The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics

Khaled El Emam, Saïda Benlarbi, Nishith Goel, and Shesh N. Rai

Abstract—Much effort has been devoted to the development and empirical validation of object-oriented metrics. The empirical validations performed thus far would suggest that a core set of validated metrics is close to being identified. However, none of these studies allow for the potentially confounding effect of class size. In this paper, we demonstrate a strong size confounding effect and question the results of previous object-oriented metrics validation studies. We first investigated whether there is a confounding effect of class size in validation studies of object-oriented metrics and show that, based on previous work, there is reason to believe that such an effect exists. We then describe a detailed empirical methodology for identifying those effects. Finally, we perform a study on a large C++ telecommunications framework to examine if size is really a confounder. This study considered the Chidamber and Kemerer metrics and a subset of the Lorenz and Kidd metrics. The dependent variable was the incidence of a fault attributable to a field failure (fault-proneness of a class). Our findings indicate that, before controlling for size, the results are very similar to previous studies: The metrics that are expected to be validated are indeed associated with fault-proneness. After controlling for size, none of the metrics we studied were associated with fault-proneness anymore. This demonstrates a strong size confounding effect and casts doubt on the results of previous object-oriented metrics validation studies. It is recommended that previous validation studies be reexamined to determine whether their conclusions would still hold after controlling for size and that future validation studies should always control for size.

Index Terms—Object-oriented metrics, software quality, metrics validation, validation methodology, object-oriented quality, coupling metrics, inheritance metrics, cohesion metrics.

1 Introduction

The validation of software product metrics[1] has received much research attention by the software engineering community. There are two types of validation that are recognized [48]: internal and external. Internal validation is a theoretical exercise that ensures that the metric is a proper numerical characterization of the property it claims to measure. External validation involves empirically demonstrating that the product metric is associated with some important external metric (such as measures of maintainability or reliability). These are also commonly referred to as theoretical and empirical validation, respectively [73], and procedures for achieving both are described in [15]. Our focus in this paper is *empirical* validation.[2]

Product metrics are of little value by themselves unless there is empirical evidence that they are associated with important external attributes [65]. The demonstration of such a relationship can serve two important purposes: early prediction/identification of high risk software components and the construction of preventative design and programming guidelines.

Early prediction is commonly cast as a binary classification problem.[3] This is achieved through a *quality model* that classifies components into either a high or low risk category. The definition of a high risk component varies depending on the context of the study. For example, a high risk component is one that contains any faults found during testing [14], [75], one that contains any faults found during operation [72], or one that is costly to correct after an error has been found [3], [13], [1]. The identification of high risk components allows an organization to take mitigating actions, such as focus defect detection activities on high risk components, for example, by optimally allocating testing resources [56], or redesign components that are likely to cause field failures or be costly to maintain. This is motivated by evidence showing that most faults are found in only a few of a system's components [86], [51], [67], [91].

A number of organizations have integrated quality models and modeling techniques into their overall quality decision making process. For example, Lyu et al. [81] report on a prototype system to support developers with software quality models, and the EMERALD system is reportedly

[1] Some authors distinguish between the terms "metric" and "measure" [2]. We use the term "metric" here to be consistent with prevailing international standards. Specifically, ISO/IEC 9126:1991 [64] defines a "software quality metric" as a "quantitative scale and method which can be used to determine the value a feature takes for a specific software product."

[2] Theoretical validations of many of the metrics that we consider in this paper can be found in [20], [21], [30].

[3] It is not, however, *always* the case that binary classifiers are used. For example, there have been studies that predict the number of faults in individual components (e.g., [69]) and that produce point estimates of maintenance effort (e.g., [78], [66]).

- *K. El Emam is with the National Research Council, Canada Institute for Information Technology, Building M-50, Montreal Rd., Ottawa, Ontario, Canada K1A OR6. E-mail: khaled.el-emam@iit.nrc.ca.*
- *S. Benlarbi and N. Goel are with Cistel Technology, Colonnade Rd., Suite 204, Nepean, Ontario, Canada K2E 7L5. E-mail: {benlarbi, ngoel}@cistel.com.*
- *S.N. Rai is with the Department of Biostatistics and Epidemiology, St. Jude Children's Research Hospital, N. Lauderdale St., Memphis, TN, 38105-2794. E-mail: shesh.rai@stjude.org.*

Manuscript received 19 Aug. 1999; accepted 24 Mar. 2000.
Recommended for acceptance by S.L. Pfleeger.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 110473.

0098-5589/01/$10.00 © 2001 IEEE

routinely used for risk assessment at Nortel [62], [63]. Ebert and Liedtke describe the application of quality models to control the quality of switching software at Alcatel [46].

The construction of design and programming guidelines can proceed by first showing that there is a relationship between say a coupling metric and maintenance cost. Then, proscriptions on the maximum allowable value on that coupling metric are defined in order to avoid costly rework and maintenance in the future. Examples of cases where guidelines were empirically constructed are [1], [3].[4] Guidelines based on anecdotal experience have also been defined [80] and experience-based guidelines are used directly in the context of software product acquisition by Bell Canada [34].

Concordant with the popularity of the object-oriented paradigm, there has been a concerted research effort to develop object-oriented product metrics [8], [17], [30], [80], [78], [27], [24], [60], [106] and to validate them [4], [27], [17], [19], [22], [78], [32], [57], [89], [106], [8], [25], [10]. For example, in [8], the relationship between a set of new polymorphism metrics and fault-proneness is investigated. A study of the relationship between various design and source code measures using a data set from student systems was reported in [4], [17], [22], [18], and a validation study of a large set of object-oriented metrics on an industrial system was described in [19]. Another industrial study is described in [27] where the authors investigate the relationship between object-oriented design metrics and two dependent variables: the number of defects and size in LOC. Li and Henry [78] report an analysis where they related object-oriented design and code metrics to the extent of code change, which they use as a surrogate for maintenance effort. Chidamber et al. [32] describe an exploratory analysis where they investigate the relationship between object-oriented metrics and productivity, rework effort, and design effort on three different financial systems, respectively. Tang et al. [106] investigate the relationship between a set of object-oriented metrics and faults found in three systems. Nesi and Querci [89] construct regression models to predict class development effort using a set of new metrics. Finally, Harrison et al. [57] propose a new object-oriented coupling metric and compare its performance with a more established coupling metric.

Despite minor inconsistencies in some of the results, a reading of the object-oriented metrics validation literature would suggest that a number of metrics are indeed "validated" in that they are strongly associated with outcomes of interest (e.g., fault-proneness) and that they can serve as good predictors of high-risk classes. The former is of course a precursor for the latter. For example, it has been stated that some metrics (namely the Chidamber and Kemerer—henceforth, CK—metrics of [30]) "have been empirically proven to be useful for the prediction of fault-prone modules" [106]. A recent review of the literature stated that "Existing data suggests that there are important relationships between structural attributes and external quality indicators" [23].

However, almost all of the validation studies that have been performed thus far completely ignore the potential confounding impact of class size. This is the case because the analyses employed are univariate: They only model the relationship between the product metric and the dependent variable of interest. For example, recent studies used the bivariate correlation between object-oriented metrics and the number of faults to investigate the validity of the metrics [57], [10]. Also, univariate logistic regression models are used as the basis for demonstrating the relationship between object-oriented product metrics and fault-proneness in [22], [19], [106]. The importance of controlling for potential confounders in empirical studies of object-oriented products has been emphasized [23]. However, size, the most obvious potential confounder, has not been controlled in previous validation studies.

The objective of this paper is to investigate the confounding effect of class size on the validation of object-oriented product metrics. We first demonstrate, based on previous work, that there is potentially a size confounding effect in object-oriented metrics validation studies and we present a methodology for empirically testing this. We then perform an empirical study on an object-oriented telecommunications framework written in C++ [102]. The metrics we investigate consist of the CK metrics suite[5] [30] and some of the metrics defined by Lorenz and Kidd [80]. The external metric that we validate against is the occurrence of a fault, which we term the *fault-proneness* of the class. In our study, a fault is detected due to a field failure.

Briefly, our results indicate that by using the commonly employed univariate analyses our results are consistent with previous studies. After controlling for the confounding effect of class size, *none* of the metrics is associated with fault-proneness. This indicates a strong confounding effect of class size on some common object-oriented metrics. The results cast serious doubt that many previous validation studies demonstrate more than that size is associated with fault-proneness.

Perhaps the most important practical implication of these results is that design and programming guidelines based on previous validation studies are questioned. Efforts to control cost and quality using object-oriented metrics as early indicators of problems may be achieved just as well using early indicators of size. The implications for research are that data from previous validation studies should be reexamined to gauge the impact of the size confounding effect, and future validation studies should control for size.

In Section 2, we provide the rationale behind the confounding effect of class size and present a framework for its empirical investigation. Section 3 presents our research method, and Section 4 includes the results of the study. We conclude the paper in Section 5 with a summary and directions for future work.

[4] It should be noted that the construction of guidelines requires the demonstration of a causal relationship rather than a mere association.

[5] It has been stated that for historical reasons the CK metrics are the most referenced [23]. Most commercial metrics collection tools available at the time of writing also collect these metrics.

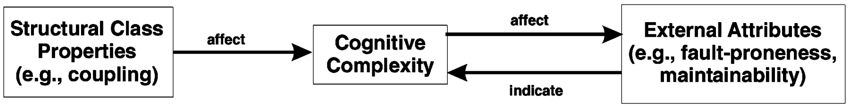
Structural Class Properties (e.g., coupling)
affect
Cognitive Complexity
affect
indicate
External Attributes (e.g., fault-proneness, maintainability)

Fig. 1. Theoretical basis for the development of object-oriented product metrics.

2 Background

This section is divided into two parts. First, we present the theoretical and empirical basis of the object-oriented metrics that we attempt to validate. Second, we demonstrate that there is a potentially strong size confounding effect in object-oriented metrics validation studies.

2.1 Theoretical and Empirical Basis of Object-Oriented Metrics

2.1.1 Theoretical Basis and Its Empirical Support

The primary reason why there is an interest in the development of product metrics in general is exemplified by the following justification for a product metric validity study: "There is a clear intuitive basis for believing that complex programs have more faults in them than simple programs" [87]. However, an intuitive belief does not make a theory. In fact, the lack of a strong theoretical basis driving the development of traditional software product metrics has been criticized in the past [68]. Specifically, Kearney et al. [68] state that "One of the reasons that the development of software complexity measures is so difficult is that programming behaviors are poorly understood. A behavior must be understood before what makes it difficult can be determined. To clearly state what is to be measured, we need a theory of programming that includes models of the program, the programmer, the programming environment, and the programming task."

In the arena of object-oriented metrics, a slightly more detailed articulation of a theoretical basis for developing quantitative models relating product metrics and external quality metrics has been provided in [19] and is summarized in Fig. 1. 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.[6]

Certain structural features of the object-oriented paradigm have been implicated in reducing the understandability of object-oriented programs, hence raising cognitive complexity. We describe these below.

Distribution of Functionality. In traditional applications developed using functional decomposition, functionality is localized in specific procedures, the contents of data structures are accessed directly, and data central to an application is often globally accessible [110]. Functional decomposition makes procedural programs easier to understand because it is based on a hierarchy in which a top-level function calls lower level functions to carry out smaller chunks of the overall task [109]. Hence, tracing through a program to understand its global functionality is facilitated.

In one experimental study with students and professional programmers [11], the authors compared maintenance time for three equivalent programs (implementing three different applications, therefore we have nine programs): They consisted of a straight serial structure (i.e., one main function), a program developed following the principles of functional decomposition, and an object-oriented program (without inheritance). In general, it took the students more time to change the object-oriented programs, and the professionals exhibited the same effect, although not as strongly. Furthermore, both the students and professionals noted that they found that it was most difficult to recognize program units in the object-oriented programs, and the students felt that it was also most difficult to find information in the object-oriented programs. Widenbeck et al. [109] make a distinction between program functionality at the local level and at the global (application) level. At the local level, they argue that the object-oriented paradigm's concept of encapsulation ensures that methods are bundled together with the data that they operate on, making it easier to construct appropriate mental models and specifically to understand a class' individual functionality. At the global level, functionality is dispersed among many interacting classes, making it harder to understand what the program is doing. They support this in an experiment with equivalent small C++ (with no inheritance) and Pascal programs whereby the subjects were better able to answer questions about the functionality of the C++ program. They also performed an experiment with larger programs. Here, the subjects with the C++ program (with inheritance) were unable to answer questions about its functionality much better than guessing. While this study was done with novices, it supports the general notions that high cohesion makes object-oriented programs easier to understand and high coupling makes them more difficult to understand. Wilde et al.'s [110] conclusions based on an interview-based study of two object-oriented systems at Bellcore implemented in C++ and an investigation of a PC Smalltalk environment, all in different application domains, are concordant with this finding, in that programmers have to understand a method's context of use by tracing back through the chain of calls that reach it, and tracing the chain of methods it uses. When there are many interactions, this exacerbates the understandability problem. An investigation of a C and a C++ system, both developed by the same staff in the same organization, concluded that "The developers found

[6] To reflect the likelihood that not only structural properties affect a component's external qualities, some authors have included additional metrics as predictor variables in their quantitative models, such as reuse [69], the history of corrected faults [70], and the experience of developers [72], [71]. However, this does not detract from the importance of the primary relationship between product metrics and a component's external qualities.

it much harder to trace faults in the OO C++ design than in the conventional C design. Although this may simply be a feature of C++, it appears to be more generally observed in the testing of OO systems, largely due to the distorted and frequently nonlocal relationships between cause and effect: The manifestation of a failure may be a 'long way away' from the fault that led to it. [...] Overall, each C++ correction took more than twice as long to fix as each C correction." [59].

