Empirical Studies of Quality Models in Object-Oriented Systems[1]

Lionel C. Briand
Software Quality Engineering Lab.
Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, K1S 5B6 Canada

Jürgen Wüst
Fraunhofer IESE
Sauerwiesen 6
67661 Kaiserslautern
Germany

Abstract

Measuring structural design properties of a software system, such as coupling, cohesion, or complexity, is a promising approach towards early quality assessments. To use such measurement effectively, quality models are needed that quantitatively describe how these internal structural properties relate to relevant external system qualities such as reliability or maintainability. This chapter has for objective to summarize, in a structured and detailed fashion, the empirical results that have been reported so far with modeling external system quality based on structural design properties in object-oriented systems. We perform a critical review of existing work in order to identify lessons learned regarding the way these studies are performed and reported. Constructive guidelines are also provided to facilitate the work of future studies, thus facilitating the development of an empirical body of knowledge.

Table of Contents

[1] To be published in Advances in Computers, Academic Press, edited by Marvin Zelkowitz

1 Introduction

As object-oriented programming languages and development methodologies moved forward, a significant research effort was also dedicated to defining specific quality measures and building quality models based on those measures. Quality measures of object-oriented code or design artifacts usually involve analyzing the structure of these artifacts with respect to the interdependencies of classes and components as well as their internal elements (e.g., inner classes, data members, methods). The underlying assumption is that such measures can be used as objective measures to predict various external quality aspects of the code or design artifacts, e.g., maintainability, reliability. Such prediction models can then be used to help decision-making during development. For example, we may want to predict the fault-proneness of components in order to focus validation and verification effort, thus finding more defects for the same amount of effort. Furthermore, as predictive measures of fault-proneness, we may want to consider the coupling, or level of dependency, between classes.

A large number of quality measures have been defined in the literature. Most of them are based on plausible assumptions but one key question is to determine whether they are actually useful, significant indicators of any relevant, external quality attribute. We also need to investigate how they can be applied in practice, whether they lead to cost-effective models in a specific application context. Though numerous empirical studies have been performed and reported in order to address the abovementioned questions, it is difficult to synthesize the current body of knowledge and identify future research directions. One of the main reasons is the large variety of measures investigated and the lack of consistency and rigor in the reporting of results.

This chapter has for objective to summarize, in a structured and detailed fashion, the results that have been reported so far. Overall, though not all the results are easy to interpret, there is enough consistency across studies to identify a number of strong conclusions. We also perform a critical review of existing work in order to identify lessons learned regarding the way these studies are performed and reported. Constructive guidelines are also provided to facilitate the work of future studies, thus facilitating the development of an empirical body of knowledge.

Section 2 summarizes existing studies and they main characteristics. Section 3 describes the most important principles and techniques regarding the analysis of software quality data and structural measurement. A recommended analysis procedure is also provided. Section 4 summarizes, in great detail, the results of the studies discussed in Section 2. These results are discussed and conclusions are provided in Section 5.

2 Overview of Existing Studies

This section presents a first overview of the existing studies relating OO design measurement and system quality, and highlights their commonalities and differences. A comparison of their results is performed in Section 4.

2.1 Classification of studies

Despite a large number of papers regarding the quality measurement of object-oriented systems, the number of articles that empirically investigate the relationship between design properties and various external quality attributes is relatively small. These studies fall into two categories:

1. Correlational Studies. These are studies which by means of univariate or multivariate regression analysis try to demonstrate a statistical relationship between one or more measures of systems´ structural properties (as independent variables) and an external system quality (as dependent variable).
2. Controlled experiments. These are studies that control for the structural properties of a set of systems (independent variables, mostly related to the use of the OO inheritance mechanism), and measure the performance of subjects undertaking software development tasks in order to demonstrate a causal relationship between the two. So far such studies have mostly been performed with students and have focused on the impact of inheritance on maintenance tasks.

Correlational studies are by far more numerous as they are usually the only option in industrial settings. Outside these two categories, published empirical work typically falls into two further categories:

3. Application of a set of design measures to one or more systems; with a discussion of the obtained distributions of the measures within one system, or a comparison of distributions across two or more systems, e.g., [AGE95], [Bar98], [CK94], [SC93]. For instance, [SC93] develop two versions of a brewery

control system to identical specifications, one following a data-driven approach [SM88], the other following a responsibility-driven approach [WWW90]. They apply the set of design measures by Chidamber and Kemerer [CK91] to the two resulting systems. They find the system resulting from the responsibility-driven approach to display more desirable structural properties. They conclude the responsibility-driven to be more effective for the production of maintainable, extensible and reusable software.
Conclusions in such studies of course are only supported when a relationship of the design measures used with the aforementioned system qualities is established. Considered in isolation, such studies are not suitable to demonstrate the usefulness of the structural measures, or draw conclusions from their measurement.

4. Apply a set of design measures to one or more systems and investigate relationships between these design measures, by investigating pairwise correlations and performing factor analysis (e.g., [AGE95], [CN00], [LC94]).

Besides empirical studies, the literature is concerned with the following topics:

5. Definition of new sets of measures (e.g., [BDM97], [BK95], [CK91], [CK94], [Hen96], [HM95], [LH93], [LK94], [LLWW95], [Li98])
6. Definition of measurement frameworks for one or more structural properties, which provide guidelines how these properties can, in principle, be measured [EKS94], [HM95], [BDW98], [BDW99].
7. Criticism/theoretical analysis of existing measures and measurement frameworks; in particular, there is an interest in defining, for measures of various structural properties, necessary mathematical properties these measures must possess in order for them to be valid measures of the properties they purport to measure [BMB96], [KPF95], [Whi97], [Zus98].

Our discussions in this article will focus on categories 1) and 2), with a strong emphasis on the former as these studies are by far the most numerous.

2.2 Measurement

In this section, we provide some examples of measures for object-oriented designs, to give the reader new to the field an impression of what measures of OO structural properties usually are about. We summarize here the measures by Chidamber and Kemerer ([CK94], in the following referred to as C&K). As we will see, these are the measures having received the widest attention in empirical studies and will be frequently mentioned in the subsequent Sections.

Chidamber and Kemerer define a suite of six measures (CBO, RFC, LCOM, DIT, NOC, WMC), to quantify the coupling, cohesion, inheritance relationships, and complexity of a class in an OO system:

- CBO (Coupling between Objects) - a count of the number of non- inheritance related couples with other classes. An object of a class is coupled to another, if methods of one class use methods or attributes of the other.
- RFC (Response for class) - RFC = |*RS*| where *RS* is the response set for the class. The response set can be expressed as $\{M\}\bigcup_{all\,i}\{R_i\}$, where $\{R_i\}$ is the set of methods called by method *i*, and *{M}* is the set of all methods in the class. The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class.
- LCOM (Lack of Cohesion in Methods) - Consider a Class C_1 with methods $M_1, M_2, ..., M_n$. Let $\{I_i\}$ = set of instance variables used by method M_i. There are *n* such sets $\{I_1\}, ..., \{I_n\}$. Let $P=\{(I_i,I_j) \mid I_i \bullet I_j=\emptyset\}$ and Q=$\{(I_i,I_j) \mid I_i \bullet I_j \bullet \emptyset\}$. If all *n* sets $\{I_1\}, ..., \{I_n\}$ are Ø, then let *P=Ø*.

$$LCOM = \begin{cases} |P|-|Q|, if\ |P|>|Q| \\ 0,\ otherwise \end{cases}$$

- DIT (depth in inheritance tree) - The depth of a class in the inheritance tree is the maximum length from the node to the root of the tree.
- NOC (Number of children) - the number of classes derived from a given class.
- WMC (Weighted Method Complexity) - Consider a class C_1, with methods $M_1, M_2, ..., M_n$. Let $c_1, c_2, ..., c_n$ be the complexity of the methods. Then:

$$WMC = \sum_{i=1}^{n} c_i$$

The complexities c_i were intentionally left undefined. Two versions of WMC were suggested and are frequently used:

- In [LH93] and [LHKS95], c_i is defined as McCabe's cyclomatic complexity of method M_i [McC76].
- In [BBM96], each c_i is set to one. In other words, this version of WMC counts the (non-inherited) methods of the class.

The Appendix provides short definitions for all measures mentioned in this chapter.

2.3 Survey of Studies

This section is divided into two distinct subsections. The first one presents correlational studies whereas the second one focuses on experiments. The studies are summarized in terms of their settings, dependent variable, independent variables, and analysis techniques.

2.3.1 Correlational Studies

Table 1 provides for each study a brief description of

- The external system quality of interest that was used as dependent variable of the study.
- The measures of structural properties used as independent variables of the study.
- A brief characterization of the system(s) from which the data sets were obtained (language, size, application domain). The systems are described once the first time they are reported and are denoted by acronyms in the remainder of the table.
- The types of analyses that were performed. Most importantly, this includes
  - univariate analysis (what modeling technique was used, if any),
  - multivariate analysis (what modeling technique was used, if any),
  - what kind of validation of the multivariate model was performed, if any.

Table 1: Overview of correlational studies

Reference	Dependent Variable	Independent Variable	Data Set	Univariate analysis	Multivariate Analysis	Model Evaluation
AM96	defect density, fault density, rework effort (system wide)	MOOD metrics	UMD: 8 C++ systems, university setting, from students, 4 to 15 KLOC	Pearson r	linear ordinary least- squares regression (LS)	R-square (R-sq);
BBM96	fault-proneness (faults from acceptance testing)	C&K [CK94]; Code Metrics Nesting Level, FunctDef, FunctCall	UMD, see [AM96]	Logistic Regression (LR)	Logistic regression (LR)	Contingency Table, Correctness/ Completeness
BWDP00	fault-proneness (acceptance testing)	~50 measures including C&K, C-FOOD [BDM97].	UMD, see [AM96]	LR	LR	R-sq, Correct ness/Completeness, 10- cross validation (CV)
BWL01	fault-proneness (field faults)	~50 measures including C&K, CFOOD.	LALO: C++ sys tem, commercial, 40 KLOC	LR	LR	R-sq, Correct ness/Completeness, 10- CV
BMW01	fault-proneness (field faults)	Suite of polymorphism measures [BM99], C&K, Part of C-FOOD	XPOSE: 144 classes, commercial	LR	LR, MARS	R-sq, Correct ness/Completeness, 10- CV, cross-sys tem validation, cost-benefit model
BW01	development effort	~50 measures including C&K, C-FOOD.	LIOO, university setting, public domain, 103 classes, 17 KLOC	negative binomial regression	Poisson Regression, Hybrid with regression trees	10-CV, ARE, MRE
BEDL99	expert opinion (EO): 'perceived complexity' 0- 100%	CACM (author's own), LCOM (2 versions - CK94, LH93	C++, 18 classes, from GUI pack ages	Spearman rho		

