintenance is the modification of a software product after delivery to improve performance or maintainability [7]. Fowler and Beck have informally described bad smells in code as bad or inconsistent parts of the design of an Object-oriented System. Flower has identified 22 code smells and associated each of them with the refactoring transformations that may be applied to improve the structure of code [3]. Refactoring increases the code readability and maintainability. Refactoring applied at any level depends on the type of design defect found in the system and has a direct influence on the software maintenance cost [6].

The process of refactoring is applied in three different stages which are identification of problematic area, choice of appropriate refactoring technique and application of refactoring techniques. The object-oriented software metrics were investigated and the most suitable metrics is evaluated. The metrics were selected on the basis of their ability to predict different aspects of object-oriented design. Then, tool was used to measure the metric quality. An adequate result means the code base must consist of both well and badly designed systems. The logistic regression is used to investigate the threshold effects of the CK metrics suite, and then validate the thresholds to identify faulty classes. This model is used to find the threshold effects in Eclipse 2.0. [5]. In this research, the aim is to find the threshold values of object oriented metrics based on the bad smell. Then the validation of these metrics threshold values will be done. Only the significantly associated metrics were considered for finding the threshold values. The need is to find the threshold values of metrics using bad smell successfully and then validate this threshold values will be done.

2. LITERATURE REVIEW

In this section, we review the previous works that are related to the validation and threshold effects of Object-Oriented metrics. Shatnawi [14]. In this paper, we use a statistical model, derived from the logistic regression, to identify threshold values for the Chidamber and Kemerer (CK) metrics. The methodology is validated empirically on a large open-source system—the Eclipse project. The empirical results indicate that the CK metrics have threshold effects at various risk levels and validated the use of these thresholds on the next release of the Eclipse project—Version 2.1—using decision trees. In addition, the selected threshold values were more accurate than those were selected based on either intuitive perspectives or on data distribution parameter [5]. Bad smells are used as a means to identify problematic classes in object-oriented systems for refactoring. Although there is a plethora of empirical studies linking software metrics to errors and error proneness of classes in object-oriented systems, the link between the bad smells and class error probability in the evolution of object-oriented systems after the systems are released has not been explored. There has been no empirical evidence linking the bad smells with class error probability so far. This paper presents the results from an empirical study that investigated the relationship between the bad smells and class error probability in three error-severity levels in an industrial-strength open source system. The research, which was conducted in the context of the post-release system evolution process, showed that some bad smells were positively associated with the class error probability in the three error-severity levels. This finding supports the use of bad smells as a systematic method to identify and refactor problematic classes in this specific context. Rosenberg[4] identified the Object oriented technology uses objects and not algorithms as its fundamental building blocks, the approach to software metrics for object oriented programs must be different from the standard metrics set. Some metrics, such as lines of code and cyclomatic complexity, have become accepted as "standard" for traditional functional/ procedural programs, but for object oriented, there are many proposed object oriented metrics in the literature. The question is, "Which object oriented metrics should a project use, and can any of the traditional metrics be adapted to the object oriented environment?" Basili et al.(1996) presented the results of a study in which we empirically investigated the suite of object-oriented (OO) design metrics introduced. More specifically, our goal is to assess these metrics as predictors of fault-prone classes and, therefore, determine whether they can be used as early quality indicators. This study is complementary to the work described where the same suite of metrics had been used to assess frequencies of maintenance changes to classes. To perform our validation accurately, we collected data on the development of eight medium-sized information management systems based on identical requirements. All eight projects were developed using a sequential life cycle model, a well-known OO analysis/design method and the C++ programming language. Based on empirical and quantitative analysis, the advantages and drawbacks of these OO metrics are discussed. Several of Chidamber and Kemerer's OO metrics appear to be useful to predict class fault-proneness during the early phases of the life-cycle. Also, on the data set, they are better predictors than "traditional" code metrics, which can only be collected at a later phase of the software development processes. Bender[10] described the method for quantitative risk assessment in epidemiological studies investigating threshold effects is proposed. The simple logistic regression model is used to describe the association between a binary response variable and a continuous risk factor. By defining acceptable levels for the absolute risk and the risk gradient the corresponding benchmark values of the risk factor can be calculated by means of nonlinear functions of the logistic regression coefficients. Standard errors and confidence intervals of the benchmark values are derived by means of the multivariate delta method. The proposed approach is compared with the threshold model of ULM (1991) for assessing threshold values in epidemiological studies. Gyimothy et al. (2005) found significant association between some of the CK metrics and the fault-proneness of classes, expect for the NOC metrics.[19] Rosenberg suggested a set of threshold values for the CK metrics that can be used to select classes for inspection or redesign[4]. Bender [10] pointed out that the estimated threshold values should only be considered suitable if the assumption of the regression model, (i.e., a constant risk below the threshold) is plausible. Bender redefined the threshold effects as an acceptable risk level. Thus far, there is no consensus on the threshold values for software metrics, and perhaps not even for what are the best methods to use in the search for the threshold effects. In this research we assess the use of a quantitative methodology that was proposed to find the threshold effects, which are redefined as the acceptable risk level.