Inheritance Complications. As noted in [43], there has been a preoccupation within the community with inheritance and, therefore, more studies have investigated that particular feature of the object-oriented paradigm.

Inheritance introduces a new level of delocalization, making the understandability even more difficult. It has been noted that "Inheritance gives rise to distributed class descriptions. That is, the complete description for a class C can only be assembled by examining C as well as each of C's superclasses. Because different classes are described at different places in the source code of a program (often spread across several different files), there is no single place a programmer can turn to get a complete description of a class" [77]. While this argument is stated in terms of source code, it is not difficult to generalize it to design documents. Wilde et al.'s study [110] indicated that to understand the behavior of a method one has to trace inheritance dependencies, which is considerably complicated due to dynamic binding. A similar point was made in [77] about the understandability of programs in languages that support dynamic binding, such as C++.

In a set of interviews with 13 experienced users of object-oriented programming, Daly et al. [40] noted that if the inheritance hierarchy is designed properly then the effect of distributing functionality over the inheritance hierarchy would not be detrimental to understanding. However, it has been argued that there exists increasing conceptual inconsistency as one travels down an inheritance hierarchy (i.e., deeper levels in the hierarchy are characterized by inconsistent extensions and/or specializations of superclasses) [45], therefore, inheritance hierarchies may not be designed properly in practice. In one study, Dvorak [45] found that subjects were more inconsistent in placing classes deeper in the inheritance hierarchy when compared to at higher levels in the hierarchy.

An experimental investigation found that making changes to a C++ program with inheritance consumed more effort than a program without inheritance, and the author attributed this to the subjects finding the inheritance program more difficult to understand based on responses to a questionnaire [26]. A contradictory result was found in [41], where the authors conducted a series of classroom experiments comparing the time to perform maintenance tasks on a "flat" C++ program and a program with three levels of inheritance. This was premised on a survey of object-oriented practitioners showing 55 percent of respondents agreeing that inheritance depth is a factor when attempting to understand object-oriented software [39]. The result was a significant reduction in maintenance effort for the inheritance program. An internal replication by the same authors found the results to be in the same direction, albeit the p-value was larger. The second experiment in [41] found that C++ programs with five levels of inheritance took more time to maintain than those with no inheritance, although the effect was not statistically significant. The authors explain this by observing that searching/tracing through the bigger inheritance hierarchy takes longer. Two experiments that were partial replications of the Daly et al. experiments produced different conclusions [107]. In both experiments, the subjects were given three equivalent Java programs to make changes to, and the maintenance time was measured. One of the Java programs was "flat," one had an inheritance depth of three, and one had an inheritance depth of five. The results for the first experiment indicate that the programs with inheritance depth of three took longer to maintain than the "flat" program, but the program with inheritance depth of five took as much time as the "flat" program. The authors attribute this to the fact that the amount of changes required to complete the maintenance task for the deepest inheritance program was smaller. The results for a second task in the first experiment and the results of the second experiment indicate that it took longer to maintain the programs with inheritance. To explain this finding and its difference from the Daly et al. results, the authors showed that the "number of methods relevant for understanding" (which is the number of methods that have to be traced in order to perform the maintenance task) was strongly correlated with the maintenance time, and this value was much larger in their study compared with the Daly et al. programs. The authors conclude that inheritance depth per se is not the factor that affects understandability, but the number of methods that have to be traced.

Summary. The current theoretical framework for explaining the effect of the structural properties of object-oriented programs on external program attributes can be justified empirically. To be specific, studies that have been performed indicate that the distribution of functionality across classes in object-oriented systems, and the exacerbation of this through inheritance, potentially makes programs more difficult to understand. This suggests that highly cohesive, sparsely coupled, and low inheritance programs are less likely to contain a fault. Therefore, metrics that measure these three dimensions of an object-oriented program would be expected to be good predictors of fault-proneness or the number of faults.

The empirical question is then whether contemporary object-oriented metrics measure the relevant structural properties well enough to substantiate the above theory. Below, we review the evidence on this.

2.1.2 Empirical Validation of Object-Oriented Metrics

In this section, we review the empirical studies that investigate the relationship between the ten object-oriented metrics that we study and fault-proneness (or number of faults). The product metrics cover the following dimensions: coupling, cohesion, inheritance, and complexity. These dimensions are based on the definition of the metrics, and may not reflect their actual behavior.

Coupling metrics characterize the static usage dependencies among the classes in an object-oriented system [21].

Cohesion metrics characterize the extent to which the methods and attributes of a class belong together [16]. Inheritance metrics characterize the structure of the inheritance hierarchy. Complexity metrics, as used here, are adaptations of traditional procedural paradigm complexity metrics to the object-oriented paradigm.

Current methodological approaches for the validation of object-oriented product metrics are best exemplified by two articles by Briand et al. [19], [22]. These are validation studies for an industrial communications system and a set of student systems, respectively, where a considerable number of contemporary object-oriented product metrics were studied. We single out these studies because their methodological reporting is detailed and because they reflect what can be considered the best methodological practice to date.

The basic approach starts with a data set of product metrics and binary fault data for a complete system or multiple systems. The important element of the Briand et al. methodology that is of interest to us here is the univariate analysis that they stipulate should be performed. In fact, the main association between the product metrics and fault-proneness is established on the basis of the univariate analysis. If the relationship is statistically significant (and in the expected direction) then a metric is considered validated.[7] For instance, in [22], the authors state a series of hypotheses relating each metric with fault-proneness. They then explain "Univariate logistic regression is performed, for each individual measure (independent variable), against the dependent variable to determine if the measure is statistically related, in the expected direction, to fault-proneness. This analysis is conducted to test the hypotheses." Subsequently, the results of the univariate analysis are used to evaluate the extent of evidence supporting each of the hypotheses. Reliance on univariate results as the basis for drawing validity conclusions is common practice (e.g., see [4], [10], [17], [18], [57], [106]).

In this review, we first present the definition of the metrics as we have operationalized them. The operationalization of some of the metrics is programming language dependent. We then present the magnitude of the coefficients and p values computed in the various studies. Validation coefficients were either the change in odds ratio as a measure of the magnitude of the metric to fault-proneness association from a logistic regression (see the appendix, Section 7) or the Spearman correlation coefficient. Finally, this review focuses only on the fault-proneness or number of faults dependent variable. Other studies that investigated effort, such as [32], [89], [78], are not covered as effort is not the topic of the current paper.

WMC. This is the Weighted Methods per Class metric [30] 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 [30]. We weight it using cyclomatic complexity as did [78]. However, other authors did not adopt a weighting scheme [4], [106]. Methods from ancestor classes are not counted and neither are "friends" in C++. This is similar to the approach taken in, for example, [4], [31]. To be precise, WMC was counted after preprocessing to avoid undercounts due to macros [33].[8]

One study found WMC to be associated with fault-proneness on three different subsystems written in C++ with p-values 0.054, 0.0219, and 0.0602, and change in odds ratio 1.26, 1.45, and 1.26 [106].[9] A study that evaluated WMC on a C++ and a Java application found WMC to have a Spearman correlation of 0.414 and 0.456 with the number of faults due to field failures, respectively, and highly significant p-values (<0.0001 and <0.0056) [10]. Another study using student systems found WMC to be associated with fault-proneness with a p-value for the logistic regression coefficient of 0.0607 [4].[10]

DIT. The Depth of Inheritance Tree [30] 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.

The DIT metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was related to fault-proneness (p=0.0074) with a change in odds ratio equal to 0.572 when measured on nonlibrary classes. The second study [22] also found it to be associated with fault-proneness (p=0.0001) with a change in odds ratio of 2.311. Another study using student systems found DIT to be associated with fault-proneness with a p-value for the logistic regression coefficient <0.0001 [4].

It will be noted that in the first study a negative association was found between DIT and fault-proneness. The authors explain this by stating that in the system studied classes located deeper in the inheritance hierarchy provide only implementations for a few specialized methods and are therefore less likely to contain faults than classes closer to the root [19]. This was a deliberate strategy to place as much functionality as close as possible to the root of the inheritance tree. Note that, for the latter two investigations, the same data set was used and, therefore, the slightly different coefficients may have been due to removal of outliers.

One study using data from an industrial system found that classes involved in an inheritance structure were more likely to have defects (found during integration testing and within 12 months postdelivery) [27]. Another study did not find DIT to be associated with fault-proneness on three different subsystems written in C++, where faults were based on three years' worth of trouble reports [106]. One study that evaluated DIT on a Java application found that it

[7] Briand et al. use logistic regression and consider the statistical significance of the regression parameters.

[8] 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 [31], would undercount methods known at design time.

[9] In this study, faults were classified as either object-oriented type faults or traditional faults. The values presented here are for all of the faults, although the same metrics were found to be significant for both all faults and the object-oriented only faults. Furthermore, the change in odds ratio reported is based on a change of one unit of the metric rather than a change in the standard deviation.

[10] This study used the same data set as in [22], except that the data was divided into subsets using different criteria. The results presented here are for all of the classes.

had a Spearman correlation of 0.523 (p<0.0015) with the number of faults due to field failures [10].

NOC. This is the Number of Children inheritance metric [30]. 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).

The NOC metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was not related to fault-proneness. Conversely, the second study [22] found it to be associated with fault-proneness (p=0.0276) with a change in odds ratio of 0.322. Another study using student systems found NOC to be associated with fault-proneness with a p-value for the regression coefficient <0.0001 [4]. Note that, for the latter two investigations, the same data set was used and, therefore, the slightly different coefficients may have been due to removal of outliers. In both studies, NOC had a negative association with fault-proneness and this was interpreted as indicating that greater attention was given to these classes (e.g., through inspections) given that many classes were dependent on them.

Another study did not find NOC to be associated with fault-proneness on three different subsystems written in C++, where faults were based on three years' worth of trouble reports [106]. NOC was not associated with the number of faults due to field failures in a study of two systems, one implemented in C++ and the other in Java [10].

CBO. This is the Coupling Between Object Classes coupling metric [30]. A class is coupled with another if methods of one class uses methods or attributes of the other, or vice versa. In this definition, uses can mean a member type, parameter type, or 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).

The CBO metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was related to fault-proneness (p<0.0001) with a change in odds ratio equal to 5.493 when measured on nonlibrary classes. The second study [22] also found it to be associated with fault-proneness (p<0.0001) with a change in odds ratio of 2.012 when measured on nonlibrary classes. Another study did not find CBO to be associated with fault-proneness on three different subsystems written in C++, where faults were based on three years worth of trouble reports [106]. This was also the case in a recent empirical analysis on two traffic simulation systems, where no relationship between CBO and the number of known faults was found [57], and a study of a Java application where CBO was not found to be associated with faults due to field failures [10]. Finally, another study using student systems found CBO to be associated with fault-proneness with a p-value for the logistic regression coefficient <0.0001 [4].

RFC. This is the Response for a Class coupling metric [30]. 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.

The RFC metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was related to fault-proneness (p=0.0019) with a change in odds ratio equal to 1.368 when measured on nonlibrary classes. The second study [22] also found it to be associated with fault-proneness (p<0.0001) with a change in odds ratio of 3.208 when measured on nonlibrary classes. Another study found RFC to be associated with fault-proneness on two different subsystems written in C++ with p-values 0.0401 and 0.0499, and change in odds ratio 1.0562 and 1.0654 [106].[11] A study that evaluated RFC on a C++ application and a Java application found RFC to have a Spearman correlation of 0.417 and 0.775 with the number of faults due to field failures, respectively, and highly significant p-values (both <0.0001) [10]. Another study using student systems found RFC to be associated with fault-proneness with a p-value for the logistic regression coefficient <0.0001 [4].

LCOM. This is a cohesion metric that was defined in [30]. 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.

The LCOM metric was empirically evaluated in [19], [22]. In [19], the authors found it to be associated with fault-proneness (p=0.0.249) with a change in odds ratio of 1.613. Conversely, the second study [22] did not find it to be associated with fault-proneness.