Reference	Dependent Variable	Independent Variable	Data Set	Univariate analysis	Multivariate Analysis	Model Evaluation
BEGR00	fault-proneness (from field faults)	C&K, without LCOM	see [EBGR01] and [EBGR99]	LR		R-Sq; LL Ratio Test to compare models
BM99	fault-proneness (from field faults)	Suite of polymorphism measures by authors, C&K, Part of C-FOOD	LALO, see [BWL01]	LR	LR	goodness of fit (chi-square)
BS96?	EO: agreement with experts' preference between two design alternatives	CDM, CBO, DAC, NCC, NSSR, efferent/afferent coupling	8 system specs, 2-3 design alternatives each, 20 designs in total	ad hoc, count agreement with expert judgment (no statistical testing)		
BS98	? "maintenance data", number of faults (from field);	CDM, C&K (with out LCOM), NCC,NSSR, CHNL, NCIM	C++ System (patient care mgmnt), 113 cls., 82KLOC; file transfer facility, 29 Java cls., 6KLOC;	Spearman rho		
BWL99	likelihood of ripple changes	CBO, part of C-Food, 10 more coupling measures	LALO, see [BWL01]	LR	ranking-based model	
CDK98	productivity (Size/ effort), rework effort, design effort	C&K, LOC, dichotomized versions of CBO and LCOM	3 banking IS: C++, 15KLOC, 45 cls. Objective C, 3KLOC, 27cls. Design docs, 25 cls.		linear LS	R-square
CL93	EO: subjective summative scale across 9 items (understandability., maintainability., etc)	author's own (operation/ attribute complexity, coupling, cohesion, inheritance)	12 system designs developed by 6 students to 2 identical specs		linear LS	
CS00	#changes due to defects from integration tests/field usage	DIT, NOC, counts of elements in Shlaer/Mellor design; LOC from code	telecom RT sub system, C++, 32 cls., 130KLOC	Spearman rho linear LS	linear LS	R-square; contingency table
EBD99	EO: 'perceived complexity' 0- 100%	author's own: AMC, CDE (class design entropy); C&K's WMC	C++, 17 classes from GUI pack ages	Spearman rho		
EDW98	EO: 'perceived class cohesion' 0- 100%	LCOM (2 versions: [CK94, LH93])	see BEDL99	LS		R-square
EBGR01	fault-proneness (from field faults)	C&K; Mthds, NMO,SIX, avg. #parameters, LOC	C++, commercial telecom frame work, 174 classes	LR		R-square
EBML00	fault-proneness (from field faults)	Stmts, Mthds, Attrs;	see [EBGR01], [EMM01], plus commercial telecom app, C++, 85 classes,	LR (Threshold models)		
EBGR99	fault-proneness (from field faults)	C&K, C-Food	see EBML00	LR	LR	R-square, leave-one-out CV, receiver-operator curve (ROC)
EMM01	fault-proneness (from field faults)	part of C-Food, DIT, NOC, Attrs	v0.5 and v0.6 of Jwriter (comm. text processor), 69 and 42 Java cls. (no inner cls.)	LR	LR	R-square, leave-one-out CV, ROC; between-version validation (v0.5 fit data, v0.6 test data)
GEMM00	fault-proneness (from field faults)	DIT, NOC, Mthds, part of C-Food	Xpose, 145 Java cls.;	LR (model including con founding variables).	LR	R-square; leave-one-out CV, ROC curve, Cost Savings Model
LH93, LHKS95	#lines changed over 3 years (maintenance effort surrogate)	C&K - CBO, MPC, DAC, 2xSize (Stmts, Mth+Attr)	2 Ada Systems (UIMS, QUES), commercial, 39/ 71 cls.		linear LS	R-square; between-system CV

Reference	Dependent Variable	Independent Variable	Data Set	Univariate analysis	Multivariate Analysis	Model Evaluation

HC98,
HCN98
	#faults (from testing), fault density, EO: Subjective complexity (understandability) on a 1-5 scale	DIT, NOC, NMO, NMI, CBO, NAS	GNU C++ library (197 cls.);
			LEDA lib. (97 cls.);
			3 sets of student systems, 113, 172, 317 cls. resp; syst. in [HSDL96]
				Spearman rho		

HN96	EO: Subjective complexity [HC98]	C&K, LOC	C++ system, 13 cls., 500LOC, poss. Comm.	Spearman, Kendall, Pear son's r		
HSDL96	EO: Subjective complexity (HC98); #faults in testing, #modification requests; time to modify	LOC, lib/non-lib functions called, depth in call graph, #function declarations /definitions	C++ system, 109 functions, 2.5KLOC. Image processing SW developed by one person.	Spearman, Kendall, Pear son's r for all pairs DVxIV		
MHM99	effort (project- wide)	DV: Dome System Meter (based on a special notation for business models, elements of size, export coupling), Function Points	37 IS projects, 1.5-100 man months, 1-10 developers, C++, Smalltalk, 4GL	quadratic LS		cross validation (fit/test data set of 24/ 13 projects.), compare MREs of FPA and System Meter
MT98	effort (project- wide)	#classes, Mthds, LOC	7 projects from a small SW company; C++, 15- 135 cls.; 3-22 person months dev.	Pearson's r, linear and exponential LS		R-squared
NQ98	effort (class level)	set of 50 measures, mostly size/ complexity of class methods/ attributes	3 C++ systems: Case Tool, GUI lib, control-SW; total 524 cls., 47KLOC		linear LS	R-sq., between-system validation using 4th system (LIOO)
RL92	EO: Ranking of classes by 'perceived complexity'	4 Coupling measures (author's own)	5 C++ systems, 12-70 cls., 15- 160KLOC	Pearson's r		
TKC99	fault-proneness (from system tests and maintenance phase)	C&K without LCOM, LOC, author's own (inheritance based coupling, memory allocation)	3 C++ sub systems from a telecom application; 6 KLOC/20 cls., 21 KLOC/45 cls., 6 KLOC/27 cls.	LR, (separate for each mea sure, system, type of fault). LR-R-sq.		
WH98	effort for field fault fixes; effort for functional enhancements	C&K measures; various interpretations for WMC (McCabe, Hal stead)	conferencing system, 114 C++ cls., 25KLOC	linear LS	linear LS	R-squared
WK00	#ripple changes a class participates in, proportion ripple changes/ changes in a class	CBO, #public methods, #methods	conferencing system, 114 C++ cls., 25KLOC	Kruskal Wallis, Kendall's tau		

Without delving into the details of the studies in Table 1, we can draw a number of observations:

- **Choice of dependent variable.** The dependent variables investigated are either fault-proneness (probability of fault detection) or the number of faults or changes in a class, effort for various development activities, or expert opinion/judgment about the psychological complexity of a class.
  - Fault-proneness or the number of defects detected in a class is the most frequently investigated dependent variable. Sources of faults are either from unit/system testing or field failures.
    This choice of dependent variable is by far the most common in the literature. One reason is that using fault-proneness as a dependent variable is an indirect way of looking at reliability, which is an important external quality to consider. Another explanation is that the collection of fault data (including the assignment of faults to classes) is less difficult than collecting other data related to classes (e.g., effort) and this makes it a convenient choice to investigate the impact of structural measures on the cognitive complexity of classes.
  - Less frequently investigated is effort for various activities: either total development effort, rework effort/functional enhancements, effort to fix faults. For some studies, effort for individual classes was collected, which is, in practice, is a difficult undertaking. Other studies collected system/project wide

    effort, which is easier to account for but leads to other practical problems. If systems become the unit of analysis then it becomes difficult to obtain enough data to perform multivariate analysis.
    - Two studies [BWL99], [WH98] used the likelihood or number of ripple effects to other classes when a change is performed to a class. The goal was to provide a model to support impact analysis. These studies are not described in the remainder of this chapter as they are the only one of their kind and more studies are needed to confirm the observed trends.
    - In the absence of hard quality data obtained from development projects, subjective data are sometimes used. For instance, the following have been used: expert opinion about the perceived complexity or cohesion of a class, preference ranking between design alternatives.
      There are a number of problems associated with the use of subjective measurement. Determining what constitute an "expert" is one. Moreover, it is a priori unclear to which degree experts' judgment correlates with any external system quality attribute. Eliciting expert opinion is a difficult undertaking and must be carefully planned to provide meaningful results and the procedures used must be properly reported. Though this is outside of the scope of this article, some abundant literature exists on the subject [MB91].
      An interesting question that, to our knowledge, has not been investigated in depth to date is whether structural measures can perform as well as or better than experts in predicting quality attributes such as fault-proneness.
- **Choice of independent variables.** Existing measures receive a varying amount of attention in the empirical studies. The measures by Chidamber and Kemerer [CK94] were investigated the most. One reason is that this was one of the first publications on the measurement of object-oriented systems. The relative difficulty of collecting more complex measures through static analyzers may also be an explanation. Last the profusion of papers proposing new measures, using a different terminology and formalism, has made any selection of meaningful measures a difficult undertaking. Some recently published measurement frameworks [BDW98, BDW99] may help choose appropriate measures based on their properties. A careful selection of measures, based on a clear rationale, is indispensable to maintain the complexity of the data analysis within reasonable limits and lower the chances of finding significant relationships by chance [Mil81]. However, in the early stage of investigation, it is common for studies to investigate large numbers of alternatives, as they tend to be exploratory.
- **Building prediction models**. Only about half of the studies employ some kind of multivariate analysis in order to build an accurate prediction model for the dependent variable. The remaining studies only investigate the impact of individual measures on system quality, but not their combined impact. Depending on the measurement scale of the dependent variable, different regression techniques are being used. Furthermore, a number of detailed technical issues regarding the data analysis can be observed and are discussed in Section 3. One noticeable pattern is the number of studies that only investigate linear relationships between structural measures and the dependent variable of interest. Though there is no rationale to support this, data sets are often not large enough to investigate non-linear relationships or interactions. In addition, because of the lack of supporting theory, it is often difficult to know what to search for. Exploratory techniques, such as regression trees or MARS [Fri91], have been used in some studies to determine non-linearities and variable interactions, with some degree of success [BMW01][BW01].
- **Evaluating prediction models**. From the studies that perform multivariate analysis, only half of these studies perform some kind of cross validation [Sto74], where the prediction performance of the multivariate prediction model in a relevant application context is investigated. The other studies only provide a measure of the goodness of fit of the prediction model (e.g., R-square). As a consequence, the potential benefits of using such prediction models are not always clear, especially from a practical standpoint. Very few studies attempts to build a model on a system and apply it to another one, within one environment. As studies move away from exploration and investigate the practical applications of measurement-based models, cross-system predictions will require more attention. One practical difficulty is to obtain consistent data from different projects of comparable nature.
- **Data sets.** Data sets with fault or effort data at the class level are rare. As a consequence, these data sets tend to be repeatedly used for various studies, for example investigating different sets of measures, or using different modeling techniques. On the one hand, this allows for better comparison between studies but it is also detrimental to building an increased body of knowledge, as replication of individual studies in many different contexts rarely take place. Instead, we find a large number of different studies using a small number of data sets.

2.3.2 Experiments

Table 2 provides an overview of the controlled experiments investigating the relationship between structural design properties and system quality in object-oriented systems. For each study, we state the literature source, a characterization of the dependent and independent variables investigated, the systems used for the experiment and the participants involved. The rightmost column indicates what experimental design was employed and the analysis techniques used to test the research hypotheses. For an introduction to experimental designs, see e.g., [Spe81].

