A Validation of Object-Oriented Metrics

El-Emam, Khaled; Benlarbi, S.; Goel, N.; Rai, S.

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

A Validation of Object-Oriented Metrics *

El-Emam, K., Benlarbi, S., Goel, N., Rai, S.
October 1999

* published as NRC/ERB-1063. October 1999. 20 pages. NRC 43607.

Canada

National Research Council Canada
Conseil national de recherches Canada
Institute for Information Technology
Institut de Technologie de l'information
ERB-1063

NRC·CNRC

A Validation of Object-oriented Metrics

Khaled El Emam, Saida Beniarbi,
Nishith Goel, and Shesh Rai
October 1999

Canada
NRC 43607

A Validation of Object-Oriented Metrics

Khaled El Emam
National Research Council, Canada
Institute for Information Technology
Building M-50, Montreal Road
Ottawa, Ontario
Canada K1A OR6
Khaled.El-Emam@iit.nrc.ca

Saida Benlarbi
Nishith Goel
Cistel Technology
210 Colonnade Road
Suite 204
Nepean, Ontario
Canada K2E 7L5
{benlarbi,ngoel}@cistel.com

Shesh Rai
Statistics Canada
24-J R.H. Coats Bldg.
Ottawa, Ontario
Canada K1A OT6
raishes@statcan.ca

Abstract

Many object-oriented metrics have been proposed, and at least fourteen empirical validations of these metrics have been performed. However, recently it was noted that without controlling for the effect of class size in a validation study, the impact of a metric may be exaggerated. It thus becomes necessary to re-validate contemporary object-oriented metrics after controlling for size. In this paper we perform a validation study on a telecommunications C++ system. We investigate 24 metrics proposed by Chidamber and Kemerer and Briand et al.. Our dependent variable was the incidence of faults due to field failures (fault-proneness). Our results indicate that out of the 24 metrics (covering coupling, cohesion, inheritance, and complexity), only four are actually related to faults after controlling for class size, and that only two of these are useful for the construction of prediction models. The two selected metrics measure coupling. The best prediction model exhibits high accuracy.

1 Introduction

A large number of object-oriented class metrics have been proposed over the last decade, for example [6][2][10][19][44][43][17][15][35][59]. The basic premise behind the development of object-oriented metrics is that they can serve as early predictors of classes that contain faults (found during testing or that result in field failures) or that are costly to maintain. A growing body of work has attempted to empirically validate this premise, for example [2][17][1][10][12][14][43][21][34][49][59][6][16][7].

A typical empirical validation of object-oriented metrics proceeds by investigating the relationship between each metric and the outcome of interest. In our case we will consider only fault-proneness as an outcome.[1] If this relationship is found to be statistically significant, then the conclusion is drawn that the metric is empirically validated. For example, recent studies used the bivariate correlation between object-oriented metrics and the number of faults to investigate the validity of the metrics [34][7]. Also, univariate logistic regression models are used as the basis for demonstrating the relationship between object-oriented metrics and fault-proneness in [2][14][12][59].

However, this validation approach completely ignores the confounding effect of class size. This can be illustrated with reference to Figure 1. Path (a) is the main hypothesized relationship between the object-oriented metric and fault-proneness. Here we see that class size (for example, measured in terms of SLOC) is associated with the object-oriented metric (path (c)). Evidence supporting this for many contemporary object-oriented metrics is found in [14][12][17]. Also, class size is known to be associated with the incidence of faults in object-oriented systems, path (b), and is supported in [17][33][14]. The pattern of relationships shown in Figure 1 exemplifies a classical confounding effect [55][8]. This means that the relationship between the object-oriented metric and fault-proneness will be inflated due the effect of size. This also means that without controlling for the effect of class size, previous validation studies have been optimistic about the validity of the object-oriented metrics that they investigated.

A recent study highlighted this potential confounding effect [25] and demonstrated it on the Chidamber and Kemerer metrics [19] and a subset of the Lorenz and Kidd [44] metrics. Specifically, this

[1] We define fault-proneness as the probability of a class having a fault.

demonstration showed that without controlling for the confounding effect of class size one obtains results that are consistent with previous empirical validation studies. After controlling for class size, all the metrics that were studied were no longer associated with fault-proneness. This made clear that it is necessary to update our knowledge about the validity of object-oriented metrics through new validation studies that account for class size.

In this paper we perform a validation study on a C++ telecommunications systems using the Chidamber and Kemerer metrics [19] and the Briand et al. metrics [10]. Although these metrics have been empirically evaluated in the past (e.g., [2][14][10][12]), these studies did not control the effect of class size, and it therefore becomes necessary to revisit them and perform further validations.

Briefly, our results indicate that out of the 24 object-oriented metrics that we study, only four are associated with fault-proneness after accounting for the influence of size, and only two are useful for the construction of prediction models to identify classes that are fault-prone. Both are coupling metrics: CBO and ACMIC.

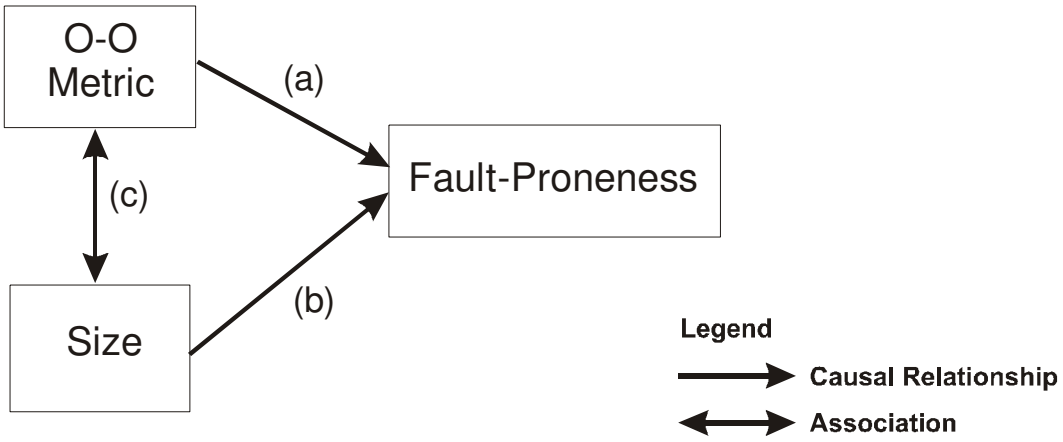

Figure 1: Path diagram illustrating the confounding effect of class size on the validity of object-oriented metrics.

The paper is organized as follows. In the next section we provide an overview of the object-oriented metrics that we evaluate, and our hypotheses. In Section 3 we present our research method in detail, and in Section 4 the detailed results, their implications, and limitations. We conclude the paper in Section 5 with a summary and directions for future research.

2 Background

In our study we consider two sets of metrics, those of Chidamber and Kemerer [19] and those of Briand et al. [10]. We summarize these metrics in this section.

2.1 The Chidamber and Kemerer Metrics

2.1.1 WMC

This is the Weighted Methods per Class metric [19], and can be classified as a traditional complexity metric. It is a count of the methods in a class. The developers of this metric leave the weighting scheme as an implementation decision [19]. We weight it using cyclomatic complexity as did [43]. However, other authors did not adopt a weighting scheme [2][59]. Methods from ancestor classes are not counted and

neither are “friends” in C++. This is similar to the approach taken in, for example, [2][20]. To be precise, WMC was counted after preprocessing to avoid undercounts due to macros [22].[2]

2.1.2 DIT

The Depth of Inheritance Tree [19] metric is defined as the length of the longest path from the class to the root in the inheritance hierarchy. It is stated that as one goes further down the class hierarchy the more complex a class becomes, and hence more fault-prone.

2.1.3 NOC

This is the Number of Children inheritance metric [19]. This metric counts the number of classes which inherit from a particular class (i.e., the number of classes in the inheritance tree down from a class).

2.1.4 CBO

This is the Coupling Between Object Classes coupling metric [19]. A class is coupled with another if methods of one class use methods or attributes of the other, or vice versa. In this definition, uses can mean as a member type, parameter type, method local variable type or cast. CBO is the number of other classes to which a class is coupled. It includes inheritance-based coupling (i.e., coupling between classes related via inheritance).

