

Predicting Error Probability in the Eclipse Project

Raed Shatnawi, Wei Li, and Huaming Zhang
Computer Science Department
The University of Alabama in Huntsville
Huntsville, AL 35899

Abstract - Object-oriented software metrics have been shown to be able to predict various software quality factors. This paper investigated whether the metrics could predict class error probability and whether the predicted probability could group classes in the object-oriented design. We found, in an open source system, that a set of object-oriented metrics could predict class error probability and the probability could be used to group classes into error and no-error categories with reasonable accuracies.

Keywords: OO design, bad smells, empirical study, open source system.

1.0 Introduction

In the object-oriented (OO) paradigm, empirical studies have shown that OO metrics could predict OO quality factors such as error proneness [2] [4] [9], maintenance effort [1], design effort [6] [14] and project progress [11]. In this article, we report the results of an empirical study on using object-oriented (OO) metrics to predict class error probability in an open source system.

2.0 Research Objectives

In this study, we investigate the relationship between the OO metrics values of a class and the class' error probability in an open source system—the Eclipse Project. We refer to the error probability of a class as the error proneness of the class—the higher the probability, the more error-prone the class. We summarize the research objectives in the following hypotheses:

- Hypothesis 1: OO coupling metrics (CTA, CTM, CBO, and RFC) cannot predict the error proneness of classes.
- Hypothesis 2: OO complexity metrics (WMC) cannot predict the error proneness of classes.
- Hypothesis 3: OO inheritance metrics (DIT and NOC) cannot predict the error proneness of classes.
- Hypothesis 4: OO polymorphism metrics (NOAM and NOOM) cannot predict the error proneness of classes
- Hypothesis 5: OO cohesion metrics (LCOM) cannot predict the error proneness of classes.
- Hypothesis 6: Size metrics (NOA and NOO) cannot predict the error proneness of classes.

3.0 The Data Collection

The Eclipse Project is available in public domain [7]. We collected and analyzed the data from one official release of the Eclipse project: Version 2.0; its source code and error data are available in public domain. The error data for the Eclipse release was logged on Bugzilla, an open source error logging tool. The Bugzilla database contains all errors (bugs) that have been found in the lifetime of the Eclipse Project. We went through the Bugzilla system and extracted the error data for Eclipse 2.0. We marked each class

as either *erroneous* or *not erroneous*, depending on whether there was at least one error recorded for the class in the release.

We used the Borland Together for Eclipse—a plug-in tool for Eclipse—to calculate the OO metrics. Together compiled the source code and reported each class name and the set of OO metrics. Appendix A lists all the metrics that we collected and used in the study.

Table 1 shows the descriptive statistics for Eclipse 2.0. The NOC values were zeros in at least 75% of the classes, the largest DIT value was eight, and 75% of classes had at most three levels of inheritance. This low variability in the DIT and NOC metrics agrees with the findings of Chidamber and Kemerer [6]. We also noticed that 50% of classes had low LCOM values (less than four) and 25% of the classes had zero in the LCOM metrics value; Basili et al. [2] and Briand et al. [4] also noted some problems in the original LCOM definition. Therefore, we did not include the LCOM metrics in our study.

Table 1. Eclipse 2.0 Descriptive Statistics

Metrics	Mean	Std. Dev.	Min	Max	Percentiles		
					25%	50%	75%
CBO	9.72	11.36	0	156	3	6	13
CTA	1.18	1.85	0	33	0	1	2
CTM	35.14	62.37	0	1142	5	16	39
RFC	41.77	71.28	0	1361	8	20	47
WMC	23.17	36.22	0	677	5	12	26
DIT	1.98	1.37	0	8	1	2	3
NOC	1.39	8.85	0	218	0	0	0
NOAM	6.40	10.99	0	299	1	3	8
NOOM	3.24	7.54	0	205	0	1	4
LCOM	85.26	476.29	0	18246	0	3	36
NOA	3.06	4.47	0	63	0	2	4
NOO	9.57	13.91	0	300	3	6	11

4.0 The Statistical Models

We used the Univariate Binary Regression (UBR) to examine whether there was any significant association between a metrics and the error proneness in the classes. We used 0.25 as the cutoff P-value [8] in testing the significance of each metrics We then built an error proneness prediction model by using the Multivariate Logistic Regression (MLR) [8]. The model used the binary dependent variable which was defined as “the class is faulty” or “the class was not faulty”. The generic MLR model (the UBR model is a special case of MLR when n is one) is as follows:

$$\pi(Y=1|X_1, X_2, \dots, X_n) = \frac{e^{g(x)}}{1 + e^{g(x)}}$$

Where:

$g(x) = B_0 + B_1 * X_1 + B_2 * X_2 + \dots + B_n * X_n$ is the logit function;

π : the probability of a class being faulty;

Y: the dependent variable—it is a binary variable;

X_i ($1 \leq i \leq n$): are the independent variables, which are the OO metrics investigated in this study;

B_i ($0 \leq i \leq n$) are the estimated coefficients from maximizing the log-likelihood.

The B_i 's are used to calculate the *odd ratios* ($OR = e^{B_i}$) for each independent variable that contributes to the model. The OR value describes the type of association between the dependent variable and an independent variable. The *positive association* ($OR > 1$) means the likelihood of the dependent variable changes in the same direction as the independent variable changes. The *negative association*

(OR<1) means the likelihood of the dependent variable changes in the opposite direction of the independent variable change. The *even association* (OR=1) means that the changes in the independent variable has no impact on the dependent variable. The interpretation of the OR magnitude depends on the scale of the independent variable. If the independent variable is a continuous variable, such as the CBO, then “OR=2” means that for each unit change of CBO, the error probability changes twice as much in the same direction in its scale. If the independent variable is dichotomous and coded as one or zero, then “OR=2” means that the error probability is twice more likely to occur in a class when the independent variable is present than otherwise.

Collinearity might exist among the OO metrics used in the study; the existence of collinearity violates the assumption of independence among the independent variables of the models. Therefore, we had to diagnose and eliminate the collinearity before we built the MLR model. We analyzed the collinearity through three tests: the Spearman correlation analysis, the Variance Inflation Factor (VIF) analysis [13], and the Condition Number analysis (we used 30 as the cutoff value) [3]. We first calculated the Spearman correlation in the metrics group. After the high correlations were noted, we calculated the condition number and the VIFs for the group. If the condition number was greater than 30, we dropped the metrics that had the highest VIF value, and recalculated the condition number again; this process continued until we had the condition number below 30; the remaining metrics did not suffer from collinearity and were selected as candidates to enter the MLR model. (These candidate metrics would be examined and selected by the *stepwise forward selection* process of the MLR model for the best set of predictors.)

4.1 The UBR Analysis

Table 2 shows the UBR analysis results for Eclipse 2.0. We noticed that all coupling metrics—CBO, CTA, CTM, and RFC—were good predictors of class error proneness; therefore we rejected hypothesis 1 and concluded that the coupling metrics could predict class error proneness. The WMC was a significant predictor of error proneness; therefore we rejected hypothesis 2 and concluded that the complexity metrics could predict class error proneness. DIT and NOC metrics were not significant predictors of errors proneness; therefore we could not reject Hypothesis 3. But since the DIT and NOC were not predictors of error proneness, they were not included as error-proneness predictors in the MLR model. The NOOM and NOAM metrics were significant predictors; therefore we rejected hypothesis 4 and concluded that the polymorphism metrics could predict class error proneness. The NOA metrics was not significant predictor of error proneness, therefore we could not reject hypothesis 6 and we did not include it in the MLR model, while NOO was significant predictor of error proneness.

Table 2. Univariate Binary Analysis for Eclipse 2.0

Metrics	B	P-value
CBO	0.024	0.000
CTA	0.071	0.005
CTM	0.004	0.000
RFC	0.004	0.000
WMC	0.006	0.000
DIT	0.051	0.223
NOC	-0.026	0.174
NOAM	0.020	0.000
NOOM	0.015	0.000
NOA	0.001	0.932
NOO	0.017	0.000

Our conclusions agree with that reached by Gyimothy et. el. [9]; their work was done on one release of another open source system (Mozilla) using only the Chidamber and Kemerer (CK) metrics [5].

4.2 The Collinearity Analysis

Table 3 shows the correlations of the metrics. We considered the correlations that were larger than 0.8 as high, in between 0.5 and 0.8 as moderate, and below 0.5 as low [15]. We noticed that there were many high correlations among the NOO, CTM, RFC, and WMC metrics. The correlations among the coupling metrics—CBO, CTA, RFC, and CTM—were either high or moderate. The correlation between RFC and CTM was very high. These high correlations might cause collinearity. Therefore, we calculated the condition number and the VIF values for the metrics. Table 4 shows the condition-number and VIF analyses.