NMO. This is an inheritance metric that has been defined in [80] and measures the number of inherited methods overriden by a subclass. A large number of overriden methods indicates a design problem [80]. Since a subclass is intended to specialize its parent, it should primarily extend the parent's services [94]. This should result in unique new method names. Numerous overrides indicate subclassing for the convenience of reusing some code and/or instance variables when the new subclass is not purely a specialization of its parent [80].

The NMO metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was related to fault-proneness (p=0.0082) with a change in odds ratio equal to 1.724. The second study [22] also found it to be associated with fault-proneness (p=0.0243) with a change in odds ratio of 1.948.

Lorenz and Kidd [80] caution that, in the context of frameworks, methods are often defined specifically for reuse or that are meant to be overriden. Therefore, for our study, there is already an a priori expectation that this metric may not be a good predictor.

NMA. This is an inheritance metric that has been defined in [80] and measures the number of methods added by a subclass (inherited methods are not counted). As this value becomes larger for a class, the functionality of that class becomes increasingly distinct from that of the parent classes.

[11] In this study, faults were classified as either object-oriented type faults or traditional faults. The values presented here are for all of the faults, although the same metrics were found to be significanct for both all faults and the object-oriented only faults. Furthermore, the change in odds ratio reported is based on a change of one unit of the metric rather than a change in the standard deviation.

The NMO metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was related to fault-proneness (p=0.0021) with a change in odds ratio equal to 3.925, a rather substantial effect. The second study [22] also found it to be associated with fault-proneness (p=0.0021) with a change in odds ratio of 1.710.

SIX. This is an inheritance metric that has been defined in [80] and consists of a combination of inheritance metrics. It is calculated as the product of the number of overriden methods and the class hierarchy nesting level normalized by the total number of methods in the class. The higher value for SIX, the more likely that a particular class does not conform to the abstraction of it's superclasses [94].

The SIX metric was empirically evaluated in [19], [22]. In [19], the authors found that this metric was not related to fault-proneness. Conversely, the second study [22] found it to be associated with fault-proneness (p=0.0089) with a change in odds ratio of 1.337.

NPAVG. This can be considered as a coupling metric and has been defined in [80] and measures the average number of parameters per method (not including inherited methods). Methods with a high number of parameters generally require considerable testing (as their input can be highly varied). Also, large numbers of parameters lead to more complex, less maintainable code.

Summary. The current empirical studies do provide some evidence that object-oriented metrics are associated with fault-proneness or the incidence of faults. Though, the evidence is equivocal. For some of the inheritance metrics that were studied (DIT and NOC), some studies found a positive association, some found a negative association, and some found no association. The CBO metric was found to be positively associated with fault-proneness in some studies and not associated with either the number of faults found or fault-proneness in other studies. The RFC and WMC metrics were consistently found to be associated with fault-proneness. The NMO and NMA metrics were found to be associated with fault-proneness, but the evidence for the SIX metric is more equivocal. The LCOM cohesion metric also has equivocal evidence supporting its validity.

It should be noted that the differences in the results obtained across studies may be a consequence of the measurement of different dependent variables. For instance, some treat the dependent variable as the (continuous) number of defects found. Other studies use a binary value of incidence of a fault during testing or in the field, or both. It is plausible that the effects of product metrics may be different for each of these.

An optimistic observer would conclude that the evidence of the predictive validity of most of these metrics is good enough to recommend their practical usage.

2.2 The Confounding Effect of Size

In this section, we take as a starting point the stance of an optimistic observer and assume that there is sufficient empirical evidence demonstrating the relationship between the object-oriented metrics that we study and fault-proneness. We already showed that previous empirical studies drew their conclusions from univariate analyses. Below, we make the argument that univariate analyses ignore the potential confounding effects of class size. We show that if there is indeed a size confounding effect, then previous empirical studies could have harbored a large positive bias.

For ease of presentation, we take, as a running example, a coupling metric as the main metric that we are trying to validate. For our purposes, a validation study is designed to determine whether there is an association between coupling and fault-proneness. Furthermore, we assume that this coupling metric is appropriately dichotomized: Low Coupling (LC) and High Coupling (HC). This dichotomization assumption simplifies the presentation, but the conclusions can be directly generalized to a continuous metric.

2.2.1 The Case Control Analogy

An object-oriented metrics validation study can be easily seen as an unmatched case-control study. Case-control studies are frequently used in epidemiology to, for example, study the effect of exposure to carcinogens on the incidence of cancers [95], [12].[12] The reason for using case-control studies as opposed to randomized experiments in certain instances is that it would not be ethically and legally defensible to do otherwise. For example, it would not be possible to have deliberately composed "exposed" and "unexposed" groups in a randomized experiment when the exposure is a suspected carcinogen or toxic substance. Randomized experiments are more appropriately used to evaluate treatments or preventative measures [52].

In applying the conduct of a case-control study to the validation of an object-oriented product metric, one would first proceed by identifying classes that have faults in them (the cases). Then, for the purpose of comparison another group of classes without faults in them are identified (the controls). We determine the proportion of cases that have, say High Coupling and the proportion with Low Coupling. Similarly, we determine the proportion of controls with High Coupling and the proportion with Low Coupling. If there is an association of coupling with fault-proneness then the prevalence of High Coupling classes would be higher in the cases than in the controls. Effectively then, a case-control study follows a paradigm that proceeds from effect to cause, attempting to find antecedents that lead to faults [99]. In a case-control study, the control group provides an estimate of the frequency of High Coupling that would be expected among the classes that do not have faults in them.

In an epidemiological context, it is common to have "hospital-based cases" [52], [95]. For example, a subset or all patients that have been admitted to a hospital with a particular disease can be considered as cases. Controls can also be selected from the same hospital or clinic.[13] The selection of controls is not necessarily a simple affair. For example, one can match the cases with controls on some confounding variables, for instance, on age and sex. Matching ensures that the cases and controls are similar on the matching variable and, therefore, this variable cannot

[12] Other types of studies that are used are cohort-studies [52], but we will not consider these here.

[13] This raises the issue of generalizability of the results. However, as noted by Breslow and Day [12], generalization from the sample in a case-control study depends on nonstatistical arguments. The concern with the design of the study is to maximize internal validity. In general, replication of results establishes generalizability [79].

TABLE 1
A Contingency Table Showing the Results of a Hypothetical Validation Study

		Fault Proneness	
		Faulty	Not Faulty
Coupling	HC	91	19
	LC	19	91

be considered a causal factor in the analysis. Alternatively, one can have an unmatched case-control study and control for confounding effects during the analysis stage.

In an unmatched case-control study, the determination of an association between the exposure (product metric) and the disease (fault-proneness) proceeds by calculating a measure of association and determining whether it is significant. For example, consider the contingency Table 1 that is obtained from a hypothetical validation study.

For this particular data set, the odds ratio is 22.9 (see the appendix, Section 7, for a definition of the odds ratio), which is highly significant, indicating a strong positive association between coupling and fault-proneness.

2.2.2 The Potential Confounding Effect of Size

One important element that has been ignored in previous validation studies is the potential confounding effect of class size. This is illustrated in Fig. 2.

The path diagram in Fig. 2 depicts a classic textbook example of confounding in a case-control study [99], [12].[14] The path (a) represents the current causal beliefs about product metrics being an antecedent to fault-proneness. The path (b) depicts a positive causal relationship between size and fault-proneness. The path (c) depicts a positive association between product metrics and size.

If this path diagram is concordant with reality, then size distorts the relationship between product metrics and fault-proneness. Confounding can result in considerable bias in the estimate of the magnitude of the association. Size is a positive confounder, which means that ignoring size will always result in the association between say coupling and fault-proneness to be more positive than it really is.

The potential confounding effect of size can be demonstrated through an example (adapted from [12]). Consider the table in Table 1 that gave an odds ratio of 22.9. As mentioned earlier, this is representative of the current univariate analyses used in the object-oriented product metrics validation literature (which explicitly exclude size as a covariate nor employ a stratification on size).

Now, let us say that if we analyze the data seperately for small and large classes, we have the data in Table 2 for the large classes and the data in Table 3 for the small classes.[15]

In both of the tables, the odds ratio is one. By stratifying on size (i.e., controlling for the effect of size), the association between coupling and fault-proneness has been reduced dramatically. This is because size was the reason why there was an association between coupling and fault-proneness in the first place. Once the influence of size is removed, the example shows that the impact of the coupling metric disappears.

Therefore, an important improvement on the conduct of validation studies of object-oriented metrics is to control for the effect of size, otherwise one may be getting the illusion that the product metric is strongly associated with fault-proneness, when, in reality, the association is much weaker or nonexistent.

2.2.3 Evidence of a Confounding Effect

Now, we must consider whether the path diagram in Fig. 2 can be supported in reality.

There is evidence that object-oriented product metrics are associated with size. For example, in [22], the Spearman rho correlation coefficients go as high as 0.43 for associations between some coupling and cohesion metrics with size and 0.397 for inheritance metrics and both are statistically significant (at an alpha level of say 0.1). Similar patterns emerge in the study reported in [19], where relatively large correlations are shown. In another study [27], the authors display the correlation matrix showing the Spearman correlation between a set of object-oriented metrics that can be collected from Shlaer-Mellor designs and C++ LOC. The correlations range from 0.563 to 0.968, all statistically significant at an alpha level 0.05. This also indicates very strong correlations with size.

Associations between size and defects have been reported in nonobject-oriented systems [58]. For object-oriented programs, the relationship between size and defects is clearly visible in the study of [27], where the Spearman correlation was found to be 0.759 and statistically significant. Another study of image analysis programs written in C++ found a Spearman correlation of 0.53 between size in LOC and the number of errors found during testing [55] and was statistically significant at an alpha level of 0.05. Briand et al. [22] find statistically significant associations between six different size metrics and fault-proneness for C++ programs, with a change in odds ratio going as high as 4.952 for one of the size metrics.

General indications of a confounding effect are seen in Fig. 3, which shows the associations between a set of coupling metrics and fault-proneness, and with size from a recent study [22]. The association between coupling metrics and fault-proneness is given in terms of the change in the odds ratio and the p-value of the univariate logistic regression parameter. The association with size is in terms of the Spearman correlation. As can be seen in Fig. 3, all the metrics that had a significant relationship with fault-proneness in the univariate analysis also had a significant correlation with size. Furthermore, there is a general trend of increasing association between the coupling metric and fault-proneness as its association with size increases.

[14] We make the analogy to a case-control study because it provides us with a well-tested framework for defining and evaluating confounding effects, as well as for conducting observational studies from which one can make stronger causal claims (if all known confounders are controlled). However, for the sole purposes of this paper, the characteristics of a confounding effect have been described and exemplified in [61] without resort to a case-control analogy.

[15] Note that, in this example, the odds ratio of the size to fault-proneness association is 100 and the size to coupling association is 81.3. Therefore, it follows the model in Fig. 2.

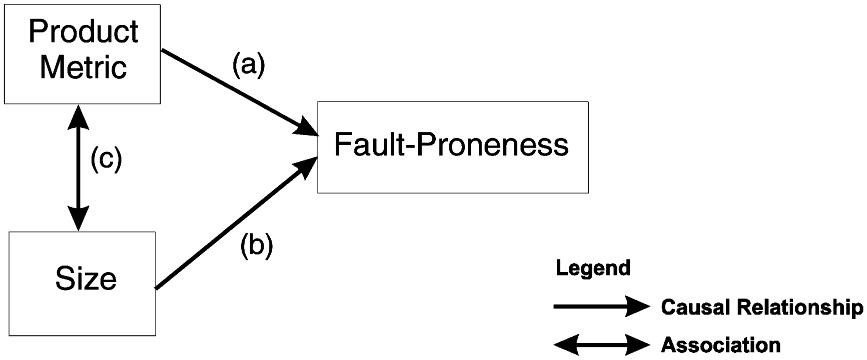

Fig. 2. Path diagram illustrating the confounding effect of size.

This leads us to conclude that, potentially, previous validation studies have overestimated the impact of object-oriented metrics on fault-proneness due to the confounding effect of size.

2.3 Summary

In this section, the theoretical basis for object-oriented product metrics was presented. This states that cognitive complexity is an intervening variable between the structural properties of classes and fault-proneness. Furthermore, the empirical evidence supporting the validity of the object-oriented metrics that we study was presented, and this indicates that some of the metrics are strongly associated with fault-proneness or the number of faults. We have also demonstrated that there is potentially a strong size confounding effect in empirical studies to date that validate object-oriented product metrics. This makes it of paramount importance to determine whether such a strong confounding effect really exists.

If a size confounding effect is found, this means that previous validation studies have a positive bias and may have exaggerated the impact of product metrics on fault-proneness. The reason is that studies to date relied exclusively on univariate analysis to test the hypothesis that the product metrics are associated with fault-proneness or the number of faults. The objective of the study below then is to directly test the existence of this confounding effect and its magnitude.