3. RESEARCH METHOD

In this section, we introduce the research methodology in which first, to find the threshold values of object-oriented metrics based on the bad smell.Two version (1.0.0 pre1 and 1.0.1) of jfreechart were taken for anaylsis. First, we collected the CK metrics and badsmell databases of these two versions of jfreechart from Analyst4j tool. .The objective is to use the logistic regression to investigate the threshold effects of the CK metrics. Only the significantly associated metrics were considered for finding the threshold values using bad smell at various risk level.

3.1 Software Measurements

We have selected the CK metrics as evidenced by previous empirical studies. The CK metrics are defined as follows:-
Coupling between Object (CBO) metric counts the number of classes to which it is coupled Line of Code (LOC) is calculated as the sum of no. of fields, the no. of nodes and the no. of instructions in a given class. Response for Class (RFC) is the count of set of all the methods that can be invoked in response to a message to an object of a class. Weighted Method Complexity (WMC) metric is the sum of all the complexities of all the methods in a class. Lack of cohesion of methods (LCOM) measure the dissimilarity of methods in a class. Depth of inheritance (DIT) is the maximum no. of steps from the class node to the root of the tree.

3.2 Bad Smell Measurements

Identify bad smells in code helps to refactor the code. Refactoring of software code is a very tedious problem and applying it manually is yet more difficult. A number of surveys have been done for refactoring and maintainability. There is a need of external attributes to refactor the code for better understandability. Metrics provide solid information regarding object oriented properties. Research results show the relationship between structural attributes and external quality metrics. In this paper we find five different bad smell categories in which different bad smells are identified and refractor to ensure maintainability. Table1 shows the different bad smell categorizations which contain various bad smells in it.

4. ANALYSIS METHODS

The method that we use to perform this analysis is based upon the Logistic regression model. The logistic regression is used to validate the metrics and to calculate the threshold values.

4.1 Univariate Binary Logistic Regression Analysis

In the Univariate Binary Logistic Regression (UBR) Analysis significant metrics for predicting bad smell were selected. Analysis was done at the 95 percent confidence level (P-value < 0.05). In the Univariate Binary Logistic Regression (UBR) Analysis significant metrics for predicting bad smell were selected.

Table 1. Bad Smell Categorization

SNo.	Bad Smell Category	Bad Smells
1.	Blob Class	• Large Objects • Large Attributes • Long Methods • Large Class • Long Parameter List
2.	Undocumented Code	• No proper Documentation • Comments
3.	Using Inheritance	• Parallel Inheritance Hierarchies • Feature Envy
4.	Procedure oriented Design	• Switch Statements • Alternative classes with different interfaces
5.	Complex Class	• Duplicate Code • Data Class

Analysis was done at the 95 percent confidence level(P-value < 0.05).If the metrics has p-value greater than 0.05 then it is neglected, it means that we can’t calculate the threshold values for the next step. Table 2. shows the significance levels(p-values) for the Univariate Logistic Regression for the CK metrics of two versions of jfreechart.

Table 2. Univariate Binary Regression Analysis

Metrics	jfreechart(1.0.0 pre1)		jfreechart(1.0.1)	
	B	p-value	B	p-value
LOC	.009	.000	0.008	.000
WMC	.034	.000	0.46	.000
RFC	.025	.000	0.027	.000
LCOM	N/A	N/A	N/A	N/A
CBO	.249	.000	0.207	.000
DIT	1.128	.000	0.703	.000

It was notice from the UBR analysis that the LOC, WMC, RFC, CBO and DIT metrics are significant predictors of the bad smell between classes at the 95 % confidence level ($P < 0.05$). That's why, we calculate the threshold values only for these metrics. *Analyst4j* data set does not allow identifying the threshold values for it.

4.2 Threshold Effects Analysis

The threshold values are calculated with the help of Value of Acceptable Risk Level (VARL) using equation (1).Only above mention metrics are calculated with this formula. Table 3 shows the threshold values of selected metrics at different five risk levels (0.5, 0.55, 0.6, 0.65, 0.7) of two different versions of jfreechart.

$$\text{VARL}= p^{-1} (p_o) = 1/\beta (\log (p_o/1-p_o) - \alpha) \qquad \text{Equation (1)}$$

Table 3. VARL Threshold Values

METRIC	β	α	VARL (p_o =0.50)	VARL (p_o =0.55)	VARL (p_o =0.60)	VARL (p_o =0.65)	VARL (p_o=0.70)
LOC	0.009	-0.003	0	23	45	69	94
WMC	0.034	0.246	-7	-1	5	11	18
RFC	0.025	0.036	-1	7	15	23	32
DIT	1.128	-0.696	1	1	1	1	1
CBO	0.249	-0.733	3	4	5	5	6

Table 4. VARL Threshold Values

METRIC	β	α	VARL (p_o =0.50)	VARL (p_o =0.55)	VARL (p_o =0.60)	VARL (p_o =0.65)	VARL (p_o=0.70)
LOC	0.008	-0.025	3	28	54	81	109
WMC	0.047	0.177	-4	1	5	9	14
RFC	0.027	0.018	-1	7	14	22	31
DIT	0.72	-0.153	0	0	1	1	1
CBO	0.208	-0.669	3	4	5	6	7

The Threshold values of selected metrics are given with the VARL formula, in which α and β are the coefficient estimates

and the probability p_o is suggested with different five risk levels i.e. ($p_o = 0.5$ to $p_o = 0.7$). Table 3 and Table 4 represent the Threshold values with equation 1 based on bad smell at different five risk levels. Result shows some metrics have effective threshold values for the metrics.