Table 2: Overview of controlled experiments

Referenc e	Dependent variable	Independent variables	Systems/Subjects	Exp. Design/ An. Technique
Bar98	reusability (subjective perception of)	C&K, LOC, Methods, Attributes, meaningfulness of variable names (subjective measure)	2 systems, 3 and 2 classes, one designed to be reusable, the other not.	ad hoc comparison
BBDD97	understandability, correctness, completeness, modification rate of Impact Analysis	procedural vs. OO design; adherence to common principles of good design	2x2 systems (30pages reqs&design); 13 student subjects, 2 runs	2x2 fact. Design; ANOVA, paired t- test
BBD97	understandability, correctness, completeness, modification rate of Impact Analysis	adherence to common principles of good design	2x2 systems (30pages reqs&design); 31 student subjects, 2 runs	2x2 fact. Design; ANOVA, paired t- test
HCN00	maintainability, understandability	DIT (flat vs deep inheritance structure)	4 versions of a university admin IS system, 2x0, 1x3, 1x5 levels of inheritance; 4x12 student subjects	4x12 between subject; chi square to compare groups
LC92	understandability, modifiability, “debugability” (time, correctness, completeness to perform these tasks)	flat vs deep inheritance structure	2 groups (5&6 students)	within subject, 2 groups, three diff. tasks
WDMR99	maintainability (time for maintenance task)	flat vs deep inheritance structure	3x2 systems, C++, 400-500 LOC; 31 student subjects, 3 runs	blocked design, 1 internal rep.; Wilcoxon sign rank and rank sum test

Controlled experiments are far fewer in number than correlational studies. The studies mostly investigate aspects of system understandability and maintainability as dependent variable, and usage of inheritance as the independent variable. Also, we see that the controlled experiments are usually performed in a university setting with student subjects.

The qualitative results of these experiments will be summarized in Section 4.2.

2.4 Discussion

From the tables above, we can see there is a large number of studies that have already been reported. The great majority of them are correlational studies. One of the reasons is that it is difficult to perform controlled experiments in industrial settings. Moreover, preparing the material for such experiments (e.g., alternative, functional designs) is usually costly. With correlational studies, actual systems and design documents can be used.

Another important observation is that the analysis procedures that are followed in across the correlational studies vary a great deal. To some extent, some variation is to be expected, as alternative analysis procedures are possible. But many of the studies are actually not optimal in terms of the techniques being used. For instance, [AM96] overfits the data and performs a great number of statistical tests without using appropriate techniques for repeated testing. We therefore need to facilitate the comparison of studies, to ensure that the data analysis is complete and properly reported. Only then it will be possible to build upon every study and develop a body of knowledge that will allow us to determine how to use structural measurement to build quality models of object-oriented software. Section 3 provides a detailed procedure that has been first used (with minor differences) in a number of articles [BWDP00, BWL01, BMW01, BW01]. Such a procedure will make the results of a study more interpretable—and thus easier to compare—and the analysis more likely to obtain accurate prediction models.

3 Data Analysis Methodology

Recall that our focus here is to explain the relationships between structural measures of object-oriented designs and external quality measures of interest. In this section, because we focus on data analysis procedures and multivariate modeling, we will refer to these measures as independent and dependent variables, respectively. For the sake of brevity they will be denoted as IVs and DVs.

Our goal here is not to paraphrase books on quantitative methods and statistics but rather to clearly make the mapping between the problems we face and the techniques that exist. We also provide clear, practical justifications for the techniques we suggest should be used.

On a high level, the procedure we have used [BWDP00, BWL01, BMW01, BW01] consists of the following steps.

1. Descriptive statistics [Hay94]
   Analysis of the frequency distributions of the IVs. This will help to explain some of the results observed in subsequent steps and is also crucial to explain differences across studies.
2. Principal component analysis (PCA) [Dun89]
   In the investigation of measures of structural properties, it is common to have much collinearity between measures capturing similar underlying phenomena. PCA is a standard technique to determine the dimensions captured by our IVs. PCA will help us better interpret the meaning of our results in subsequent steps
3. Univariate analysis
   Univariate regression analysis looks at the relationships between each of the IVs and the DV under study. This is a first step to identify types of IVs that are significantly related to the DV and thus are *potential* predictors to be used in the next step.
4. Prediction model construction (multivariate analysis)
   Multivariate analysis also looks at the relationships between IVs and the DV, but considers the former in combination, as covariates in a multivariate model, in order to better explain the variance of the DV and ultimately obtain accurate predictions. To measure the prediction accuracy, different modeling techniques (e.g., OLS [Lew80], logistic regression, Poisson regression [Lon97]) have specific measures of goodness of fit of the model.
5. Prediction model evaluation
   In order to get an estimate of the predictive power of the multivariate prediction models that is more realistic than goodness-of-fit, we need to apply models to data sets other than those from which they were derived. A set of procedures known as cross-validation [Sto74] should be carried out. Typically, such a procedure consists in dividing the data set into V pieces and use them in turn as test data sets, using the remainder of the data set to fit the model. This is referred to as V-cross validation and allows the analyst to get a realistic accuracy prediction even when a data set of limited size is available. Based on the results of the cross-validation, the benefit of using the model in a usage scenario should then be demonstrated.

The above procedure is aimed at making studies and future replications repeatable and comparable across different environments. In the following, we describe and motivate each step in more detail.

3.1 Descriptive statistics

Within each case study, the distribution (mean, median, and interquartile ranges) and variance (standard deviation) of each measure is examined. Low variance measures do not differentiate classes very well and therefore are not likely to be useful predictors. The range and distribution of a measure determines the applicability of subsequent regression analysis techniques. Analyzing and presenting the distribution of measures is important for the comparison of different case studies[2]. It allows us to determine if the data collected across studies stem from similar populations. If not, this information will likely be helpful to explain different findings across studies.

Also, this analysis will identify measures with potential outlying values, which will be important in the subsequent regression analyses. Univariate and multivariate outlier analysis are discussed in their respective sections.

3.2 Principal component analysis

It is common to see groups of variables in a data set that are strongly correlated. These variables are likely to capture the same underlying property of the object to be measured. Principal component analysis (PCA) is a standard technique to identify the underlying, orthogonal dimensions (which correspond to properties that are directly or indirectly measured) that explain relations between the variables in the data set. For example, analyzing a data set using PCA may lead to the conclusions that all your measures come down to measuring some aspect of class size and import coupling.

Principal components (PCs) are linear combinations of the standardized IVs. The sum of the square of the weights in each linear combination is equal to one. PCs are calculated as follows. The first PC is the linear combination of all standardized variables that explain a maximum amount of variance in the data set. The second and subsequent PCs are linear combinations of all standardized variables, where each new PC is orthogonal to all previously calculated PCs and captures a maximum variance under these conditions. Usually, only a subset of all variables shows large weights and therefore contributes significantly to the variance of each PC. To better identify these variables, the *loadings* of the variables in a given PC can be considered. The loading of a variable is its correlation with the PC. The variables with high loadings help identify the dimension the PC is capturing but this usually requires some degree of interpretation. In other words, one assigns a meaning or property to a PC based on the variables that show a high loading. For example, one may decide that a particular PC mostly seems to capture the size of a class.

In order to further ease interpretation of the PCs, we consider the *rotated components*. This is a technique where the PCs are subjected to an orthogonal rotation in the sample space. As a result, the resulting principal components (referred to as rotated components) show a clearer pattern of loadings, where the variables either have a very low or high loading, thus showing either a negligible or a significant impact on the PC. There exist several strategies to perform such a rotation, the *varimax* rotation, being the most frequently used strategy in the literature.

For a set of *n* measures there are, at most, *n* orthogonal PCs, which are calculated in decreasing order of variance they explain in the data set. Associated with each PC is its eigenvalue, which is a measure of the variance of the PC. Usually, only a subset of the PCs is selected for further analysis (interpretation, rotated components, etc.). A typical stopping rule that we also use in our studies is that only PCs whose eigenvalue is larger than 1.0 are selected. See [Dun89] for more details on PCA and rotated components.

We do not consider the PCs themselves for use as independent variables in the prediction model. Although this is often done with ordinary least- square (OLS) regression, in the context of logistic regression, this has shown to result in models with a sub-optimal goodness of fit (when compared to models built using the measures directly), and is not current practice. In addition, principal components are always specific to the particular data set on which they have been computed, and may not be representative of other data sets. A model built using principal components is not likely to be applicable across different systems.

Still, it is interesting to interpret the results from regression analyses (see next sections) in the light of the results from PCA, e.g., analyze from which PCs the measures that are found significant stem from. This shows which dimensions are the main drivers of fault-proneness, and may help explain why this is the case.

[2] Note that one strong conclusion that comes from our experience analyzing data and building models is that we will only be able to draw credible conclusions regarding what design measures to use if we are able to replicate studies across a large number of environments and compare their results.

Regarding replicated studies, it is interesting to see which dimensions are also observable from PCA results in other systems, and find possible explanations for differences in the results, e.g., a different design methodology. We would expect to see consistent trends across systems for the strong PCs that explain a large percentage of the data set variance, and can be readily interpreted. From such observations, we can also derive recommendations regarding which measures appear to be redundant and need not be collected, without losing a significant amount of design information.

As an example of an application of PCA, and the types of conclusions we can draw from it, Table 3 shows the rotated components obtained from cohesion measures applied to the system in [BW01]. The measures mostly capture two orthogonal dimensions (the rotated components PC1 and PC2) in the sample space formed by all measures. Those two dimensions capture 81.5% of the variance in the data set. Analyzing the definitions of the measures with high loadings in PC 1 and PC2 yields the following interpretations of the cohesion dimensions:

- *PC1*: Measures LCOM5, COH, CO, LCC, TCC are all *normalized* cohesion measures, i.e., measures that have a notion of maximum cohesion.
- *PC2*: Measures LCOM1-LCOM3, and ICH are non-normalized cohesion measures, which have no upper bound.

Table 3: Rotated components for cohesion measures (from [BW01])

	PC1	PC2
EigenValue:	4.440	3.711
Percent:	44.398	37.108
CumPercent:	44.398	81.506
LCOM1	0.084	**0.980**
LCOM2	0.041	**0.983**
LCOM3	-0.218	**0.929**
LCOM4	-0.604	0.224
LCOM5	**-0.878**	0.057
Coh	**0.872**	-0.113
Co	**0.820**	0.139
LCC	**0.869**	0.320
TCC	**0.945**	0.132
ICH	0.148	**0.927**

As discussed in [BDW98], many of the cohesion measures are based on similar ideas and principles. Differences in the definitions are often intended to improve of shortcomings of other measures (e.g., behavior of the measure in some pathological cases). The results show that these variations, based on careful theoretical consideration, do not make a substantial difference in practice. By and large, the measures investigated here either capture normalized or non-normalized cohesion, measures of the latter category having been shown to be related to the size of the class in past studies ([BWDP00, BWL01]).

3.3 Univariate Regression analysis

Univariate regression is performed for each individual IV against the DV, in order to determine if the measure is a *potentially* useful predictor. Univariate regression analysis is conducted for two purposes:

- to test the hypotheses that the IVs have a significant statistical relationship with the DV.
- to screen out measures that are not significantly related to the DV and not likely to be significant predictors in multivariate models. Only measures that are significant at significance level, say α=0.25 [HL89], should be considered for the subsequent multivariate analysis.

Note that some IV may be significantly related to the DV for various reasons. It may capture a causal relationship or be the result of a confounding effect with another IV. Because of the repeated testing that is taking place during univariate analysis, there is a non-negligible chance to obtain a spurious relationship obtained by chance. Though a number of techniques exist to deal with repeated testing (e.g., Bonferroni [Mil81]), this is not an issue here as we are not trying to demonstrate or provide evidence for a causal relationship. Our goal is to pre-select a number of potential predictors for multivariate analysis, which will tell us in turn which IVs seem to be useful predictors. Causality cannot really be demonstrated in this context and only a careful definition of the design measures used as IVs, along with plausible mechanisms to explain causality, can be provided.

