Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults

Yuming Zhou and Hareton Leung, *Member*, *IEEE Computer Society*

Abstract—In the last decade, empirical studies on object-oriented design metrics have shown some of them to be useful for predicting the fault-proneness of classes in object-oriented software systems. This research did not, however, distinguish among faults according to the severity of impact. It would be valuable to know how object-oriented design metrics and class fault-proneness are related when fault severity is taken into account. In this paper, we use logistic regression and machine learning methods to empirically investigate the usefulness of object-oriented design metrics, specifically, a subset of the Chidamber and Kemerer suite, in predicting fault-proneness when taking fault severity into account. Our results, based on a public domain NASA data set, indicate that 1) most of these design metrics are statistically related to fault-proneness of classes across fault severity, and 2) the prediction capabilities of the investigated metrics greatly depend on the severity of faults. More specifically, these design metrics are able to predict low severity faults in fault-prone classes better than high severity faults in fault-prone classes.

Index Terms—Object-oriented, faults, fault-proneness, metrics, prediction, cross validation.

1 Introduction

Given the complexity of modern software and the commercial constraints under which it is produced, it is not unusual for customers to be delivered software with serious faults. One cost-effective way to deal with this problem is apply fault prediction models based on object-oriented (OO) design metrics to detect software faults in classes in developed systems before delivery. Such models can be used to guide the decision-making of software development managers seeking to produce high-quality software on time and within budget [25], and indeed the last three decades has seen proposed many software fault prediction models [1], [2], [4], [7], [8], [9], [10], [11], [12], [13], [15], [16], [17], [23], [24], [25], [26], [27]. Software faults, however, may vary considerably in their severity, some having potentially catastrophic consequences and others being little more than cosmetic issues; yet, to date no software fault prediction models have been proposed that are able to distinguish among faults according to their severity.

Previous research on OO design metrics has shown some of them to be useful for predicting the fault-proneness of classes [1], [2], [3], [4], [5], [6], [7], [8], [9], yet has also made the point that this usefulness depends upon their being valid whatever the degree of severity of the fault [6]. These studies did not, however, distinguish among faults by their severity and, to the best of our knowledge, there has been no work published to date in this very important area. There would seem to be two reasons for this: First, it is time-consuming and expensive to collect qualitative data on fault severity [30]; second, it is difficult to find personnel with the knowledge to accurately rate fault severity [16].

In this paper, as a first step in this area, we investigate the accuracy of the fault-proneness predictions of six widely used OO design metrics, especially a subset of the Chidamber and Kemerer (CK) metrics suite [14], with particular focus on how accurately they predict faults when taking fault severity into account. Each of these metrics is tested against a null hypothesis using logistic regression and three machine learning methods on a public domain NASA data set. To simplify the process of fault-categorization, we provide only two categorizations: faults are either of high severity or of low severity. We attempt to answer six questions:

1. How are the metrics related to the occurrence of high severity faults in classes?
2. Which metrics are strongly related to the occurrence of high severity faults in classes?
3. How are the metrics related to the occurrence of low severity faults in classes?
4. Which metrics are strongly related to the occurrence of low severity faults in classes?
5. How accurately do the metrics predict high severity faults?
6. How accurately do the metrics predict low severity faults?

This paper makes a number of contributions. First, using a publicly available data set, we present new evidence indicating an association between OO design metrics and fault-proneness, thereby providing valuable data in an important area for which limited experimental data is available. Second, we validate the association between OO design metrics and fault-proneness of classes across fault

• *The authors are with the Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China. E-mail: csyzhou@comp.polyu.edu.hk, cshleung@inet.polyu.edu.hk.*

Manuscript received 18 Jan. 2006; revised 25 July 2006; accepted 29 Aug. 2006; published online 19 Oct. 2006.
Recommended for acceptance by B. Littlewood.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0011-0106.

TABLE 1
Definitions of Metrics

Metric	Definition
WMC (weighted methods per class)	The number of methods implemented within a given class
DIT (depth of inheritance tree)	The length of the longest path from a given class to the root in the inheritance hierarchy
RFC (response for a class)	The number of methods that can potentially be executed in response to a message being received by an object of a given class. In this study, RFC is the number of methods implemented within a class plus the number of methods accessible to an object class due to inheritance
NOC (number of children)	The number of classes that directly inherit from a given class
CBO (coupling between object classes)	The number of distinct non-inheritance-related classes on which a given class is coupled. A class is said to be coupled to another class if it uses methods or attributes of the other
LCOM (lack of cohesion in methods)	For each attribute in a given class, calculate the percentage of the methods in the class using that attribute. Average the percentages, then subtract from 100 percent
SLOC (Source lines of code)	The number of all nonblank and noncomment lines in the body of a given class and all of its methods

severity. To the best of our knowledge, in spite of its importance, there has been no such previous research. Third, on the methodological front, the fault-proneness of classes is analyzed using not just the familiar method of logistical regression but also applied machine learning techniques. This study makes two main findings. First, most of the OO design metrics in this study are statistically related to the fault-proneness of classes across fault severity. Second, the predictive abilities of these metrics strongly depend on the severity of faults. More specifically, the design metrics are better predictors of low severity faults in classes than of high severity faults.

The rest of this paper is organized as follows: Section 2 introduces the six OO design metrics we will investigate and states our hypotheses about their expected connections with fault-proneness of classes. Section 3 describes the research method used for this study, including data source, data preprocessing, and data analysis methodology. Section 4 presents in detail the analytical results. Section 5 discusses related work. Section 6 concludes the paper and outlines directions for future work.

2 The Metrics and Hypotheses

The six metrics to be discussed here have been selected for study because they have received considerable attention from researchers and are also being increasingly adopted by practitioners [2]. In particular, they are being incorporated into development tools such as Rational Rose and JBuilder. Five of these metrics, WMC, DIT, RFC, NOC, and CBO, were proposed by Chidamber and Kemerer [14], and the other, LCOM, was proposed by Rosenberg and Hyatt [18]. All are defined at the class level, which captures the concepts of cohesion, coupling, and inheritance. We also include the well-known SLOC as a "baseline" metric to compare against the six object-oriented design metrics.[1] Table 1 provides definitions of these metrics.

This study will test seven hypotheses, which relate six OO design metrics and one traditional code-size metric to fault-proneness according to fault severity. To simplify the process of fault-categorization, these hypotheses allow for only two categories of software fault in a software system: Faults are either high severity or low severity. Note that the basis of this categorization system is described in Section 3.1. On the other hand, to give a complete picture and facilitate comparing our results with previous work, we also investigate the relationships between OO design metrics and fault-proneness without distinguishing the severity of faults. As such, ungraded faults are used to denote all faults in the system without distinguishing their severity. For each metric M, the hypothesis in this study was the following (the relevant null hypothesis is given in parentheses), where the capital H indicates "hypothesis":

H-M. *A class with a high M value is more likely to have high/low/ungraded severity faults than a class with a low M value.* (*Null hypothesis: A class with a high M value is no more likely to have high/low/ungraded severity faults than a class with a low M value.*)

3 Research Method

In this section, we present the research method. We introduce first the data source and then the data preprocessing process. Finally, we describe our data analysis methodologies, specifically, logical regression analysis and machine learning models, and our model evaluation criteria.

3.1 Data Source

This study makes use of the public domain data set KC1 from the NASA Metrics Data Program.[2] The data in KC1 was collected from a storage management system for receiving/processing ground data, which was implemented in the C++ programming language. This system consists of 145 classes that are comprised of 2,107 methods, with a total of roughly 40,000 lines of code.

KC1 provides both method- and class-level static metrics. At the method level, there are 21 software product metrics based on the product's size, complexity, and vocabulary. There are also five types of defect information, such as the number of defects and the density of defects. At the class-level, there are 10 metrics including the 6 OO design metrics under study here. Of particular interest are three files in KC1, one representing the association between classes and methods, one describing the severity rating of

1. We wish to thank an anonymous reviewer for making this suggestion.

2. Available from http://mdp.ivv.nasa.gov/.

TABLE 2
Distributions of Faulty Classes and Faults at Each Severity Rating in KC1

Severity rating	Number of faulty classes	Percentage of faulty classes	Number of faults	Distribution of faults in %
1	16	11.03%	34	6.19%
2	56	38.62%	421	76.68%
3	31	21.38%	92	16.76%
4	1	0.69%	1	0.18%
5	1	0.69%	1	0.18%

each defect and the association between defects and methods, and one stating defect data that remains constant throughout the life cycle of each defect, including a specific reason for the closure of each defect. The severity rating value for each defect quantifies its impact on the overall environment, with 1 being the most severe and 5 being the least severe. In total, KC1 contains 669 defects.

3.2 Data Preprocessing

The focus of our study is to investigate the predictiveness of the OO design metrics with regard to the severity of faults. Consequently, we are interested only in the number of faults at each severity rating for each class. The defect data provided in KC1, however, refers to information parsed from error reports. A defect was generated due to an error, no matter what source it comes from (either from COTS/OS, design, or source code) or even if it was not a bug. We thus first preprocessed KC1 by removing all defects that are not bugs or not from the source code. This reduced the total number of defects from 669 to 549.

The next step was to categorize faults according to their severity and class. There were 549 faults associated with 145 classes. Of these 145, 58 classes (40 percent) were faulty, having between 1 and 90 faults. The faults were rated for severity on a scale of 1 to 5. Severity 1 faults require immediate correction in order for the system to continue to operate properly [28]. More than 90 percent of faults were classified at the severity 2 and severity 3 levels. Faults with severity ratings of 4 and 5 were regarded as trivial. Only 34 faults (6.19 percent) were classified with a severity rating of 1. These were distributed in 16 classes (11.03 percent). This result is somewhat similar to the findings in [28], where 78 of 4,743 faults (1.6 percent of all faults) were classified at the severity 1 level. Table 2 shows the distributions of faulty classes and faults at each severity rating in the KC1 data set after preprocessing.

All faults in KC1 were categorized as of either a high or low severity. Intuitively, a system should contain many more low severity faults than high severity faults. On that assumption, this study classified faults with a severity rating of 1 as high severity faults, and all faults with severity rating 2, 3, 4, or 5 as low severity faults. This gave us 34 high severity faults (6.19 percent) and 515 low severity faults (93.81 percent).

Figs. 1a and 1b show the distribution of high severity faults and low severity faults among the 145 classes. It can be seen that 11 percent of all the classes (i.e., 16 classes) contain high severity faults and may contain between one and eight faults. On the other hand, 39 percent of all the classes (i.e., 57 classes) contain low severity faults and may contain between 1 and 86 faults.