4.3 Assessing Threshold Effectiveness

Accuracy of each software metrics was calculated with the help of confusion matrix at five risk levels. It is observed that some metrics gives the proper accuracy at five risk levels. Actually, it depends on the threshold values of selected software metrics. Both the versions of jfreechart packages gave the different accuracy at different risk levels. It is observed that highest accuracy is at p_o = 0.55 risk level which was above 70% at this risk levels.

Table 5. Accuracy for jfreechart version

METRICS	P_O=0.5	P_O = 0.55	P_O = 0.6	P_O = 0.65	P_O = 0.7
LOC	-	77.3	73.7	69.5	69.5
WMC	-	77.3	75.5	69.5	69.5
RFC	-	77.7	76.1	69.5	69.5
DIT	75.3	75.3	75.3	69.5	69.5
CBO	78.5	77.3	76.7	75.1	74.1

Table 6. Accuracy for jfreechart version (1.0.0 to 1.0.1)

METRICS	P_O=0.5	P_O = 0.55	P_O = 0.6	P_O = 0.65	P_O = 0.7
LOC	-	75.6	68.5	67.9	67.9
WMC	-	77.7	72.3	67.9	67.9
RFC	-	77.3	74.4	67.9	67.9
DIT	75.4	75.4	75.4	67.9	67.9
CBO	80	79.6	77.5	75.8	74.8

Table 7. Accuracy for jfreechart package

S.No.	Metrics	P_O = 0.55	VARL Threshold values
1.	LOC	75.6	28
2.	WMC	77.7	1
3.	RFC	77.3	7
4.	DIT	75.4	1
5.	CBO	79.6	4

The confusion matrix can be used to select the best results among the different risk levels. The Thresholds for po = 0.55 shows the best accuracy values. Therefore, we consider the threshold values for po = 0.55 as the most effective. These threshold values are LOC = 28, WMC= 1, RFC = 7, DIT = 0 and CBO = 4. The LOC, WMC, RFC and CBO metrics shows the best among the five metrics. Only DIT metrics gives the less accuracy as compared to other selected metrics. However, each of these metrics serves a different purpose for both developers and testers of a software system. The CBO metric can be used to identify the classes that are excessively coupled to other classes, and the RFC metric can be used to identify the classes that have large responsibilities, whereas the WMC metric can be used to identify the classes that have excessive complexity. In general, a class with significantly more methods than its peers is more complex, tends to be more application-specific, and often hosts a greater number of bad smells present in the class.

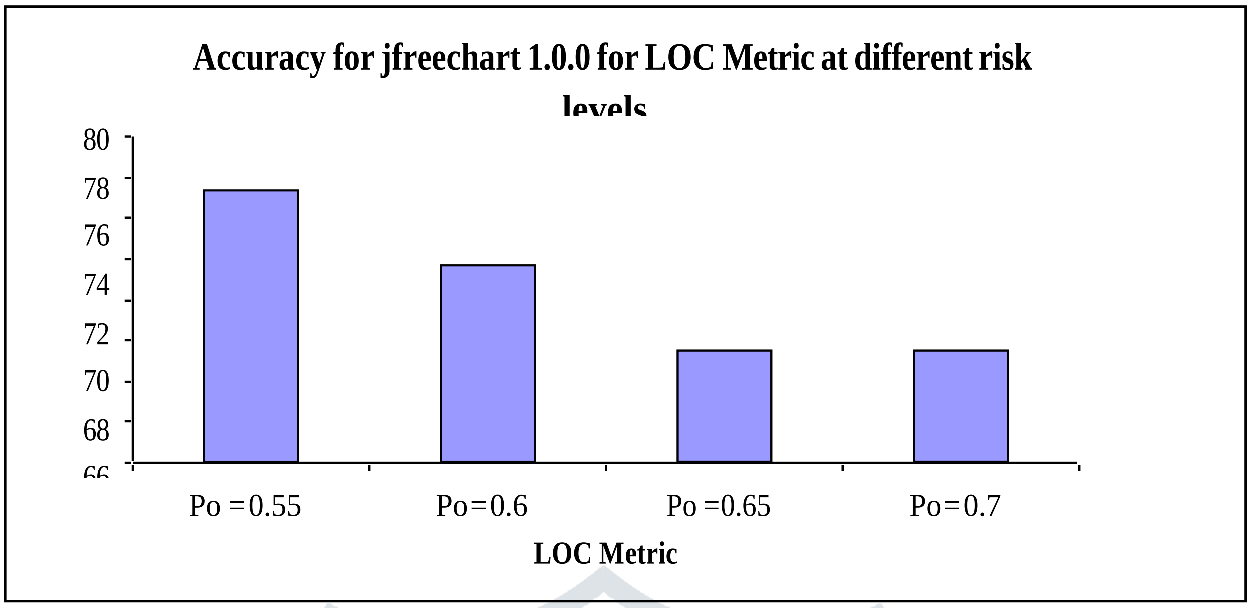
To get insight into the accuracy of two version of jfreechart package. Accuracy of two versions was calculated and it is observed that some metrics gives the highest accuracy at different risk levels. Then, apply one version to another version to predict the accuracy of each software metrics and after comparing the result, it is observed that accuracy after applying one version to another version also gave highest accuracy at the same risk level which is more than 70%. Therefore, the prediction accuracy can be improved if these thresholds are applied of one version to another version.