The choice of modeling technique for univariate analysis (and also the multivariate analysis that follows) is mostly driven by the nature of the DV: its distribution, measurement scale, whether it is continuous or discrete. Examples from the literature include:

- Logistic regression to predict the likelihood for an event to occur, e.g., fault detection [BWDP00, GEMM00].
- Ordinary least-squares regression, often combined with monotonic transformation (logarithmic, quadratic) of the IVs and/or DV, to predict interval/ratio scale DVs [BW01, CS00].
- Negative binomial regression (of which poisson regression is a special case) to predict discrete DVs that have low averages and whose distribution is skewed to the right [BW01].
- Parametric and non-parametric measures of correlation (Spearman Rho, Pearson r) are sometimes used. But they can only provide a rough picture and are not as practical as they do not account for non-linearities and are not comparable to the multivariate modeling techniques we present below.

3.3.1 Univariate Outliers

Outliers are data points that are located in an empty part of the sample space [BP95]. Inclusion or exclusion of outliers can have a large influence on the analysis results. It is important that conclusions drawn are not solely dependent on a few outlying observations, otherwise, the resulting prediction models are unstable and cannot be reliably used. When comparing results across replicated studies, it is particularly crucial to ensure that differences in observed trends are not due to singular, outlying data points. For this reason it is necessary to identify outliers, test their influence, and possibly remove them to obtain stable results.

For univariate analysis, all observations must be checked for outlying values in the distribution of any one of the measures used in the study. The influence of the identified observation is tested: an outlier is *influential*, if the significance of the relationship between the measure and the DV depends on the absence or presence of the outlier. Such influential outliers should not be considered in the univariate analysis results. Outliers may be detected from scatterplots, and their influence systematically tested.

For many regression techniques, specific diagnostics to automatically identify outliers were proposed, e.g., Cooks Distance for OLS [BP95], Pregibon beta for logistic regression [Pre81].

3.4 Prediction Model Construction

Multivariate regression is performed to build prediction models of the DV. This analysis is conducted to determine how well we can predict the DV, when the design measures are used in combination. For the selection of measures to be used in the model, a strategy must be employed that

- Select an appropriate number of independent variables in the model. Overfitting a model increases the standard error of the model’s prediction, making the model more dependent on the data set it is based on and thus less generalizable.
- Reduces multicollinearity [BKW80], i.e., independent variables which are highly correlated. High multicollinearity results in large standard errors for regression coefficient estimates and may affect the predictive power of the model. It also makes the estimate of the impact of one IV on the DV difficult to derive from the model.

3.4.1 Stepwise selection process

Often, the validation studies described here are exploratory in nature, that is, we do not have a strong theory that tells us which variables should be included in the prediction model and which not. In this situation, a stepwise selection process can be used, where prediction models are built in a stepwise manner, where each step consists of one variable entering or leaving the model.

The two major stepwise selection processes used for regression model fitting are forward selection and backward elimination [HL89]. The general forward selection procedure starts with a model that includes the intercept only. Based on certain statistical criteria, variables are selected one at a time for inclusion in the model, until a stopping criterion is fulfilled. Similarly, the general backward elimination procedure starts with a model that includes all independent variables. Variables are selected one at a time to be deleted from the model, until a stopping criterion is fulfilled.

When investigating a large number of independent variables, the initial model in a backward selection process would contain too many variables and could not be interpreted in a meaningful way. In that case, we use a stepwise forward selection procedure to build the prediction models. In each step, all variables not already in the model are tested: the most significant variable is selected for inclusion in the model. If this causes a variable already in the model to become not significant (at some level of significance α_{Exit}), it is deleted from the model. The process stops when adding the best variable no longer improves the model significantly (at some significance level $\alpha_{Enter}< \alpha_{Exit}$).

A commonly used procedure to reduce the number of independent variables to make possible the use of a backward selection process is by pre- selecting variables using the results from principal component analysis: the highest loading variables for each principal component is selected and then the backward selection process run on this reduced set of variables. In our studies [BWDP00, BWL00], within the context of logistic regression, this strategy showed the goodness of fit of the models thus obtained to be poorer than the models obtained from the forward stepwise procedure, hence favoring the use of latter.

The choice of significance levels for measures to enter and exit the model is an indirect means to control the number of variables in the final model. A rule of thumb for the number of covariates is to have at least ten data points per independent variable.

Criticism of stepwise selection heuristics

Stepwise selection procedures have been criticized for a number of reason: (1) the inclusion of noise variables in the presence of multicollinearity - clearly an issue with our design measures, (2) the number of variables that are selected is a function of the sample size and is often too large. This casts doubt on the trustworthiness of a model built in such a fashion. In [EBGR01], the authors state that "variables selected through such a procedure cannot not be construed as the best object-oriented metrics, nor even as good predictors of the DV". However, many IVs can typically be replaced other related IVs (i.e., confounded measures, belonging to the same principal component in PCA) without a significant loss of fit. In addition, our studies show that trends between design measures and system quality frequently vary across systems [BMW01, BWL01], and a prediction model built from one system is likely to be representative of only a small number of systems developed in the same environment. The particular measures selected for a model are not of much *general* importance. Therefore, the goal of building multivariate models using stepwise heuristics is not to determine what are the "best" metrics or whether they are the only or best predictors. The most we can hope for is that the properties / dimensions (i.e., principal components) captured by the measures are relevant, frequently represented in the predictive models, and can explain most of the variance in the DV. In short, our aim here is only to obtain an optimal predictive model, as defined in Section 3.5.

Stepwise variable selection is a standard technique frequently used in the literature. It is certainly true that the output from such a stepwise selection heuristic cannot be blindly relied upon. It is necessary to perform a number of sanity checks on the resulting model: (1) the number of covariates is reasonable considering the size of the data set, (2) the degree of collinearity among covariates is acceptable [BKW80], (3) no outlier is overinfluential with respect to the selection of covariates. If violations of these principles are detected, they can be amended by adjusting (1) inclusion/ exclusion thresholds, (2) removing covariates or (3) dismissing data points. We think the results obtained from a model that passes these checks, and also performs reasonably well in the subsequent model evaluation (see Section 3.5), are trustworthy at least in that they indicate the order of magnitude of the benefits that we can expect to achieve from a prediction model built in the same fashion in any given environment.

3.4.2 Capturing non-linear or local trends and interactions

When analyzing and modeling the relationship between IVs and DVs, one of the main issues is that relationships between these variables can be complex (non-linear) and involve interaction effects (the effect of one variable depends on the value of one of more other variables). Because we currently know little about what to expect and because such relationships are also expected to vary from one organization and family of systems to another, identifying non-linear relationships and interaction effects is usually a rather complex, exploratory process.

Data mining techniques such CART Regression tree analysis [BFOS84, BW01] makes no functional form assumption about the relationship of IVs and DV. In addition, the tree construction process naturally explores interaction effects. Another recent technique, MARS (Multivariate Adaptive Regression Splines) [Fri91], attempts to approximate complex relationships by a series of linear regressions on different intervals of the independent variable ranges and automatically searches for interactions. Both techniques can be combined with traditional regression modeling [BW01].

Hybrid Models with Regression Trees

Adapting some of the recommendations in [SC99], traditional regression analysis and regression trees can be combined into a hybrid model as follows:

- Run regression trees analysis, with some restrictions on the minimum number of observations in each terminal nodes (in order to ensure that samples will be large enough for the next steps to be useful).
- Add dummy variables (binary) to the data set by assigning observations to terminal nodes in the regression trees, i.e., assign 1 to the dummy variable for observations falling in its corresponding terminal node. There are as many dummy variables as terminal nodes in the tree.
- Together with the IVs based on design measures, the dummy variables can be used as additional covariates in the stepwise regression.

This procedure takes advantage of the modeling power of regression analysis while still using the specific interaction structures that regression trees can uncover and model. As shown in [BW01], such properties may significantly improve the predictive power of multivariate models.

Multivariate Adaptive Regression Splines (MARS)

As previously discussed, building quality models based on structural, design measures is an exploratory process. MARS is a novel statistical method that has shown to be useful in helping the specification of appropriate regression models in an exploratory context. This technique is presented in [Fri91] and is supported by a recent tool developed by Salford Systems[3]. At a high level, MARS attempts to approximate complex relationships by a series of linear regressions on different *intervals* of the independent variable ranges (i.e., subregions of the independent variable space). It is very flexible as it can adapt any functional form and is thus suitable to exploratory data analysis. Search algorithms find the appropriate intervals on which to run independent linear regressions, for each independent variable, and identify interactions while avoiding overfitting the data. Though these algorithms are complex and out of the scope of this paper, MARS is based on a number of simple principles. MARS identifies optimal *basis functions* based on the IVs and these basis functions are then used as candidate covariates to be included in the regression model. When we are building, for example, a classification model (such as a fault-proneness model), we use MARS in two steps: (1) Use the MARS algorithms to identify relevant basis functions, (2) Refit the model with logistic regression, using the basis functions as covariates [BW01]. Our experience has shown that MARS was helpful in building more accurate predictive models [BW01, BMW01].

3.4.3 Multivariate outliers

Just as univariate analysis results are susceptible to univariate outliers, multivariate models can be strongly influenced by the absence or presence of individual observations. Our set of *n* independent variables spans an *n*-dimensional sample space. To identify multivariate outliers in this sample-space, we calculate, for each data point, the Mahalanobis Jackknife [Eve93] distance from the sample space centroid. The Mahalanobis Distance is a measure that takes correlations between measures into account. Multivariate outliers are data points with a large distance from the sample space centroid. Again, a multivariate outlier may be over-influential and therefore removed, if the significance of any of the *n* variables in the model depends on the absence or presence of the outlier. A subtle point here occurs when dismissing an outlier causes one or more covariates in the model resulting from a stepwise selection heuristic to become insignificant. In that case, our strategy is to rerun the stepwise selection heuristic from scratch excluding the outlier from the beginning. More detailed information on outlier analysis can be found in [BP95].

[3] www.salford-systems.com

3.4.4 Test for multicollinearity

Multivariate models should be tested for multicollinearity. In severe cases, multicollinearity results in inflated standard errors for the estimated coefficients, which renders predicted values of the model unreliable. The presence of multicollinearity also makes the interpretation of the model difficult, as the impact of individual covariates on the dependent variable can no longer be judged independently from other covariates.

According to [HL89], tests for multicollinearity used in least-squares regression are also applicable in the context of logistic regression. They recommend the test suggested by Belsley et al. [BKW80], which is based on the conditional number of the correlation matrix of the covariates in the model. This conditional number can conveniently be defined in terms of the eigenvalues of principal components as introduced in Section 3.2.

Let X_1,..., X_n be the covariates of our model. We perform a principal component analysis on these variables, and set l_{max} to be the largest eigenvalue, l_{min} the smallest eigenvalue of the principal components. The conditional number is then defined as $\lambda = \sqrt{l_{\max} / l_{\min}}$ A large conditional number (i.e., discrepancy between minimum and maximum eigenvalue) indicates the presence of multicollinearity. A series of experiments showed that the degree of multicollinearity is harmful, and corrective actions should be taken, when the conditional number exceeds 30 [BKW80].

3.4.5 Evaluating goodness of fit

The purpose of building multivariate models is to predict the DV as accurately as possible. Different regression techniques provide specific measures for a model's *goodness of fit*, for instance R-squared for OLS or methods based on maximum likelihood estimation such as logistic regression. While these allow, to some degree, for comparison of accuracy between studies, such measures are abstract mathematical artifacts that do not illustrate very well the potential benefits of using the prediction model for decision-making.