2.1.5 RFC

This is the Response for a Class coupling metric [19]. The response set of a class consists of the set M of methods of the class, and the set of methods invoked directly by methods in M (i.e., the set of methods that can potentially be executed in response to a message received by that class). RFC is the number of methods in the response set of the class.

2.1.6 LCOM

This is a cohesion metric that was defined in [19]. This measures the number of pairs of methods in the class using no attributes in common minus the number of pairs of methods that do. If the difference is negative it is set to zero.

2.2 The Briand et al. Coupling Metrics

These coupling metrics are counts of interactions between classes. The metrics distinguish between the relationship amongst classes (i.e., friendship, inheritance, or another type of relationship), different types of interactions, and the locus of impact of the interaction [10].

The acronyms for the metrics indicate what types of interactions are counted:

- The first or first two letters indicate the relationship (A: coupling to ancestor classes; D: Descendents; F: Friend classes; IF: Inverse Friends; and O: other, i.e., none of the above)
- The next two letters indicate the type of interaction between classes c and d (CA: there is a class-attribute interaction between classes c and d if c has an attribute of type d; CM: there is a class-method interaction between classes c and d if class c has a method with a parameter of type class d; MM: there is a method-method interaction between classes c and d if c invokes a method of d, or if a method of class d is passed as parameter to a method of class c).
- The last two letters indicate the locus of impact (IC: Import Coupling; and EC: Export Coupling). A class c exhibits import coupling if it is the using class (i.e., client in a client-server relationship), while it exhibits export coupling if is the used class (i.e., the server in a client-server relationship).

Based on the above, the authors define a total of 18 different coupling metrics.

[2] Note that macros embodied in #ifdef’s are used to customize the implementation to a particular platform. Therefore, the method is defined at design time but its implementation is conditional on environment variables. Not counting it, as suggested in [20], would undercount methods known at design time.

2.3 Hypotheses

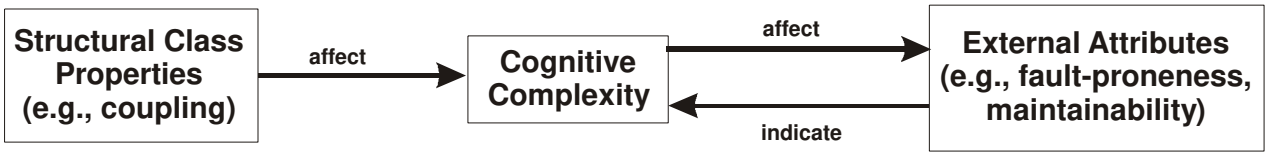

Figure 2: Theoretical basis for the development of object oriented product metrics.

An articulation of a theoretical basis for developing quantitative models relating object-oriented metrics and external quality metrics has been provided in [12], and is summarized in Figure 2. There, it is hypothesized that the structural properties of a software component (such as its coupling) have an impact on its cognitive complexity. Cognitive complexity is defined as the mental burden of the individuals who have to deal with the component, for example, the developers, testers, inspectors, and maintainers. High cognitive complexity leads to a component exhibiting undesirable external qualities, such as increased fault-proneness and reduced maintainability.

Therefore, our general hypothesis is that the metrics that we validate, and that were described above, are positively associated with the fault-proneness of classes. This means that higher values on these metrics represent structural properties that increase the probability that a class will have a fault that causes a field failure.

2.4 Empirical Validations of Metrics

Since our focus in this paper is with fault-proneness (as opposed to maintainability or cost of rework for example), we will only consider studies that evaluate the relationship between the above object-oriented metrics and fault-proneness.

To our knowledge, the only study that validated any of the above metrics after controlling for the effect of class size is [25]. This study was performed with telecommunications software written in C++, and only the Chidamber and Kemerer metrics were evaluated (i.e., not the Briand et al. metrics). It was found that after controlling for size, none of the metrics were associated with fault-proneness. It is therefore of importance to perform further validations to determine if these results hold, and to expand the scope of enquiry to cover other metrics. This is the purpose of our study.

3 Research Method

3.1 Data Source and Measurement

The data set that we used comes from the development of a telecommunications system developed in C++, and has been in operation for approximately seven years. This system has been deployed around the world in multiple sites. In total six different developers had worked on its development and evolution. It consists of 85 different classes that we analyzed.

All of the object-oriented metrics mentioned above were collected from the source code using a static analysis tool. The class size measure that we used is source lines of code: SLOC. Since the system has been evolving in functionality over the years, we selected a specific version for analysis where reliable fault data could be obtained.

Fault data was collected from the configuration management system. This documented the reason for each change made to the source code, and hence it was easy to identify which changes were due to faults. We focused on faults that were due to failures reported from the field. In total, 31 classes had one or more fault in them that was attributed to a field failure. It has been argued that considering faults causing field failures is a more important question to address than faults found during testing [7]. In fact, it

has been argued that it is the *ultimate* aim of quality modeling to identify post-release fault-proness [27]. In at least one study it was found that pre-release fault-proneness is not a good surrogate measure for post-release fault-proness, the reason posited being that pre-release fault-proness is a function of testing effort [29].

3.2 Data Analysis Methods

Our data anslysis approach consists of two stages. In the first stage we evaluate the relationship between each of the object-oriented metrics and fault-proneness after controlling for the size effect. In the second stage we construct prediction models with the validated metrics from the first stage. For both of these we use logistic regression as the modeling technique. Below we explain the analysis procedures used for both.

3.2.1 Evaluating the Relationship with Fault-Proneness

3.2.1.1 Logistic Regression Model

Binary logistic regression (henceforth LR) is used to construct models when the dependent variable is binary, as in our case. The general form of an LR model is:

$$\pi = \frac{1}{1 + e^{-\left(\beta_0 + \sum_{i=1}^{k} \beta_i x_i\right)}} \qquad \textbf{Eqn. 1}$$

where π is the probability of a class having a fault, and the x_i's are the independent variables. The β parameters are estimated through the maximization of a log-likelihood [36].

As noted in [25], it is necessary to control for the confounding effect of class size when validating object-oriented metrics. A measured confounding variable can be controlled through a regression adjustment [8][55]. A regression adjustment entails including the confounder as another independent variable in a regression model. Our logistic regression model is therefore:

$$\pi = \frac{1}{1 + e^{-(\beta_0 + \beta_i x_1 + \beta_2 x_2)}} \qquad \textbf{Eqn. 2}$$

where x_1 is the object-oriented metric being validated, and x_2 is size, measured as SLOC. We construct such a model for each object-oriented metric being validated.

It should be noted that the object-oriented metric and the size confounding variable are not treated symmetrically in this model. Specifically, the size confounder (i.e., variable x_2) should always be included in the model, irrespective of its statistical significance [8]. If inclusion of the size confounder does not affect the parameter estimate for the object-oriented metric (i.e., variable x_1), then we still get a valid estimate of the impact of the metric on fault-proneness. The statistical significance of the parameter estimate for the object-oriented metric, however, is interpreted directly since this is how we test our hypotheses.

In constructing our models, we follow previous literature in that we do not present results for interactions nor higher order terms (for example, see [25][2][6][10][11][12][14][59]). This is to some extent justifiable given that as yet there is no clear theoretical basis to assume any of the above.

The magnitude of an association can be expressed in terms of the change in odds ratio as the x_1 variable (i.e., object-oriented metric) changes by one standard deviation, and is denoted by $\Delta\Psi$. This is given by:

$$\Delta\Psi = \frac{\Psi(x_1 + \sigma)}{\Psi(x_1)} = e^{\beta_1 \sigma} \qquad \textbf{Eqn. 3}$$

where σ is the standard deviation of the x_1 variable, and Ψ is the odds ratio.

3.2.1.2 Diagnosing Collinearity

Since we control for the size confounder through regression adjustment, careful attention should be paid to the detection and mitigation of potential collinearity. Strong collinearity can cause inflated standard errors for the estimated regression parameters.