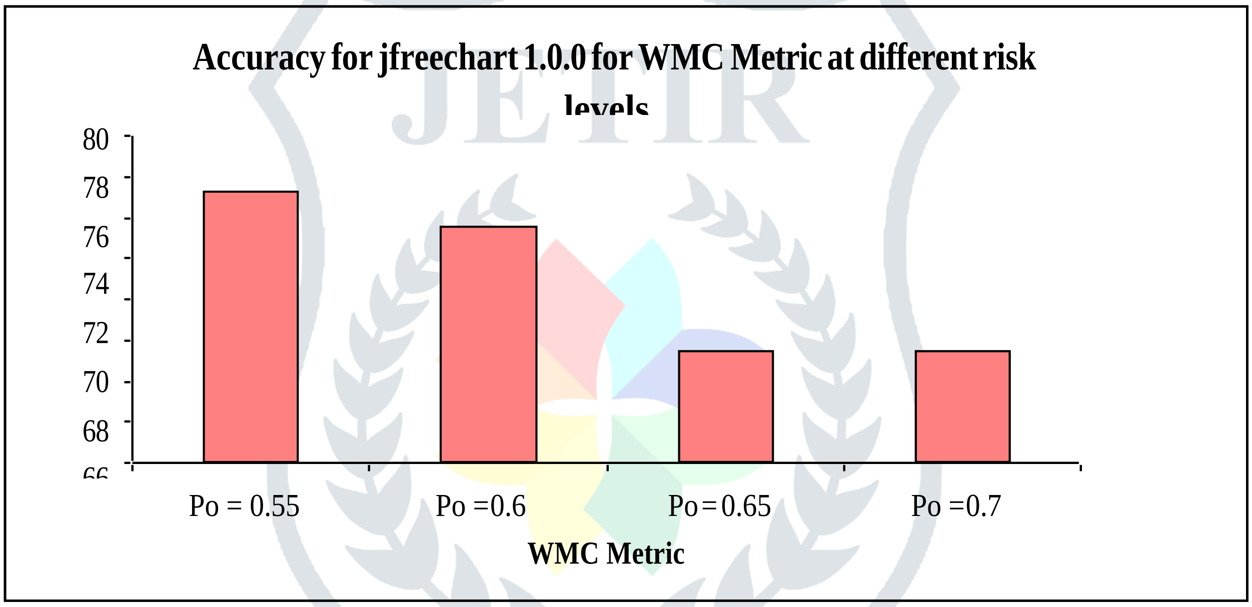
Accuracy Evaluation

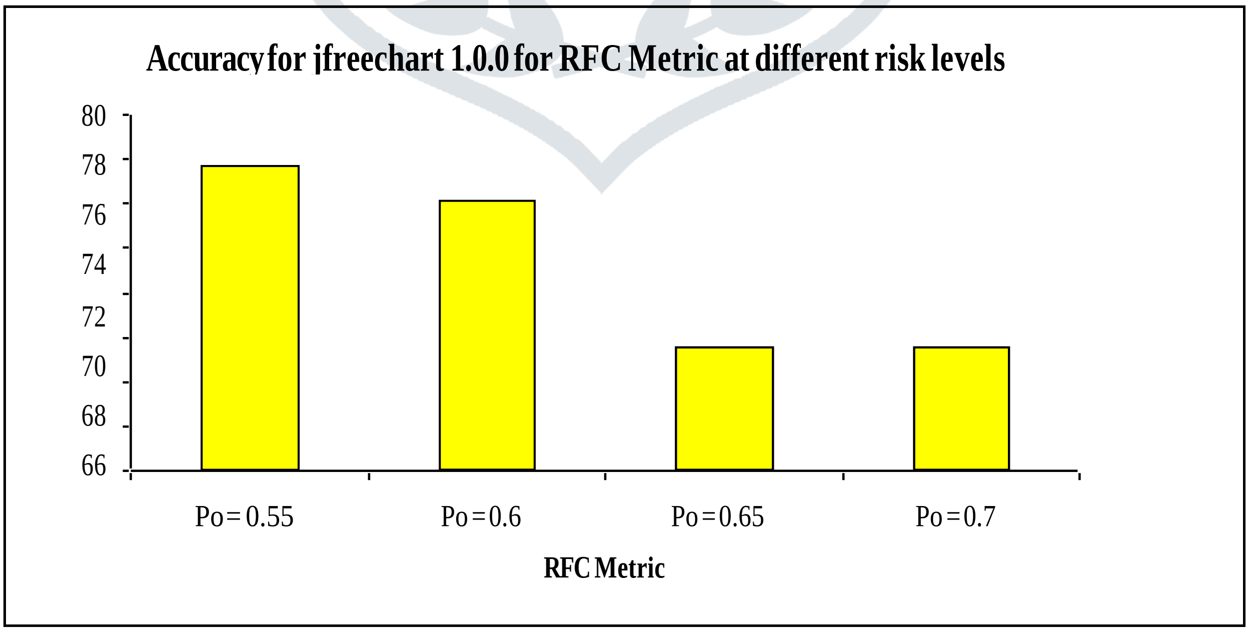
We need to check the accuracy or effectiveness of the above selected metrics values. Accuracy of each software metrics was calculated with the help of confusion matrix at five risk levels of two versions of jfreechart. It is observed that some metrics gives the proper accuracy at five risk levels. Actually, it depends on the threshold values of selected software metrics. Both the versions of jfreechart packages give the different accuracy at different risk levels. The confusion matrix can be used to select the best results among the five risk levels. This matrix can be used to calculate the percentage of each selected metrics and also the confusion matrix gave the observed and predicted values through which it is easy to calculate the accuracy. The percentage accuracy can be calculated at each five different risk levels. It is observed that accuracy can be calculated only at some risk levels of selected software metrics. It can be observed from the figure 6.1 that only CBO and DIT metrics are calculated at all five different risk levels whereas WMC,LOC and RFC metrics can't gave any accuracy at level po=0.5 but these metrics gave good accuracy at other four risk levels. The accuracy values of version (1.0.1) of selected metrics are summarized in Fig.6.1. It was observed that accuracy of this jfreechart version of selected threshold metrics gave good result and also highest accuracy at the po = 0.55 level that are more than 70% for all the selected metrics. Some metrics did not give accurate value at particular risk levels. But on the other hand, selected metrics gave the highest accuracy at five risk levels.