We provide below a quick summary of goodness of fit measures that users of prediction models tend to use to evaluate the practicality of using a model. There are two main cases that have to be dealt with in practice: (1) classification (such as classifying components as fault-prone or not), (2) predicting a continuous DV on an interval or ratio scale. We will use an example of each category to illustrate practical measures of goodness of fit.

Classifying Fault-proneness

To evaluate the model's goodness of fit, we can apply the prediction model to the classes of our data set from which we derived the model[4]. A class is classified fault-prone, if its predicted probability to contain a fault is higher than a certain threshold p_0. Assume we use this prediction to select classes to undergo inspection. Further assume that inspections are 100% effective, i.e., all faults in a class are found during inspection.

We then compare the predicted fault-proneness of classes to their actual fault-proneness. We then use the following measures of the goodness of fit of the prediction model:

- *Completeness:*
  Completeness, in this context, is defined as the number of faults in classes classified as fault-prone, divided by the total number of faults in the system. It is a measure of the percentage of faults that would have been found if we used the prediction model to drive inspections. Low completeness indicates that, despite the use of the classification model, many faults are not detected. These faults would then slip to subsequent development phases, where they are more expensive to correct.
  We can always increase the completeness of our prediction model by lowering the threshold p_0 used to classify classes as fault-prone ($\pi > p_0$). This causes more classes to be classified as fault-prone, thus completeness increases. However, the number of classes that are incorrectly being classified as fault-prone also increases. It is therefore important to consider the *correctness* of the prediction model.
- *Correctness:*
  Correctness is the number of classes correctly classified as fault- prone, divided by the total number of classes classified as fault-prone. Low correctness means that a high percentage of the classes being classified as fault-prone do not actually contain a fault. We want correctness to be high, as inspections of classes that do not contain faults is an inefficient use of resources.

[4] This is, of course, an optimistic way to assess a model. This is why the term *goodness-of-fit* is used, as opposed to *predictive power*. This issue will be addressed in Section 3.5.

These definitions of completeness and correctness have straightforward, practical interpretations. They can be used to other application contexts where a classification models is required. A drawback of these measures is that they depend on a particular classification threshold. The choice of threshold is system-dependent and, to a large degree, arbitrary. To achieve comparability between studies and models, we can, however, employ a consistent strategy for threshold selection, such as using prior probability (proportion) of fault-prone classes, or selecting threshold p_0 so as to balance the number of actually faulty and predicted fault-prone classes. Plotting the correctness and completeness curves as a function of the selected threshold p_0 is also a good, common practice [BWDP00], as shown in Figure 1.

As an example, we show here the fault-proneness classification results from a model ("Linear" logistic regression Model) built in [BMW01]:

Table 4: Fault-proneness classificationi results (Linear Model in [BMW01])

		Predicted		Σ
		π<0.5	π>0.5	
Actual	No fault	108	5	113
	Fault	17 (50 faults)	14 (82 faults)	31 (132 faults)
Σ		125	19	144

The model identifies 19 out of 144 classes as fault-prone (i.e., 13% of all classes). Of these, 14 actually are faulty (74% correctness), and contain 82 out of 132 faults (62% completeness).

The above figures are based on a cutoff value of π=0.5 for predicted fault- prone/not fault-prone classes, and the table only gives a partial picture, as other cut-off values are possible. Figure 1 shows the correctness and completeness numbers (vertical axis) as a function of the threshold π (horizontal axis).

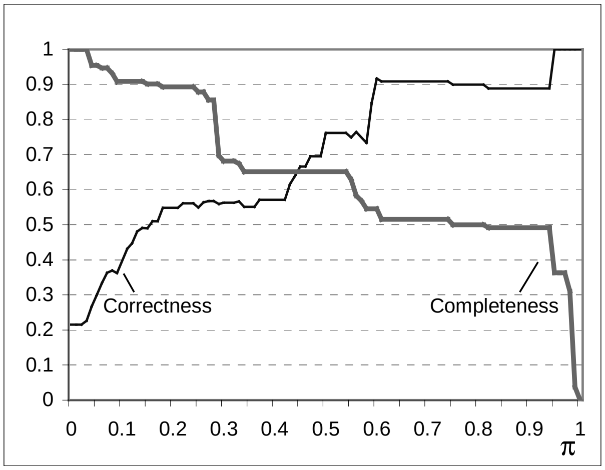

Figure 1: Correctness/completeness graph (for "Linear model" in [BMW01])

Standard measures of the goodness of fit used in the context of logistic regression models are sensitivity, specificity, and the area under the receiver-operator-curve (ROC) [GS74]. Sensitivity is the fraction of observed positive outcome cases that are correctly classified (i.e., the fraction of faulty classes correctly classified fault-prone, which similar to completeness as defined above). Specificity is the fraction of observed negative outcomes cases correctly classified (i.e., the fraction of non- faulty classes correctly classified not fault-prone). Calculating sensitivity and specificity too requires a selection of a particular threshold *p*. The receiver-operator curve is a graph of sensitivity versus 1-specificity as the threshold *p* is varied. The area under the ROC is a common measure of the goodness of fit of the model - a large area under the ROC indicates that high values for both sensitivity and specificity can be achieved. The advantage of the area under the ROC is that this measure does not necessitate the selection of a particular threshold. The drawback is that its interpretation (i.e., probability that a randomly selected faulty class has a higher predicted fault-proneness than a randomly selected not faulty class) has no immediate interpretation in the context of a practical application of the model.

Predicting development effort

We use here development effort as an example for the prediction of continuous, interval/ratio scale DVs. In the area of effort estimation, the most commonly used measures of prediction accuracy are the absolute relative error (ARE) and the magnitude of relative error (MRE) of the effort prediction. If eff is the actual effort (e.g., for a class or system), and $\hat{eff}$ the predicted effort. Then,

$ARE = | eff - \hat{eff} |$, and $MRE = | eff - \hat{eff} | / eff /$

The percentage (or absolute value in terms of person hours) that a predicted effort is on average off is immediately apparent to a practitioner and can be used to decide whether the model can be of any practical help. ARE and MRE measures can readily be used in other contexts than effort estimation.

3.4.6 The impact of design size

The size of an artifact (e.g., class designs) is a necessary part of any model predicting a property (e.g., fault proneness) of this artifact. This is mostly justified by the fact that size determines, to some extent, many of its external properties such as fault-proneness or effort. On the one hand, we want our predictive models to account for size. But in many cases, e.g., in the case of fault-proneness models, and for practical reasons, we need them to capture *more* than size effects. Using again our inspection example, a model that systematically identifies bigger classes as more fault-prone would a priori be less useful: the predicted fault-prone classes are likely to cover a larger part of the system and the model could not help focus inspection and testing efforts very well.

In our studies [BWDP00, BWL01, BW01], we compare models built from (1) size measures only and (2) models allowing all measures (size, coupling, cohesion, inheritance) to enter the model. With these models, we seek to find answers to the following questions:

- Are coupling, cohesion, and inheritance (CCI) measures complementary predictors of the DV as compared to size measures alone?
- How much more accurate is a model including the more difficult to collect coupling, cohesion, and inheritance measures[5]? If it is not significantly better, then the additional effort of calculating these more expensive measures instead of some easily collected size measures would not be justified.

When the DV is class fault-proneness and the measures are collected based on design information, the results [BWL01, BMW01] so far have shown that

- Models including CCI measures clearly outperform models based on size measures only. Even though they may be related to size, CCI measures therefore capture information related to fault-proneness that cannot be explained by size alone.
- There is no significant difference between models based on CCI measures only and models based on both CCI and size measures. This indicates that all size aspects that have a bearing on fault-proneness are also accounted for by the set of CCI measures investigated. In other words, the CCI measures are not just complementary to the size measures, they subsume them.

When the DV is effort [BW01], however, it appears that the size accounts for most of the variation in effort, and more sophisticated CCI measures do not help to substantially improve the model's predictive capability.

In the model building strategy proposed by El Emam et al [EBGR01], a size measure is forced in the predictive model by default. Measures that are confounded by size are not considered for inclusion in the model. This is an alternative strategy and which one to use depends on your purpose. If you want to build an optimal prediction model and determine what are useful predictors, then the procedure we outlined above is fine. If your goal is to demonstrate that a given measure is related to fault-proneness, or any other DV, and that this relationship cannot be explained by size effects, then the procedure in [EBGR01] is appropriate.

[5] Such measures usually require the use of complex static analyzers

3.5 Prediction model evaluation

We discussed above the notion of goodness-of-fit and practical ways to measure it. However, though such measures are useful to compare models built on a given data set, they present two limitations

- They are optimistic since we have to expect the model's predictive accuracy to deteriorate when it is applied to different data sets than the one it is built on.
- They still do not provide information that can be used directly to assess whether a model can be useful under given circumstances.

Those two issues are addressed by the next two subsections.

3.5.1 Cross validation

One of the commonly encountered problems in software engineering is that our data sets are usually of limited size, i.e., a few hundred observations when we are lucky. Dividing the available data into a modeling set and a test set is usually difficult as it implies that either the test set is going to be too small to obtain representative and reliable results or the modeling set is going to be too small to build a refined predictive model. One reasonable compromise is to use a *cross-validation* procedure. To get an impression of how well the model performs when applied to different data sets, i.e., its prediction accuracy, a cross-validation should be carried out.

Depending on the availability and size of the data set, various cross-validation techniques can be used

- V-cross validation [Sto74] is what we used in our studies [BWDP00, BW01, BMW01]. For the V-cross validation, the *n* data points of each data set are randomly split into V partitions of roughly equal size (n/V). For each partition, we re-fit the model using all data points *not* included in the partition, and then apply the resulting model to the data points in the partition. We thus obtain for all *n* data points a predicted probability of their fault-proneness (or predicted development effort).
- Leave-one-out cross validation, a special case of V-cross validation, where V=n-1, used for very small data sets.
- For larger data sets, one can randomly partition the data set in a fit/ modeling data partition (usually 2/3 of all observations) used to fit the model, and a test data partition (all remaining observations).

The ideal situation is where separate data sets, derived from different systems stemming from similar environments, are available. The prediction model is built from one system that is used in turn to make prediction for the other system. This is the most effective demonstration of the practical use of a prediction model. Typically, models are built on past systems and used to predict properties of new systems or their components. System factors may affect the predictive power of a model and, therefore, it is important to validate the model under conditions that resembles as closely as possible its usage conditions. [BMW01] report such a study where the authors introduce a cost-effectiveness model for fault-proneness models. This is described further in the next section.

3.5.2 Cost-benefit model for class fault-proneness prediction

Goodness-of-fit or predictive power does not give the potential users of a model a direct means to assess whether the model can be practically useful to them. We need to develop cost-benefit models that are based on realistic assumptions and that use parameters that can be either measured or estimated. Though it is difficult to further specify general rules to build such models in our context, we will use an example to illustrate the principles to follow: How can we determine whether a fault-proneness model would be economically viable if used to drive inspections? The first step is to identify all the parameters that the model will be based on. At the same time, list all the assumptions on which the model will be based regarding these parameters. Such assumptions are usually necessary to help simplify the cost-benefit model. Some of these assumptions will inevitably be specific to an environment and can very well be unrealistic in others. What we present here is based on a study reported in [BMW01]:

- All classes predicted fault-prone are inspected.
- Usually, an inspection does not find all faults in a class. We assume an average inspection effectiveness *e*, $0 \le e \le 1$, where $e = 1$ means that all faults in inspected classes are being detected.