3.3 Data Analysis Methodology

The goal of this study is to empirically analyze six design metrics for the purpose of evaluating whether or not these metrics are useful for predicting fault-prone classes when the severity of faults is taken into account. The definition of what constitutes a fault-prone class must have regard to the context in which a class is used. When investigating the fault-proneness prediction capabilities of the metrics in the context of high severity faults, a class is said to be fault-prone if it has at least one high severity fault. Similarly, in the context of low severity faults, a class is said to be fault-prone if it has at least one low severity fault. In the context of faults of any severity, a class is said to be fault-prone if it has at least one (ungraded severity) fault. In other words, we distinguish three types of prediction models: high severity fault models, low severity fault models, and ungraded severity fault models.

The classification of classes into the two categories of fault-prone and not fault-prone is done using a software quality classification model constructed with logistic regression and machine learning methods. This classification model takes the category (fault-prone or not fault-prone) as the dependent variable and the metrics as the independent variables. The model will allow us to evaluate the abilities of the metrics to predict fault-proneness.

In logistic and machine learning analyses, we also include SLOC as a baseline metric to compare against the investigated design metrics. In particular, we compare models built with design metrics only, built with SLOC

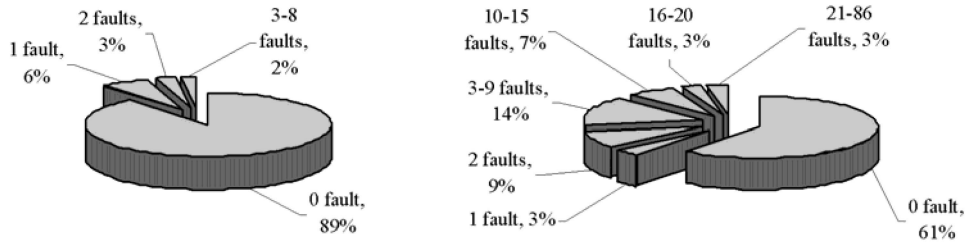


Fig. 1. Distributions of high and low severity faults among all the classes. (a) High severity faults and (b) low severity faults.

only, and one allowing all measures (the design metrics and SLOC) to enter the model. With these comparisons, we can determine: 1) whether some (or even all) of the design metrics are better than SLOC in terms of their predictive accuracy on faults or vice versa, and 2) whether the design metrics are complementary to SLOC metric as fault-proneness predictors.

3.3.1 Logistic Regression Model

Logistic regression is a standard statistical modeling method in which the dependent variable can take on only one of two different values. It is suitable for building software quality classification models because the classes under consideration are divided into two categories: fault-prone and not fault-prone.

A multivariate logistic regression model is based on the following equation:

$$\pi(X_1, X_2, \ldots, X_n) = \frac{e^{C_0+C_1X_1+\ldots+C_nX_n}}{1+e^{C_0+C_1X_1+\ldots+C_nX_n}},$$

where the X_is are the metrics included as independent variables, the C_is are the regression coefficients corresponding to X_is, and π is the probability that a fault was found in a class during validation. The univariate logistic regression model is a special case of the multivariate logistic regression model, where there is only one independent variable [6], [9].

When building a logistic regression model, all observations are assumed to be statistically independent. As in [6] and [9], we regard the (non)detection of a fault in a C++ class as an observation. Each fault (non)detection is assumed to be an event independent from other fault (non)detections. In particular, if more than one fault is found in a given class, each fault detection is treated as a separate observation in our data set. Thus, the number of faults found in a class is taken into account when we fit our models.

In our study, both univariate and multivariate logistic regression are used. Univariate regression analysis is used to examine the effect of each metric separately, identifying which metrics are significantly related to fault-proneness of classes and identifying potential fault-proneness predictors to be used in multivariate analysis. Multivariate regression analysis is used to examine the effectiveness of the metrics when used together. It tells us what levels of predictive performance can be achieved and which metrics may play a more dominant, significant role in predicting fault-proneness [21]. All results will be checked for the presence of influential observations by using the Cook's distance [20].

3.3.2 Machine Learning Model

To predict the fault-proneness of classes, we use three common machine learning methods, naive Bayes network, random forest, and NNge (nearest neighbor with generalization). All the methods are able to predict the fault-proneness of classes using just one metric. This is similar to univariate logistic regression. It is also possible to make predictions by using several metrics together, as in multivariate logistic regression.

TABLE 3
Descriptive Statistics of the Classes

Metric	N	Max.	75%	Median	25%	Min.	Mean	Std. dev.
WMC	145	100	22.00	12	8.00	0	17.42	17.45
DIT	145	6	1.50	1	0.00	0	1.00	1.26
RFC	145	222	44.50	28	10.00	0	34.38	36.20
NOC	145	5	0.00	0	0.00	0	0.21	0.70
CBO	145	24	14.00	8	3.00	0	8.32	6.38
LCOM	145	100	96.00	84	56.50	0	68.72	36.89
SLOC	145	2313	235.50	108	8.00	0	211.25	345.55

3.3.3 Model Evaluation

To facilitate the comparison of the performances of different models, each model is calibrated to make it satisfy the following condition: The number of classes misclassified as fault-prone and the number of classes misclassified as not fault-prone are roughly the same [9], [17]. The performance of a given prediction model is evaluated with the following three measures:

- *Precision*: the number of classes that are correctly classified, divided by the total number of classes;
- *Correctness*: the number of classes correctly classified as fault-prone, divided by the total number of classes classified as fault-prone; and
- *Completeness*: the sum of all faults in classes classified as fault-prone, divided by the total number of faults in the system.

To obtain a realistic estimate of the predictive power of a model when it is applied to data sets other than those from which the model was derived, we will employ leave-one-out (LOO) cross validation. For a data set with n observations, an LOO cross validation divides the data set into n parts, each part being used to assess a model built on the remainder of the data set. We also consider the important question, "Could these results of high/low severity fault models be obtained by some simpler strategy?" [31] We therefore perform the following comparisons: high severity fault model versus a simple prediction model based on class size and low severity fault model versus a simple prediction model based on class size.

4 Experimental Results

In this section, we present in detail the experimental results. In Section 4.1, we report the distribution and correlation analysis results of the investigated metrics on the pre-processed KC1 data set. In Section 4.2, we describe the results of univariate and multivariate logistic regression analyses for ungraded, high severity, and low severity faults. In Section 4.3, we show the results of three machine learning-based methods, namely, naive Bayes network, random forest, and NNge. In Section 4.4, our findings are used to validate the hypotheses proposed in Section 2.

4.1 Descriptive Analysis of the Data Set

Table 3 presents common descriptive statistics of the data set. A low median was found for DIT and NOC. This

TABLE 4
Correlations between the Metrics

KC1	WMC	DIT	RFC	NOC	CBO	LCOM	SLOC
WMC	1	0.14	0.63*	0.04	0.25*	0.32*	0.62*
DIT		1	0.65*	−0.03	0.47*	0.26*	0.35*
RFC			1	−0.05	0.39*	0.33*	0.51*
NOC				1	−0.03	−0.03	−0.04
CBO					1	0.26*	0.57*
LCOM						1	0.24*
SLOC							1

** Correlation is significant at the 0.01 level (2-tailed).*

TABLE 5
Result of Univariate Logistic Regression—Ungraded Severity Faults

Metric	Constant	Coefficient	Std. err.	*p*-value	R^2
WMC	0.366	0.069	0.012	0.000	0.196
DIT	1.747	0.094	0.093	0.312	0.003
RFC	0.748	0.026	0.004	0.000	0.137
NOC	2.007	−2.187	0.447	0.000	0.102
CBO	−0.790	0.284	0.030	0.000	0.367
LCOM	1.079	0.010	0.003	0.001	0.027
SLOC	−0.160	0.010	0.001	0.000	0.390

indicates that inheritance was not used a lot in the system under study. Similar results were found in [6], [9], and [14]. A high mean and median were found for CBO, suggesting that the coupling between classes, which is not caused by inheritance, is relatively high. This result is similar to what was found in [1] and [6]. A high mean and median were found for LCOM, implying that the cohesion of classes is relatively low. For all metrics except NOC, there are large differences between the lower 25th percentile, the median, and the 75th percentile, thus showing strong variations across classes. For NOC, there are 18 nonzero data points, which is larger than the threshold (five nonzero data values) suggested by [9]. All these metrics will be used in the following analysis.

Table 4 shows the results of linear Pearson's correlations analysis. Assuming a reasonably sized data set, Hopkins calls a correlation value less than 0.1 trivial, 0.1–0.3 minor, 0.3–0.5 moderate, 0.5–0.7 large, 0.7–0.9 very large, and 0.9–1.0 almost perfect [19]. From Table 4, it can be seen that three of the six design metrics are related to each other. We conclude that these metrics are not totally independent and represent redundant information. This result is different from what was found in [6], where the correlations between these metrics were very weak. Also, we can see that all the design metrics except NOC are significantly related to SLOC. In particular, SLOC has a large correlation with WMC, with CBO, and with RFC. The result is somewhat similar to what was found in [1].

4.2 Logistic Regression Models

4.2.1 Univariate Analyses

Table 5 summarizes the results of the univariate analysis with regard to fault-proneness prediction in terms of ungraded severity faults. The *Coefficient* is the estimated regression coefficient. The larger the absolute value of the coefficient, the stronger the impact (positive or negative, according to the sign of the coefficient) of the independent variable on the probability of a fault being detected in a class. The *p-value* is related to the statistical hypothesis and tells us whether the corresponding coefficient is significant or not. We use the $\alpha = 0.05$ significance level to assess the p-value we obtained. R^2 is defined as the proportion of the total variation in the dependent variable that is explained by the model. The higher R^2 is, the higher the effect of the model's independent variables, and the more accurate the model. However, as stated in [9], we should not interpret the logistic regression R^2s using the usual heuristics for least-square regression R^2s since they are built upon very different formulas. As a result, high R^2s are rare in logistic regression.

One influential observation for metric NOC was removed since the corresponding Cook's distance is larger than 2. From Table 5, we can see that six of the seven metrics are very significant $(\text{p-value} < 0.002)$ and DIT is the only metric which is not significant $(\text{p-value} = 0.312)$. SLOC has the largest R^2 value, which suggests that SLOC is the best predictor. CBO has the second largest R^2 value, which is slightly lower than that of SLOC. All metrics except NOC have positive regression coefficients. NOC has a negative coefficient, which means classes with higher NOC are less fault-prone. This result is in line with findings in [6] and [9]. However, the result is inconsistent with what was found in [1], where all metrics except NOC were significant and CBO had the largest R^2 value.