Previous studies have shown that outliers can induce collinearities in regression models [45][54]. But also, it is known that collinearities may mask influential observations [3]. This has lead some authors to recommend addressing potential collinearity problems as a first step in the analysis [3], and this is the sequence that we follow.

Belsley et al. [3] propose the *condition number* as a collinearity diagnostic in the case of ordinary least squares regression[3] First, let $\hat{\boldsymbol{\beta}}$ be a vector of the LR parameter estimates, and $\mathbf{X}$ is a $n \times (k+1)$ matrix of the x_{ij} raw data, with $i = 1 \ldots n$ and $j = 1 \ldots k$, where n is the number of observations and k is the number of independent variables. Here, the $\mathbf{X}$ matrix has a column of ones to account for the fact that the intercept is included in the models.

The condition number can be obtained from the eigenvalues of the $\mathbf{X}^T\mathbf{X}$ matrix as follows:

$$\eta = \sqrt{\frac{\mu_{\max}}{\mu_{\min}}} \qquad \textbf{Eqn. 4}$$

where $\mu_{\max} \geq \cdots \geq \mu_{\min}$ are the eigenvalues. Based on a number of experiments, Belsley et al. suggest that a condition number greater than 30 indicates mild to severe collinearity.

Belsley et al. emphasize that in the case where an intercept is included in the model, the independent variables should not be centered since this can mask the role of the constant in any near dependencies. Furthermore, the $\mathbf{X}$ matrix must be column equilibrated, that is, each column should be scaled for equal Euclidean length. Without this, the collinearity diagnostic produces arbitrary results. Column equilibration is achieved by scaling each column in $\mathbf{X}$, $\mathbf{X}_j$, by its norm [4]: $\mathbf{X}_j \Big/ \left\| \mathbf{X}_j \right\|$.

This diagnostic has been extended specifically to the case of LR models [23][60] by capitalizing on the analogy between the independent variable cross-product matrix in least-squares regression to the information matrix in maximum likelihood estimation, and therefore it would certainly be parsimonious to use the latter.

The information matrix in the case of LR models is [36]:

$$\hat{\mathbf{I}}\left(\hat{\boldsymbol{\beta}}\right) = \mathbf{X}^T \hat{\mathbf{V}} \mathbf{X} \qquad \textbf{Eqn. 5}$$

where $\hat{\mathbf{V}}$ is the diagonal matrix consisting of $\hat{\pi}_{ii}\left(1 - \hat{\pi}_{ii}\right)$ where $\hat{\pi}_{ii}$ is the probability estimate from the LR model for observation i. Note that the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ is given by $\hat{\mathbf{I}}^{-1}\left(\hat{\boldsymbol{\beta}}\right)$. One can then compute the eigenvalues from the information matrix after column equilibrating and compute the condition number as in Eqn. 4. The general approach for non-least-squares models is described by Belsley [5]. In this case, the same interpretive guidelines as for the traditoinal condition number are used [60].

We therefore use this as the condition number in diagnosing collinearity in our models that include the size confounder, and will denote it as η_{LR}. In principle, if severe collinearity is detected we would use logistic ridge regression to estimate the model parameters [53][54]. However, since we do not detect severe collinearity in our study, we will not describe the ridge regression procedure here.

---

[3] Actually, they propose a number of diagnostic tools. However, for the purposes of our analysis with only two independent variables we will consider the condition number only.

3.2.1.3 Hypothesis Testing

The next task in evaluating the LR model is to determine whether any of the regression parameters are different from zero, i.e., test $H_0 : \beta_1 = \beta_2 = \cdots = \beta_k = 0$. This can be achieved by using the likelihood ratio G statistic [36]. One first determines the log-likelihood for the model with the constant term only, and denote this l_0 for the 'null' model. Then the log-likelihood for the full model with the k parameters is determined, and denote this l_k. The G statistic is given by $2(l_k - l_0)$ which has a χ^2 distribution with k degrees of freedom.[4]

If the likelihood ratio test is found to be significant at an $\alpha = 0.05$ then we can proceed to test each of the individual coefficients. This is done using a Wald statistic, $\frac{\hat{\beta}_j}{s.e.(\hat{\beta}_j)}$, which follows a standard normal distribution. These tests were performed at a one-tailed alpha level of 0.05. We used one-tailed test since all of our alternative hypotheses are directional: there is a positive association between the metric and fault-proneness.

In previous studies another descriptive statistic has been used, namely an R^2 statistic that is analogous to the multiple coefficient of determination in least-squares regression [14][12]. This is defined as $R^2 = \frac{l_0 - l_k}{l_0}$ and may be interpreted as the proportion of uncertainty explained by the model. We use a corrected version of this suggested by Hosmer and Lemeshow [2]. It should be recalled that this descriptive statistic will in general have low values compared to what one is accustomed to in a LS regression. In our study we will use the corrected R^2 statistic as an indicator of the quality of the LR model.

3.2.1.4 Influence Analysis

Influence analysis is performed to identify influential observations (i.e., ones that have a large influence on the LR model). This can be achieved through deletion diagnostics. For a data set with n observations, estimated coefficients are recomputed n times, each time deleting exactly one of the observations from the model fitting process. Pergibon has defined the $\Delta\beta$ diagnostic [50] to identify influential groups in logistic regression. The $\Delta\beta$ diagnostic is a standardized distance between the parameter estimates when a group of observations with the same x_i values are included and when they are not included in the model.

We use the $\Delta\beta$ diagnostic in our study to identify influential groups of observations. For groups that are deemed influential we investigate this to determine if we can identify substantive reasons for the differences. In all cases in our study where a large $\Delta\beta$ was detected, its removal, while affecting the estimated coefficients, did not alter our conclusions.

3.2.2 Prediction Model Evaluation

By building and testing the model described in Section 3.2.1.1, we can identify the individual metrics that are associated with fault-proneness. The next step is to combine these metrics into a multivariate LR model.[5] Such a model follows the same formulation as in Section 3.2.1.1. The prediction model can be

---

[4] Note that we are not concerned with testing whether the intercept is zero or not since we do not draw substantive conclusions from the intercept in a validation study. If we were, we would use the log-likelihood for the null model which assigns a probability of 0.5 to each response.

[5] In principle, we could have just used an automated selection procedure to identify the useful metrics for prediction out of the set of 24. However, a Monte Carlo simulation of forward selection indicated that in the presence of collinearity amongst the independent variables, the proportion of 'noise' variables that are selected can reach as high as 74% [24]. Furthermore, some general guidelines

practically applied in identifying which classes are likely to contain a fault. In this subsection we explain how we evaluate the accuracy of the prediction model.

3.2.2.1 Accuracy Estimation Approach

There are a number of different approaches that can be used for estimating the accuracy of a prediction model, for example, using a hold-out sample, bootstrapping, or leave-one-out cross-validation. It has been recommended that in studies where sample sizes are less than 100, as in our case, a leave-one-out approach provides reliable estimates of accuracy [61], and therefore we use this approach for accuracy estimation.

3.2.2.2 Some Notation

It is common that prediction models using object-oriented metrics are cast as a binary classification problem. We first present some notation before discussing accuracy measures.

Table 1 shows the notation in obtained frequencies when a binary classifier is used to predict the class of unseen observations in a confusion matrix. We consider a class as being high risk if it has a fault and low risk if it does not have a fault.

		Predicted Risk		
		Low	**High**	
Real Risk	**Low**	n_{11}	n_{12}	N_{1+}
	High	n_{21}	n_{22}	N_{2+}
		N_{+1}	N_{+2}	N

Table 1: Notation for a confusion matrix.

Such a confusion matrix also appears frequently in the medical sciences in the context of evaluating diagnostic tests (e.g., see [30]). Two important parameters have been defined on such a matrix that will be used for our exposition, namely sensitivity and specificity.[6]

The *sensitivity* of a binary classifier is defined as:

$$s = \frac{n_{22}}{n_{21} + n_{22}} \qquad \textbf{Eqn. 6}$$

This is the proportion of high risk classes that have been correctly classified as high risk classes.