Accuracy Evaluation for other version

The accuracy values of version (1.0.0) of selected metrics are summarized in Fig.   It was observed that accuracy of this jfreechart version of selected threshold metrics gave good result and also highest accuracy at the p_o = 0.55 level that are more than 70% for all the selected metrics. Some metrics did not give accurate value at particular risk levels. But on the other hand, selected metrics gave the highest accuracy at five risk levels.


Accuracy for jfreechart 1.0.0 for LOC Metric at different risk levels
80
78
76
74
72
70
68
66
Po =0.55
Po=0.6
Po =0.65
Po=0.7
LOC Metric


Accuracy for jfreechart 1.0.0 for WMC Metric at different risk levels
80
78
76
74
72
70
68
66
Po = 0.55
Po =0.6
Po=0.65
Po =0.7
WMC Metric


Accuracy for jfreechart 1.0.0 for RFC Metric at different risk levels
80
78
76
74
72
70
68
66
Po=0.55
Po=0.6
Po=0.65
Po=0.7
RFC Metric

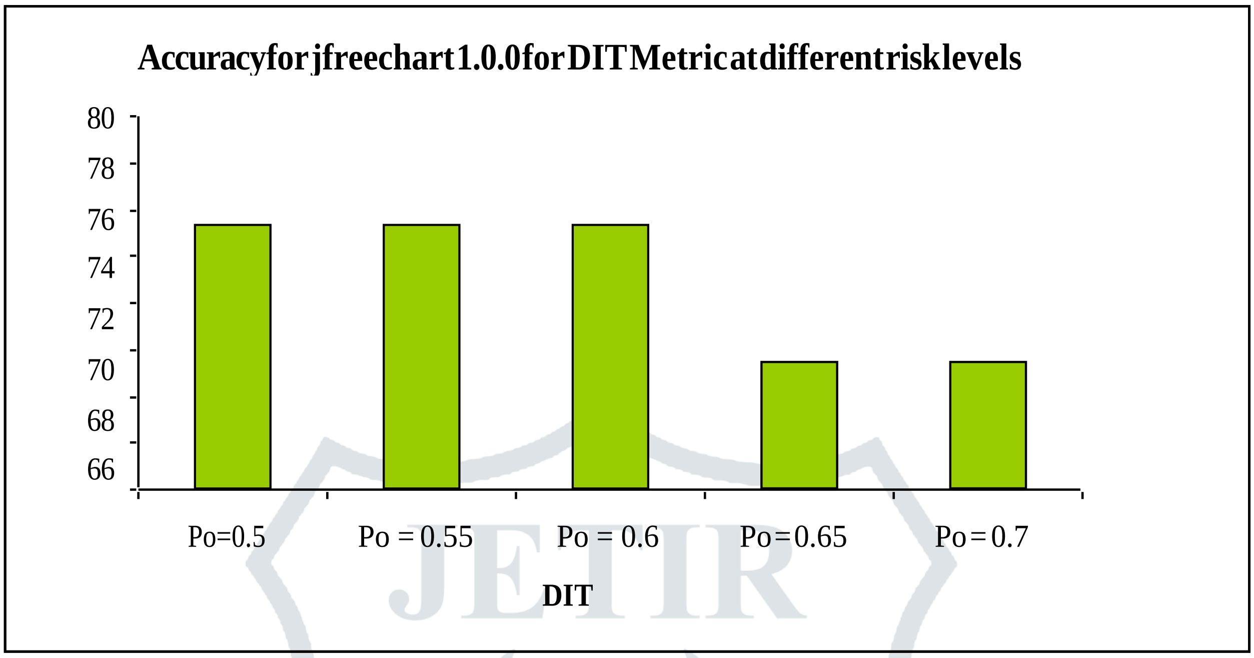
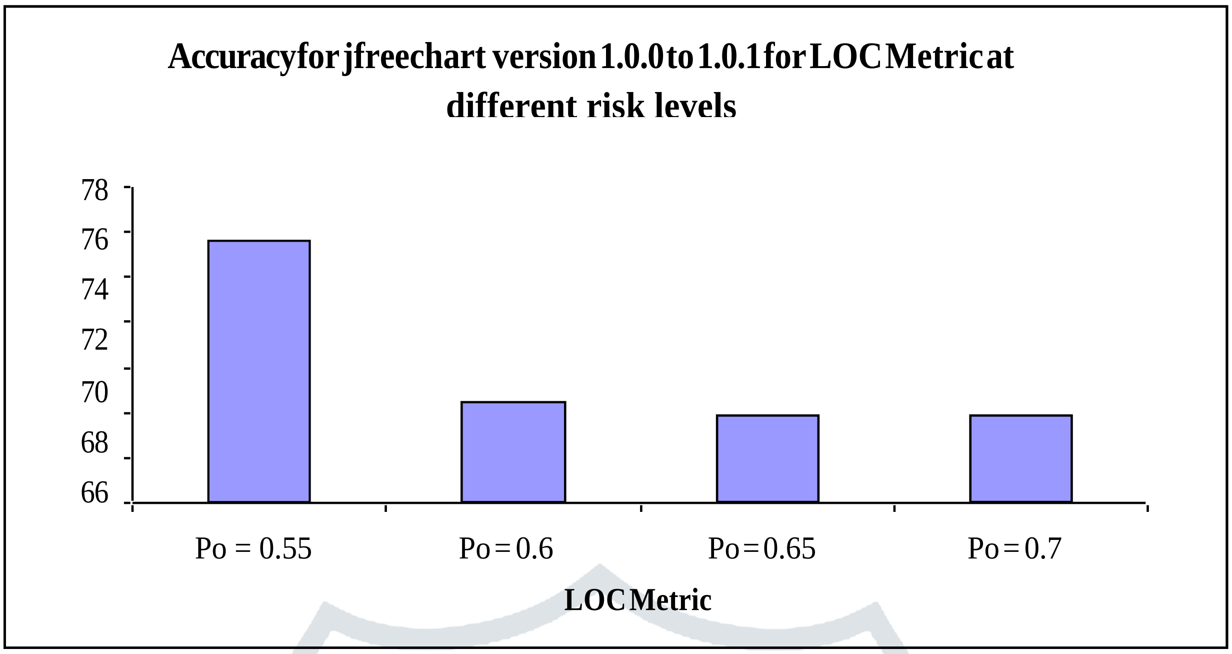
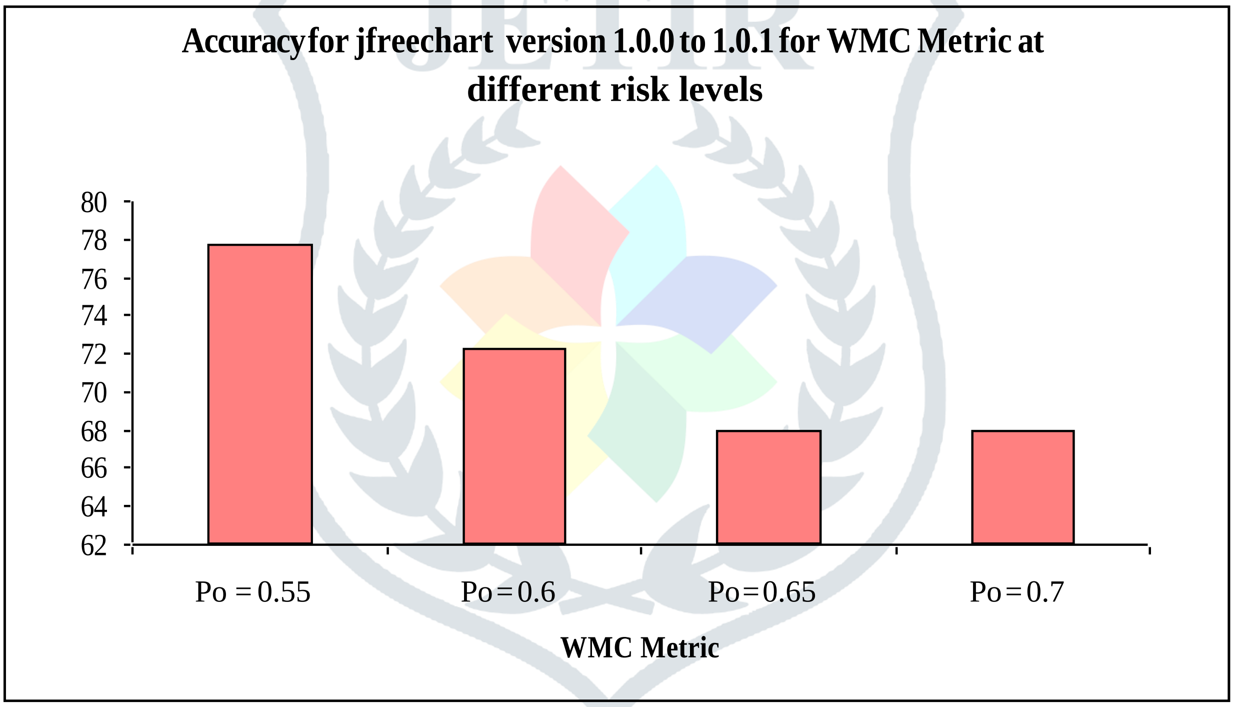
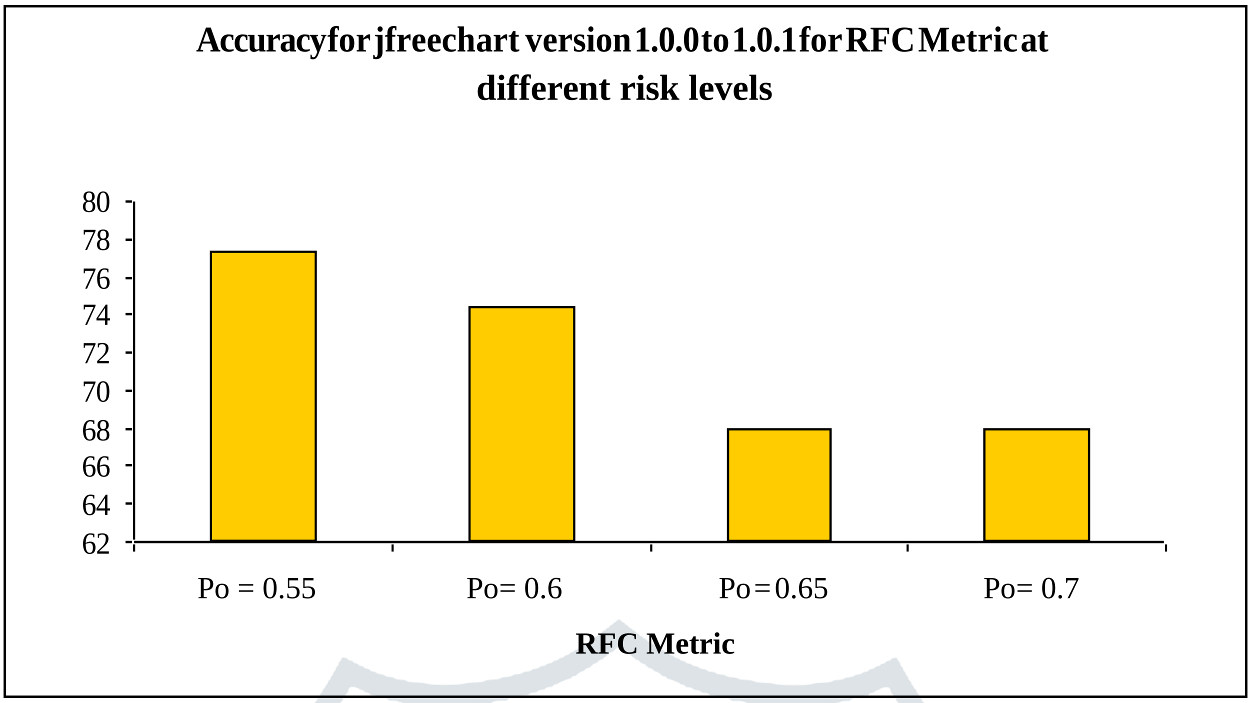