Table 6 summarizes the results of the univariate analysis with regard to fault-proneness prediction in terms of high severity faults. None of the univariate outliers was found to be influential. From Table 6, it can be seen that WMC, RFC, CBO, LCOM, and SLOC are very significant $(\text{p-value} < 0.001)$ and DIT is not significant $(\text{p-value} = 0.116)$. Again, SLOC has the largest R^2 value and CBO has the second largest R^2 value. The R^2 value of DIT is significantly smaller than the others, thus indicating that it is less useful. The coefficients in a logistic regression model are estimated based on the maximization of a likelihood function. SPSS uses iterative procedures involving approximations to obtain parameter

TABLE 6
Result of Univariate Logistic Regression—High Severity Faults

Metric	Constant	Coefficient	Std. err.	*p*-value	R^2
WMC	−2.229	0.040	0.011	0.000	0.172
DIT	−1.597	0.224	0.143	0.116	0.023
RFC	−2.444	0.023	0.005	0.000	0.228
NOC	–	–	–	–	–
CBO	−3.366	0.173	0.034	0.000	0.297
LCOM	−3.520	0.027	0.010	0.000	0.114
SLOC	−2.451	0.003	0.001	0.000	0.350

TABLE 7
Result of Univariate Logistic Regression—Low Severity Faults

Metric	Constant	Coefficient	Std. err.	p-value	R^2
WMC	0.313	0.068	0.012	0.000	0.200
DIT	1.684	0.082	0.092	0.375	0.002
RFC	0.694	0.026	0.004	0.000	0.135
NOC	1.928	−2.119	0.444	0.000	0.100
CBO	−0.862	0.284	0.030	0.000	0.369
LCOM	1.067	0.009	0.003	0.003	0.024
SLOC	−0.229	0.010	0.002	0.000	0.394

estimates for testing hypotheses and predictions. However, if the data are completely or partially separated, it may not be possible to obtain reliable maximum likelihood estimates since convergence may not occur [22]. In our data set, there is a quasi-complete separation for NOC. Therefore, we were unable to perform univariate logistic regression analysis on NOC.

Table 7 summarizes the results of the univariate analysis with regard to fault-proneness prediction in terms of low severity faults. One influential observation for metric NOC was removed since the corresponding Cook's distance is larger than 2. From Table 7, we can see that six of the seven metrics are very significant $(\text{p-value} < 0.004)$ and DIT is the only metric which is not significant $(\text{p-value} = 0.375)$. Overall, the results are very similar to what were obtained from ungraded severity faults.

4.2.2 Multivariate Analyses

In this section, we introduce and describe the performance of three multivariate fault-proneness prediction models built from design metrics only: one for ungraded severity faults, one for high severity faults, and a third for low severity faults. The main goal was to examine how well we could predict the fault-proneness of classes when the design metrics were used in combination. However, only those design metrics that were significant $(\text{p-value} < 0.05)$ in the univariate regression analyses were used to build the multivariate regression models. Also, to further investigate the prediction capability of these models, we compared them to the univariate model built with SLOC and one multivariate model built from all design metrics and SLOC. Since the examined metrics capture redundant information (see Table 4), all multivariate logistic regression models were built by a forward stepwise selection procedure for selecting the relevant metrics.

Ungraded severity fault model. In the univariate analysis, six metrics (WMC, RFC, NOC, CBO, LCOM, and SLOC) were statistically related to ungraded severity faults. We built two multivariate logistic regression models for ungraded severity faults: one built from five design metrics only and the other built from all the six metrics. We will refer to the former model as "model I" and the latter model as "model I′," which are shown in Tables 8a and 8b, respectively. As can be seen, all metrics that are significant in univariate analysis were selected in model I. The R^2 value of model I is 0.502. The signs of the coefficients of metrics RFC and LCOM are negative, though they were positive in univariate analysis. As stated in [9], this is due to suppressor relationships between the variables, which are commonly observed in multivariate regression analysis. One influential observation was found, since the corresponding Cook's distance is larger than 2. Thus, it was removed for the final model fitting. For model I′, no influential observation was found and the R^2 value is 0.508. Compared to model I, model I′ includes one more metric SLOC but excludes WMC.

Tables 9a, 9b, and 9c show the classification results when model I, the SLOC USF model (i.e., the univariate ungraded severity fault model built with SLOC), and model I′ are applied to the 145 classes, respectively. For each model, a class was classified "predicted fault-prone" if the predicted probability that it contained an ungraded severity fault was larger than or equal to a given threshold. In general, a low threshold tends to cause high completeness and low correctness and vice versa. As stated in [29], the best strategy is to seek a threshold that represents the trade-off between the correctness and completeness that is most appropriate for the given application. As in [9] and [17], we did not want this threshold to be arbitrary so, because we wanted to be systematic for all models, the threshold was selected to roughly balance the number of classes misclassified as fault-prone and not fault-prone. More specifically, we set the threshold at 0.865 for model I, at 0.79 for the SLOC USF model, and at 0.86 for model I′.

As can be seen, model I correctly classified 99 $(64 + 35)$ of the 145 classes. That is, it achieved 68.28 percent precision. In particular, 35 out of the 58 classes that were predicted to be fault-prone actually did contain an ungraded severity

TABLE 8
Ungraded Severity Fault Models: (a) Model I and (b) Model I′

Metric	Coefficient	Std. err.	p-value
Constant	−0.660	0.339	0.052
WMC	0.055	0.012	0.000
RFC	−0.016	0.006	0.004
NOC	−2.099	0.601	0.000
CBO	0.284	0.034	0.000
LCOM	−0.009	0.004	0.039

(a)

Metric	Coefficient	Std. err.	p-value
Constant	−0.232	0.331	0.484
SLOC	0.006	0.002	0.001
RFC	−0.014	0.006	0.021
NOC	−1.091	0.301	0.000
CBO	0.206	0.004	0.044
LCOM	−0.008	0.004	0.039

(b)

TABLE 9
Good-of-Fit: (a) Model I, (b) SLOC USF Model, and (c) Model I′

	Predicted	
Actual	no fault ($\pi < 0.865$)	fault ($\pi \geq 0.865$)
no fault	64	23
fault	23(114)	35(435)

(a)

	Predicted	
Actual	no fault ($\pi < 0.79$)	fault ($\pi \geq 0.79$)
no fault	66	21
fault	21(126)	37(423)

(b)

	Predicted	
Actual	no fault ($\pi < 0.86$)	fault ($\pi \geq 0.86$)
no fault	66	21
fault	21(128)	37(421)

(c)

fault (60.34 percent correctness) and 435 out of the 549 ungraded severity faults in the system (79.23 percent completeness). The prediction performance of model I is similar to what was found in [1], where the multivariate logistic regression model (without cross validation) achieved 69.61 percent precision, 72.57 percent correctness, and 65.24 percent completeness. Compared to model I, the SLOC USF model has a slightly higher precision (71.03 percent), a slightly higher correctness (63.79 percent), and a slightly lower completeness (77.05 percent). Model I′ does not outperform the SLOC USF model, thus indicating that the investigated design metrics may not be complementary to SLOC in predicting fault-proneness of classes.

Table 10 shows the values for the precision, correctness, completeness, and threshold of the univariate models and, in the last two rows, the multivariate logistic regression models. The model built with NOC has a very high completeness (98.36 percent), meaning that almost all the ungraded severity faults could be discovered by this metric. However, the low correctness (42.52 percent) and precision (47.59 percent) means that a large number of classes that do not contain any ungraded severity faults would have in that case been inspected. This would have been a large waste of testing resources. Both the models corresponding to DIT and LCOM have a poor precision, correctness, and completeness. These values correspond to the low R^2 values of DIT, LCOM, and NOC. On the other hand, the values of the models corresponding to WMC and RFC are more or less equal and are more acceptable than those models corresponding to DIT, LCOM, and NOC. SLOC appears to be the best predicator because the precision and correctness of the model built with SLOC are the largest and its completeness is also good. Compared to the model built with SLOC, the model built with CBO has a slightly lower precision and correctness but has a slightly higher completeness, indicating CBO has a prediction performance competitive with SLOC.

TABLE 10
Precision, Correctness, and Completeness—Ungraded Severity Faults

Metric	Precision	Correctness	Completeness	π
WMC	62.07%	52.38%	72.31%	0.80
DIT	51.03%	41.56%	52.46%	0.86
RFC	61.38%	51.72%	67.58%	0.84
NOC	47.59%	42.52%	98.36%	0.80
CBO	68.97%	60.32%	79.96%	0.88
LCOM	53.79%	42.37%	60.84%	0.88
SLOC	71.03%	63.79%	77.05%	0.79
Model I	68.28%	60.34%	79.23%	0.865
Model I'	71.03%	63.79%	76.68%	0.86

To obtain a realistic estimate of the predictive power of the models when it is applied to different data sets, we performed a LOO cross validation of model I, the SLOC USF model, and model I′, respectively. For the LOO cross validation, the 145 data points were divided into 145 partitions, each consisting of only one class. For each partition, we refitted each model using all data points not included in the partition, and then applied the model to the data point in the partition. We thus again obtain for all 145 classes a predicted probability of their fault-proneness.

Tables 11a, 11b, and 11c show the results from LOO cross validations of model I, the SLOC USF model, and model I′, respectively. For model I, the precision (68.97 percent), correctness (60.66 percent), and completeness (80.51 percent) all increase slightly. The prediction performance of model I in LOO cross validation is similar to what was found in [23], where the multivariate logistic regression model achieved 73.6 percent correctness and 62 percent completeness in 10-cross validation. Note that both our model and the model presented in [23] have a lower prediction performance than what was found in [9], where the best model achieved 78 percent correctness and 92 percent completeness in 10-cross validation. As stated in [23], one explanation for the high performance in the model presented in [9] is that it used a more complete set of design metrics (more than 50 metrics). For the SLOC USF model, the precision, correctness, and completeness all remain unchanged. For model I′, the precision (69.66 percent), correctness (62.07 percent), and completeness (76.14 percent) all decrease slightly. Overall, these results indicate that 1) these models are viable when applied to a comparable data set other than the one used for model fitting and their prediction capabilities for ungraded severity faults are acceptable, and 2) the SLOC USF model has performance competitive with the other two models in terms of precision, correctness, and completeness.