TABLE 2
A Contingency Table Showing the Results for Only *Large* Classes of a Hypothetical Validation Study

		Fault Proneness	
		Faulty	Not Faulty
Coupling	HC	90	9
	LC	10	1

TABLE 3
A Contingency Table Showing the Results for Only Small *Classes* of a Hypothetical Validation Study

		Fault Proneness	
		Faulty	Not Faulty
Coupling	HC	1	10
	LC	9	90

3 Research Method

3.1 Data Source

Our data set comes from a telecommunications framework written in C++ [102]. The framework implements many core design patterns for concurrent communication software. The communication software tasks provided by this framework include event demultiplexing and event handler dispatching, signal handling, service initialization, interprocess communication, shared-memory management, message routing, dynamic (re)configuration of distributed services, and concurrent execution and synchronization. The framework has been used in applications such as electronic medical imaging systems, configurable telecommunications systems, high-performance real-time CORBA, and web servers. Examples of its application include in the Motorola Iridium global personal communications system [101] and in network monitoring applications for telecommunications switches at Ericsson [100]. A total of 174 classes from the framework that were being reused in the development of commercial switching software constitute the system that we study. A total of 14 different programmers were involved in the development of this set of classes.[16]

3.2 Measurement

3.2.1 Product Metrics

All product metrics are defined on the class and constitute design metrics, and have been presented in Section 2.1.2. In our study, the size variable was measured as a noncomment source LOC for the class. Measurement of product metrics used a commercial metrics collection tool that is currently being used by a number of large telecommunications software development organizations.

3.2.2 Dependent Variable

For this product, we obtained data on the faults found in the library from actual field usage.[17] Each fault was due to a

[16] This number was obtained from the different login names of the version control system associated with each class.

[17] It has been argued that considering faults causing field failures is a more important question to address than faults found during testing [9]. In fact, it has been argued that it is the ultimate aim of quality modeling to predict postrelease fault-proness [50]. In at least one study, it was found that prerelease fault-proneness is not a good surrogate measure for postrelease fault-proness, the reason posited being that prerelease fault-proness is a function of testing effort [51].

Metric	Relationship with fault-proneness		Relationship with size	
	Change in odds Ratio	p-value	rho	p-value
CBO	2.012	<0.0001	0.3217	<0.0001
CBO'	2.062	<0.0001	0.3359	<0.0001
RFC1	3.208	<0.0001	0.3940	<0.0001
RFC$_{\infty}$	8.168	<0.0001	0.4310	<0.0001
MPC	5.206	<0.0001	0.3232	<0.0001
ICP	7.170	<0.0001	0.3168	<0.0001
IH-ICP	1.090	0.5898	-0.124	0.1082
NIH-ICP	9.272	<0.0001	0.3455	<0.0001
DAC	1.395	0.0329	0.1753	0.0163
DAC'	1.385	0.0389	0.1958	0.0088
OCAIC	1.416	0.0307	0.1296	0.0785
FCAEC	1.206	0.3213	0.0297	0.7010
OCMIC	1.133	0.3384	0.0493	0.4913
OCMEC	0.816	0.252	-0.0855	0.2528
IFMMIC	1.575	0.0922	0.2365	0.0019
AMMIC	1.067	0.6735	-0.1229	0.1115
OMMIC	4.937	<0.0001	0.2765	0.0001
OMMEC	1.214	0.2737	-0.0345	0.6553

Fig. 3. Relationship between coupling metrics and fault-proneness, and between coupling metrics and size from [22]. This only covers coupling to nonlibrary classes. This also excludes the following metrics because no results pertaining to the relationship with fault-proneness were presented: ACAIC, DCAEC, IFCMIC, ACMIC, IFCMEC, and DCMEC. The definition of these metrics is provided in the appendix.

unique field failure and represents a defect in the program that caused the failure. Failures were reported by the users of the framework. The developers of the framework documented the reasons for each delta in the version control system, and it was from this that we extracted information on whether a class was faulty.

A total of 192 faults were detected in the framework at the time of writing. These faults occurred in 70 out of 174 classes. The dichotomous dependent variable that we used in our study was the detection or nondetection of a fault. If one or more faults are detected, then the class is considered to be faulty and, if not, then it is considered not faulty.

3.3 Data Analysis Methods

3.3.1 Testing for a Confounding Effect

It is tempting to use a simple approach to test for a confounding effect of size: examine the association between size and fault-proneness. If this association is not significant at a traditional alpha level, then conclude that size is not different between cases and controls (and hence has no confounding effect) and proceed with a usual univariate analysis.

However, it has been noted that this is an incorrect approach [38]. The reason is that traditional significance testing places the burden of proof on rejecting the null hypothesis. This means that one has to prove that the cases and controls do differ in size. In evaluating confounding potential, the burden of proof should be in the opposite direction: Before discarding the potential for confounding, the researcher should demonstrate that cases and controls do not differ on size. This means controlling the Type II error rather than the Type I error. Since one usually has no control over the sample size, this means setting the alpha level to 0.25, 0.5, or even larger.

A simpler and more parsimonious approach is as follows: For an unmatched case-control study, a measured confounding variable can be controlled through a regression adjustment [12], [99]. A regression adjustment entails including the confounder as another independent variable in a regression model. If the regression coefficient of the object-oriented metric changes dramatically (in magnitude and statistical significance) with and without the size variable, then this is a strong indication that there was indeed a confounding effect [61]. This is further elaborated below.

3.3.2 Logistic Regression Model

Binary logistic regression is used to construct models when the dependent variable can only take on two values, as in our case. It is most convenient to use a logistic regression (henceforth LR) model rather than the contingency table analysis used earlier for illustrations since the model does not require dichotomization of our product metrics.

The general form of an LR model is:

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

where π is the probability of a class having a fault, and the xs are the independent variables. The β parameters are estimated through the (unconditional)[18] maximization of a log-likelihood [61].

[18] Conditional logistic regression is used when there has been matching in the case-control study and each matched set is treated as a stratum in the analysis [12].

In a univariate analysis only one x_i, x_1, is included in the model, and this is the product metric that is being validated:

$$\pi = \frac{1}{1 + e^{-(\beta_0 + \beta_i x_1)}}. \tag{2}$$

When controlling for size, a second x_i, x_2, is included that measures size:

$$\pi = \frac{1}{1 + e^{-(\beta_0 + \beta_i x_1 + \beta_2 x_2)}}. \tag{3}$$

In constructing our models, we could follow the previous literature and not consider interaction effects nor consider any transformations (for example, see [4], [8], [17], [18], [19], [22], [106]). To err on the conservative side, however, we did test for interaction effects between the size metric and the product metric for all product metrics evaluated. In none of the cases was a significant interaction effect identified. Furthermore, we performed a logarithmic transformation on our variables[19] and reevaluated all the models. Our conclusions would not be affected by using the transformed models.[20] Therefore, we only present the detailed results for the untransformed model.

The magnitude of an association can be expressed in terms of the change in odds ratio as the x_1 variable changes by one standard deviation. This is explained in the appendix (Section 7) and is denoted by $\Delta\Psi$. Since we construct two models, as shown in (2) and (3), without and with controlling for size, respectively, we will denote the change in odds ratio as $\Delta\Psi_{x_1}$ and $\Delta\Psi_{x_1+x_2}$, respectively. As suggested in [74], we can evaluate the extent to which the change in odds ratio changes as an indication of the extent of confounding. We operationalize this as follows:

$$\Delta^2\psi = \left| \frac{\Delta\psi_{x_1} - \Delta\psi_{x_1+x_2}}{\Delta\psi_{x_1+x_2}} \right| \times 100. \tag{4}$$

This gives the percent change in $\Delta\Psi_{x_1+x_2}$ by removing the size confounder. If this value is large then we can consider that class size does indeed have a confounding effect. The definition of "large" can be problematic, however, as will be seen in the results, the changes are sufficiently big in our study that, by any reasonable threshold, there is little doubt.

3.3.3 Diagnostics and Hypothesis Testing

The appendix of this paper presents the details of the model diagnostics that were performed and the approach to hypothesis testing. Here, we summarize these.

The diagnostics concerned checking for collinearity and identifying influential observations. We compute the condition number specific to logistic regression, η_{LR}, to determine whether dependencies amongst the independent variables are affecting the stability of the model (collinearity). The $\Delta\beta$ value provides us an indication of which observations are overly influential. For hypothesis testing, we use the likelihood ratio statistic, G, to test the significance of the overall model, the Wald statistic to test for the significance of individual model parameters, and the Hosmer and Lemeshow R^2 value as a measure of goodness of fit. Note that for the univariate model the G statistic and the Wald test are statistically equivalent, but we present them both for completeness. All statistical tests were performed at an alpha level of 0.05.

4 Results

4.1 Descriptive Statistics

Box and whisker plots for all the product metrics that we collected are shown in Fig. 4. These indicate the median, the 25th and 75th quantiles. Outliers and extreme points are also shown in the figure.[21]

As is typical with product metrics, their distributions are clearly heavy tailed. Most of the variables are counts, and therefore their minimal value is zero. Variables NOC, NMO, and SIX have less than six observations that are nonzero. Therefore, they were excluded from further analysis. This is the approach followed in [22].

The fact that few classes have NOC values greater than zero indicates that most classes in the system are leaf classes. Overall, 76 of the classes had a DIT value greater than zero, indicating that they are subclasses. The remaining 98 classes are at the root of the inheritance hierarchy. The above makes clear that the inheritance hierarchy for this system was "flat." Variable DIT has a small variation, but this is primarily due to there not being a large amount of deep inheritance in this framework. Shallow inheritance trees, indicating sparse use of inheritance, have been reported in a number of systems thus far [27], [30], [32].

The LCOM values may seem to be large. However, examination of the results from previous systems indicate that they are not exceptional. For instance, in the C++ systems reported in [22], the maximum LCOM value was 818, the mean 43, and the standard deviation was 106. Similarly, the system reported in [19] had a maximum LCOM value of 4988, a mean of 99.6, and standard deviation of 547.7.

4.2 Correlation with Size

Table 4 shows the correlation of the metrics with size as measured in LOC. As can be seen, all of the associations are statistically significant except DIT. But, DIT did not have much variation and, therefore, a weak association with size is not surprising. All metrics except LCOM and NPAVG have a substantial correlation coefficient, indicating a nontrivial assocation with size.

4.3 Validation Results

The results of the univariate analyses and the models controlling for size for each of the remaining metrics are presented in this section. The complete results are presented in Table 5.

[19] Given that product metrics are counts, an appropriate transformation to stablize the variance would be the logarithm.

[20] We wish to thank an anonymous reviewer for making this suggestion.

[21] As will be noted that, in some cases, the minimal value is zero. For metrics such as CBO, WMC, and RFC, this would be because the class was defined in a manner similar to a C struct, with no methods associated with it.

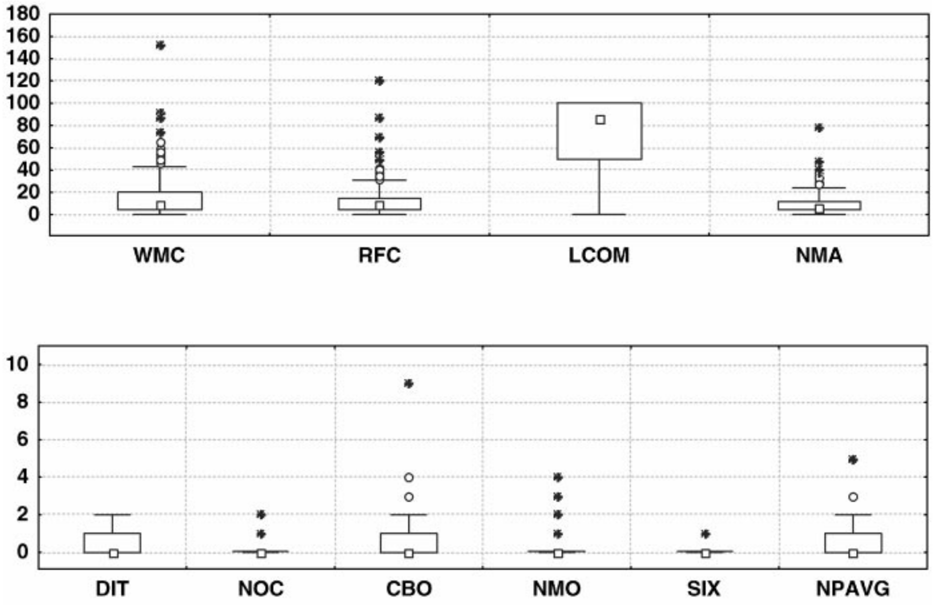

Fig. 4. Box and whisker plots for all the object-oriented product metrics. Two charts are shown to allow for the fact that two y-axis scales are required due to the different ranges.