The *specificity* of a binary classifier is defined as:

$$f = \frac{n_{11}}{n_{11} + n_{12}} \qquad \textbf{Eqn. 7}$$

This is the proportion of low risk classes that have been correctly classified as low risk classes.

Ideally, both the sensitivity and specificity should be high. A low sensitivity means that there are many low risk classes that are classified as high risk. Therefore, the organization would be wasting resources reinspecting or focusing additional testing effort on these classes. A low specificity means that there are many high risk classes that are classified as low risk. Therefore, the organization would be passing high

---

on the number of variables to consider in an automatic selection procedure given the number of 'faulty classes' are provided in [32]. We have a much larger number of variables than these guidelines. Therefore, clearly the variables selected through such a procedure should not be construed as the best object-oriented metrics nor even as good predictors of fault-proneness.

[6] The terms sensitivity and specificity were originally used by Yerushalmy [62] in the context of comparing the reading of X-rays, and have been used since.

risk classes to subsequent phases or delivering them to the customer. In both cases the consequences may be expensive field failures or costly defect correction later in the life cycle.

3.2.2.3 Common Measures of Accuracy

Common measures of accuracy for binary classifiers that have been applied in software engineering are the proportion correct value, reported in for example [1][41][56] (also called correctness in [51][52]), completeness, for example [9][51] (which is equal to the senstivity), correctness which is the true positive rate [9][12][13][14] (also called consistency in [51][52]), the Kappa coefficient [12][14], Type I and Type II misclassifications [37][38][40][39] (Type I misclassification rate is $1-f$, and the Type II misclassification rate is $1-s$), and some authors report sensitivity and specificity directly [1].

Proportion correct is an intuitively appealing measure of prediction performance since it is easy to interpret. With reference to Table 1, it is defined as:

$$\frac{n_{11}+n_{22}}{N} \qquad \textbf{Eqn. 8}$$

However, in [26] it was shown that such an accuracy measure does not provide generalisable results. This can be illustrated through an example. Consider a classifier with a sensitivity of 0.9 and a specificity of 0.6, then the relationship between the prevalence of high risk classes in the data set and proportion correct is as depicted in Figure 3. Therefore, if we evaluate a classifier with a data set where say 50% of the classes are high risk then we would get a proportion correct value of 0.75. However, if we take the same classifier and apply it to another data set with say 10% of the classes are high risk, then the proportion correct accuracy is 0.63.[7] Therefore, proportion correct accuracy is sensitive to the proportion of high risk components (i.e., prevalence of high risk components), which limits its generalisability. Similar demonstrations were made for other measures of accuracy, such as correctness and Kappa [26].

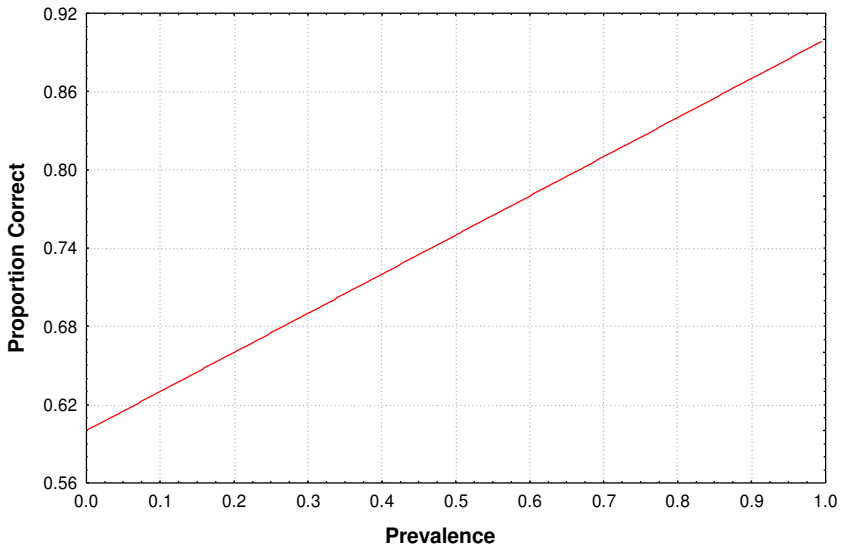

Figure 3: Relationship between prevalence (the proportion of high risk classes) and proportion correct for a classifier with sensitivity of 0.9 and specificity of 0.6.

[7] This is based on the assumption that sensitivity and specificity are stable. If this assumption is not invoked then there is no reason to believe that any results are generalisable, irrespective of the accuracy measure used.

Therefore, accuracy measures based on sensitivity and specificity are desirable (such as Type I and Type II misclassification rates). However, it is also desirable that a single accuracy measure be used, as opposed to two. The reason is best exemplified by the results in [39]. Here the authors were comparing nonparametric discriminant analysis with case-based reasoning classification. The Type I misclassification rate was lower for discriminant analysis, but the Type II misclassification rate was lower for the case-based reasoning classifier. From these results it is not clear which modeling technique is superior. Therefore, using multiple measures of accuracy can give confusing results that are difficult to interpret.

3.2.2.4 The J Coefficient

The J coefficient of Youdon [63] was suggested in [26] as an appropriate measure of accuracy for binary classifiers in software engineering. This is defined as:

$$J = s + f - 1 \qquad \textbf{Eqn. 9}$$

This coefficient has a number of desirable properties. First, it is prevalence independent. For example, if our classifier has specificity and sensitivity equal to $f = 0.9$ and $s = 0.7$, then its J value is 0.6 irrespective of prevalence. The J coefficient can vary from minus one to plus one, with plus one being perfect accuracy and –1 being the worst accuracy. A guessing classifier (i.e., one that guesses High/Low risk with a probability of 0.5) would have a J value of 0. Therefore, J values greater than zero indicate that the classifier is performing better than would be expected from a guessing classifier.

However, when the modeling technique that is being used is logistic regression, the J coefficient has a disadvantage. Recall that a logistic regression model makes predictions as a probability rather than a binary value (i.e., if we use a LR model to make a prediction, the predicted value is the probability of the occurrence of a fault). Previous studies have used a plethora of cutoff values to decide what is high risk or low risk, for example, 0.5 [13][12][2][48], 0.6 [12], 0.65 [12][14], 0.66 [11], 0.7 [14], and 0.75 [14]. In fact, and as noted by some authors [48], the choice of cutoff value is arbitrary, and one can obtain different results by selecting different cutoff values (we illustrate this using our results later in the paper). This creates interpretation difficulties for the J coefficient as well because it requires the continuous probability prediction to be dichotomized.

3.2.2.5 Receiver Operating Characteristic (ROC) Curves

A general solution to the arbitrary thresholds problem mentioned above is Receiver Operating Characteristic (ROC) curves [47]. One selects many cutoff points, from 0 to 1 in our case, and calculates the sensitivity and specificity for each cutoff value, and plots sensitivity against 1-specificity as shown in Figure 4. Such a curve describes the compromises that can be made between sensitivity and specificity as the cutoff value is changed. The main advantages of expressing the accuracy of our prediction model (or for that matter any diagnostic test) as an ROC curve are that it is independent of prevalence (proportion of high risk classes), and therefore the conclusions drawn are general, and it is independent of the cutoff value, and therefore no arbitrary decisions need be made as to where to cut off the predicted probability to decide that a class is high risk [64]. Furthermore, using an ROC curve, one can easily determine the optimal operating point, and hence obtain an optimal cutoff value for an LR model.

For our purposes, we can obtain a summary accuracy measure from an ROC curve by calculating the area under the curve using a trapezoidal rule [31]. The area under the ROC curve has an intuitive interpretation [31][58]: it is the estimated probability that a randomly selected class with a fault will be assigned a higher predicted probability by the logistic regression model than another randomly selected class without a fault Therefore, an area under the curve of say 0.8 means that a randomly selected faulty class has an estimated probability larger than a randomly selected not faulty class 80% of the time.

When a model cannot distinguish between faulty and not faulty classes, the area will be equal to 0.5 (the ROC curve will coincide with the diagonal). When there is a perfect separation of the values of the two groups, the area under the ROC curve equals 1 (the ROC curve will reach the upper left corner of the plot).