The above results are based on a particular threshold for the distinction of fault-prone/not fault-prone. In other words, Table 11 gives only a partial picture of these models, as other thresholds are possible in practice. To give a complete picture, Figs. 2a, 2b, and 2c show correctness and completeness as a function of the threshold π for model I, the SLOC USF model, and model I′, respectively. As can be seen, the correctness/completeness graph for model I differs from that for the SLOC USF model. The former has a slightly lower completeness curve for threshold less than

TABLE 11
Results from LOO Cross-Validation: (a) Model I, (b) SLOC USF Model, and (c) Model I′

	Predicted	
Actual	no fault ($\pi < 0.865$)	fault ($\pi \geq 0.865$)
no fault	63	24
fault	21(107)	37(442)

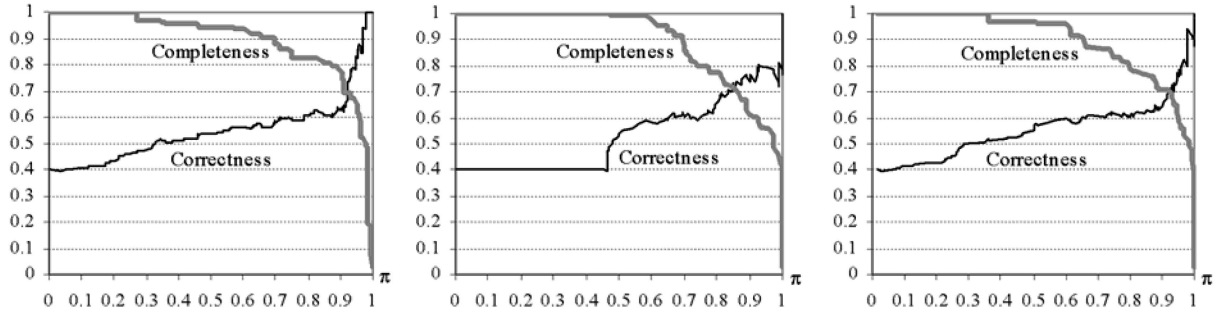
(a)

	Predicted	
Actual	no fault ($\pi < 0.79$)	fault ($\pi \geq 0.79$)
no fault	66	21
fault	21(126)	37(423)

(b)

	Predicted	
Actual	no fault ($\pi < 0.86$)	fault ($\pi \geq 0.86$)
no fault	65	22
fault	22(131)	36(418)

(c)



(a) (b) (c)

Fig. 2. Correctness and completeness of ungraded severity fault models. (a) Model I, (b) SLOC USF model, and (c) Model I′.

0.696 but has a higher correctness curve for thresholds less than 0.495 and for thresholds larger than 0.934. For thresholds between 0.696 and 0.934, model I achieves a higher completeness value and a lower correctness value than the SLOC USF model. On the other hand, the correctness/completeness graph for model I does not show a strong difference from that for model I′. Overall, within the range of practically useful values of π, there are no substantial differences in prediction performance for model I, the SLOC USF model, and model I′.

High severity fault model. Table 12 gives the results of the multivariate analysis with regard to fault-proneness prediction in terms of high severity faults. We will refer to the model built from design metrics only as "model II" (shown in Table 12a) and the model built from both design metrics and SLOC as "model II′" (shown in Table 12b). As can be seen, the metrics selected in model II are WMC and CBO but the metrics selected in model II′ are SLOC and CBO. Model II has an R^2 value of 0.382, while model II′ has an R^2 value of 0.409. No influential observation was found for either model.

Tables 13a, 13b, and 13c show the classification results when model II, the SLOC HSF model (i.e., the univariate high severity fault model built with SLOC), and model II′

TABLE 12
High Severity Fault Models: (a) Model II and (b) Model II′

Metric	Coefficient	Std. err.	p-value
Constant	−4.049	0.601	0.000
WMC	0.032	0.010	0.001
CBO	0.163	0.036	0.000

(a)

Metric	Coefficient	Std. err.	p-value
Constant	−3.446	0.543	0.000
SLOC	0.002	0.001	0.001
CBO	0.114	0.040	0.004

(b)

TABLE 13
Good-of-Fit: (a) Model II, (b) SLOC HSF Model, and (c) Model II′

	Predicted	
Actual	no fault ($\pi < 0.33$)	fault ($\pi \geq 0.33$)
no fault	118	11
fault	11(14)	5(20)

(a)

	Predicted	
Actual	no fault ($\pi < 0.27$)	fault ($\pi \geq 0.27$)
no fault	120	9
fault	9(12)	7(22)

(b)

	Predicted	
Actual	no fault ($\pi < 0.317$)	fault ($\pi \geq 0.317$)
no fault	118	11
fault	11(14)	5(20)

(c)

TABLE 14
Precision, Correctness, and Completeness—High Severity Faults

Metric	Precision	Correctness	Completeness	π
WMC	83.45%	21.43%	38.24%	0.28
DIT	79.31%	18.18%	35.29%	0.28
RFC	84.83%	31.25%	47.06%	0.26
NOC	–	–	–	–
CBO	81.38%	21.05%	50.00%	0.33
LCOM	67.59%	0.00%	0.00%	0.31
SLOC	87.59%	43.75%	64.71%	0.27
Model II	84.83%	31.25%	58.82%	0.33
Model II'	84.83%	31.25%	58.82%	0.317

are applied to the 145 classes, respectively. As can be seen, model II has the same good-of-fit (84.43 percent precision, 31.25 percent correctness, and 58.82 percent completeness) as model II′, both of which have a lower performance than the SLOC HSF model (87.59 percent precision, 43.75 percent correctness, and 64.71 percent completeness). For the SLOC HSF model, although the precision is high, both the correctness and completeness are low.

Table 14 shows the values for the precision, correctness, completeness, and thresholds of the univariate models and, in the last two rows, the multivariate logistic regression models. Since the univariate analysis cannot be performed on NOC, it is not possible to obtain the corresponding precision, correctness, and completeness. In the case of LCOM, all classes are classified as not fault-prone. Of these design metrics, the models corresponding to CBO and RFC have better classification performance. However, none of the design metrics is superior to the code size metric SLOC for predicting high severity faults.

Tables 15a, 15b, and 15c show the results from LOO cross validations of model II, the SLOC HSF model, and model II′, respectively. As can be seen, for model II, the precision (83.45 percent), correctness (25.00 percent), and completeness (47.06 percent) all fell. This is also true for the SLOC HSF model (86.70 percent precision, 40.00 percent correctness, and 50.00 percent completeness). For model II′, the precision remains unchanged (84.83 percent) but both the correctness (33.33 percent) and completeness (61.76 percent) increase slightly. Of the three models, model II′ has the highest completeness. While model II′ is capable of identifying 18 out of 145 classes (i.e., 12 percent of all classes), which contain more than 61 percent of all high severity faults, the fact that we still miss 39 percent of the high severity faults if we rely on the model prediction may hamper the model's practicality. Furthermore, the low correctness means that most classes that actually do not contain any high severity faults will undergo thorough but unnecessary inspection.

Figs. 3a, 3b, and 3c show the correctness and completeness as a function of π for model II, the SLOC HSF model, and model II′, respectively. As can be seen, the correctness/completeness graph for model II differs from that for the SLOC HSF model. The main difference is that the latter has a higher correctness/completeness curve for thresholds larger than 0.6. On the other hand, we can see that the correctness/completeness graph for model II does not show a strong difference from that for model II′. For any of these three models, it is difficult to select a threshold that produces acceptable correctness and completeness. In

TABLE 15
Results from LOO Cross-Validation: (a) Model II, (b) SLOC HSF Model, and (c) Model II′

	Predicted	
Actual	no fault ($\pi < 0.33$)	fault ($\pi \geq 0.33$)
no fault	117	12
fault	12(18)	4(16)

(a)

	Predicted	
Actual	no fault ($\pi < 0.27$)	fault ($\pi \geq 0.27$)
no fault	120	9
fault	10(17)	6(17)

(b)

	Predicted	
Actual	no fault ($\pi < 0.317$)	fault ($\pi \geq 0.317$)
no fault	117	12
fault	10(13)	6(21)

(c)

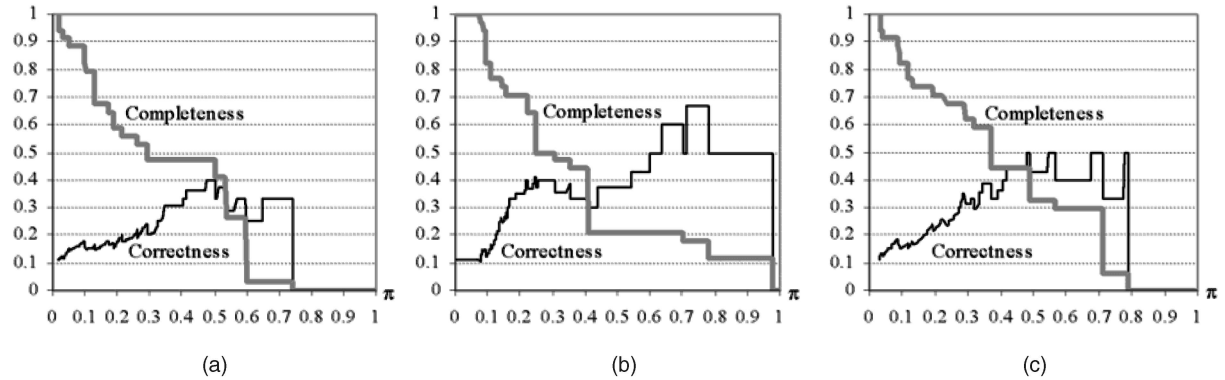


Fig. 3. Correctness and completeness of high severity fault models. (a) Model II, (b) SLOC HSF model, and (c) Model II′.

TABLE 16
Low Severity Fault Models: (a) Model III and (b) Model III′

Metric	Coefficient	Std. err.	p-value
Constant	−0.713	0.343	0.038
WMC	0.057	0.012	0.000
RFC	−0.018	0.006	0.003
NOC	−2.020	0.594	0.001
CBO	0.290	0.035	0.000
LCOM	−0.010	0.004	0.020

(a)

Metric	Coefficient	Std. err.	p-value
Constant	−0.431	0.348	0.216
SLOC	0.004	0.002	0.034
WMC	0.031	0.015	0.000
RFC	−0.019	0.006	0.002
NOC	−1.063	0.302	0.000
CBO	0.224	0.040	0.000
LCOM	−0.011	0.004	0.012

(b)

TABLE 17
Good-of-Fit: (a) Model III, (b) SLOC LSF Model, and (c) Model III′

	Predicted	
Actual	no fault ($\pi < 0.87$)	fault ($\pi \geq 0.87$)
no fault	67	21
fault	22(106)	35(409)

(a)

	Predicted	
Actual	no fault ($\pi < 0.784$)	fault ($\pi \geq 0.784$)
no fault	68	20
fault	20(120)	37(395)

(b)

	Predicted	
Actual	no fault ($\pi < 0.859$)	fault ($\pi \geq 0.859$)
no fault	66	22
fault	22(131)	35(384)

(c)

particular, the correctness cannot reach the maximum value of 1 no matter what threshold is used. The reason is that the class with the maximum value of predicted fault-proneness does not actually contain high severity faults.

Low severity fault model. Table 16 gives the results of the multivariate analysis with regard to fault-proneness prediction in terms of low severity faults. We will refer to the model built from only design metrics as "model III" (shown in Table 16a) and the model built from both design metrics and SLOC as "model III′" (shown in Table 16b). As can be seen, all design metrics that are significant in univariate analysis were selected in model III. The R^2 value of model III is 0.511, which is slightly better than that of model I. One influential observation was found since the corresponding Cook's distance is larger than 2. Thus, it was removed for the final model fitting. Overall, this model is very similar to model I. Compared to model III, model III′ includes one more metric SLOC, which has a slightly higher R^2 value of 0.522. No influential observation was found for model III′.