TABLE 4
Spearman Correlation of the Object-Oriented Metrics (Only the Ones that Have More Than Five Nonzero Values) with Size in LOC

OO Metric	LOC	
	Rho	p-value
WMC	0.88	<0.0001
DIT	0.098	0.19
CBO	0.46	<0.0001
RFC	0.88	<0.0001
LCOM	0.24	0.0011
NMA	0.86	<0.0001
NPAVG	0.27	0.000256

4.3.1 WMC

As can be seen in Table 5, the univariate result for WMC is rather consistent with the previous literature that explicitly evaluated WMC [4], [10], [106]. The overall goodness of fit is significant, and the regression parameter is significant. Furthermore, the $\Delta\Psi$ indicates that an increase of one standard deviation in the value of WMC will increase the odds of a fault 1.74 times.

The results after the correction for size show that the regression parameter for WMC is no longer statistically significant, but the impact of size is large and statistically significant. The condition number does not indicate that collinearity is a problem. The WMC coefficient is negative, but this is not different from zero after taking sampling variability into account. Table 5 shows that the inclusion of size in the model has a dramatic impact on the validity of WMC and is confirmed by a $\Delta^2\psi$ value of 117 percent. Hence, we can say that size has a confounding effect on WMC.

4.3.2 DIT

The results for the DIT metric as shown in Table 5 indicate that there is no relationship between DIT and fault-proneness, with the overall G coefficient being nonsignificant. An analysis similar to that in [27] whereby the classes were dichotomized into those involved in inheritance and those not involved was performed. This subsequent analysis does not change the conclusions that DIT is not associated with fault-proneness in this system.

4.3.3 CBO

The univariate analysis results for CBO indicate that CBO is associated with fault-proneness with a change in odds ratio of 1.387. This is consistent with previous literature that found CBO to be associated with fault-proneness [22], [19].

After controlling for the confounding impact of size, the effect of CBO on fault-proneness diminishes considerably, with a $\Delta^2\psi$ value of 35 percent. Furthermore, CBO is no longer statistically significant. Also, note that the condition

TABLE 5
Overall Results of the Models Without Control of Size (Univariate Models) and With Control of Size

Metric	Without Size Control					Controlling for Size						
	G (p-value)	H-L R^2	η_{LR}	Coeff. (p-value)	$\Delta\Psi$	G (p-value)	H-L R^2	η_{LR}	Coeff. (p-value)	$\Delta\Psi$	Size Coeff. (p-value)	Size $\Delta\Psi$
WMC	10.1 (0.0015)	0.043	2.24	0.0280 (0.0022)	1.74	13.83 (0.001)	0.059	7.63	-0.0113 (0.3073)	0.80	0.0141 (0.0289)	2.374
DIT	0.17 (0.6814)	0.0007	2.166	-0.1224 (0.3409)	0.938	--	--	--	--	--	--	--
CBO	4.16 (0.0414)	0.018	1.792	0.2804 (0.0266)	1.387	13.59 (0.0011)	0.0579	3.59	0.0209 (0.4495)	1.024	0.0105 (0.0030)	1.9
RFC	11.48 (0.0007)	0.049	2.41	0.0408 (0.0013)	1.84	16.51 (0.0003)	0.071	7.843	-0.0207 (0.2520)	0.73	0.0171 (0.0157)	2.80
LCOM	3.24 (0.0721)	0.014	4.011	0.0074 (0.0391)	1.33	13.91 (0.001)	0.059	4.682	0.0026 (0.2830)	1.107	0.0101 (0.0017)	1.86
NMA	12.83 (0.0003)	0.055	2.76	0.0690 (0.0007)	1.975	16.06 (0.0003)	0.069	7.47	-0.0025 (0.4783)	0.97	0.0127 (0.0402)	2.15
NPAVG	0 (0.975)	4.18e-6	1.75	0.0053 (0.4875)	1.004	--	--	--	--	--	--	--

The G value is the likelihood ratio test for the whole model. The "Coeff." columns give the estimated parameters from the logistic regression model. The "p-value" is the one-sided test of the null hypothesis for the coefficient. The R^2 values are based on the definition of R^2 provided by Hosmer and Lemeshow [61]. Hence, they are referred to as the H-L R^2 values. For the second half of the table, presenting the results of the model with size control, the coefficient for the size parameter is provided with its change in odds ratio. For the metrics where the model without size control is not significant (DIT and NPAVG), we do not present the model with size control since given the hypothesised confounding effect, the results will not be substantively different from the no size control model.

number does not indicate collinearity problems in this model.

4.3.4 RFC

The univariate results for RFC indicate that RFC is associated with fault-proneness, as one would expect given the current literature [19], [22], [10], [4], [106]. However, after controlling for size, Table 5 shows that the RFC coefficient is no longer statistically significant, the $\Delta^2\psi$ value is 152 percent, and the multivariate model is dominated by size. This indicates that there is a strong size confounding effect. The condition number for this model does not indicate collinearity problems.

4.3.5 LCOM

As will be recalled, the evidence for the validity of the LCOM metric from previous studies is equivocal. This is further evident here in Table 5, whereby the G statistic is not significant at an alpha level of 0.05, but it does approach it.

The multivariate LCOM model in Table 5 clearly indicates that, after controlling for size, there is no evidence that LCOM is associated with fault-proneness. Again, this model does not show signs of collinearity problems.

4.3.6 NMA

The univariate and multivariate results for the NMA metric present the same pattern, whereby at the univariate level this metric is highly significant, but after controlling for size its effect disappears, with a $\Delta^2\psi$ value of 103 percent.

4.3.7 NPAVG

The results for the NPAVG metric in Table 5 indicate that there is no association at all with fault-proneness and, in fact, this is the worst metric among those studied here.

4.4 Discussion of Results

The results that we obtained are remarkably consistent. They indicate that, for some of the product metrics that we studied, the univariate analyses demonstrate that they are indeed associated with fault-proneness (namely, WMC, CBO, RFC, and NMA). After controlling for the size confounder, all of these associations disappear, and the differences in the product metrics' odds ratios indicates a strong and classic confounding effect. The remaining product metrics were not associated with fault-proneness from a univariate analysis, but this was predictable from previous work.

This result has two important implications. First, it casts doubt on the validity claims made in the past about object-oriented metrics. It seems more plausible now that previous studies demonstrated an association with fault-proneness or faults because of the confounding effect of size, which by itself leads to greater fault-proneness or more faults. Second, it suggests caution in drawing validity conclusions from univariate analyses, as is currently a common practice, in validation studies of object-oriented product metrics.

To be clear, our results are not the last word on the validity of these metrics. Subsequent work with different data sets may demonstrate that they are valid *after* controlling for size. Reexamination of previous data sets may indicate that the metrics are still valid *after* controlling for size. What is clear is that researchers in this area, especially those who have demonstrated valid metrics, are urged to reexamine their data and determine the extent to which their conclusions would be contaminated after controlling for size. Similarly, future validation studies should always control for size.

Indeed, a careful examination of some previous studies indicates that there are potentially some object-oriented metrics that have an association with fault-proneness after controlling for size. For instance, Briand et al. construct two multivariate models in two studies [19], [22] using a forward selection procedure and allowing a large number of metrics to enter. The purpose was to build prediction models only and not to validate the metrics (recall that validation was demonstrated primarily through univariate results). When size metrics are allowed to enter, the result was a model with size metrics in both cases. Notwithstanding the inherent problems in using a forward selection procedure,[22] one of the models included the NMA metric [22], which had a significant impact on fault-proneness after controlling for size. This indicates that, out of the set of common object-oriented metrics, there is possibly a subset of metrics that are associated with fault-proneness for reasons other than their association with size.

More promising results were obtained after we reanalyzed the data of Cartwright and Shepperd [27]. They had collected early design metrics from Shlaer-Mellor models for a real-time telecommunications system with 32 C++ classes and data on the number of faults per class. We analyzed this data using multivariate least-squares models including LOC and each design metric in turn.[23] We found their design metrics to be strongly associated with the number of defects after controlling for size. Specifically, the DELS (count of all delete accesses by a class), ATTRIB (count of attributes per class), READS (count of all read accesses), WRITES (count of all write accesses), EVNTS (count of events per class in the state model), and STATES (count of states per class) all had significant associations with the number of defects at a one tailed alpha level of 0.0.5.[24] Of course, metrics such as READ and WRITE, and EVNTS and STATES are strongly correlated, but individually they seem to capture a structural property that has an impact on faults.

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 [49]. 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.

Even though our overall approach ought to compel caution in future validation studies, we do not claim that none of the studied object-oriented metrics are associated with fault-proneness. They were not associated in our study, but they may be in other studies. One thing is clear, however: Previous studies that did not account for the confounding effect of size overestimate the impact of the product metrics on fault-proneness.

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).

Even though we cast validation studies as case-control studies, we have been cautious in not making claims of causality. There are other potential confounders that would have to be controlled in order to gain any confidence that the product metrics are really causally affecting fault-proneness, for example, the competence of the designers and programmers.[25] Our argument has been to identify each of these variables and incrementally perform better validation studies. Eventually, once a core set of confounding variables has been identified, they ought to be controlled for in every validation study. The case-control framework provides a set of techniques for controlling the confounders in the design of the study through matching, or through various types of statistical adjustments.

5 Conclusions

The objective of the current paper was to examine if there is a confounding effect of class size in the validation of object-oriented metrics. The metrics set that we studied are the CK metrics [30] and a subset of the Lorenz and Kidd metrics [80]. Our focus in this paper was the prediction of class fault-proneness for faults found in the field.

To attain this objective, we first presented evidence from previous work indicating that there is a potential

[22] A Monte Carlo simulation of forward selection indicated that, in the presence of collinearity among the independent variables, the proportion of "noise" variables that are selected can reach as high as 74 percent [44]. It is clear that, in the object-oriented metrics, validation studies many of the metrics were correlated [19], [22]. Harrell and Lee [53] note that when statistical significance is the sole criterion for including a variable the number of variables selected is a function of the sample size and, therefore, tends to be unstable across studies. Furthermore, some general guidelines on the number of variables to consider in an automatic selection procedure given the number of "faulty classes" are provided in [54]. The studies that used automatic selection [19], [22] had 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.

[23] We used the condition number of Belsley et al. [5] as a diagnostic for detecting collinearity. We did not find any serious collinearity problems. The biggest condition number was for the STATES variable: 17.18. However, this is still smaller than the recommended threshold of 30.

[24] For this analysis, we did a careful evaluation of influential observations using Cook's distance [35], [36]. In general, we found two observations for each metric that can be considered influential. However, inclusion or removal of these observations, while changing the estimated parameters and goodness of fit, does not change our conclusions.

[25] We did perform an analysis that accounted for the potential confounding effect of programmer capability. For a subset of the 174 classes, we could reliably obtain the name of the primary developer. This subset consisted of 159 classes. To account for the potential confounding effect of individual capability, we included 13 dummy variables in each of the multivariate models to capture the effect of the 14 different developers (see [61] for a description of LR models with design variables), in addition to the product metric in question and the size metric. The estimated coefficients for the product metrics were slightly different from the values we present in this paper, but their statistical significance followed exactly the trends shown in our results. Therefore, accounting for the programmer confounder in addition to size would not change our conclusions.

TABLE 6
Contingency Table Showing the Notation for Potential Results of a Validation Study

		Fault Proneness	
		Faulty	Not Faulty
Coupling	HC	a	b
	LC	c	d

confounding effect of size in the product metrics to fault-proneness relationship. We then described and justified an empirical methodology for examining whether there is a confounding effect. Finally, we performed a study with a large C++ system to answer the confounding effect question.

When we performed our own empirical investigation, we controlled for the confounding effect of class size. Without controlling for class size, our results are consistent with what one would expect from the current literature, with most of the coefficients being statistically significant. When we controlled for class size, *all* of the effects disappeared, providing compelling evidence for a confounding effect of class size.

These results cast serious doubt on the empirical validity of object-oriented metrics since they suggest that previous empirical validity results were strongly positively biased by the confounding effect of size. We urge a reexamination of already validated object-oriented metrics after controlling for size. Our initial attempts at reanalyzing existing data (after controlling for size) indicate that some design metrics are indeed associated with faults after controlling for size. However, this ought to be performed on a broader scale.

Appendix A
The Odds Ratio

In this appendix, we provide a brief description of the odds ratio, which is a measure of association. We take it that the dependent variable is dichotomous, fault or no fault. First, we assume that the independent variable is also dichotomous, say High Coupling and Low Coupling. Second, we relax this and explain the odds ratio in the context of a logistic regression model and also the change in odds ratio. The latter is used extensively in this paper.

The odds of an event, such as High Coupling, is the ratio of the number of ways the event can occur to the number of ways the event cannot occur. Therefore, if the probability that a class has High Coupling is denoted by p, then the odds of a class having high coupling is $\frac{p}{1-p}$. Consider the contingency table (Table 6) that could have been obtained from a validation study.