Therefore, to compute the accuracy of a prediction logistic regression model, we use the area under the ROC curve, which provides a general and non-arbitrary measure of how well the probability predictions can rank the classes in terms of their fault-proneness.

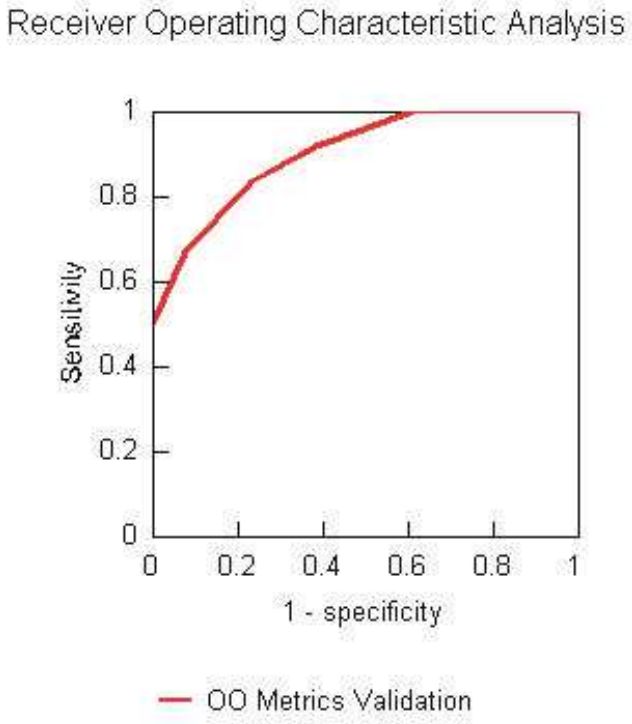

Figure 4: Hypothetical example of an ROC curve.

4 Results

4.1 Descriptive Statistics

The descriptive statistics for the object-oriented metrics and the SLOC metric are presented in Table 2. These include traditional summaries such as the mean and standard deviation. However, these summaries can be easily exaggerated by a single or a small number of observations. More robust analogs to these are the median and the inter-quartile range (IQR). The final column gives the number of observations that do not have zero values.

Two noticable points from this summary are that there does not exist any coupling to friend classes, and that the LCOM metric has a rather large value. The reason for the former is that friendship relationships were not used in the development of this system. We therefore excluded the friendship metrics from further analysis. The reason or the latter is a number of extreme observations (two classes with extremely high cohesion) that inflated the non-robust measures of central tendency and dispersion.

Variables ACAIC, and DCAEC have less than six observations that are non-zero. Therefore, they were excluded from further analysis. This is the approach followed in other validation studies [14][25].

	Mean	Median	Std. Dev.	IQR	NOBS ≠ 0
WMC	16.27	12	17.44	7	85
DIT	0.81	1	0.85	1	49
NOC	0.56	0	1.33	0	17
CBO	13.2	9	9.19	12	85
RFC	35.2	25	35.18	22	85
LCOM	55.56	0	329.287	0	10
OCAIC	1.7	1	2.040	1	66
IFCAIC	0	0	0	0	0
ACAIC	0.023	0	0.216	0	1
OCAEC	1.72	1	4.115	1	58
FCAEC	0	0	0	0	0
DCAEC	0.023	0	0.216	0	1
OCMIC	18.29	11	32.85	9	81
IFCMIC	0	0	0	0	0
ACMIC	0.89	0	1.851	1	27
OCMEC	17.43	6	35.62	8	82
FCMEC	0	0	0	0	0
DCMEC	0.894	0	5.483	0	6
OMMIC	32.87	10	77.39	17	71
IFMMIC	0	0	0	0	0
AMMIC	6.31	1	37.95	3	45
OMMEC	31.92	12	64.84	24	77
FMMEC	0	0	0	0	0
AMMEC	2.2	0	8.874	0	15
SLOC	436	280	492	244	0

Table 2: Descriptive statistics for all of the object-oriented metrics.

4.2 Validation Results

Table 3 contains all of the validation results. This includes the G value for the test of the overall hypothesis that the two estimated parameters are equal to zero, the R^2 value, the condition number η_{LR}, the parameter estimate for the object-oriented metric and its standard error, the one-sided p-value for the estimated parameter, and the change in odds ratio $\Delta\Psi$.

It can be seen that collinearity was not a problem for any of the models constructed, with the η_{LR} coefficient being always lower than 30.

The results for the six Chidamber and Kemerer metrics are consistent with the results we obtained in a previous study [25]. It was found that after controlling for size, WMC, DIT, RFC, and LCOM were not associated with fault-proneness. The NOC metric was not evaluated in the previous study because it had too few observations, although the results in Table 3 indicate that it is not related to fault-proneness. The CBO metric provides an interesting contrast. In the previous study the CBO metric was found not to be associated with fault-proneness after controlling for size [25]. Although, out of all the metrics evaluated, it was the one that was least affected by the control of class size. In our current study it is clear that CBO is associated with fault-proneness.

Metric	G (p-value)	R^2	η_{LR}	Coefficient (s.e.)	1-sided p-value	$\Delta\Psi$
WMC	13.16; (0.0014)	0.1179	7.119	0.0724 (0.0548)	0.0933	3.535
DIT	10.71 (0.0047)	0.0959	3.5617	-0.3398 (0.3015)	0.1299	0.7486
NOC	12.16 (0.0023)	0.109	3.303	0.2993 (0.1860)	0.0538	1.489
CBO	33.64 (<0.0001)	0.3016	5.0646	0.1848 (0.0475)	**<0.0001**	5.475
RFC	11.94 (0.0026)	0.107	5.3048	0.0213 (0.0157)	0.0868	2.118
LCOM	9.11 (0.0105)	0.0848	2.998	0.0258 (0.0215)	0.1151	1.6
OCAIC	11.03 (0.004)	0.098	3.386	0.1849 (0.1529)	0.1132	1.4583
OCMIC	9.16 (0.0102)	0.08369	3.8927	0.0012 (0.0171)	0.4727	1.0307
ACMIC	20.95 (<0.0001)	0.1878	3.01598	0.5838 (0.2099)	**0.0027**	2.947
OCMEC	23.83 (<0.0001)	0.213	3.15	0.0451 0.0200	**0.0120**	4.99
DCMEC	14.15 (0.0008)	0.1268	3.1376	0.2248 (0.1607)	0.0810	3.429
OMMIC	9.22 (0.01)	0.084	4.15849	-0.0017 (0.0068)	0.4022	0.9077
AMMIC	11.81 (0.0027)	0.1078	3.1679	0.1294 (0.0860)	0.0662	1.7272
OMMEC	20.81 (<0.0001)	0.186	3.258	0.0215 (0.0096)	**0.0129**	4.02372
AMMEC	14.5 (0.0007)	0.1299	3.188	0.1514 (0.1054)	0.0755	3.832

Table 3: Results of the validation, including the logistic regression parameters and diagnostics.

The Briand et al. metrics were mostly not associated with fault-proneness except for ACMIC, OCMEC, and OMMEC. Table 4 shows the Spearman correlations [57] amongst these three coupling metrics and

Chidamber and Kemerer CBO metric. We use this type of correlation coefficient since it is robust to outliers. It is seen that CBO is associated with the two export coupling metrics, and that the two export coupling metrics are strongly associated to each other, suggesting considerable redundancy and the opportunity for further elimination of metrics.

The change in odds ratio values in Table 3 indicate that CBO has the largest impact on fault-proneness out of the four validated metrics. It is therefore prudent to keep CBO. CBO is a general coupling metric, incorporating both import and export coupling.[8] However, the pattern of correlations that we see in Table 4 indicates that the variation in CBO for our system is dominated by export coupling.

	CBO	**ACMIC**	**OCMEC**	**OMMEC**
ACMIC	0.12 (0.2558)	1.00		
OCMEC	0.36 **(0.0007)**	0.20 (0.0642)	1.00	
OMMEC	0.51 **(<0.0001)**	-0.15 (0.1786)	0.65 **(<0.0001)**	1.00

Table 4: Spearman correlations amongst the metrics that were found to be associated with fault-proneness after controlling for size.