- Faults that are not discovered during inspection (faults that slipped through, faults in classes not inspected) later cause costs for isolating and fixing them. The average cost of a fault when not detected during inspection is *fc*.
- The cost of inspecting a class is assumed to be proportional to the size of the class.

In general, in order to estimate the benefits of a model, we need a comparison baseline that represents what could be achieved without the use of the model. In our example, we assume a simple model that ranks the classes by their size, and selects the *n* largest classes for inspection. The number *n* is chosen so that the total size the selected classes is roughly the same as the total size of classes selected by fault-proneness model based on design (size and CCI) measures. It is thus ensured that we compare models where the investment – the cost of inspections – are the same or similar and can be factored out.

For the specification of the model, we need some additional definitions. Let $c_1,....,c_N$ denote the *N* classes in the system. For $i=1,…,N,$ let

- f_i be the number of actual faults in class i.
- p_i indicates if class i is predicted fault-prone by the model, i.e., $p_i=1$ if class i is predicted fault-prone, $p_i=0$ otherwise
- s_i denotes the size of class i (measured in terms of the number of methods, though other measures of size are possible).

The inspection cost is $ic \cdot s_i$, where *ic* is the cost of inspection of one size unit.

The next step is to quantify the gains and losses due to using the model. In our example, they are all expressed below in effort units, i.e., the effort saved and the effort incurred assuming inspections are performed on code.

Gain (effort saved):

g_m =defects covered and found
$g\_m = e \cdot fc \cdot \Sigma_i(f_i \cdot p_i)$

Cost (effort incurred):

c_m = direct inspection cost + defects not covered + defects that escape
$c\_m = ic \cdot \Sigma_i(s_i \cdot p_i) + fc \cdot \Sigma_i(f_i \cdot (1-p_i)) + (1-e) \cdot fc \cdot \Sigma_i(f_i \cdot p_i)$

In the same way, we express the cost and gains of using the size ranking model (baseline) to select the *n* largest classes, so that their cumulative size is equal or close to $\Sigma_i(s_i \cdot p_i)$, the size of classes selected by the predictive model[6]. For $i=1,..,N$, let $p'_i=1$ if class *i* is among those *n* largest classes, and $p'_i=0$ otherwise:

$g\_s = e \cdot fc \cdot \Sigma_i(f_i \cdot p'_i)$

$c\_s = ic \cdot \Sigma_i(s_i \cdot p'_i) + fc \cdot \Sigma_i(f_i \cdot (1-p'_i)) + (1-e) \cdot fc \cdot \Sigma_i(f_i \cdot p'_i)$

We now want to assess the difference in cost and gain when using the fault-proneness model and size-ranking model, which is our comparison baseline:

$\Delta gain = g\_m – g\_s = e \cdot fc \cdot (\Sigma_i(f_i \cdot p_i) - \Sigma_i(f_i \cdot p'_i))$

$\Delta cost = c\_m - c\_s = ic \cdot (\Sigma_i(s_i \cdot p_i)-\Sigma_i(s_i \cdot p'_i)) + fc(\Sigma_i(f_i \cdot (1-p_i)) - \Sigma_i(f_i \cdot (1-p'_i))) + (1-e) \cdot fc(\Sigma_i(f_i \cdot p_i) - \Sigma_i(f_i \cdot p'_i))$

We select *n* and therefore *p'* so that $\Sigma_i(s_i \cdot p_i)-\Sigma_i(s_i \cdot p'_i) \approx 0$ (Inspected classes have roughly equal size in both situations). We can thus, as an approximation, drop the first term from the $\Delta cost$ equation. This also eliminates the inspection cost *ic* from the equation, and with it the need to make assumptions about the ratio *fc* to *ic* for calculating values of $\Delta cost$. With this simplification, we have

$\Delta cost = fc \cdot (\Sigma_i(f_i \cdot (1-p_i))- \Sigma_i(f_i \cdot (1-p'_i))) + (1-e) \cdot fc \cdot (\Sigma_i(f_i \cdot p_i) - \Sigma_i(f_i \cdot p'_i))$

By doing the multiplications and adding summands it is easily shown that

$\Delta cost = - e \cdot fc \cdot (\Sigma_i(f_i \cdot p_i) - \Sigma_i(f_i \cdot p'_i)) = - \Delta gain$

[6] We may not be able to get the exact same size, but we should be sufficiently close so that we perform the forthcoming simplifications. This is usually not difficult as the size of classes composing a system usually represent a small percentage of the system size. In practice, we can therefore make such an approximation and find an adequate set of *n* largest classes.

The benefit of using the prediction model to select classes for inspection instead of selecting them according to their size is

benefit = Δgain - Δcost = 2Δgain = 2 · e · fc · ($\Sigma_i(f_i \cdot p_i) - \Sigma_i(f_i \cdot p'_i)$)

Thus, the benefit of using the fault-proneness model is proportional to the number of faults it detects above what the size-based model can find (if inspection effort is about equal to that of the baseline model, as is the case here). The factor 2 is because the difference between not finding a fault and having to pay *fc*, and finding a fault and not having to pay *fc* is 2*fc*.

Once such a model is developed, parameters *e* and *fc* are estimated in a given environment, and we can determine, for a given range of *e* values, the benefits (in effort unit) of using a fault-proneness model in function of *fc*, the cost of a defect slipping through inspections. Based on such information, one may decide whether using a predictive model for driving inspections can bring practical benefits.

As an example, Figure 2 shows the benefit graph for two models, referred to as “linear” and “MARS” model. The benefit of using the linear or MARS model to select classes for inspection over a simple size-based selection of classes is plotted as a function of the number *n* of classes selected for inspection. The benefit is expressed in multiples of *fc*, assuming an inspection effectiveness e=80%.

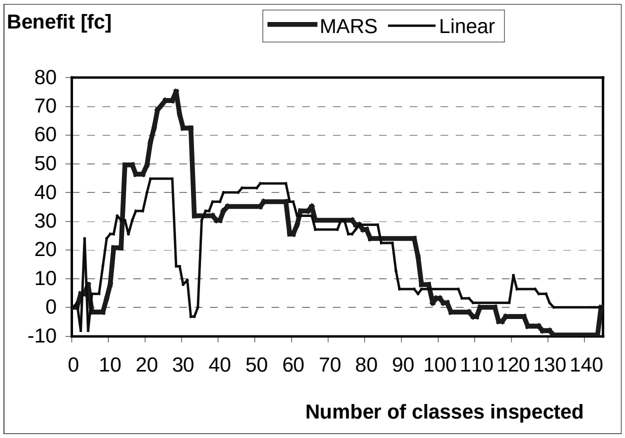

Figure 2: Benefit graph for linear and MARS model from [BMW01]

Besides the economical viability of a model, such a figure effectively demonstrates the advantages of one model over the other. It also helps to identify ranges for the number of selected classes, *n*, at which the model usage has its greatest pay-off.

To decide whether a prediction model is worth using, some additional costs and constraints may also be accounted for such as, in our example, the cost of deploying the model: automation and training.

4 Summary of Results

This section presents a summary of the empirical results reported in the studies of Section 2. It attempts to identify consistent trends across the results reported and discuss inconsistencies when they arise.

4.1 Correlational Studies

We focus here on correlational studies where the dependent variable is related to some measure of fault-proneness. The reason is that this is the only dependent variable for which a large-enough number of studies exist and hence a cross examination of results is possible.

4.1.1 Univariate Analysis of Fault-Proneness

Tables Table 5 to Table 8 show the results from univariate analysis in studies using fault- proneness or the number of faults as dependent variable, for size, coupling, cohesion, and inheritance measures, respectively. The Table for size measures also includes measures presented as "complexity" measures, which in practice are often strongly correlated to simple size measures. The inheritance measures capture various properties of inheritance such as its depth or level of overloading. Each column provides the results for one study, each table row shows the results for one measure. The aim is to facilitate comparison of results where the same measure is investigated in several studies.

In each table, row "Tech." indicates the modeling technique used in each study - which is either univariate logistic regression (denoted as LR) or Spearman Rho (rho). For studies that investigate more than one system, row "System", identifies the name of the system each column pertains to.

In the body of the table, the semantic of the entries is as follows:

- ++: measure has positive significant relationship at the 0.01 level
- +: measure has positive significant relationship at the 0.05 level
- O: measure has no significant relationship at the 0.05 level
- -: measure has negative significant relationship at the 0.05 level
- --: measure has negative significant relationship at the 0.01 level
- na: the measure was considered, but showed no or little variation (measure not significant and not meaningful in the respective system).

Though it may be desirable to provide more detailed results (e.g., regression coefficients, exact p-values) in the table, we chose this more compact summary for the following reasons:

- In isolation, the magnitude of a regression coefficient is not meaningful, since this depends on the range and variation of the measure in the data sample.
- These coefficients are not applicable in other environments and not really useful to report here.
- Some studies do not provide detailed p-values, but only indicate if it is below a certain threshold.
- To contain the size of the table.

To further contain the size of the tables we made a number of simplifications while presenting the results from several studies:

- In [BDW00] and [BWL01], coupling to library classes and coupling to non-library classes were measured separately. We show here the results for coupling to non-library. The results for coupling to non-library classes are mostly consistent, except that some coupling mechanisms occur less frequently there, resulting in more measures being "not applicable".
- In [EBGR01, EBGR99, EMM01, GEMM00], univariate analysis controlled for size (see subsequent discussion of "confounding effects") by including both a size measure and a design measure in the regression equation. We report here the significance of the design measure only.
- In [TK99], results for different types of faults (faults related to OO features or not) are reported, we show here the results for correlation against "all faults", which should be consistent with the nature of faults accounted for in the other studies.
- In [ABM96], three dependent variables are considered and we show here the results for defect density only. The results for the other DVs differ only marginally

Table 5: Summary of univariate analysis results for size measures

	BDW00	BWI01	BMW01	BS98		CS00	EBGR01		EBGR99	RL92	TKC99		
Tech.	LR	LR	LR	rho	rho	rho	LR	LR	LR	rho	LR	LR	LR
System				CCM	EFT						A	B	C
ATTRIB						O							
STATES						++							
EVNT						++							
READS						++							
WRITES						++							
DELS						++							
RWD													
LOC					O	++				++			
LOC_B						++							
LOC_H					++	++							
WMC-ss										++			
WMC-1 / NMA / NMImp	++	++	++	++	++		++	O			O	+	O
WMC-CC					++		++	O	O	++			
NOMA											O	+	O
AMC											+	++	O
Stmts	++												
NAImp	++	++	++										
NMpub	++												
NMNpub	+		++										
NumPara	++	++											
NAInh		o											
NMInh		o											
TotPrivAtrib			++										
TotMethod			++										