The odds of a faulty class having High Coupling is:

$$\frac{a/a+c}{c/a+c} = \frac{a}{c}. \tag{5}$$

The odds of a not-faulty class having High Coupling is given by:

$$\frac{b/b+d}{d/b+d} = \frac{b}{d}. \tag{6}$$

The ratio of the odds that the faulty class has High Coupling to the odds that the not-faulty class has High Coupling is the odds ratio [47]:[26]

$$\psi = \frac{a/c}{b/d} = \frac{ad}{bc}. \tag{7}$$

If coupling is not related to fault-proneness, then the odds ratio is equal to one. If there is a positive association, then the odds ratio will be greater than one, and if there is a negative association, then the odds ratio will be a fraction.

Now, assume that we are performing a univariate logistic regression with one coupling metric. Let D denote the presence of a fault ($D = 1$) or absence ($D = 0$), and let x be our coupling metric. Then,

$$p_x = \Pr(D = 1|x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}} \tag{8}$$

is the probability of a fault given the value of x. The probability of there not being a fault given the value of x is:

$$\begin{aligned} q_x &= 1 - p_x = \\ \Pr(D = 0|x) &= 1 - \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}} = \frac{e^{-(\beta_0+\beta_1 x)}}{1 + e^{-(\beta_0+\beta_1 x)}}. \end{aligned} \tag{9}$$

One can then claim that out of every say 100 classes, $1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \times 100$ classes will have a fault. The odds of a class having a fault given the value of x is

$$\frac{p_x}{q_x} = \Psi(x) = e^{\beta_0+\beta_1 x)}. \tag{10}$$

The odds of a class having a fault if the product metric value x is increased by one standard deviation is:

$$\Psi(x + \sigma) = e^{\beta_0+\beta_1(x+\sigma))}. \tag{11}$$

The change in odds by increasing the value of x by one standard deviation is:

$$\Delta\Psi = \frac{\Psi(x + \sigma)}{\Psi(x)} = e^{\beta_1 \sigma}. \tag{12}$$

Appendix B
Definition of Metrics

The Table 7 gives the a definition for the metrics that appear in Fig. 3.

Appendix C
Data Analysis Details

This appendix presents the details of the data analysis that we performed. Specifically, it covers how we determine the extent of collinearity, identify influential observations, and perform hypothesis tests.

C.1 Diagnosing Collinearity

Since in our study we control for the size confounder through regression adjustment, careful attention should be

[26] This is the estimate of the population odds ratio.

TABLE 7
The Definitions for the Metrics in Fig. 3

Metric Acronym	**Definition**
CBO	This is defined in the main text of the paper.
CBO'	Same as CBO except that inheritance-based coupling is not counted [29].
RFC1	This is defined in the main text of the paper.
RFCoo	Same as RFC1 except that methods indirectly invoked by methods in M are not included in the response set [29].
MPC	The number of method invocations in a class [78].
ICP	The number of method invocations in a class, weighted by the number of parameters of the invoked methods [76].
IH-ICP	The same as ICP, but counts invocations of methods of ancestors of classes (i.e., inheritance-based coupling) only [76].
NIH-ICP	The same as ICP, but counts invocations to classes not related through inheritance [76].
DAC	The number of attributes in a class that have as their type another class [78].
DAC'	The number of different classes that are used as types of attributes in a class [78].
OCAIC	These coupling metrics are counts of interactions between classes. The metrics distinguish between the classes (friendship, inheritance, none), different types of interactions, and the locus of impact of the interaction [17]. 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)
FCAEC	
OCMIC	
OCMEC	
IFMMIC	
AMMIC	
OMMIC	
OMMEC	

paid to the detection and mitigation of potential collinearity. Previous studies have shown that outliers can induce collinearities in regression models [83], [98]. But also, it is known that collinearities may mask influential observations [5]. This has lead some authors to recommend addressing potential collinearity problems as a first step in the analysis [5] and this is what we do.

We first briefly review some common approaches for diagnosing collinearity in the context of ordinary least squares regression (henceforth, LS regression for short). This is important because the previously suggested diagnostic for validating object-oriented metrics [22] was initailly defined for the LS case. We then present some further diagnostics for the LR case.

One of the requirements for properly interpreting a LS regression model is that not one of the independent variables are perfectly linearly correlated to one or more other independent variables. This situation is referred to as perfect collinearity. When there is perfect collinearity then the regression surface is not even defined. However, perfect collinearity is rare and, therefore, one talks about the *degree* of collinearity. Recall that the confounder is associated with the product metric by definition and, therefore, one should not be surprised if strong collinearity exists.

The larger the collinearity, the greater the standard errors of the coefficient estimates. This means that t-statistics for significance testing tend to be small and confidence intervals tend to be wide [28], [92]. One implication of this is that the conclusions drawn about the relative impacts of the independent variables based on the regression coefficient estimates from the sample are less stable.

In the context of LR regression, Hosmer and Lemeshow [61] have illustrated the detrimental effects of collinearity among the independent variables and between the independent variables and the constant term. Specifically, the parameter standard errors tend to be grossly inflated. An empirical investigation with a large data set indicated large errors and variability in maximum-likelihood parameter estimates [97]. Schaefer [98] performed a Monte Carlo simulation showing that for two independent variables (which is our context) the impact of collinearity on the LR maximum likelihood estimates was severe but reduced as sample size increases. In some cases, the effect became almost negligible for sufficiently large sample sizes (n=200) [98]. Even though our sample size approaches this number, to err on the conservative side, we explicitly diagnose and deal with collinearity problems if they are detected.

Hosmer and Lemeshow [61] suggest that diagnostics originally devised for LS regression can also be applied for diagnosing collinearity for LR regression. Mennard [84] proposes using the *tolerance* statistic to detect collinearity. This is defined as $1 - R_j^2$, where the R_j^2 is the multiple coefficient of determination after regressing the $j\text{th}$ independent variable on all of the remaining independent variables [104]. He notes that a tolerance of less than 0.2 is cause for concern and a tolerance of less than 0.1 indicates a serious collinearity problem. This diagnostic is the reciprocal of the more common Variance Inflation Factor (VIF) [28], where a value greater than 10 is taken as signaling a collinearity problem. A common operationalization of the VIF is as follows: Assume that the linear regression model is given by:

$$\mathbf{y} = \widehat{\beta_0}\ \mathbf{?} + \hat{\beta}_{\mathbf{x}}\mathbf{Z}, \tag{13}$$

where $\mathbf{y}$ is the dependent variable vector, $\mathbf{?}$ is a vector of ones, $\hat{\beta}_{\mathbf{x}}$ are the LS parameter estimates, and $\mathbf{Z}$ is a $n \times k$ matrix of the x_{ij} raw data, with $i = 1 \dots n$ and $j = 1 \dots k$. If we transform each of the x_{ij} values using the correlation transform as follows [90]:

$$\frac{1}{\sqrt{n-1}}\left(\frac{x_{ij} - \bar{x}_j}{s_j}\right), \tag{14}$$

where s_j is the standard deviation of the $j\text{th}$ variable, then the correlation matrix is given by $\mathbf{Z}^T\mathbf{Z}$ and the VIFs are the diagonals of $\left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}$.

Belsley et al. [5] notes a number of important disadvantages of using the VIF as a collinearity diagnostic. Namely, the common way of computing the VIF as above does not account for collinearities involving the intercept, it cannot diagnose the number of near dependencies among the independent variables, and there are no guidelines for interpreting the VIF values that are not ad hoc.

Belsley et al. [5] propose the *condition number* as a collinearity diagnostic.[27] First, it is easier if we reformulate (13) as follows:

$$\mathbf{y} = \hat{\beta}\mathbf{X}. \tag{15}$$

Here, there is a column of ones in the $n \times (k+1)\mathbf{X}$ matrix to account for the fact that the intercept is included in the models. The condition number is defined as follows:

$$\eta = \frac{\lambda_{\max}}{\lambda_{\min}}, \tag{16}$$

where the λs are singular values obtained from the decomposition of the $\mathbf{X}$ matrix:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{G}^T, \tag{17}$$

where the λs are the nonnegative diagonal values of $\mathbf{D}$. Based on a number of experiments, Belsley et al. suggest that a condition number greater than 30 indicates mild to severe collinearity.

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

$$\eta = \sqrt{\frac{\mu_{\max}}{\mu_{\min}}}, \tag{18}$$

where $\mu_{\max} \geq \cdots \geq \mu_{\min}$ are the eigenvalues. 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 [7]: $\mathbf{X}_j / \|\mathbf{X}_j\|$.

Briand et al. [22] suggest using the condition number as the basis for diagnosing collinearity in multivariate LR models when validating object-oriented product metrics. They propose using the eigenvalues obtained from a principal components analysis in conjunction with (18). However, this approach suffers from the problem of ignoring the intercept[28] and its impact on near dependencies. It is known that collinearities can exist between the independent variables and the intercept, jointly[29] and individually [104]. Therefore, the Briand et al. approach can underestimate the extent of collinearity in a particular data set. For example, from our data set, the condition number computed from the results of a principal components analysis for a model with the variables WMC, RFC, CBO, and LOC is 6.94, whereas computing it directly using eigenvalues of the $\mathbf{X}^T\mathbf{X}$ or a singular value decomposition of the $\mathbf{X}$ matrix gives a condition number of 10.62. It is therefore prudent to compute the eigenvalues or singular values directly from $\mathbf{X}$ after the appropriate scaling.

Even though the condition number diagnostic was originally designated for LS regression, it has been applied in some instances for diagnosing collinearity in LR regression [105]. However, this diagnostic has been extended to the case of LR regression [42], [108] by capitalizing on the analogy between the independent variable cross-product matrix in LS 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 regression is [61]:

$$\hat{\mathbf{I}}(\hat{\beta}) = \mathbf{X}^T\hat{\mathbf{V}}\mathbf{X}, \tag{19}$$

where $\hat{\mathbf{V}}$ is the diagonal matrix consisting of $\hat{\pi}_{ii}(1 - \hat{\pi}_{ii})$, where $\hat{\pi}_{ii}$ is the probability estimate from the LR model for observation i. Note that the variance-covariance matrix of $\hat{\beta}$ is given by $\hat{\mathbf{I}}^{-1}(\hat{\beta})$. One can then compute the eigenvalues from the information matrix after column equilibrating and compute the condition number as in (18). The general approach for non-LS models is described by Belsley [6]. In

[27] 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.

[28] Recall that only the independent variables are used in a principal components analysis.

[29] Several columns of the data matrix become jointly, but not individually, collinear with the intercept, for example, when a linear combination of several nonconstant independent variables is nearly constant.

this case, the same interpretive guidelines as for the LS condition number are used [108].

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[30] to estimate the model parameters [97], [98]. However, since we do not detect severe collinearity in our study, we will not describe the ridge regression procedure here.

C.2 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 [61]. One first determines the log-likelihood for the model with the constant term only and denote this l_0 for the "null" model.[31] Then, the log-likelihood for the full model with the k parameters is determined and denoted as l_k.[32] The G statistic is given by $2(l_k - l_0)$, which has a χ^2 distribution with k degrees of freedom.[33]

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 LS regression [22], [19]. 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. Ideally, since the analogy to LS regression is the driving force behind such a statistic, it should vary between 1 and 0. However, Hosmer and Lemeshow [61] note that when there are groups in the data the maximum value will not be 1. A group occurs when there are multiple observations that have exactly the same values on the independent variables. In validation studies of product metrics, this frequently happens (for example, the data clustering in very few groups with only one independent variable) because the product metrics are counts (i.e., integers). To be specific, in our data set there were 48 groups for the WMC metric (recall that we have 174 observations), 40 groups for the RFC metric, 6 groups for the CBO metric, 39 groups for the LCOM metric, and 31 groups for the NMA metric.

Hosmer and Lemeshow [61] suggest an alternative formulation for R^2 that corrects for this and still allows for the reduction in uncertainty interpretation. 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.[34] In our study, we will use the corrected Hosmer and Lemeshow R^2 statistic as an indicator of the quality of the LR model.

C.3 Influence Analysis

Influence analysis is performed to identify influential observations (i.e., ones that have a large influence on the LR regression 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. This is similar to the approach advocated by Briand et al. [22] for identifying outliers in multivariate LR models used in validating object-oriented metrics. Specifically, they use the Mahalanobis distance from the centroid, removing each observation in turn before computing the centroid.

However, this approach has some deficiencies. First, it is important to note that an outlier is not necessarily an influential observation [96]. For example, an observation may be further away from the other observations but may be right on the regression surface. Furthermore, influential observations may be clustered in groups. For example, if there are say two independent variables in our model and there are five out of the n observations that have exactly the same values on these two variables, then these five observations are a group. Now, assume that this group is multivariately different from all of the other $n - 5$ observations. Which one of these five would be considered an outlier? A leave-one-out approach may not even identify any one of these observations as being different from the rest since in each run the remaining four observations will be included in computing the centroid. Such masking effects are demonstrated in [5].