To further investigate the possibility of reducing the number of metrics down from 4, we constructed a model with only size (SLOC) and CBO, then added the ACMIC metric, and then added the two export coupling metrics. The results of an analysis of deviance [46] comparing these different models is shown in Table 5. This clearly indicates that the two export coupling metrics are not providing additional information after entering the CBO and ACMIC metrics. We therefore exclude these two variables from further analysis.[9]

#	**Model**	**Deviance**	**p-value**
1.	SLOC + CBO	24.27315	.000001
2.	SLOC + CBO + ACMIC	7.20513	.007272
3.	SLOC + CBO + ACMIC + OMMEC + OCMEC	1.22794	.541202

Table 5: Deviance table for different logistic regression models, each time adding new variables.

4.3 Prediction

At this juncture we have identified two metrics that have value additional to class size, and that carry complementary information about the impact of class structure on fault-proneness. By constructing different prediction models and evaluating them, we can further determine whether both of these metrics are necessary, or whether we can eliminate one of them (if it does not add to prediction accuracy).

We first evaluated the J coefficient for different cutoff values for the predicted probability from the LR model. The results are shown graphically in Figure 5 for two different models: numbers 1 and 2 in Table 5. Two points are clear from this figure. First, by intelligently selecting the cutoff values, one can easily

[8] However, note that CBO is not simply the sum of different coupling metrics, since different types of coupling can occur with the same class, but are counted only once in CBO.

[9] We also evaluated the usefulness of these two export coupling metrics for constructing prediction models (i.e., by evaluating prediction accuracy), and the results led to the same conclusion: that they do not add value after considering CBO and ACMIC. Furthermore, the prediction accuracy of models that use these two export coupling metrics are systematically inferior to any model that incorporates CBO.

increase the J accuracy values. Second, when we are comparing two models the conclusion is different depending on which cutoff value one chooses. It can therefore be seen that the choice of cutoff value can give different conclusions about the prediction accuracy. This clarifies why we chose not to rely on the J coefficient to draw conclusions in our study.

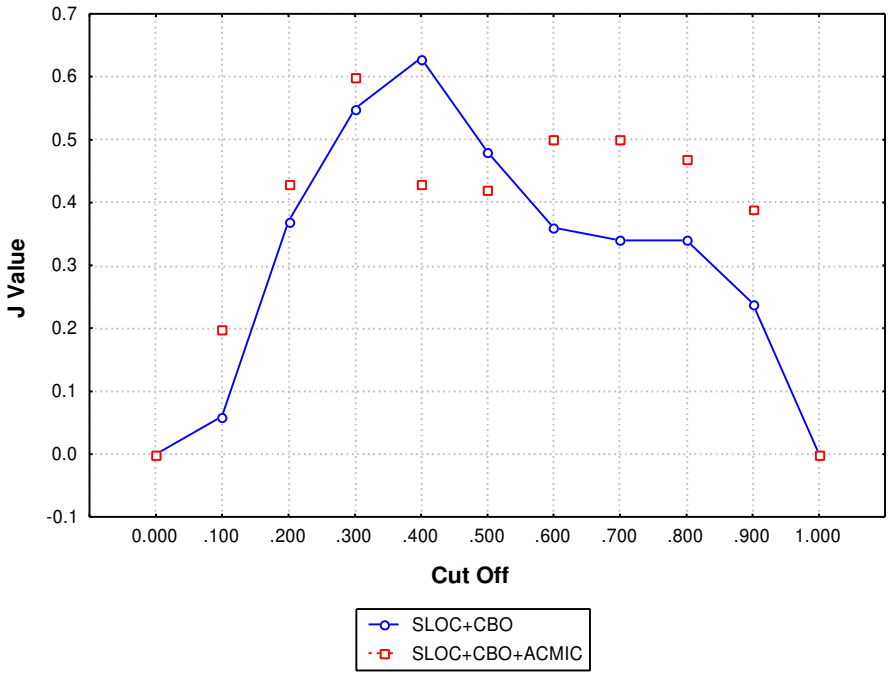

Figure 5: Variation on the J coefficient value as the cutoff value applied to the logistic regression prediction is varied. Note that the values were calculated using the leave-one-out procedure, as described in Section 3.2.2.1.

The results of evaluating the area under the ROC curve are shown in Table 6 for the model with only CBO and the model with ACMIC added. As can be seen, the addition of ACMIC improves the prediction accuracy of the model and therefore this can be considered to be a useful metric to include in the prediction model. We conclude then that model number 2 from Table 5 is the best prediction model. Also, the value of 0.813 indicates that a randomly selected faulty class has an estimated probability that is larger than a randomly selected not faulty class approximately 81% of the time, indicating a rather good prediction accuracy for this simple model.

#	Model	Area Under ROC Curve
1	SLOC + CBO	0.79
2	SLOC + CBO + ACMIC	0.813

Table 6: Area under the ROC curve for the two competing models.

The model 2 parameters from Table 6 are shown in Table 7. It can be seen that the two object oriented metrics have sizeable effects in terms of the change in odds ratio parameter. It will also be noted that the SLOC variable is not significant in this model, and in actuality its removal has minor influence on the prediction accuracy parameters. However, as noted earlier, it is preferable to include it anyway as its inclusion provides a more realistic estimate of the object-oriented metrics' individual influence on fault-proneness.

G	R^2		η_{LR}	
40.84 (p<0.0001)	0.366		5.369	
	Intercept	**SLOC**	**CBO**	**ACMIC**
Coefficient	-3.46	0.00017	0.178	0.643
1-sided p-value	<0.0001	0.4133	0.0003	0.0374
$\Delta\Psi$		1.09	5.16	3.29

Table 7: Logistic regression results for the best model.[10]

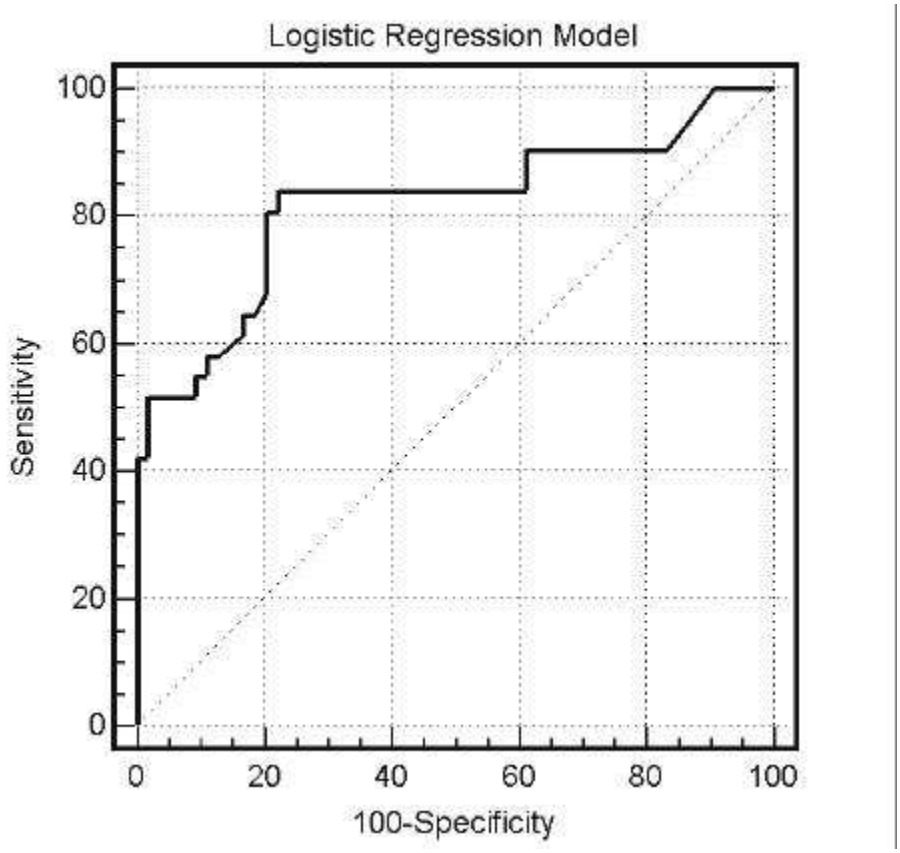

Figure 6: ROC curve for the best predictive model: FAULT ~ SLOC + CBO +ACMIC. The sensitivity and specificity values are multiplied by 100 in this chart.