Table 3. Spearman Correlation among Metrics in Eclipse 2.0

Metrics	CBO	CTA	CTM	RFC	WMC	NOAM	NOOM
CTA	0.54						
CTM	0.85	0.52					
RFC	0.81	0.52	0.96				
WMC	0.68	0.49	0.86	0.92			
NOAM	0.49	0.48	0.65	0.73	0.78		
NOOM	0.34	0.19	0.37	0.40	0.37	-0.06	
NOO	0.59	0.48	0.76	0.87	0.91	0.82	0.42

Table 4. Multicollinearity Diagnoses for Eclipse 2.0

Metrics	VIF	VIF(w/o NOO)	VIF(w/o RFC)
CBO	4	4	4
CTA	4	4	4
CTM	647	637	6
RFC	856	840	N/A
WMC	9	9	9
NOAM	3725	20	4
NOOM	717	4	2
NOO	4874	N/A	N/A
Condition Number	123	115	10

The first VIF column shows the VIF analysis and the condition number for all the metrics. The second VIF column shows the result after the NOO metrics was dropped and the third VIF column shows the result after the NOO and RFC metrics were both dropped. We noticed that the condition number was less than 30 after dropping the NOO and RFC metrics. Succu suggested not using the RFC in conjunction with any of the NOO, CBO and LCOM metrics in prediction models [15]. Consequently, we dropped the NOO and RFC metrics in the subsequent analyses. Therefore, the set of metrics that was considered as independent and did not suffer from collinearity included CBO, CTA, CTM, WMC, NOAM, and NOOM.

4.3 The MLR Analysis

Eliminating the collinearity in the metrics sets does not necessarily mean that we had the best set of metrics in the MLR. Therefore, we used the forward stepwise selection process [8] to select the best subset of the metrics to use, and the MLR model was shown in Tables 5. The table shows the coefficients, standard error, P-value and odd ratio of the metrics included in the model. Table 6 shows the result of the Likelihood Ratio Test (LRT) to verify the overall significance of the model regardless of the significance of each individual independent variable.

Table 5. Eclipse 2.0 MLR Model

Metrics	B	Std. Error	P-value	OR
CTA	0.063	.025	.014	1.065
NOOM	0.014	.005	.004	1.014
Constant	-2.755	.072	.000	.064

The forward selection selected the combination of CTA, and NOOM as the predictors of class error proneness in Eclipse 2.0. The LRT (in Table 6) showed that the model was significant at 95% level. The condition number of the set of independent variables was 4.6, which was below the threshold of collinearity. Therefore, the model in Table 5 was valid and did not suffer from collinearity.

Table 6. Log-Likelihood Ratio Test for MLR Model

		Eclipse 2.0
Likelihood Ratio Test	P-Value	<0.001

We evaluated the accuracy of the prediction model by using the classification table [8]. We used the means of the error probabilities, shown in Table 7, as the cutoff values to differentiate between classes that contained errors and the classes that did not. If the error probability of a class was larger than the cutoff value, we assigned one (as the predicted value) to the class; zero otherwise. We built the classification table of the actual and predicted erroneous classes. The classification table shows the number of correct classifications of classes into error or no-error categories as well as the number of misclassifications. The sensitivity and specificity [8] were also reported.

Table 7. The Means of the Predicted Error Probabilities for the MLR Model

		Eclipse 2.0
Cutoff Value		0.068

Table 8. Classification Table for the Eclipse 2.0 MLR Model

		Predicted		Total
		No error	Error	
Actual	No error	2945	1207	4152
	Error	176	126	302
Total		3361	1093	4454

The sensitivity of the model (classes predicted erroneous) was 42% (126/302), the specificity (classes predicted not erroneous) was 71% (2945/4152), and the percentage of correctly classified classes was 69%. We therefore concluded that the predicted probabilities could be used to group classes into error and no-error categories with reasonable success rate. The grouping of classes to error and no-error categories is significant in the software engineering process, especially when directing and allocating the testing effort.

5. Conclusion

We found that the selected OO metrics could predict the class error probability (error proneness) in Eclipse 2.0, and we could use the predicted probability to group classes into error and no-error groups. The fact that these research results were obtained from an open source system makes the results verifiable.