Therefore, a more principled approach to the detection of influential observations is necessary. The first step is to identify the groups of observations with the same values on the independent variables. Similar to Cook's distance diagnostics [35], [36] that are commonly used in LS regression to identify influential observations, Pergibon has defined the $\Delta\beta$ diagnostic [93] to identify influential

[30] Ridge regression produces biased estimates of the parameters, but they are more robust in situations with high collinearity.

[31] This is defined as: $l_0 = \sum_{i=1}^{n}\{n_1\ln(n_1 + n_0\ln(n_0) - n\ln(n)\}$, where $n_1 = \sum y_i$ and $n_0 = \sum(1 - y_i)$.

[32] This is defined as: $l_k = \sum_{i=1}^{n}\{y_i\ln(\hat{\pi}(x_i)) + (1 - y_i)\ln(1 - \hat{\pi}(x_i))\}$.

[33] 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.

[34] Hosmer and Lemeshow [61] recommend against using the R^2 statistic stating that "R^2 is nothing more than an expression of the likelihood ratio test and, as such, is not a measure of goodness-of-fit. This likelihood ratio test compares fitted values under two models rather than comparing observed values to those fitted under one model." However, Menard [85] notes that the comparison of fitted values under two models is one way to interpret the traditional R^2 coefficient of determination in OLS regression. Therefore, if there is no conceptual objection to its use for OLS regression, there ought to be none for logistic regression. An alternative measure of goodness-of-fit was proposed by Maddala [82]. In an OLS context, this can be interpreted in terms of the geometric mean reduction in error per observation produced by the full model as opposed to the model with only the intercept [37]. Nagelkerke [88] notes that this statistic cannot have a value of one even when the model fits the data perfectly and proposed a correction for the maximum value. However, this coefficient measures a criterion that is not optimized in maximum-likelihood logistic regression and, therefore, the Hosmer and Lemeshow [88] is preferred. Furthermore, Menard [85], based on a conceptual review and an empirical demonstration, recommended the Hosmer and Lemeshow coefficient because it is the most independent of the base rate (in our case, this is the proportion of classes that are faulty), making it a more generally useful measure of goodness-of-fit.

groups in LR regression. The $\Delta\beta$ diagnostic is a standardized distance between the parameter estimates when a group is included and when it is not included in the model.

We use the $\Delta\beta$ diagnostic in our study to identify influential groups. For groups that are deemed influential, we investigate this to determine if we can identify substantive reasons for the differences. We found one observation that consistently had a large $\Delta\beta$ value. However, its removal does not change our conclusions.

Acknowledgments

The authors wish to thank Janice Singer, Hakan Erdogmus, Oliver Laitenberger, Morven Gentleman, James Miller, Barbara Kitchenham, and Anatol Kark for their comments on an earlier draft of this paper. The authors also wish to thank the anonymous reviewers whose suggestions have improved the paper.

References

[1] M. Almeida, H. Lounis, and W. Melo, "An Investigation on the Use of Machine Learned Models for Estimating Correction Costs," *Proc. 20th Int'l Conf. Software Eng.*, pp. 473-476, 1998.

[2] A. Baker, J. Bieman, N. Fenton, D. Gustafson, A. Mellon, and R. Whitty, "A Philosophy for Software Measurement," *J. Systems and Software*, vol. 12, pp. 277-281, 1990.

[3] V. Basili, S. Condon, K. El Emam, R. Hendrick, and W. Melo, "Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components," *Proc. 19th Int'l Conf. Software Eng.*, 1997.

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

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

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

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

[8] S. Benlarbi and W. Melo, "Polymorphism Measures for Early Risk Prediction," *Proc. 21st Int'l Conf. Software Eng.*, pp. 334-344, 1999.

[9] A. Binkley and S. Schach, "Prediction of Run-Time Failures Using Static Product Quality Metrics," *Software Quality J.*, vol. 7, pp. 141-147, 1998.

[10] A. Binkley and S. Schach, "Validation of the Coupling Dependency Metric as a Predictor of Run-Time Fauilures and Maintenance Measures," *Proc. 20th Int'l Conf. Software Eng.*, pp. 452-455, 1998.

[11] D. Boehm-Davis, R. Holt, and A. Schultz, "The Role of Program Structure in Software Maintenance," *Int'l J. Man-Machine Studies,* vol. 36, pp. 21-63, 1992.

[12] N. Breslow and N. Day, *Statistical Methods in Cancer Research—The Analysis of Case Control Studies,* vol. 1. Int'l Agency for Research on Cancer, 1980.

[13] L. Briand, W. Thomas, and C. Hetmanski, "Modeling and Managing Risk Early in Software Development," *Proc. Int'l Conf. Software Eng.*, pp. 55-65, 1993.

[14] L. Briand, V. Basili, and C. Hetmanski, "Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components," *IEEE Trans. Software Eng.*, vol. 19, no. 11, pp. 1028-1044, Nov. 1993.

[15] L. Briand, K. El Emam, and S. Morasca, "Theoretical and Empirical Validation of Software Product Measures," Technical Report ISERN-95-03, Int'l Software Eng. Research Network, 1995.

[16] L. Briand, J. Daly, and J. Wuest, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," *Empirical Software Eng.*, vol. 3, pp. 65-117, 1998.

[17] L. Briand, P. Devanbu, and W. Melo, "An Investigation into Coupling Measures for C++," *Proc. 19th Int'l Conf. Software Eng.*, 1997.

[18] L. Briand, J. Daly, V. Porter, and J. Wuest, "Predicting Fault-Prone Classes with Design Measures in Object Oriented Systems," *Proc. Int'l Symp. Software Reliability Eng.*, pp. 334-343, 1998.

[19] L. Briand, J. Wuest, S. Ikonomovski, and H. Lounis, "A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study," Technical Report ISERN-98-29, Int'l Software Eng. Research Network, 1998.

[20] L. Briand, J. Daly, and J. Wuest, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," *Empirical Software Eng.*, vol. 3, pp. 65-117, 1998.

[21] L. Briand, J. Daly, and J. Wuest, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Trans. Software Eng.*, vol. 25, no. 1 pp. 91-121, Jan. 1999.

[22] L. Briand, J. Wuest, J. Daly, and V. Porter, "Exploring the Relationships Between Design Measures and Software Quality in Object-Oriented Systems," *J. Systems and Software,* vol. 51, pp. 245-273, 2000.

[23] L. Briand, E. Arisholm, S. Counsell, F. Houdek, and P. Thevenod-Fosse, "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Direction," *Empirical Software Eng.*, vol. 4, no. 4, pp. 387-404, 1999.

[24] F. Brite e Abreu and R. Carapuca, "Object-Oriented Software Engineering: Measuring and Controlling the Development Process," *Proc. Fourth Int'l Conf. Software Quality,* 1994.

[25] F. Brito e Abreu and W. Melo, "Evaluating the Impact of Object-Oriented Design on Software Quality," *Proc. Third Int'l Software Metrics Symp.*, pp. 90-99, 1996.

[26] M. Cartwright, "An Empirical View of Inheritance," *Information and Software Technology,* vol. 40, pp. 795-799, 1998.

[27] M. Cartwright and M. Shepperd, "An Empirical Investigation of an Object-Oriented Software System," *IEEE Trans. Software Eng.*, vol. ?, no. ?, Mon? Year?

[28] S. Chaterjee and B. Price, *Regression Analysis by Example.* John Wiley and Sons, 1991.

[29] S. Chidamber and C. Kemerer, "Towards a Metrics Suite for Object Oriented Design," *Proc. Conf. Object Oriented Programming Systems, Languages and Applications (OOPSLA '91),* SIGPLAN Notices, vol. 26, no. 11, pp. 197-211, 1991.

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

[31] S. Chidamber and C. Kemerer, "Authors' Reply," *IEEE Trans. Software Eng.*, vol. 21, no. 3, p. 265, Mar. 1995.

[32] S. Chidamber, D. Darcy, and C. 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.

[33] N. Churcher and M. Shepperd, "Comments on 'A Metrics Suite for Object Oriented Design,'" *IEEE Trans. Software Eng.*, vol. 21, no. 3, pp. 263-265, Mar. 1995.

[34] F. Coallier, J. Mayrand, and B. Lague, "Risk Management in Software Product Procurement," *Elements of Software Process Assessment and Improvement,* K. El Emam and N.H. Madhavji, eds., 1999.

[35] R. Cook, "Detection of Influential Observations in Linear Regression," *Technometrics,* vol. 19, pp. 15-18, 1977.

[36] R. Cook, "Influential Observations in Linear Regression," *J. Amer. Statistical Assoc.*, vol. 74, pp. 169-174, 1979.

[37] D. Cox and N. Wermuth, "A Comment on the Coefficient of Determination for Binary Responses," *The Amer. Statistician,* vol. 46, pp. 1-4, 1992.

[38] L. Dales and H. Ury, "An Improper Use of Statistical Significance Testing in Studying Covariables," *Int'l J. Epidemiology,* vol. 7, no. 4, pp. 373-375, 1978.

[39] J. Daly, J. Miller, A. Brooks, M. Roper, and M. Wood, "Issues on the Object-Oriented Paradigm: A Questionnaire Survey," Research Report EFoCS-8-95, Dept. of Computer Science, Univ. of Strathclyde, 1995.

[40] J. Daly, M. Wood, A. Brooks, J. Miller, and M. Roper, "Structured Interviews on the Object-Oriented Paradigm," Research Report EFoCS-7-95, Dept. of Computer Science, Univ. of Strathclyde, 1995.

[41] J. Daly, A. Brooks, J. Miller, M. Roper, and M. Wood, "Evaluating Inheritance Depth on the Maintainability of Object-Oriented Software," *Empirical Software Eng.*, vol. 1, no. 2, pp. 109-132, 1996.

[42] C. Davies, J. Hyde, S. Bangdiwala, and J. Nelson, "An Example of Dependencies Among Variables in a Conditional Logistic Regression," *Modern Statistical Methods in Chronic Disease Edpidemiology,* S. Moolgavkar and R. Prentice, eds., John Wiley and Sons, 1986.

[43] I. Deligiannis and M. Shepperd, "A Review of Experimental Investigations into Object-Oriented Technology," *Proc. Fifth IEEE Workshop Empirical Studies of Software Maintenance,* pp. 6-10, 1999.

[44] S. Derksen and H. Keselman, "Backward, Forward and Stepwise Automated Subset Selection Algorithms: Frequency of Obtaining Authentic and Noise Variables," *British J. Mathematical and Statistical Psychology,* vol. 45, pp. 265-282, 1992.

[45] J. Dvorak, "Conceptual Entropy and Its Effect on Class Hierarchies," *Computer,* pp. 59-63, 1994.

[46] C. Ebert and T. Liedtke, "An Integrated Approach for Criticality Prediction," *Proc. Sixth Int'l Symp. Software Reliability Eng.,* pp. 14-23, 1995.

[47] B. Everitt, *The Analysis of Contingency Tables.* Chapman and Hall, 1992.

[48] N. Fenton, "Software Metrics: Theory, Tools and Validation," *Software Eng. J.,* pp. 65-78, Jan. 1990.

[49] N. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Trans. Software Eng.,* vol. 25, no. 5, pp. 676-689, May 1999.

[50] N. Fenton and M. Neil, "Software Metrics: Successes, Failures, and New Directions," *J. Systems and Software,* vol. 47, pp. 149-157, 1999.

[51] N. Fenton and N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," *IEEE Trans. Software Eng.,* vol. 26, no. 8, pp. 797-814, Aug. 2000.

[52] L. Gordis, *Epidemiology.* W.B. Saunders, 1996.

[53] F. Harrell and K. Lee, "Regression Modelling Strategies for Improven Prognostic Prediction," *Statistics in Medicine,* vol. 3, pp. 143-152, 1984.

[54] F. Harrell, K. Lee, and D. Mark, "Multivariate Prognostic Models: Issues in Developing Models, Evaluating Assumptions and Adequacy, and Measuring and Reducing Errors," *Statistics in Medicine,* vol. 15, pp. 361-387, 1996.

[55] R. Harrison, L. Samaraweera, M. Dobie, and P. Lewis, "An Evaluation of Code Metrics for Object-Oriented Programs," *Information and Software Technology,* vol. 38, pp. 443-450, 1996.

[56] W. Harrison, "Using Software Metrics to Allocate Testing Resources," *J. Management Information Systems,* vol. 4, no. 4, pp. 93-105, 1988.

[57] R. Harrison, S. Counsell, and R. Nithi, "Coupling Metrics for Object Oriented Design," *Proc. Fifth Int'l Symp. Software Metrics,* pp. 150-157, 1998.

[58] L. Hatton, "Is Modularization Always a Good Idea?" *Information and Software Technology,* vol. 38, pp. 719-721, 1996.