The ROC curve for the best model from Table 6 is shown in Figure 6.[11] Given that the curve is above the diagonal, it is clear that the model provides value above just simple guessing. The optimal cutoff point for

[10] We also constructed the LR model after subtracting the ACMIC value from CBO such that CBO does not account for this type of import coupling. The model parameters changed only slightly, and therefore there is no advantage from disentangling the ACMIC effect from CBO.

[11] Note that in this figure the ROC curve is not smoothed.

this model is 0.28. The optimal point maximizes both sensitivity and specificity. The optimal sensitivity is 0.84, and the optimal specificity is 0.78. It should be noted that this optimal point is specific to this model and data set, but does indicate that the conventional cutoff points for $\hat{\pi}$ may not necessarily be the best ones.

In practice, it is also informative to calculate the proportion correct accuracy for a prediction model *when used in a particular context*. The following equation formulates the relationship between sensitivity, specificity, prevalence of faulty classes, and proportion correct accuracy:

$$A = (s \times h) + (f \times (1 - h)) \qquad \textbf{Eqn. 10}$$

where A is the proportion correct accuracy, and h is the proportion of faulty classes. For example, if our prediction model is to be used on an actual project where only 10% of the classes are expected to have faults in them, then the proportion correct accuracy would be approximately 0.79.

4.4 Discussion of Results

Our results indicate that, in addition to a simple size metric, the CBO and ACMIC metrics can be useful indicators of fault-prone classes. They are both associated with fault-proneness after controlling for size, and when combined in a multivariate model can provide accurate predictions. The added advantage of both of the above metrics is that SLOC and CBO are available in almost all contemporary tools that collect object-oriented metrics, and the ACMIC metric is quite easy to collect in practice. Also, interestingly, a large majority of the remaining metrics that we evaluated were not associated with fault-proneness after controlling for size. Hence, the net outcome of our study was to filter out the metrics that are not useful indicators of fault-proneness.

We have also added to the methodology initially presented in [25] by providing techniques for drawing general conclusions about the accuracy of prediction models, namely through the use of Receiver Operating Characteristic curves.

4.5 Limitations

This study has a number of limitations which should be made clear in the interpretation of our results. These limitations are not unique to our study, but are characteristics of most of the product metrics validation literature. However, it is of value to repeat them here.

This study did not account for the severity of faults. A failure that is caused by a fault may lead to a whole network crash or to an inability to interpret an address with specific characters in it. These types of failures are not the same, the former being more serious. Lack of accounting of fault severity was one of the criticisms of the quality modeling literature in [28]. In general, unless the organization has a reliable data collection program in place where severity is assigned, it is difficult to retrospectively obtain this data. Therefore, the prediction models developed here can be used to identify classes that are prone to have faults that cause any type of failure.

It is also important to note that our conclusions are pertinent only to the fault-proneness dependent variable, albeit this seems to be one of the more popular dependent variables in validation studies. We do not make claims about the validity (or otherwise) of the studied object-oriented metrics when the external attributes of interest are, for example, maintainability (say measured as effort to make a change) or reliability (say measured as mean time between failures).

It is unwise to draw broad conclusions from the results of a single study. Our results indicate that the general coupling metric, CBO, is associated with fault-proneness. However, we also concluded that its influence stems from an association with two export coupling metrics: method-method interactions and class-method interactions. We also found that ancestor based class-method import interactions are important contributors to fault-proneness. While these results provide guidance for future research on the impact of coupling on fault-proneness, they should not be interpreted as the last word on the subject. Further validations with different industrial systems are necessary so that we can accumulate knowledge and draw stronger conclusions.

5 Conclusions

In this paper we performed a validation of 24 different object-oriented metrics on a telecommunications C++ system. The objective of the validation was to determine which of these metrics were associated with fault-proneness, and hence can be used for predicting the classes that will be fault-prone. Our results indicate that out of the 24 metrics only 4 are associated with fault-proneness, and only two out of these are useful predictors: CBO and ACMIC. Furthermore, the prediction model that we constructed with the remaining two metrics has good accuracy.

While this is a single study, it does suggest that perhaps many of the contemporary object-oriented metrics may not be associated with fault-proneness, and that good prediction accuracy may be attained by careful selection of a small number of metrics. This conclusion is encouraging from a practical standpoint, and hence urges further studies to corroborate (or otherwise) our findings and conclusions.

6 Acknowledgements

We wish to thank Dave Card for his comments on an earlier version of this paper.

7 References

[1] M. Almeida, H. Lounis, and W. Melo: "An Investigation on the Use of Machine Learned Models for Estimating Correction Costs". In *Proceedings of the 20th International Conference on Software Engineering*, pp. 473-476, 1998.

[2] V. Basili, L. Briand and W. Melo: "A Validation of Object-Oriented Design Metrics as Quality Indicators". In *IEEE Transactions on Software Engineering*, 22(10):751-761, 1996.

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

[4] D. Belsley: "A Guide to Using the Collinearity Diagnostics". In *Computer Science in Economics and Management,* 4:33-50, 1991.

[5] D. Belsley: *Conditioning Diagnostics: Collinearity and Weak Data in Regression*. John Wiley and Sons, 1991.

[6] S. Benlarbi and W. Melo: "Polymorphism Measures for Early Risk Prediction". In *Proceedings of the 21st* International *Conference on Software Engineering*, pages 334-344, 1999.

[7] A. Binkley and S. Schach: "Validation of the Coupling Dependency Metric as a Predictor of Run-Time Fauilures and Maintenance Measures". In *Proceedings of the 20th International Conference on Software Engineering*, pages 452-455, 1998.

[8] N. Breslow and N. Day: *Statistical Methods in Cancer Research – Volume 1 – The Analysis of Case Control Studies*, IARC, 1980.

[9] L. Briand, V. Basili, and C. Hetmanski: "Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components". In *IEEE Transactions on Software Engineering*, 19(11):1028-1044, 1993.

[10] L. Briand, P. Devanbu, and W. Melo: "An Investigation into Coupling Measures for C++". In *Proceedings of the 19th International Conference on Software Engineer*ing, 1997.

[11] L. Briand, J. Daly, V. Porter, and J. Wuest: "Predicting Fault-Prone Classes with Design Measures in Object Oriented Systems". In *Proceedings of the International Symposium on Software Reliability Engineering*, pages 334-343, 1998.

[12] L. Briand, J. Wuest, S. Ikonomovski, and H. Lounis: "A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study". International Software Engineering Research Network technical report ISERN-98-29, 1998.

[13] L. Briand, J. Wuest, S. Ikonomovski, and H. Lounis: "Investigating Quality Factors in Object-Oriented Designs: An Industrial Case Study". In *Proceedings of the International Conference on Software Engineering*, 1999.

[14] L. Briand, J. Wuest, J. Daly, and V. Porter: "Exploring the Relationships Between Design Measures and Software Quality in Object Oriented Systems". To appear in *Journal of Systems and Software*.