Tables 17a, 17b, and 17c show the classification results when model III, the SLOC LSF model (i.e. the univariate low severity fault model built with SLOC), and model III′ are applied to the 145 classes, respectively. As can be seen, model III (70.34 percent precision, 62.50 percent correctness, and 79.42 percent completeness) has a slightly better good-of-fit than model III′ (69.66 percent precision, 61.40 percent correctness, and 74.56 percent completeness). Compared to model III, the SLOC LSF model has a slightly higher precision (72.41 percent), a slightly higher correctness (64.91 percent), and a slightly lower completeness (76.70 percent).

Table 18 shows the values for the precision, correctness, completeness, and thresholds of the univariate models and, in the last two rows, the multivariate logistic regression models. Overall, the results are very similar to those given in Table 10 for ungraded severity faults. Again, the results indicate that SLOC and CBO are the best two predictors. Compared to the results in Table 10, we can see that the prediction performances of most univariate models for low severity faults fall slightly. The prediction performance of the multivariate regression model built with design metrics only increases slightly, while the prediction performance of the multivariate regression model built with design metrics and SLOC decreases slightly.

Tables 19a, 19b, and 19c show the results from LOO cross validations of model III, SLOC LSF model, and model III′, respectively. As can be seen, compared against the results in Table 17, for any of the models, the precision, correctness, and completeness do not change much. Overall, the results indicate that SLOC LSF model has a

TABLE 18
Precision, Correctness, and Completeness—
Low Severity Faults

Metric	Precision	Correctness	Completeness	π
WMC	62.76%	52.63%	70.10%	0.80
DIT	50.34%	40.26%	51.65%	0.85
RFC	60.69%	50.00%	67.57%	0.83
NOC	46.21%	41.73%	98.25%	0.80
CBO	69.66%	60.32%	79.81%	0.87
LCOM	53.10%	40.68%	60.00%	0.871
SLOC	72.41%	64.91%	76.70%	0.784
Model III	70.34%	62.50%	79.42%	0.87
Model III'	69.66%	61.40%	74.56%	0.859

TABLE 19
Results from LOO Cross Validation: (a) Model III, (b) SLOC LSF Model, and (c) Model III′

	Predicted	
Actual	no fault ($\pi < 0.87$)	fault ($\pi \geq 0.87$)
no fault	65	23
fault	21(100)	36(415)

(a)

	Predicted	
Actual	no fault ($\pi < 0.784$)	fault ($\pi \geq 0.784$)
no fault	66	22
fault	20(120)	37(395)

(b)

	Predicted	
Actual	no fault ($\pi < 0.859$)	fault ($\pi \geq 0.859$)
no fault	67	21
fault	24(138)	33(377)

(c)

competitive performance to the other two models in terms of precision, correctness, and completeness.

Figs. 4a, 4b, and 4c show the correctness and completeness as a function of the threshold π for model III, the SLOC LSF model, and model III′, respectively. As can be seen, the correctness/completeness graph for model III differs from that for the SLOC LSF model. The main difference is that the former has a slightly lower completeness curve for thresholds less than 0.666 but has a higher correctness curve for thresholds less than 0.5 and for thresholds larger than 0.923. For thresholds between 0.666 and 0.923, model III achieves a higher completeness value and a lower correctness value than SLOC LSF model. On the other hand, the correctness/completeness graph for model III does not show a strong difference from that for model III′. Overall, within the range of practically useful values of π, there are no substantial differences in prediction performance for model III, the SLOC LSF model, and model III′. We can also see that model III is slightly better than model I in terms of fault-proneness prediction performance.

4.2.3 Applicability of Logistic Regression Models to Fault-Proneness Ranking of Classes

The practical purpose of building fault-proneness prediction models is to apply them to software systems. In Section 4.2.2, a multivariate logistic regression model was built on one system by selecting a certain threshold so that it achieved a good prediction performance within LOO cross validation. The straightforward way to apply such a model to another system is to calculate the predicted fault-proneness for each class in the new system. If it is above the predetermined threshold, the class is classified as fault-prone. Unfortunately, empirical results indicated that the predetermined threshold might be inappropriate for the new system because of system differences [23].

Studies have suggested, however, that the benefits of logistic regression models were still across systems when such models were applied to class ranking [23], [24]. In this context, classes are ranked by their predicted fault-proneness. A fault-proneness ranking lists classes from the most to the least fault-prone. Software managers would find such a ranking very helpful since they can simply select from the list as many potential faulty classes as available resources will allow [25].

In what follows, we examine the applicability of models II and III to class ranking in two scenarios. Assume that the n classes with the highest predicted fault-proneness in the ranking list are selected for quality enhancement. In the first scenario, the question is how the maximum number of actual faulty classes that can be potentially detected increase with n. In the second scenario, the question is how the maximum number of faults that can be potentially detected increase with n.

In both scenarios, software managers are interested in knowing how the fault-proneness ranking of classes generated by the logistic regression models II and III compared with a ranking obtained using some simpler strategy. We, therefore, compare the fault-proneness ranking performance of the High severity fault model versus a simple prediction model based on class size and of the Low severity fault model versus a simple prediction model based on class size.

In previous sections, we used SLOC as the class size metric. However, SLOC can only be obtained in the

Fig. 4. Correctness and completeness of low severity fault models. (a) Model III, (b) SLOC LSF model, and (c) Model III′.

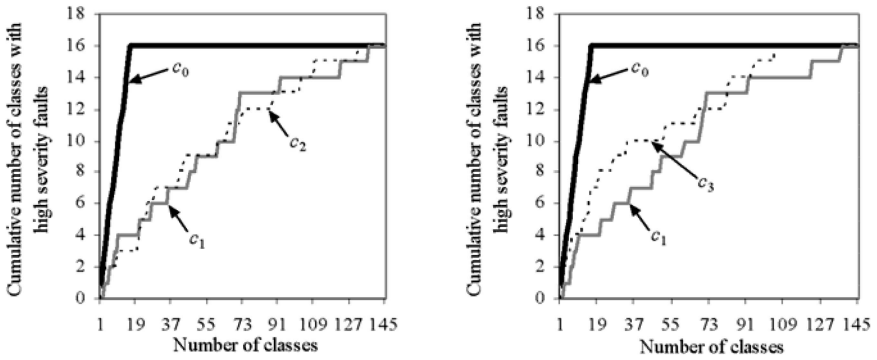


Fig. 5. Cumulative number of classes with high severity faults when classes are sorted in decreasing order with respect to different criteria. (a) Number of faults, model II, and NOM, and (b) Number of faults, model II, and SLOC.

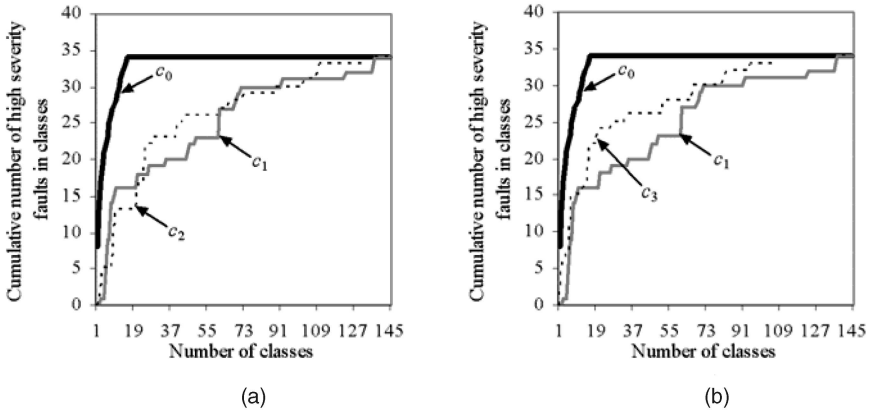


Fig. 6. Cumulative number of high severity faults in the classes when the classes are sorted in decreasing order with respect to different criteria. (a) Number of faults, model II, and NOM, and (b) Number of faults, model II, and SLOC.

implementation phase. Since all the metrics investigated in this study are design metrics, we are also interested in a class size metric that can be obtained in the design phase. In [23], Briand et al. used the Number of Methods (NOM) as the class size metric, by which the classes were ranked in decreasing order to investigate the cumulative faults. In the following, in addition to SLOC, we also use NOM as the class size metric. We will compare the rankings generated by the regression models II and III with those obtained by ordering the classes according to NOM/SLOC.

Figs. 5a and 5b, respectively, plot the cumulative number of classes with high severity faults when the classes are ranked in decreasing order according to different criteria.[3] In both figures, the bold line c_0 represents the cumulative number of classes with high severity faults with respect to the class ranking r_0 produced by the number of high severity faults. The thin line c_1 represents the cumulative number of classes with high severity faults with respect to the ranking r_1 produced by the predicted fault-proneness of model II. Note that the values for the predicted fault-proneness were taken from the LOO cross validation. The dashed lines c_2 in Figs. 5a and c_3 in Fig. 5b, respectively, represent the cumulative number of classes with high severity faults with respect to the ranking r_2 produced by NOM and the ranking r_3 produced by SLOC. Figs. 5a and 5b slightly differ from Alberg diagrams [28], where the y-axis represents the cumulative percentage of faults rather than the cumulative number of faults.

From Fig. 5a, we can see that c_1 and c_2 interweave. In practice, software managers are interested in the prediction performance of a fault-proneness ranking of classes only at a preferred cutoff percentile value [25]. To examine whether there is a significant difference between the prediction performances of r_1 and r_2, we performed a paired-samples t test between all pairs of data values in c_1 and c_2 that fall above the following representative cutoff percentile values: top 10 percent, top 20 percent, top 30 percent, top 40 percent, top 50 percent, and top 60 percent. All the p-values for these cutoff points were larger than 0.05. Therefore, in this context, model II does not outperform the simple model based on NOM. When similar analysis was performed on c_1 and c_3 in Fig. 5b, however, all the p-values for these cutoff points were less than 0.05. This indicates that the simple model based on SLOC is significantly better than model II.

Figs. 6a and 6b show the cumulative number of high severity faults in the classes when the classes are ranked in decreasing order according to different criteria, respectively. c_0, c_1, c_2, and c_3, respectively, represent the cumulative number of high severity faults in the classes

3. There are actually four criteria: the rankings produced by the number of faults, by the predicted fault-proneness, by NOM, and by SLOC. We use two subfigures to describe them, each having three curves, rather than use only one figure with four curves. The reason is that these curves interweave and, thus, separating descriptions enables a clearer understanding of our analyses.