[59] L. Hatton, "Does OO Sync with How We Think?" *IEEE Software,* pp. 46-54, May/June 1998.

[60] B. Henderson-Sellers, *Software Metrics.* Prentice-Hall, 1996.

[61] D. Hosmer and S. Lemeshow, *Applied Logistic Regression.* John Wiley and Sons, 1989.

[62] J. Hudepohl, S. Aud, T. Khoshgoftaar, E. Allen, and J. Mayrand, " MERALD: Software Metrics and Models on the Desktop," *IEEE Software,* vol. 13, no. 5, pp. 56-60, 1996.

[63] J. Hudepohl, S. Aud, T. Khoshgoftaar, E. Allen, and J. Mayrand, "Integrating Metrics and Models for Software Risk Assessment," *Proc. Seventh Int'l Symp. Software Reliability Eng.,* pp. 93-98, 1996.

[64] ISO/IEC 9126, *Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for their Use.* Int'l Organization for Standardization and the Int'l Electrotechnical Commission, 1991.

[65] ISO/IEC DIS 14598-1, *Information Technology—Software Product Evaluation; Part 1: Overview.* Int'l Organization for Standardization and the Int'l Electrotechnical Commission, 1996.

[66] M. Jorgensen, "Experience with the Accuracy of Software Maintenance Task Effort Prediction Models," *IEEE Trans. Software Eng.,* vol. 21, no. 8, pp. 674-681, Aug. 1995.

[67] M. Kaaniche and K. Kanoun, "Reliability of a Commercial Telecommunications System," *Proc. Int'l Symp. Software Reliability Eng.,* pp. 207-212, 1996.

[68] J. Kearney, R. Sedlmeyer, W. Thompson, M. Gray, and M. Adler, "Software Complexity Measurement," *Comm. ACM,* vol. 29, no. 11, pp. 1044-1050, 1986.

[69] T. Khoshgoftaar, E. Allen, K. Kalaichelvan, and N. Goel, "The Impact of Software Evolution and Reuse on Software Quality," *Empirical Software Eng.: An Int'l J.,* vol. 1, pp. 31-44, 1996.

[70] T. Khoshgoftaar, E. Allen, R. Halstead, G. Trio, and R. Flass, "Process Measures for Predicting Software Quality," *Computer,* vol. 31, no. 4, pp. 66-72, Apr. 1998.

[71] T. Khoshgoftaar, E. Allen, W. Jones, and J. Hudepohl, "Which Software Modules Have Faults that will be Discovered by Customers?" *J. Software Maintenance: Research and Practice,* vol. 11, no. 1, pp. 1-18, 1999.

[72] T. Khoshgoftaar, E. Allen, W. Jones, and J. Hudepohl, "Classification Tree Models of Software Quality Over Multiple Releases," *Proc. Int'l Symp. Software Reliability Eng.,* pp. 116-125, 1999.

[73] B. Kitchenham, S.-L. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation," *IEEE Trans. Software Eng.,* vol. 21, no. 12, pp. 929-944, Dec. 1995.

[74] D. Kleinbaum, L. Kupper, and H. Morgenstern, *Epidemiologic Research: Principles and Quantitative Methods.* Van Nostrand Reinhold, 1982.

[75] F. Lanubile and G. Visaggio, "Evaluating Predictive Quality Models Derived from Software Measures: Lessons Learned," *J. Systems and Software,* vol. 38, pp. 225-234, 1997.

[76] Y. Lee, B. Liang, S. Wu, and F. Wang, "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow," *Proc. Int'l Conf. Software Quality,* 1995.

[77] M. Leijter, S. Meyers, and S. Reiss, "Support for Maintaining Object-Oriented Programs," *IEEE Trans. Software Eng.,* vol. 18, no. 12, pp. 1045-1052, Dec. 1992.

[78] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," *J. Systems and Software,* vol. 23, pp. 111-122, 1993.

[79] R. Lindsay and A. Ehrenberg, "The Design of Replicated Studies," *The Amer. Statistician,* vol. 47, no. 3, pp. 217-228, 1993.

[80] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics.* Prentice-Hall, 1994.

[81] M. Lyu, J. Yu, E. Keramides, and S. Dalal, "ARMOR: Analyzer for Reducing Module Operational Risk," *Proc. 25th Int'l Symp. Fault-Tolerant Computing,* pp. 137-142, 1995.

[82] G. Maddala, *Limited-Dependent and Qualitative Variables in Econometrics.* Cambridge Univ. Press, 1983.

[83] R. Mason and R. Gunst, "Outlier-Induced Collinearities," *Technometrics,* vol. 27, pp. 401-407, 1985.

[84] S. Menard, *Applied Logistic Regression Analysis.* Sage Publications, 1995.

[85] S. Menard, "Coefficients of Determination for Multiple Logistic Regression Analysis," *The Amer. Statistician,* vol. 54, no. 1, pp. 17-24, 2000.

[86] K.-H. Moller and D. Paulish, "An Empirical Investigation of Software Fault Distribution," *Proc. First Int'l Software Metrics Symp.,* pp. 82-90, 1993.

[87] J. Munson and T. Khoshgoftaar, "The Detection of Fault-Prone Programs," *IEEE Trans. Software Eng.,* vol. 18, no. 5, pp. 423-433, May 1992.

[88] N. Nagelkerke, "A Note on a General Definition of the Coefficient of Determination," *Biometrika,* vol. 78, no. 3, pp. 691-692, 1991.

[89] P. Nesi and T. Querci, "Effort Estimation and Prediction of Object-Oriented Systems," *J. Systems and Software,* vol. 42, pp. 89-102, 1998.

[90] J. Neter, W. Wasserman, and M. Kunter, *Applied Linear Statistical Models.* Irwin, 1990.

[91] 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.

[92] E. Pedhazur, *Multiple Regression in Behavioral Research.* Harcourt Brace Jovanovich, 1982.

[93] D. Pergibon, "Logistic Regression Diagnostics," *The Annals of Statistics,* vol. 9, no. 4, pp. 705-724, 1981.

[94] R. Pressman, *Software Engineering: A Practitioner's Approach.* McGraw Hill, 1997.

[95] K. Rothman and S. Greenland, *Modern Epidemiology.* Lippincott-Raven, 1998.

[96] P. Rousseeuw and A. Leroy, *Robust Regression and Outlier Detection.* John Wiley and Sons, 1987.

[97] R. Schaefer, L. Roi, and R. Wolfe, "A Ridge Logistic Estimator," *Comm. Statistics—Theory and Methods,* vol. 13, no. 1, pp. 99-113, 1984.

[98] R. Schaefer, "Alternative Estimators in Logistic Regression when the Data are Collinear," *The J. Statistical Computation and Simulation,* vol. 25, pp. 75-91, 1986.

[99] J. Schlesselman, *Case-Control Studies: Design, Conduct, Analysis.* Oxford Univ. Press 1982.

[100] D. Schmidt and P. Stephenson, "Experiences Using Design Patterns to Evolve System Software Across Diverse OS Platforms," *Proc. Nineth European Conf. Object Oriented Programming,* 1995.

[101] D. Schmidt, "A System of Reusable Design Patterns for Communication Software," *The Theory and Practice of Object Systems,* S. Berzuk ed., 1995.

[102] D. Schmidt, "Using Design Patterns to Develop Reusable Object-Oriented Communication Software," *Comm. ACM,* vol. 38, no. 10, pp. 65-74, 1995.

[103] S. Shlaer and S. Mellor, *Object-Oriented Systems Analysis: Modelling the World in Data.* Prentice-Hall, 1988.

[104] S. Simon and J. Lesage, "The Impact of Collinearity Involving the Intercept Term on the Numerical Accuracy of Regression," *Computer Science in Economics and Management,* vol. 1, pp. 137-152, 1988.

[105] K. Smith, M. Slattery, and T. French, "Collinear Nutrients and the Risk of Colon Cancer," *J. Clinical Epidemiology,* vol. 44, no. 7, pp. 715-723, 1991.

[106] 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.

[107] B. Unger and L. Prechelt, "The Impact of Inheritance Depth on Maintenance Tasks—Detailed Description and Evaluation of Two Experiment Replications," Technical Report 19/1998, Fakultat fur Informatik, Univ. Karlsruhe, 1998.

[108] Y. Wax, "Collinearity Diagnosis for Relative Risk Regression Analysis: An Application to Assessment of Diet-Cancer Relationship in Epidemiological Studies," *Statistics in Medicine,* vol. 11, pp. 1273-1287, 1992.

[109] S. Wiedenbeck, V. Ramalingam, S. Sarasamma, and C. Corritore, "A Comparison of the Comprehension of Object-Oriented and Procedural Programs by Novice Programmers," *Interacting with Computers,* vol. 11, no. 3, pp. 255-282, 1999.

[110] N. Wilde, P. Matthews, and R. Huitt, "Maintaining Object-Oriented Software," *Software,* pp. 75-80, Jan. 1993.

Khaled El Emam received the PhD degree from the Department of Electronic Engineering, King's College, the University of London (UK) in 1994. He is currently at the National Research Council in Ottawa. He is the editor of the *IEEE TCSE Software Process Newsletter*, the current International Trials Coordinator for the SPICE Trials, which is empirically evaluating the emerging ISO/IEC 15504 Int'l Standard world-wide, co-editor of ISO's project to develop an international standard defining the software measurement process, member of the CMMI product development team, focusing on empirical evaluation of the CMMI product set, and Knowledge Area specialist for the Software Engineering Process in the IEEE's project to define the Software Engineering Body of Knowledge. He is also an adjunct professor at the School of Computer Science at McGill University, and an adjunct professor at the Department of Computer Science at the University of Quebec in Montreal. Previously, he worked in both small and large software research and development projects for organizations such as Toshiba International Company, Yokogawa Electric, and Honeywell Control Systems. He was previously the head of the Quantitative Methods Group at the Fraunhofer Institute for Experimental Software Engineering in Germany, a research scientist at the Centre de recherche informatique de Montreal (CRIM), and a research assistant in the Software Engineering Laboratory at McGill University.

Saïda Benlarbi received the PhD degree in computer science from the University of Sherbrooke, and the MSc degree in computer science from Laval University, Quebec. She also holds an engineering degree in computer engineering and an associate degree in mathematics and physics, both from the Faculty of Sciences, University of Tunis, Tunisia. She is currently heading the Software Engineering Division of Cistel Technology Inc., where she leads the Research and Development activities. Previously, she worked as a research scientist at the Centre de Recherche Informatique de Montréal in the Software Engineering group. Her research interests include software architecture and software reuse, software dependability evaluation and prediction, statistical and adaptive prediction models for software measurement, definition and validation of software metrics, neural nets modeling, supervised and nonsupervised learning in multilayered neural nets.

Nishith Goel received the bachelor's degree in engineering from University of Jodhpur, Jodhpur, India in 1976, the MASc degree in electrical engineering and the PhD degree in systems design engineering from the University of Waterloo, Canada, in 1978 and 1983, respectively. He joined Nortel Networks in 1984 as a member of the scientific staff where he worked in various areas of semiconductor development, design, and manufacturing. In 1994, he joined the software engineering assurance lab (SEAL) at Nortel Networks, where he was engaged in the research and development of software engineering methods and tools for developing large software systems. He is currently president of the Cistel Technology Inc., a company he founded in 1995, providing research and consulting in the areas of software engineering and information technology.

Shesh N. Rai received the PhD degree from the University of Waterloo. He is currently a faculty member at the Department of Biostatistics and Epidemiology, St. Jude Children's Research Hospital, Memphis Tennessee, USA. He is also an adjunct professor at the Department of Biostatistics and Community Medicine, the University of Ottawa, Ottawa, Canada. He is on a leave from Statistics Canada, Ottawa, Canada. At Statistics Canada, he has developed procedures to identify vulnerable population (at health risk) and have applied to the National Population Health Survey data. He has also developed methods for the analysis of hierarchical models for complex survey data. Previously he worked as a postdoctoral fellow (NSERC) at Health Canada, Ottawa, Canada, where he developed methods for characterization and analysis of uncertainty and variability in risk models. The methods first present rigorous integrated approach to uncertainty analysis, distinguishing formally between uncertainty (lack of knowledge about the value of a particular parameter or variable) and variability (variation among individuals in the population of interest). The proposed methods also allow for the complex interrelationships between subsets of variables in multivariate models. He has conducted detailed uncertainty analysis on two important practical problems: estimating proportion of the lung cancer burden attributable to radon in homes and estimating cancer risks due to trace levels of chemical contaminants in drinking water. His work is published in *Human and Ecological Risk Assessment* and *Risk Analysis J.s*. In his PhD thesis at the University of Waterloo, he developed methods for the analysis of interval censored failure time data, such as that arising in occult tumour trials (results are published in *Biometrics, the Canadian Journal of Statistics, the Journal of the Royal Statistical Society*, and *Biometrical Journal*).