Table 6: Summary of univariate analysis results for coupling measures

	ABM96	BDW00	BWI01	BMW01	BS98		CS00	EBGR01		EBGR99	EMM01	GEMM00	HCN98		RL92	TKC99		
Tech.	rho	LR	LR	LR	rho	rho	rho	LR	LR	LR	LR[8]	LR[8]	rho	rho	rho	LR	LR	LR
System					CCM	EFT							SEG1	SEG2		A	B	C
COF	+																	
CBO' CK91																		
CBO CK94						O		+	O	++			O	O		O	O	O
RFC-1		++	++		++	++		++	O	O						+	+	O
RFC		++	++															
MPC		++	++															
ICP		++	++															
IH-ICP		o	++															
NIH-ICP		++	++															
DAC		+	+															
DAC'		+	+															
ACAIC		na	na	O						na	na	O						
OCAIC		+	+	++						O	O	++						
DCAEC		na	na	na						na	na							
OCAEC		--	+	O						na	++	+						
ACMIC		na	+	+						++	na	O						
OCMIC		o	++	++						O	+	++						
DCMEC		na	o	na						O	na							
OCMEC		o	++	+						+	++	+						
AMMIC		o	+							O								
OMMIC		++	++							O								
DMMEC		o	o							O								
OMMEC		o	++							+								
IFCAIC		o	na															
FCAEC		o	na															
IFCMIC		na	na															
FCMEC		na	na															
IFMMIC		o	na															
FMMEC		o	na															
NAS													O	-				
CC															++			
AMC															++			
CDM					++	++												
Fan-in					++	++												
Fan-out					++	++												
IC																O	++	O
CBM																O	+	O

Table 7: Summary of univariate analysis results for cohesion measures

	ABM96	BDW00	BWI01	EBGR01		EBGR99
Tech.	rho	LR	LR	LR	LR	LR
MHF	O					
AHF	O					
LCOM1-CK91		o	++			
LCOM2-CK94		o	+	+	O	O
LCOM3		+	+			
LCOM4		o	o			
LCOM5		o	o			
Coh		--	o			
Co		o	o			
LCC		o	+			
TCC		o	o			
ICH		++	++			

Table 8: Summary of univariate analysis results for inheritance and polymorphism-related measures

	ABM96	BDW00	BWI01	BMW01	BM99	BS98		CS00	EBGR01		EBGR99	EMM01	GEMM00	HCN98			RL92	TKC99		
Tech.	rho	LR	LR	LR	LR	rho	rho	rho	LR	LR	LR	LR	LR	rho	rho	rho	rho	LR	LR	LR
System						CCM	EFT							SEG1	SEG2	SEG3		A	B	C
DIT		++	--	O			++	++	O		O	++	(qm)	-	--	o		O	O	O
AID		++	--																	
CLD		--	o																	
NOC		-	o	O		O	na	O	na	na	O	O	+	O	O	O		O	O	O
NOP		++	o																	
NOD		-	o			O														
NOA		++	o																	
NMO		+	++											-	++	O				
NMI		+	o											O	O	O				
NMA		++	++																	
SIX		++	o																	
CHNL						++														
MIF	-																			
AIF	O																			
POF	O																			
OVO				O	O															
SPA				+	O															
SPD					O															
DPA				++	--															
DPD					O															
SP				+	-															
DP				++	-															
NIP				++	O															

Tables 5 to 8 are sparsely populated - most measures have been used only in the context of one or two studies. The only set of measures that have received a wider attention are the six measures by Chidamber and Kemerer [CK94]. This profusion of similar but different measures is of course a problem if we want to converge towards a stable body of knowledge. But it was unavoidable, in the past recent years, as the research was at an exploratory stage. The focus should know be on investigating further the use of measures that have shown significance in at least one study.

Measures of complexity and size are consistently associated with fault- proneness or the number of faults, in the expected direction: the larger/ more complex the class, the more fault-prone.

For coupling measures, we find a mixed picture

- CBO is significant in only 3 out of 10 instances
- RFC in 6 out of 8 instances (two insignificant cases due to controlling for size). This is not surprising as RFC has been shown to be a combination of size and import coupling [BDW99].
- RFC-1, OCAIC, OCMEC, OMMIC, and OMMEC are mostly significant in the expected direction. Therefore, coupling due to parameter typing and method invocation are probably worth to investigate further in the future.

Cohesion measures are rarely investigated empirically. An explanation may be that these measures are difficult to obtain from static analysis, as they require a detailed analysis of the class attribute usage by methods. The results show that overall, cohesion measures appear to have no significant relationship to fault-proneness, in particular normalized cohesion measures that have a notion of maximum cohesion [BDM98].

For measures related to inheritance, the results related to the depth of a class (DIT) and the number of children are inconclusive. The use of inheritance mechanisms can increase the fault-proneness of deeper classes, decrease it, or have no effect on fault-proneness at all.

Overall, a number of size and coupling measures consistently show a significant relationship in the expected direction. Only for inheritance-related measures DIT and NOC do we find entirely inconsistent trends, indicating that the use of inheritance has effects on fault-proneness that depend on other factors such as design methodology and experience [BWL01].

In general, even assuming consistent data analysis procedures, we have to expect some inconsistency in results across studies. This is due to the nature of the data set, different programming practices causing certain measures to display little variance in some cases, or, in some cases, different interpretation and implementation of the same measures (e.g., LCOM and CBO).

Confounding Effects

In [EBGR01], the authors discuss the notion of confounding effects of design measures with size. The idea is that a measure of a given design measure (e.g., import coupling) can show both a significant relationship to, say class fault-proneness, and class size. Class size also is an indicator of fault-proneness. The question then is whether the relationship between the design measures and fault-proneness is due the measures' association with size, or to a causal relationship between the design property and fault-proneness.

To test this possible confounding effect, the authors fit models including both a measure of size and a design measure, and investigate the significance of the design measure in this model.

When controlling for size, the significance of some measure, in particular import coupling measures, drops, indicating that their relationship to fault- proneness is due these measures also being associated with size. This may be expected, as larger classes are likely to import more services and has been shown in a number of previous paper [BWDP00, BWL01]. Export coupling measures are also shown not to be affected by this.

The authors then conclude results cast doubts on empirical validity claims made in the past as these empirical results were possibly biased by the confounding effect with size. However, it is interesting to note that controlling for size does not systematically "invalidate" measures in their studies. For example, OCMIC is found insignificant after controlling for size in [EBGR99], but significant after controlling for size in [GEMM00]. Furthermore, in the next section, where multivariate models are reported to predict fault-proneness, a number of studies have shown that models that are based on both coupling and size perform significantly better than size models alone

[BWPD00, BWL01]. But again, we have to expect variations across systems and another important point to consider is whether one's objective is to build the most accurate prediction model possible or demonstrate a causal relationship. In some circumstances, coupling measures may not have main effect on say, fault-proneness, but an interaction effect with size.

OO Measures and thresholds

One reoccurring suggested use of product measures is that they can be used to build simple quality benchmarks based on thresholds. If a product measure exceeds a certain threshold (e.g., size measured as the number of public methods), the class or module is either rejected and must be redesigned, or at least flagged as "critical".

In [BEGR00] and [EBML00], the authors conduct a refined univariate analysis to investigate if this assumption is supported by empirical data. To do this, they modified the regular univariate logistic regression models to force the inclusion of thresholds and compared those models with typical logistic regression models. In each case, the authors find no significant difference in goodness of fit between the threshold model and the simpler no-threshold models. This result is rather intuitive as it is difficult to imagine why a threshold effect would exist between, for example, size measures and fault-proneness. This would imply a sudden, steep increase in fault-proneness in a certain size value range, something that would be difficult to explain as it would imply, for example, a sharp increase in fault-proneness above a certain size threshold. Note that none of the other empirical studies reported here assume the existence of thresholds.

4.1.2 Multivariate Prediction Models

This section reports on what are possibly the results of highest practical importance: the prediction models, e.g., fault detection probability. Those models are usually multivariate models in the sense that they integrate the effect of more one structural measure.

As we will see, many different measures are used in the various multivariate models, and these measures do not necessarily all capture different aspects of a class. Therefore, we first identify what are common dimensions captured by structural measures, and then investigate whether any of these dimensions are predominantly represented in the multivariate models.

Principal Component Analysis

As introduced in Section 3.2, Principal Component Analysis (PCA) is a technique aimed, in our context, at identifying the dimensions captured by structural measures. For example, does a coupling measure really capture what it is intended to or is that part of the dimension capturing the size of a class.

We performed PCA for a large set of measures in the context of four different data sets [BWDP00, BWL01, BMW01, BW01]. Though there are differences in the dimensions identified from system to system, and how measures are allocated to dimensions, some dimensions reoccurred in two, three, or all studies, and we summarize them here as follows:

1. Import coupling (through method invocation). Measures such as MPC, OMMIC, which count invocations of methods of other classes. Note that when distinguishing coupling to library and non-library classes, which tend to be orthogonal dimensions. However, this distinction was only made in two studies, so we do not separate them here.

2. Class size, measured in terms of the number of methods (NMImp, WMC), method parameters (NumPar,...), class attributes, executable/declaration statements.

3. Normalized cohesion measures (TCC, LCC etc), i.e., cohesion measures with an upper bound that represents maximal cohesiveness of methods and attributes.

4. Export coupling to other classes - degree to which a class is used by other classes, be it as class parameter, class attribute, or via method invocation.

5. Depth of inheritance tree below a class / degree to which a class is extended (NOC, CLD, etc.).

6. Depth of class in inheritance tree / number of ancestor classes (DIT, NOP, etc.).

7. Import coupling from ancestor classes. Degree to which the added methods of a derived class make use of inherited methods (AMMIC, IH- ICP,...).

8. Inherited size. Number of attributes/methods a class inherits (NMInh, NAInh,...).

In the following discussion, we assign each measure to one of the above dimensions, in order to abstract from the individual measure and focus on the concept it captures. In cases were, based on empirical results across the four studies, a measure cannot be uniquely assigned to a particular dimension, we assign it to the dimension whose interpretation best matches the definition of the measures.

Overview of Multivariate Models

The following tables summarize the multivariate prediction models for fault-proneness or number of faults (Table 9) and effort (Table 10) as dependent variable. Each row provides the details for one prediction model. The first three columns indicate the literature source, the modeling technique, and the procedure that was used to arrive at the model. The following eight columns list the covariates of the model, each column representing one of the above structural dimension. Each covariate is listed in the column of the dimension it was assigned to. Column T (total) reports the total number of covariates in the model. The last three columns reproduce whatever measures of goodness of fit of the model were reported, what type of cross-validation was performed, if any, and how well the model performed in cross-validation (i.e., its predictive power).