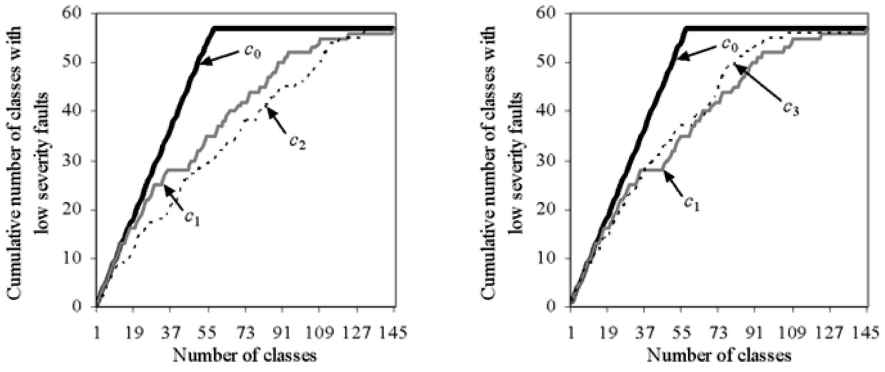


Fig. 7. Cumulative number of classes with low severity faults when classes are sorted in decreasing order with respect to different criteria. (a) Number of faults, model III, and NOM, and (b) Number of faults, model III, and SLOC.

with respect to the ranking r_0 produced by the number of high severity faults, the ranking r_1 produced by the predicted fault-proneness of model II, the ranking r_2 produced by NOM, and the ranking r_3 produced by SLOC. We performed a paired-samples t test between all pairs of data values in c_1 and c_2 that fall above the following representative cutoff percentile values: top 10 percent, top 20 percent, top 30 percent, top 40 percent, top 50 percent, and top 60 percent. The results show that there is no significant difference between the prediction performances of r_1 and r_2 for the cutoff points top 10 percent, top 20 percent, and top 30 percent. However, the prediction performance of r_1 is significantly worse than that of r_2 for the cutoff points top 40 percent, top 50 percent, and top 60 percent. Therefore, in this context, we also conclude that model II does not outperform the simple model based on NOM. When similar analysis was performed on c_1 and c_3 in Fig. 6b, however, the results indicate that the simple model based on SLOC is significantly better than model II.

Figs. 7a and 7b, respectively, show the cumulative number of classes with low severity faults, when the classes are ranked in decreasing order according to the number of known low severity faults, the predicted fault-proneness by model III, and NOM/SLOC. Here, c_0, c_1, c_2, c_3, r_0, r_1, r_2, and r_3 have the same meanings as in Fig. 5 except that low severity faults rather than high severity faults are involved. The first top 13 classes of r_1 are actually faulty and most of the next top 16 classes of r_1 are faulty. We also reach a maximum of 14 percent (8 out of 57 classes) difference between c_1 and c_2. Furthermore, we performed a paired-samples t test between all pairs of data values in c_1 and c_2 that fall above the following representative cutoff percentile values: top 10 percent, top 20 percent, top 30 percent, top 40 percent, top 50 percent, and top 60 percent. The results show that the prediction performance of r_1 is significantly better than that of r_2 for all the cutoff points ($\alpha = 0.05$). Therefore, in this context, we conclude that model III outperforms the simple model based on NOM. When similar analysis was performed on c_1 and c_3 in Fig. 7b, the results show that the prediction performance of r_1 is significantly better than that of r_3 for the cutoff points top 10 percent, top 20 percent, and top 30 percent, has no significant difference with that of r_3 for the cutoff points top 40 percent, top 50 percent, and is significantly worse than that of r_3 for the cutoff points top 60 percent. Overall, in this context, we conclude that model III also outperforms the simple model based on SLOC.

Figs. 8a and 8b, respectively, show the cumulative number of low severity faults in the classes when the classes are ranked in decreasing order according to the

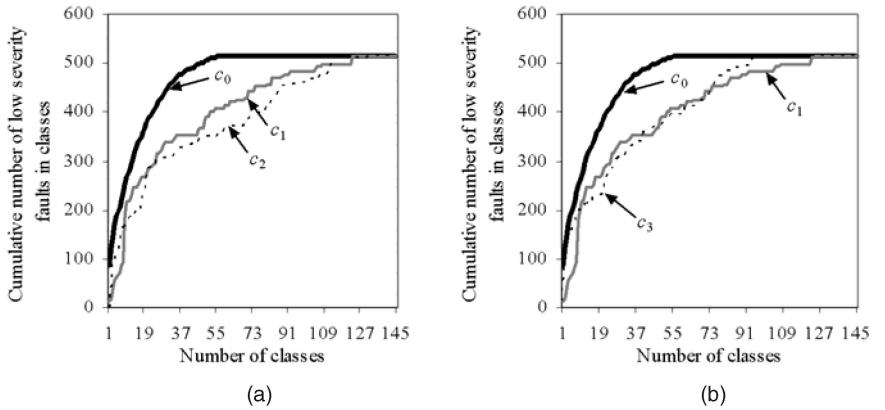


Fig. 8. Cumulative number of low severity faults in the classes when the classes are sorted in decreasing order with respect to different criteria. (a) Number of faults, model III, and NOM, and (b) Number of faults, model III, and SLOC.

TABLE 20
Results of Machine Learning Methods—Ungraded Severity Fault Models

Metric	Naive Bayes			Random forest			NNge		
	Prec.	Correct.	Comp.	Prec.	Correct.	Comp.	Prec.	Correct.	Comp.
WMC	61.38%	58.33%	32.42%	55.86%	44.83%	45.17%	55.86%	44.83%	43.72%
DIT	57.24%	16.67%	0.36%	62.76%	62.50%	22.04%	55.86%	44.83%	62.66%
RFC	64.14%	75.00%	38.43%	57.24%	46.67%	61.02%	60.00%	50.00%	48.45%
NOC	60.00%	–	0.00%	60.00%	50.00%	1.64%	55.86%	45.16%	59.38%
CBO	70.34%	64.15%	58.11%	66.90%	58.06%	58.83%	68.97%	61.02%	58.65%
LCOM	60.00%	–	0.00%	55.86%	44.64%	52.09%	56.55%	46.03%	30.60%
SLOC	68.97%	78.26%	53.19%	73.79%	67.86%	82.15%	73.10%	66.67%	82.88%
Multi. 1	71.72%	66.04%	77.23%	71.72%	66.67%	74.13%	67.59%	59.32%	59.56%
Multi. 2	71.72%	71.79%	68.81%	75.86%	71.70%	80.87%	73.10%	67.27%	78.87%

TABLE 21
Results of Machine Learning Methods—High Severity Fault Models

Metric	Naive Bayes			Random forest			NNge		
	Prec.	Correct.	Comp.	Prec.	Correct.	Comp.	Prec.	Correct.	Comp.
WMC	88.28%	40.00%	14.71%	87.59%	33.33%	11.76%	80.00%	19.05%	17.65%
DIT	88.28%	–	0.00%	88.97%	–	0.00%	80.69%	12.50%	8.82%
RFC	87.59%	33.33%	26.47%	87.76%	20.00%	20.59%	77.93%	10.00%	11.76%
NOC	88.97%	–	0.00%	88.97%	–	0.00%	82.76%	20.00%	8.82%
CBO	88.97%	–	0.00%	86.90%	20.00%	14.70%	80.00%	11.76%	17.65%
LCOM	88.97%	–	0.00%	87.59%	33.33%	8.82%	83.45%	21.43%	29.41%
SLOC	88.28%	44.44%	44.12%	83.45%	21.43%	8.82%	82.07%	14.29%	5.88%
Multi. 1	84.83%	31.25%	47.06%	84.14%	21.05%	47.06%	80.69%	20.00%	41.18%
Multi. 2	84.14%	29.41%	47.06%	80.00%	19.05%	44.12%	83.45%	25.00%	41.18%

known low severity faults, the predicted fault-proneness of model III, and NOM/SLOC. Here, c_0, c_1, c_2, c_3, r_0, r_1, r_2, and r_3 have the same meanings as those in Fig. 6, except that low severity faults rather than high severity faults are involved. From Fig. 8a, it can be seen that we reach a maximum of 11 percent (58 out of 515 low severity faults) difference between c_1 and c_2. Furthermore, we performed a paired-samples t test between all pairs of data values in c_1 and c_2 that fall above the following representative cutoff percentile values: top 10 percent, top 20 percent, top 30 percent, top 40 percent, top 50 percent, and top 60 percent. The results show that there is no significant difference between the prediction performances of r_1 and r_2 for the cutoff points top 10 percent and top 20 percent. However, for all the other cutoff points, the prediction performance of r_1 is significantly better than that of r_2 ($\alpha = 0.05$). Therefore, overall, model III outperforms the simple model based on NOM. When similar analysis was performed on c_1 and c_3 in Fig. 8b, the results show that the prediction performance of r_1 is slightly worse than that of r_3 for the cutoff points top 10 percent (p-value $= 0.049$) and has no significant difference with that of r_3 for all the other cutoff points. Overall, in this context, the results indicate that model III has similar performance to the simple model based on SLOC.

4.3 Machine Learning Models

In this section, we present the class fault-proneness prediction results for three machine learning methods: naive Bayes network, random forest, and NNge. As in logistic regression analyses, the modeling parameters of each machine learning method are varied to roughly balance the number of classes misclassified as fault-prone and not fault-prone. These machine learning algorithms are available in the WEKA (Waikato Environment for Knowledge Analysis) open source tool.[4]

Table 20 summarizes the results of machine learning methods with regard to fault-proneness prediction in terms of ungraded severity faults, in which *Prec.*, *Correct.*, and *Comp.* denote precision, correctness, and completeness, respecitively. We show the performance for the model build with all design metrics (Multi. 1), the model built with all design metrics and SLOC (Multi. 2), and the models built with individual metrics. As can be seen, the models built with all design metrics have better precision and correctness than model I, but their completeness is inferior to that of model I. Of all the naive Bayes models built with a single metric, the model corresponding to CBO has a competitive performance to the model corresponding to SLOC. Of all the random forest and NNge models built with a single metric, the model corresponding to SLOC has the best performance. Overall, the results indicate that SLOC and CBO are the two best predictors for ungraded severity faults.

Table 21 summarizes the results of machine learning methods with regard to fault-proneness prediction in terms of high severity faults. As can be seen, all the models built

4. Available from http://www.cs.waikato.ac.nz/ml/weka/.

TABLE 22
Results of Machine Learning Methods—Low Severity Fault Models

Metric	Naive Bayes			Random forest			NNge		
	Prec.	Correct.	Comp.	Prec.	Correct.	Comp.	Prec.	Correct.	Comp.
WMC	61.38%	54.55%	31.07%	55.86%	44.44%	51.84%	57.93%	46.55%	53.20%
DIT	57.93%	16.67%	0.19%	64.83%	68.75%	24.24%	58.62%	47.46%	44.66%
RFC	64.83%	75.00%	39.03%	57.24%	45.61%	46.41%	57.24%	45.90%	46.41%
NOC	60.69%	–	0.00%	60.69%	–	0.00%	57.24%	44.44%	50.49%
CBO	69.66%	62.75%	56.70%	68.28%	59.32%	57.28%	69.66%	61.40%	69.51%
LCOM	60.69%	–	0.00%	60.00%	48.78%	39.03%	57.93%	46.77%	54.37%
SLOC	69.66%	78.26%	52.04%	74.48%	68.52%	81.94%	72.41%	66.04%	82.52%
Multi. 1	72.41%	66.04%	78.25%	71.03%	63.16%	75.92%	72.41%	64.41%	76.31%
Multi. 2	72.41%	71.79%	69.71%	76.55%	71.70%	78.83%	79.31%	73.68%	86.60%

with a single metric have low correctness and poor completeness, although their precision is good. The worst prediction performance is that of the model corresponds to DIT, which is slightly worse than the model corresponding to NOC. The two multivariate models also have poor prediction performance and are all worse than the multivariate logistic regression models.