Figure 1. The Accuracy for jfreechart version 1.0.0 at different risk levels for metrics (a) LOC (b) WMC (c) ROC and (d) DIT


Accuracy for jfreechart version 1.0.0 to 1.0.1 for LOC Metric at different risk levels
78
76
74
72
70
68
66
Po = 0.55
Po = 0.6
Po = 0.65
Po = 0.7
LOC Metric


Accuracy for jfreechart version 1.0.0 to 1.0.1 for WMC Metric at different risk levels
80
78
76
74
72
70
68
66
64
62
Po = 0.55
Po = 0.6
Po = 0.65
Po = 0.7
WMC Metric



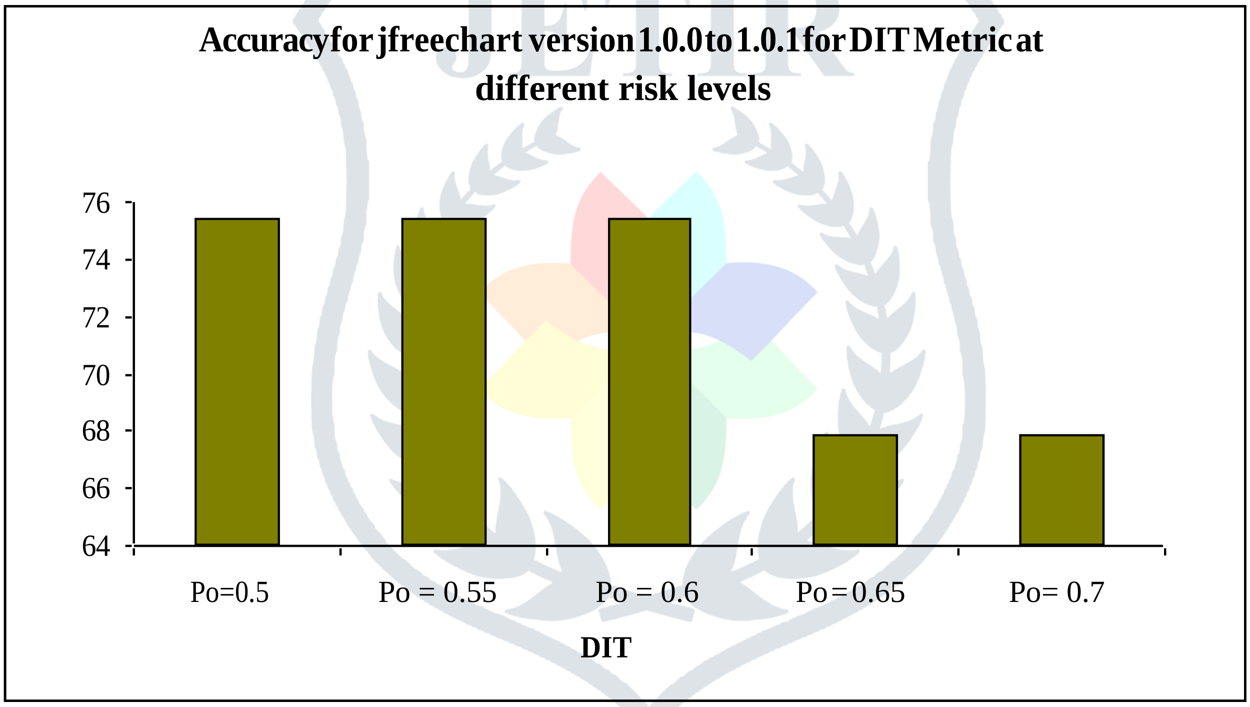


Figure 2. The Accuracy for jfreechart version 1.0.0 to 1.0.1 at different risk levels for metrics (a) LOC (b) WMC (c) ROC (d) DIT

Accuracy on other releases of jfreechart

As our goal is to predict bad smell in the successive releases of the system, one version is applied on other version to predict the bad smell and compare the accuracy of both the versions. It is observed that accuracy gave satisfactory result. In this study, accuracy of both the versions give more than 70% results and it was also observed that when one version applied on the other version then also, accuracy more than 70% at $p_o = 0.55$ for the selected metrics: LOC, WMC, RFC, CBO and DIT

CONCLUSION AND FUTURE WORK

Object-Oriented metrics are more beneficial for the software engineers. Threshold values also provide the meaning to the OO metrics and to identify the various classes at risk. In this work, we find the threshold values of Object-Oriented metrics based on bad smell by using Logistic Regression model. It was concluded that these threshold values also
helps to improve the software quality because classes having more than threshold values will increase the testing efficiency. It was observed that accuracy of jfreechart version of selected threshold metrics gave good result and also highest accuracy at the p_o = 0.55 level that are more than 70% for all the selected metrics: LOC, WMC, RFC, CBO and DIT. We found that there are effective threshold values for the selected metrics. At different risk level, we found different effects for the given metrics.

Acknowledgments: The authors would like to thank Dr. Satwinder Singh for his valuable suggestions and support on this paper.

References