[15] F. Brite e Abreu and R. Carapuca: "Object-Oriented Software Engineering: Measuring and Controlling the Development Process". In *Proceedings of the 4th International Conference on Software Quality*, 1994.
[16] F. Brito e Abreu and W. Melo: "Evaluating the Impact of Object-Oriented Design on Software Quality". In *Proceedings of the 3rd International Software Metrics Symposium*, pages 90-99, 1996.
[17] M. Cartwright and M. Shepperd: "An Empirical Investigation of an Object-Oriented Software System". To appear in *IEEE Transactions on Software Engineering*.
[18] S. Chidamber and C. Kemerer: "Towards a Metrics Suite for Object Oriented Design". In *Proceedings of the Conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA'91)*. Published in SIGPLAN Notices, 26(11):197-211, 1991.
[19] S. Chidamber and C. Kemerer: "A Metrics Suite for Object-Oriented Design". In *IEEE Transactions on Software Engineering*, 20(6):476-493, 1994.
[20] S. Chidamber and C. Kemerer: "Authors' Reply". In *IEEE Transactions on Software Engineering*, 21(3):265, 1995.
[21] S. Chidamber, D. Darcy, and C. Kemerer: "Managerial Use of Metrics for Object Oriented Software: An Exploratory Analysis". In *IEEE Transactions on Software Engineering*, 24(8):629-639, 1998.
[22] N. Churcher and M. Shepperd: "Comments on 'A Metrics Suite for Object Oriented Design'". In *IEEE Transactions on Software Engineering*, 21(3):263-265, 1995.
[23] C. Davies, J. Hyde, S. Bangdiwala, and J. Nelson: "An Example of Dependencies Among Variables in a Conditional Logistic Regression". In S. Moolgavkar and R. Prentice (eds.): *Modern Statistical Methods in Chronic Disease Edpidemiology*. John Wiley and Sons, 1986.
[24] S. Derksen and H. Keselman: "Backward, Forward and Stepwise Automated Subset Selection Algorithms: Frequency of Obtaining Authentic and Noise Variables". In *British Journal of Mathematical and Statistical Psychology*, 45:265-282, 1992.
[25] K. El Emam, S. Benlarbi, and N. Goel: "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics". Submitted for Publication, 1999.
[26] K. El Emam, S. Benlarbi, and N. Goel: "Comparing Case-Based Reasoning Classifiers for Predicting High Risk Software Components". Submitted for Publication, 1999.
[27] N. Fenton and M. Neil: "Software Metrics: Successes, Failures, and New Directions". In *Journal of Systems and Software*, 47:149-157, 1999.
[28] N. Fenton and M. Neil: "A Critique of Software Defect Prediction Models". To appear in *IEEE Transactions on Software Engineering*.
[29] N. Fenton and N. Ohlsson: "Quantitative Analysis of Faults and Failures in a Complex Software System". To appear in *IEEE Transactions on Software Engineering*.
[30] L. Gordis: *Epidemiology*. W. B. Saunders Company, 1996.
[31] J. Hanley and B. McNeil: "The Meaning and Use of the Area Under a Receiver Operating Characteristic Curve". In *Diagnostic Radiology*, 143(1):29-36, 1982.
[32] F. Harrell, K. Lee, and D. Mark: "Multivariate Prognostic Models: Issues in Developing Models, Evaluating Assumptions and Adequacy, and Measuring and Reducing Errors". In *Statistics in Medicine*, 15:361-387, 1996.
[33] R. Harrison, L. Samaraweera, M. Dobie, and P. Lewis: "An Evaluation of Code Metrics for Object-Oriented Programs". In *Information and Software Technology*, 38:443-450, 1996.
[34] R. Harrison, S. Counsell, and R. Nithi: "Coupling Metrics for Object Oriented Design". In *Proceedings of the 5th Internationsl Symposium on Software Metrics*, pages 150-157, 1998.
[35] B. Henderson-Sellers: *Software Metrics*. Prentice-Hall, 1996.
[36] D. Hosmer and S. Lemeshow: *Applied Logistic Regression*. John Wiley & Sons, 1989.
[37] T. Khoshgoftaar, E. Allen, K. Kalaichelvan, N. Goel, J. Hudepohl, and J. Mayrand: "Detection of Fault-Prone Program Modules in a Very Large Telecommunications System". In *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pp. 24-33, 1995.
[38] T. Khoshgoftaar, E. Allen, L. Bullard, R. Halstead, and G. Trio: "A Tree Based Classification Model for Analysis of a Military Software System". In *Proceedings of the IEEE High-Assurance Systems Engineering Workshop*, pages 244-251, 1996.
[39] T. Khoshgoftaar, K. Ganesan, E. Allen, F. Ross, R. Munikoti, N. Goel, and A. Nandi: "Predicting Fault-Prone Modules with Case-Based Reasoning". In *Proceedings of the Eighth International Symposium on Software Reliability Engineering*, pages 27-35, 1997.

[40] T. Khoshgoftaar, E. Allen, W. Jones, and J. Hudepohl: "Classification Tree Models of Software Quality Over Multiple Releases". To appear in *Proceedings of the International Symposium on Software Reliability Engineering*, 1999.
[41] F. Lanubile and G. Visaggio: "Evaluating Predictive Quality Models Derived from Software Measures: Lessons Learned". In *Journal of Systems and Software*, 38:225-234, 1997.
[42] Y. Lee, B. Liang, S. Wu, and F. Wang: "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow". In *Proceedings of the International Conference on Software Quality*, 1995.
[43] W. Li and S. Henry: "Object-Oriented Metrics that Predict Maintainability". In *Journal of Systems and Software*, 23:111-122, 1993.
[44] M. Lorenz and J. Kidd: *Object-Oriented Software Metrics*. Prentice-Hall, 1994.
[45] R. Mason and R. Gunst: "Outlier-Induced Collinearities". In *Technometrics*, 27:401-407, 1985.
[46] P. McCullagh and J. Nedler: *Generalized Linear Models*. Chapman & Hall, 1989.
[47] C. Metz: "Basic Principles of ROC Analysis". In *Seminars in Nuclear Medicine*, VIII(4):283-298, 1978.
[48] S. Morasca and G. Ruhe: "Knowledge Discovery from Software Engineering Measurement Data: A Comparative Study of Two Analysis Techniques". In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, 1997.
[49] P. Nesi and T. Querci: "Effort Estimation and Prediction of Object-Oriented Systems". In *Journal of Systems and Software*, 42:89-102, 1998.
[50] D. Pergibon: "Logistic Regression Diagnostics". In *The Annals of Statistics*, 9(4):705-724, 1981.
[51] A. Porter and R. Selby: "Evaluating Techniques for Generating Metric-Based Classification Trees". In *Journal of Systems and Software*, 12:209-218, 1990.
[52] A. Porter: "Using Measurement-Driven Modeling to Provide Empirical Feedback to Software Developers". In *Journal of Systems and Software*, 20:237-243, 1993.
[53] R. Schaefer, L. Roi, and R. Wolfe: "A Ridge Logistic Estimator". In *Communications in Statistics – Theory and Methods*, 13(1):99-113, 1984.
[54] R. Schaefer: "Alternative Estimators in Logistic Regression when the Data are Collinear". In *The Journal of Statistical Computation and Simulation*, 25:75-91, 1986.
[55] J. Schlesselman: *Case-Control Studies: Design, Conduct, Analysis*. Oxford University Press, 1982.
[56] N. Schneidewind: "Validating Metrics for Ensuring Space Shuttle Flight Software Quality". In *IEEE Computer*, pages 50-57, August 1994.
[57] D. Sheskin: *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, 1997.
[58] D. Spiegelhalter: "Probabilistic Prediction in Patient Management in Clinical Trials". In *Statistics in Medicine*, 5:421-433, 1986.
[59] M-H. Tang, M-H. Kao, and M-H. Chen: "An Empirical Study on Object Oriented Metrics". To appear in *Proceedings of the International Symposium on Software Metrics*, 1999.
[60] Y. Wax: "Collinearity Diagnosis for Relative Risk Regression Analysis: An Application to Assessment of Diet-Cancer Relationship in Epidemiological Studies". In *Statistics in Medicine*, 11:1273-1287, 1992.
[61] S. Weiss and C. Kulikowski: *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann Publishers, 1991.
[62] J. Yerushalmy: "Statistical Problems in Assessing Methods of Medical Diagnosis, with Special Reference to X-Ray Techniques". In *Public Health Report*, 62:1432-1449, 1947.
[63] W. Youden: "Index for Rating Diagnostic Tests". In *Cancer*, 3:32-35, 1950.
[64] M. Zweig and G. Campbell: "Receiver-Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine". In *Clinical Chemistry*, 39(4):561-577, 1993.