Table 22 summarizes the results of machine learning methods with regard to fault-proneness prediction in terms of low severity faults. As can be seen, overall, the two best metrics in this study are SLOC and CBO. Of all the naive Bayes models built with a single metric, the model corresponding to CBO has a competitive performance to the model corresponding to SLOC. Of the random forest models built with a single metric, the model corresponding to SLOC has the best precision, correctness, and completeness. This can also be seen in all the NNge models built with a single metric. Again, the values of the models corresponding to WMC and RFC are more or less equal and are more acceptable than the values of models corresponding to DIT, LCOM, and NOC.

4.4 Validation of Hypotheses and Discussion

In this section, we will use our findings to validate our hypotheses of Section 2. We will also compare the fault-proneness prediction capability of the metrics for high severity faults and low severity faults. Apart from the analysis results of the regression analysis, the precision, correctness, and completeness values can also help us in the validation process. These values were calculated using logistic regression, naive Bayes, random forest, and NNge.

- **H-WMC** (*for ungraded, high, and low severity faults*). The results of univariate regression analyses, for ungraded, high, and low severity faults, indicate that the effect of WMC on fault-proneness is positive and very significant. Therefore, the H-WMC hypotheses for ungraded, high, and low severity faults are supported: The larger the WMC, the higher the probability of fault detection. However, the classification results of the logistic regression and machine learning models indicate that WMC performs much better in predicting fault-prone classes in terms of ungraded/low severity faults than in predicting fault-prone classes in terms of high severity faults.
- **H-DIT** (*for ungraded, high, and low severity faults*). The results of univariate regression analyses, for ungraded, high, and low severity faults, indicate that the effect of DIT on fault-proneness is not significant. Therefore, the H-DIT hypotheses for ungraded, high, and low severity faults are not supported. Similar results were found in [4], [11], and [27], where the severity of faults was not distinguished. However, some previous research, such as [6] and [9], found that DIT was very significant when the severity of faults was not considered. The relatively poor classification performance of the logistic regression and machine learning models are also in line with the results of univariate analyses. DIT performs better in predicting fault-prone classes in terms of ungraded/low severity faults than in predicting fault-prone classes in terms of high severity faults.
- **H-RFC** (*for ungraded, high, and low severity faults*). The results of univariate regression analyses, for ungraded, high, and low severity faults, indicate that the effect of RFC on fault-proneness is positive and very significant. Therefore, the H-RFC hypotheses for ungraded, high, and low severity faults are supported: The larger the RFC, the higher the probability of fault detection. However, RFC performs much better in predicting fault-prone classes in terms of ungraded/low severity faults than in predicting fault-prone classes in terms of high severity faults.
- **H-NOC** (*for ungraded, high, and low severity faults*). For high severity faults, we failed to perform univariate logistic regression analysis because there is a quasi-complete separation for NOC. Therefore, we cannot validate H-NOC for high severity faults. However, the classification results of machine learning models indicate that the fault-proneness prediction capability of NOC is not better than that of DIT. For ungraded/low severity faults, NOC was found to be very significant, but in an inverse direction: The larger the NOC, the lower the probability of ungraded/low severity fault detection. This result is in line with what were found in previous research [6] and [9] but is contrary to what was found in [11].

TABLE 23
Validation Results of the Hypotheses

H-M	Faults		
	Ungraded severity	High severity	Low severity
H-WMC	√	√	√
H-DIT	×	×	×
H-RFC	√	√	√
H-NOC	×		×
H-CBO	√	√	√
H-LCOM	√	√	√
H-SLOC	√	√	√

- **H-CBO** (*for ungraded, high, and low severity faults*). CBO was shown to be very significant for ungraded, high, and low severity faults. Therefore, the H-CBO hypotheses for ungraded, high, and low severity faults are supported: the larger the CBO, the higher the probability of fault detection. Both regression analysis and machine learning models indicate that CBO has the best fault-proneness prediction performance for ungraded/low severity faults. Also, CBO performs much better in predicting fault-prone classes in terms of ungraded/low severity faults than in predicting fault-prone classes in terms of high severity faults.
- **H-LCOM** (*for ungraded, high, and low severity faults*). LCOM was found to very significant for high severity faults and significant for ungraded/low severity faults. Therefore, the H-LCOM hypotheses for ungraded, high, and low severity faults are supported: The larger the LCOM, the higher the probability of fault detection. Also, LCOM performs much better in predicting fault-prone classes in terms of ungraded/low severity faults than in predicting fault-prone classes in terms of high severity faults.
- **H-SLOC** (*for ungraded, high, and low severity faults*). SLOC was found to be very significant for high severity faults and significant for ungraded/low severity faults. Therefore, the H-SLOC hypotheses for ungraded, high, and low severity faults are supported: The larger the SLOC, the higher the probability of fault detection. Also, SLOC performs much better in predicting fault-prone classes in terms of ungraded/low severity faults than in predicting fault-prone classes in terms of high severity faults.

Table 23 summarizes the validation results of the hypotheses stated in Section 2, in which $\surd$ means that the hypothesis is supported, $\times$ means that the hypothesis is not supported, and a blank entry means that the hypothesis was not examined.

In the preceding, the hypotheses were validated using univariate analyses. In the following, we evaluate the performance of univariate and multivariate models with regard to different fault severities. As discussed in Section 4.2, for high severity faults, both the univariate and multivariate regression models have rather low correctness and completeness, although they have high precision. In particular, of all metrics, SLOC is the best predictor. When applied to fault-proneness ranking of classes, the multivariate model built only with design metrics did not outperform even a simple model that would be immediately applicable based on class size (measured by NOM or SLOC). The precision, correctness, and completeness of the machine learning models are also worse than those of the regression model. These results indicate that the fault-proneness prediction usefulness of these design metrics with regard to high severity faults is limited, although most of them are statistically related to fault-proneness of classes.

For low severity faults, the corresponding prediction performance of the univariate and multivariate regression model built only with design metrics are comparable to what was found in [1] and [23], although they are not very satisfactory. Of all design metrics, CBO has a competitive prediction capability to SLOC. When applied to the fault-proneness ranking of classes, the top fraction of the ranking produced by the multivariate regression model built only from design metrics contains more faulty classes than that of the ranking produced by a simple model based on class size (measured by NOM or SLOC). When class size is measured by NOM, the top fraction of the former ranking also contains more faults than that of the latter ranking. When class size is measured by SLOC, both the rankings have similar prediction capabilities. Overall, the results mean that the regression model is more useful to software managers than the simple model in practice. The precision, correctness, and completeness of the machine learning models are also more or less equal to those of the regression model. These results suggest that the design metrics investigated perform much better in predicting fault-prone classes in terms of low severity faults than in predicting fault-prone classes in terms of high severity faults.

5 Related Work

The earliest research that investigated the relationships between OO design metrics and fault-proneness of classes appears to be Basili et al.'s [6]. Based on a study of eight medium-sized systems developed by students in C++, they examined the effect of the CK metrics suite on fault-proneness of classes. Except for the metric LCOM, all other metrics were the same as in our study. Based on logistic regression analyses, they found that WMC was somewhat significant, CBO was significant, DIT and RFC were very significant, and LCOM was not significant. NOC was found to be very significant but in an inverse direction, i.e., a high NOC meant a low probability of fault detection.

Briand et al. carried out an industrial case study on quality factors in OO designs [26]. They found that CBO, RFC, and LCOM were statistically related to the fault-proneness of classes. Later, Birand et al. investigated the relationships based on a set of size metrics and a more complete set of OO design metrics [9]. They found that CBO, RFC, DIT, and SLOC were significant predictors of fault-proneness, and NOC was also significantly related to fault-proneness but in an inverse direction. In 2001, Briand et al. performed a replication of the study [17]. The results differed from [9] in the following aspects: 1) DIT was statistically related to fault-proneness in an inverse direction, and 2) NOC was not found to be significant.

TABLE 24
Summary of Related Work

Study	Modeling method	Language	Fault severity distinguished?		Summary of results						
					WMC	DIT	RFC	NOC	CBO	LCOM	SLOC
Basili et al. [6]	LR	C++	No		+	++	++	−−	+		
Briand et al. [26]	LR	C++	No				+		++		
Tang et al.[27]	LR	C++	No		+	0	+	0	0		
Briand et al. [9]	LR	C++	No		+	++	++	−	++		++
Briand et al. [17]	LR	C++	No		++	−	++	0	++		
El Eman et al. [4]	LR	C++	No	#1	+	0	++		+		++
				#2	0	0	0		0		++
Subramanyam et al. [2]	OLS	C++	No		+	−			+		+
		Java	No		0	−			−		++
Yu et al. [11]	OLS+LDA	Java	No		++	0	+	+	+	+	++
Gyimothy et al. [1]	LR+ML	C++	No		++	+	++	0	++		++
Our study	LR+ML	C++	High severity		++	0	++		++	++	++
			Low severity		++	0	++	−−	++	+	++
			Ungraded severity		++	0	++	−−	++	+	++

LR: Logistic regression. OLS: Ordinary least square linear regression. LDS: Linear discriminant analysis. ML: Machine learning. #1: Without size control. #2: With size control.

Tang et al. analyzed the CK metrics suite on three industrial real-time systems implemented in C++ that contained a number of natural faults reported over the preceding three years [27]. They found none of the CK metrics to be significant except WMC and RFC. Yu et al. examined the correlations of 10 metrics with fault-proneness, where CBO and RFC were divided into two different types [11]. They used the same LCOM metric as our study. The subject system was the client side of a large network service management system, which was written in Java and consisted of 123 classes. The validation was carried out using linear regression analysis and linear discriminant analysis. The results indicated that WMC, SLOC, $\mathrm{CBO}_{\mathrm{out}}$, $\mathrm{RFC}_{\mathrm{out}}$, LCOM, and NOC were significant.

El Eman et al. analyzed a large C++ telecommunication application and found that the size (i.e., SLOC) of classes confounded the effect of most OO design metrics on faults [4]. In their experiment, WMC, RFC, CBO, and LCOM were found to be significant without size control but none of these metrics was significant after controlling for the size of the system. NOC was not investigated. To further examine El Eman et al.'s findings, Subramanyam and Krisnan analyzed an e-commerce application developed in C++ and Java [2]. They performed an experiment based on 405 C++ and 301 Java classes to study the effect of the size along with the WMC, CBO, and DIT on fault-proneness of classes. Their results indicated, however, that even after controlling for the size (i.e. SLOC) of classes, some of the CK metrics were significantly associated with faults.