[1] Abreau. F. and Melo. (1996). Evaluating the impact of object oriented design on software quality, Proc. 3rd International Software metrics Symposium (Metrics96), IEEE, Berlin, Germany.

[2] Coleman. D, Ash.D, Lowther. B and Oman. P.W. (1994). Using metrics to evaluate software system maintainability, IEEE Computing Practices, vol. 27, (pp. 44-49)

[3] Fowler and Martin.(2000). Refactoring: Improving the Design of Existing Code Addison-Wisely.

[4] Rosenberg L.H.(1998). Applying and Interpreting Object Oriented Metrics Applying and Interpreting Object Oriented Metrics, Proc. Software Technology Conf.

[5] Shatnawi. R.(2007). An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution, Journal of Systems and Software, vol. 80 no. 7, (pp.1120-1128).

[6] Munro. M. (2005). Product Metrics for Automatic Identification of "Bad Smell" Design Problems in Java Source-Code, in 'METRICS '05, Proceedings of the 11th IEEE International Software Metrics Symposium.

[7] Mantyala. M. (2003). Bad Smells in Software-a Taxonomy and an empirical Study, PhD Thesis, Helsinki University of Technology.

[8] Briand. L, Daly. J, and Wust. J. (1999). A Unified Framework for Coupling Measurement in Object-Oriented Systems, IEEE Trans. Software Eng., vol. 25, no. 1, (pp. 91-121).

[9] Shatnawi. R. (2010). A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems, *IEEE* Transactions Software Engineering, vol. 36, no. 2,(pp. 216-225).

[10] Bender. R. (1999). Quantitative Risk Assessment in Epidemiological Studies Investigating Threshold Effects, Biometrical Journal, vol. 41, no. 3,(pp. 305-319).

[11] Gyimothy. T, Ferenc. R, and Siket. I. (2005) Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction, IEEE Transactions On Software Engineering, vol. 31, pp. (10-19)

[12] Basili. V, Briand.L, and Melo. W.(1996). A Validation of Object- Oriented Design Metrics as Quality Indicators, IEEE Trans. Software Eng., vol. 22, no. 10, (pp. 751-761).

[13]. Shatnawi, R. and Althebyan, Q. (2013). An Empirical Study of the Effect of Power Law Distribution on the Interpretation of OO Metrics, Hindawi Publishing Corporation ISRN Software Engineering, (pp. 18).

[14]. Shatnawi, R. (2010). A Quantitative Investigation of the Acceptable Risk Levels of Object- Oriented Metrics in Open-Source Systems, IEEE Transactions on Software Engineering, Vol. 36, No. 2, (pp. 216-225).