Appendix: OO Metrics Definitions

Metrics	Description	Definition	Reference
NOA	Number Of Attributes	Counts the number of attributes.	
NOO	Number of Operations	NOO counts the number of operations.	
LCOM	Lack of Cohesion of Methods	Takes each pair of methods in the class and determines the set of fields they each access. If they have disjoint sets of field accesses, increase the count P by one. If they share at least one field access, then increase Q by one. After considering each pair of methods $RESULT = (P > Q) ? (P - Q) : 0$	[5]
WMC	Weighted Methods Complexity	This metrics is the sum of the complexity of all methods for a class, where each method is weighted by its cyclomatic complexity.	[5]
CBO	Coupling Between Objects	Represents the number of other classes to which a class is coupled to.	[5]
CTA	Coupling Through Data Abstraction	CTA counts the number of reference types used in the attribute declarations.	[10]
CTM	Coupling Through Message Passing	CTM counts the number of method call expressions made into body of the measured method.	[10]
RFC	Response for Class	The size of the response set for the class includes methods in the class's inheritance hierarchy and methods that can be invoked on other objects.	[10]
DIT	Depth of Inheritance Hierarchy	The length of the inheritance chain from the root of the inheritance tree to the measured class is the DIT metrics for the class	[5]
NOC	Number of Child Classes	NOC counts the number of classes directly or indirectly derived from the measured class	[5]
NOAM	Number of Added Methods	NOAM counts the number of operations added by a class. Inherited and overridden operations are not counted.	[12]
NOOM	Number Of Overridden Methods	NOOM counts the number of inherited operations, which a class overrides.	[12]

6. References

- [1] M. Alshayeb and Wei Li. "An Empirical Validation of Object-Oriented Metrics in Two Iterative Processes." *IEEE Transactions on Software Engineering*, 29(11): 1043--1049, 2003.
- [2] V.L. Basili, L. Briand, and W.L. Melo. "A Validation of Object-Oriented Metrics as Quality Indicators." *IEEE Transactions on Software Engineering*, 22(10): 751--761, 1996.
- [3] B. Belesly, R. Welsch. "Regression Diagnostics: Identifying Influential Data and Sources of Collinearity." John Wiley and Sons, 1980.
- [4] L.C. Briand, J. Wust, J.W. Daly, and D.V. Porter. "Exploring the Relationship between Design Measures and Software Quality in Object Oriented Systems." *Journal Systems and Software*, 51(3): 245--273, 2000.
- [5] S.R. Chidamber and C.F. Kemerer. "A Metrics Suite for Object Oriented Design." *IEEE Transactions on Software Engineering*, 20(6): 476--493, 1994.
- [6] S.R. Chidamber, D.P. Darcy, and C.F. Kemerer. "Managerial Use of Metrics for Object Oriented Software: An Exploratory Analysis." *IEEE Transactions on Software Engineering*, 24(8): 629--639, 1998.
- [7] Eclipse Homepage, www.eclipse.org, 2004.
- [8] D. Hosmer, and S. Lemeshow. "Applied Logistic Regression." Wiley Series in Probability and Statistics, second edition, 2000.
- [9] Tibor Gyimothy, Rudolf Ferenc, and Istvan Siket. "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction." *IEEE Transactions on Software Engineering*, 31(10): 897--910, 2005
- [10] W. Li. "Another Metrics Suite for Object-Oriented Programming." *Journal of Systems and Software*, 44(2): 155--162, 1998.
- [11] W. Li, L. Etzkorn, C. Davis, and J. Talburt. "An Empirical Study of Object-Oriented System Evolution." *Information & Software Technology*, 42(6): 373--381, 2000.
- [12] M. Lorenz and J. Kidd. "Object-oriented software metrics." Prentice Hall, Englewood Cliffs, N.J., 1994.
- [13] D. Montgomery, E. Peck, and G. Vining. "Introduction to Linear Regression Analysis." Wiley Series in Probability and Statistics, third edition, 2001.
- [14] P. Nesi and T. Querci. "Effort Estimation and Prediction of Object-Oriented Systems." *Journal of Systems and Software*, 42(1): 89--102, 1998.
- [15] G. Succi, W. Pedryez, S. Djokic, P. Zuliani and B. Russo. "An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite." *Empirical Software Engineering*, 10(1), 81--103, Jan. 2005.