Gyimóthy et al. chose a large open source software system called Mozilla and collected 3,192 C++ classes [1]. They examined the effect of SLOC and all the CK metrics on fault-proneness of classes by using statistical methods (logistic regression and linear regression) and machine learning methods (decision trees and neural networks). The results indicated that NOC was not significant but SLOC and all the other CK metrics were significant.

Table 24 summarizes the related work, in which + means positively significant (++ means more positively significant), - means negatively significant (-- means more negatively significant), and 0 denotes that this metric is not significant. A blank entry means that our hypothesis was not examined or the metric was calculated in a different way. Since the definition of LCOM in our study is from Rosenberg and Hyatt [18], not from Chidamber and Kemerer [5], most entries corresponding to LCOM are blank. Also, note that CBO and RFC in Table 24 denote $\mathrm{CBO}_{\mathrm{out}}$ and $\mathrm{RFC}_{\mathrm{out}}$, respectively, for Yu et al.'s study [11].

From Table 24, we can see that: 1) most studies employed logistic regression analyses to examine the relationships between OO design metrics and fault-proneness of classes, 2) most of the studied systems were implemented in C++, i.e., there is little work across OO programming languages, 3) no previous work distinguished between the severity of faults, and 4) WMC, RFC, and CBO were almost consistently found to be statistically significant to fault-proneness of classes, but the other metrics were not.

6 Conclusion and Future Work

In real-life systems, faults can differ significantly in their impact on the operation of a software system. It would be valuable to use OO design metrics to help to identify the fault-proneness of classes when the severity of faults is taken into account. Previous studies have raised the need to validate OO design metrics across different fault severities [6]. However, little is currently known on this subject. Our study attempts to fill this gap by empirically validating OO design metrics for different fault severities.

Based on a public domain data set KC1 provided by NASA, we employed statistical (logistic regression) and

machine learning (naive Bayes, random forest, and NNge) methods to investigate the fault-proneness prediction usefulness of OO design metrics with regard to ungraded, high, and low severity faults. We analyzed six OO design metrics, five from Chidamber and Kemerer's metrics suite [5] and one from Rosenberg and Hyatt [18]. Our main results are summarized as follows:

- The CBO, WMC, RFC, and LCOM metrics are statistically significant across fault severity, while DIT is not significant for any fault severity. NOC is statistically significant with regard to ungraded/low severity faults, but in an inverse direction, i.e., a class with a large NOC means a low fault-proneness. The significance of NOC with regard to high severity faults cannot be tested because of the quasi-complete separation problem. However, the results of machine learning methods indicate that the fault-proneness prediction usefulness of NOC with regard to high severity faults is poor and, in this, is similar to that of DIT.
- The fault-proneness prediction capabilities of these metrics differ greatly depending on the fault severity used. When applied to the classification of classes as fault-prone and not fault-prone in terms of ungraded/low severity faults, the logistic regression and machine learning models based on these metrics achieve a performance comparable to previous studies. When applied to fault-proneness ranking of classes in terms of low severity faults, the logistic regression model based on these metrics outperforms the simple model based on class size. In both cases, however, the prediction usefulness of these metrics with regard to high severity faults is limited.

One limitation of our study is that the fault severity ratings in KC1 data set are subjective and may be inaccurate, which possibly limits the generalizability of our results. Also, the data used in our study is from a single project. As a result, the findings in this paper should be viewed as exploratory and indicative rather than conclusive. In future work, we will replicate this study across projects. Finally, the usefulness of these metrics for fault-proneness prediction with regard to the different severities of faults may depend on the programming language used. There is a need to validate our findings in other OO programming languages such as Java.

ACKNOWLEDGMENTS

The authors are very grateful to Pinata Winoto for assisting in the experiments and Martin Kyle for reviewing our paper. The authors also wish to thank the anonymous reviewers of this paper for their helpful comments and suggestions. This research is partly supported by Hong Kong Polytechnic University research grant G-U169 and the Hong Kong CERG grant PolyU5219/06E.

REFERENCES

[1] T. Gyimóthy, R. Ference, and L. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Trans. Software Eng.*, vol. 31, no. 10, pp. 897-910, Oct. 2005.

[2] R. Subramanyan and M.S. Krisnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," *IEEE Trans. Software Eng.*, vol. 29, no. 4, pp 297-310, Apr. 2003.

[3] M. Alshayeb and L. Wei, "An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes," *IEEE Trans. Software Eng.*, vol. 29, no. 11, pp. 1043-1049, Nov. 2003.

[4] K. El Emam, S. Benlarbi, N. Goel, and S.N. Rai, "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics," *IEEE Trans. Software Eng.*, vol. 27, no. 7, pp. 630-650, July 2001.

[5] S.R. Chidamber, D.P. Darcy, and C.F. Kemerer, "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis," *IEEE Trans. Software Eng.*, vol. 24, no. 8, pp. 629-639, Aug. 1998.

[6] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Trans. Software Eng.*, vol. 22, no. 10, pp. 751-761, Oct. 1996.

[7] M.M.T. Thwin and T.S. Quah, "Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics," *J. Systems and Software,* vol. 76, no. 2, pp. 147-156, 2005.

[8] G. Succi, W. Pedrycz, M. Stefanovic, and J. Miller, "Practical Assessment of the Models for Identification of Defect-Prone Classes in Object-Oriented Commercial Systems Using Design Metrics," *J. Systems and Software,* vol. 65, no. 1, pp. 1-12, 2003.

[9] L.C. Briand, J. Wüst, J.W. Daly, and D.V. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems," *J. Systems and Software,* vol. 51, no. 3, pp. 245-273, 2000.

[10] H. Lounis and L. Ait-Mehedine, "Machine-Learning Techniques for Software Product Quality Assessment," *Proc. Fourth Int'l Conf. Quality Software,* pp. 102-109, 2004.

[11] P. Yu, T. Systa, and H. Muller, "Predicting Fault-Proneness Using OO Metrics: An Industrial Case Study," *Proc. Sixth European Conf. Software Maintenance and Reeng.,* pp. 99-107, 2002.

[12] L.C. Briand, J. Daly, V. Porter, and J. Wüst, "A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems," *Proc. Fifth Int'l Software Metrics Symp.,* pp. 246-257, 1998.

[13] L.C. Briand and J. Wüst, "Empirical Studies of Quality Models in Object-Oriented Systems," *Advances in Computers,* D.M. Zelkowitz, ed., pp. 97-166, vol. 56, Academic, 2002.

[14] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design," *IEEE Trans. Software Eng.*, vol. 20, no. 6, pp. 476-493, June 1994.

[15] N.E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Trans. Software Eng.*, vol. 25, no. 5, pp. 675-689, May 1999.

[16] T.J. Ostrand, E.J. Weyuker, and R.M. Bell, "Predicting the Location and Number of Faults in Large Software Systems," *IEEE Trans. Software Eng.*, vol. 31, no. 4, pp. 340-355, Apr. 2005.

[17] L.C. Briand, J. Wüst, and H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," *Empirical Software Eng.*, vol. 6, no. 1, pp. 11-58, 2001.

[18] L. Rosenberg and L. Hyatt, "Software Quality Metrics for Object-Oriented System Environments," NASA Technical Report SATC-TR-95-1001, 1995.

[19] W.G. Hopkins, *A New View of Statistics.* Sport Science, 2003.

[20] D. Belsley, E. Kuh, and R. Welsch, *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity.* John Wiley and Sons, 1980.

[21] L.C. Briand and J. Wust, "Modeling Development Effort in Object-Oriented Systems Using Design Properties," *IEEE Trans. Software Eng.*, vol. 27, no. 11, pp. 963-986, Nov. 2001.

[22] A. Albert and A. Anderson, "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models," *Biometrika,* vol. 71, pp. 1-10, 1984.

[23] L.C. Briand, W.L. Melo, and J. Wust, "Assessing the Application of Fault-Proneness Models Across Object-Oriented Software Projects," *IEEE Trans. Software Eng.*, vol. 28, no. 7, pp. 706-720, July 2002.

[24] G. Denaro, M. Pezzè, and S. Morasca, "Towards Industrially Relevant Fault-Proneness Models," *Int'l J. Software Eng. and Knowledge Eng.*, vol. 13, no. 4, pp. 395-417, 2003.

[25] T.M. Khoshgoftaar, Y. Liu, and N. Seliya, "A Multiobjective Module-Order Model for Software Quality Enhancement," *IEEE Trans. Evolutionary Computation,* vol. 8, no. 6, pp. 593-608, 2004.

[26] L.C. Briand, J. Wust, S.V. Ikonomovski, and H. Lounis, "Investigating Quality Factors in Object-Oriented Designs: An Industrial Case Study," *Proc. 21st Int'l Conf. Software Eng.*, pp. 345-354, 1999.
[27] M.H. Tang, M.H. Kao, and M.H. Chen, "An Empirical Study on Object-Oriented Metrics," *Proc. Sixth Int'l Software Metrics Symp.*, pp. 242-249, 1999.
[28] N. Ohlsson and H. Alberg, "Predicting Fault-Prone Software Modules in Telephone Switches," *IEEE Trans. Software Eng.*, vol. 22, no. 12, pp. 886-894, Dec. 1996.
[29] R.S. Michalski and K.A. Kaufman, "Learning Patterns in Noisy Data: The AQ Approach," *Machine Learning and Applications*, G. Paliouras, V. Karkaletsis, and C. Spyropoulos, eds., pp. 22-38, Springer-Verlag, 2001.
[30] T.J. Ostrand and E.J. Weyuker, "The Distribution of Faults in a Large Industrial Software System," *Proc. Int'l Symp. Software Testing and Analysis*, pp. 55-64, 2002.
[31] A.A. Porter, "Using Measurement-Driven Modeling to Provide Empirical Feedback to Software Developers," *J. Systems and Software*, vol. 20, no. 3, pp. 237-243, 1993.

Yuming Zhou received the PhD degree in computer science from Southeast University in 2003. From January 2003 to December 2004, he was a postdoctoral researcher fellow at Tsinghua University. Since February 2005, he has been a researcher in the Department of Computing at Hong Kong Polytechnic University. His main research interests are software metrics, software evolution, and program understanding and analysis.

Hareton Leung received the PhD degree in computer science from the University of Alberta. He is an associate professor and director of the Laboratory for Software Development and Management in the Department of Computing at Hong Kong Polytechnic University. He currently serves on the editorial boards of the *Software Quality Journal* and the *Journal of the Association for Software Testing*. His research interests include software testing, software maintenance, quality and process improvement, and software metrics. He is a member of the IEEE Computer Society.