Table 9: Summary of multivariate models for fault-proneness / number of faults

Source	Modeling Technique	How built	1. Import coupling	2. Size	3. norm. cohesion	4. Export coupling	5. Inh. depth above	6. Inh. depth below	7. Inh. based coupling	8. Inh. size	T	Goodness of fit	Type of CV	Predictive Power
[EBGR99]	Logistic regression	size measure forced in model, include design measures not confounded with size		LOC		CBO			ACMIC		3	R^2= 0.366	Leave-one-out	Area under ROC: 0.813, sensitivity: 0.84, specificity 0.78. Proportion correct: 0.79
[EMM01]				NAImp		OCMEC	DIT				3	R^2=0.4355	Leave-one-out	AROC: 0.87, sensitivity: 0.81, specificity 0.83
													Between-system validation	AROC: 0.78, Proportion correct: 0.77
[GEMM00]			OCMIC	NMImp		OCMEC	DIT, DIT^2				5	R^2= 0.42	Leave-one-out	AROC: 0.85
[CS00]	OLS			EVNTS							1	adj. R^2= 87.2%		
				EVNTS			INHTS				2	adj. R^2= 89%		
[BWDP00]	LR	forward stepwise heuristic		NM, NMpub, NumPar							3	R^2=0.139, 67% completeness, 60% correctness		
			NIHICP, RFC, RFC1_L			FMMEC	NOP NMI	CLD			7	R^2 = 0.53, 94% completeness, 81% correctness	10-cross validation	92% completeness, 78% correctness.
			OMMIC RFC	NM, NMA NumPar		OCAEC FMMEC	NOP	CLD			9	R^2 = 0.56, 90% completeness, 79% correctness		
[BWL01]	LR	forward stepwise heuristic		NAImp NumPar							2	R^2 = 0.16, 68% completeness 68% correctness		
			ICP_L, NIHICP OCMIC OCAIC	ICH		CBO’					6	R^2 = 0.53, 91% correctness, 97% completeness	10-cross-val.	correctness and completeness about 90%

Source	Modeling Technique	How built	1. Import coupling	2. Size	3. norm. cohesion	4. Export coupling	5. Inh. depth above	6. Inh. depth below	7. Inh. based coupling	8. Inh. size	T	Goodness of fit	Type of CV	Predictive Power
			ICP_L OCAIC	NAImp, Numpar ICH		CBO’					6	R^2 = 0.56, 97% completeness, 87% correctness		
[BMW01]	LR	forward stepwise heuristic	OCMIC			OCMEC				NIP	3		10-cross-val.	74% correctness, 62% completeness
													Between-system validation	completeness and correctness about 60%
	LR with MARS		OCMIC			OCMEC	DIT				3		10-cross-val.	68% correctness, 73% completeness
													Between-system validation	completeness and correctness about 60%

Table 10: Summary of multivariate models for effort

Source	Modeling Technique, IV	How built	1. Import coupling	2. Size	3. norm. cohesion	4. Export coupling	5. Inh. depth above	6. Inh. depth below	7. Inh. based coupling	8. Inh. size	T	Goodness of fit	Type of cross-validation	Predictive Power
[CDK98]	OLS, Productivity	stepwise heuristic		LOC							2[7]	adj. R^2 = 75%		
				LOC, HILCOM		HICBO					4[7]	adj. R^2 = 82%		
	OLS, Rework effort			HILCOM		HICBO					3[7]	adj. R^2 = 60%		
	OLS Design effort			HILCOM		HICBO					2	adj. R^22 = 60%		
[LH93]	OLS, #lines added/ changed/ removed	a priori selection of variables to be included	MPC RFC DAC	WMC NOM Size1 Size2 LCOM			DIT	NOC			10	adj. R^2 87.73%(UIMS), 85.50% (QUES)		
			MPC RFC DAC	WMC NOM LCOM			DIT	NOC			8	adj. R^2 87.71% (UIMS), 85.33% (QUES)	Between-system validation	Correlation: predicted values of UIMS model applied to QUES with actual values in QUES: r=0.678 (0.650 the other way round).
				Size1 Size2							2	adj. R^2 64.29% (UIMS), 61.72% (QUES)		
[LHKS95]	OLS, #lines added/ changed/ removed	a priori selection of variables to be included	RFC DAC	NOM LCOM			DIT	NOC			5	adj. R^2 77% (UIMS) 67% (QUES).	Between-system validation	UIMS model to applied to QUES: r=0.44, other way round r=0.49

[7] The model includes an additional dummy variable indicating whether a particular developer worked on a given class.

Source	Modeling Technique, IV	How built	1. Import coupling	2. Size	3. norm. cohesion	4. Export coupling	5. Inh. depth above	6. Inh. depth below	7. Inh. based coupling	8. Inh. size	T	Goodness of fit	Type of cross-validation	Predictive Power
[BW01]	Poisson Regression (PR)	forward stepwise heuristic		NAImp, NMImp, NumPar						NA	4	mean MRE (mMRE) = 1.71	10-cross validation	mMRE = 1.71
	hybrid PR & CART			NAImp, NMImp, NumPar						NA NMInh	7[8]	mMRE = 0.72	10-cross validation	mMRE = 0.78
	PR		OCMIC MPC, ICP	NAIMP		DMMEC			ACMIC IH-ICP		7	mMRE = 1.5	10-cross validation	mMRE = 1.55
	hybrid PR & CART		MPC, OCMIC, ICP	NAImp, NMImp		PIM_EC			ACMIC, IH-ICP	NA NMInh	10	mMRE = 0.69	10-cross validation	mMRE = 0.77
	PR		OCIMC	LCOM2, LCOM3, ICH, NumPar	COH, LCC				ACMIC		8	mMRE = 0.97	10-cross validation	mMRE = 1.01
	hybrid PR & CART			NumPar NMImp, LCOM1, LCOM2	LCC	PIM_EC			ACMIC	NMInh	8	mMRE = 0.74	10-cross validation	mMRE = 0.85

[8] The CART models include dummy variables representing leaf nodes from the regression tree (cf. Section3.4.2). The measures listed here include those measures that contribute to the definition of at least one included dummy variable. Because there is no 1:1 mapping, the number of measures listed differs from the indicated number of covariates in the model.

The tables contain 13 fault-proneness models, and 14 effort models. From the tables, we can draw a number of observations:

Design measurement dimensions represented in the models

From the 13 *fault-proneness models*, size measures are represented in 10 models, export coupling in 9, import coupling in 7, depth of class in 6 models. The other inheritance dimensions are represented once or twice, normalized cohesion not at all.

For the 14 *effort models*, size is a contributor to all of them, import coupling and export coupling are represented in 6 models each, inheritance-based size and import coupling in 4 models each, inheritance depth above/below class in 3 models each, normalized cohesion in two.

It should be noted that for the above models, authors typically have investigated only a subset of measures in which some dimensions were not represented at all, or were deliberately left out (e.g., when building models based on size measures only). In those instances, the fact that a given dimension is not included in the model does not imply that it is not useful. That notwithstanding, the results seem to indicate that size, import, and export coupling appear to be the dimensions mostly likely to affect the DV and should be always be considered when building models. Dimensions concerning inheritance sometimes play a role (especially with the effort models) and are worth to be tried out, whereas normalized cohesion measures appear not to be effective.

Goodness of fit achieved by fault-proneness models

The LR *fault-proneness models* typically achieve R-squared values from 0.42 to 0.56, which is fairly high (unlike the R- squared known from OLS, the R-squared for maximum likelihood estimation techniques such as LR [HL89] seldom approaches one; from our experience, values above 0.5 usually indicate very good fit). The high goodness of fit is also visible in correctness and completeness values, which are typically well above 80%. Given that these models stem from different environments (wrt. overall system/project size, application domain, programming languages, developers, etc), these results suggest highly accurate fault-proneness models based on design measures can likely be built in most development environments.

Impact of size. In [BWPD00] and [BWL01], models based on size measures only and models based on design measures were assessed, to see if the latter help to better predict fault-proneness over what is explained by size alone. This assumption is confirmed, as the size-only models had a significantly lower fit (R-squared<0.16, completeness and correctness below 70%) than models including design measures.

Goodness of fit achieved by effort models

The OLS *effort models* aimed at predicting effort/code churn for individual classes achieve R-squared values between 60%-87% percent. Though R-squared values approaching 90% usually indicate a good model fit, it is difficult to assess from this information to which degree reliable class level effort estimates can be made - absolute and relative errors are more transparent measures of prediction error in the context of effort estimation. In [BW01], goodness of fit is expressed, amongst others, in terms of the mean MRE, which ranges from 0.7 to 1.7. In other words, on average the predicted effort is 70% to 170% off the actual effort. As such, these models are not suitable to predict development effort for individual classes. It is shown, however, that when adding up estimates for class level efforts to the system level, reliable system effort estimates with MREs between 10-30% can be achieved. Also, the hybrid regression and CART models (see Section 3.4.2) have considerably lower MREs than the regular models, indicating that this approach to capture non-linear and local trends in data can in deed help build better models.

Impact of size. Models based on size measures only, and size and design measures, were built in [LH93] and [BW01]. In [LH93], the size- only models showed a significantly lower fit than design models (R- squared 62-65% vs. 85%-87%), indicating that design measures carry additional information related to effort not contained by size measures. While this difference is statistically significant, it is not clear if it is also of practical significance. Similar findings were made for the non-hybrid models in [BW01], where adding including design measures to the models decreased mMREs somewhat. For the hybrid models, however, inclusion of design measures did not help to improve the goodness of fit.

Results from cross-validation

Where conducted, the results from cross-validation (Section 3.5.1) show a promising picture. In most cases, the performance of the models in cross-validation has not dropped much compared to the goodness of fit achieved by the models.

For the effort models in [BW01], the change in mMRE compared to goodness of fit is statistically not significant. In [LH93] and [LHKS95], the authors calculate a correlation coefficient of actual effort vs. predicted effort (prediction from a model built from another system). Depending on the model, they find correlation coefficients r between 0.44 and 0.67, significantly different from 0). Here, it is difficult to assess if such a prediction can still be practically useful, as the figures provided have no meaningful interpretation in an application context. The absolute/relative errors of the prediction could provide more insight here.

The fault-proneness models typically achieve completeness and correctness levels (or sensitivity/specificity levels) of about 80% and above in cross-validation. Recently, cost-benefit models have been proposed in [GEMM00] and [BMW01], investigating whether it is economical viable to use models of such accuracy, for instance, to focus inspections efforts on classes likely to contain faults:

- The cost-benefit model in [GEMM00] defines the savings of using a prediction model as the proportion of costs due to post-release faults that are saved due to inspection on classes predicted fault- prone (as opposed to performing no inspections at all). This model requires assumptions about the relative cost of post-release faults and class inspections.
- The cost-benefit model in [BMW01] and summarized in Section 3.5.2 expresses the benefit of using a prediction model to select classes for inspection over another class selection strategy which is used as comparison baseline (e.g., ranking based on class size). This model requires no assumptions about the relative cost of post-release faults and class inspections, thus making it practical to use. It assumes that you should compare the benefits of decision making (e.g., select class to inspect) with the predictive model with respect to the same decisions without the model, following a procedure already available. To make the comparison meaningful in terms of benefits, the cost (e.g., the cumulative size of classes inspected) should be held constant for both the procedure using the model and the baseline procedure.

In Table 9 and Table 10, a number of published models were not considered due to some methodological flaws in building these models. Including them would bias the results:

- The models in [AM96] based on 8 observations and 7 covariates suffer from overfitting (as visible in high R-squared>99.9%).
- The models in [BM99] and [NQ98] contain covariates with p-values above 0.5. Such covariates can be removed from the model without significant loss of fit.
- In [CL93], a full model with all candidate measures is fitted. From this model, not significant covariates were removed, and predicted values were calculated from the remaining significant covariates, but retaining the regression coefficients from the full model. Of course, the reduced model should have been refitted to obtain unbiased predicted values, or a proper backward elimination procedure could have been used.

4.2 Controlled Experiments

In the following, we briefly report on the qualitative results established in controlled experiments.

- In [BBDD97], students answered comprehension questions and performed change impact analysis on functionally equivalent systems with differing adherence to design principles (including low coupling, high cohesion, and simple, small classes). The time required to perform the tasks and the correctness of the students' performance were measured. The OO system displaying good object-oriented design principles was shown to be better understandable and faster to modify than the version violating the design principles. A refined experiment in [BBD97] confirmed these results. Note however that no statement can be made whether any of the design principles contributed more or less to the differences in understandability/maintainability.
- Though not a controlled experiment in the strict sense, in [Bar98] two developers were asked to implement a certain piece of software. One was instructed to write code for a specific context and of poor reusability, whereas the other to write the code in the most reusable manner possible. A set of design measures was applied to both resulting systems. The "reusable" system showed lower coupling and higher cohesion (as measured by CBO and LCOM) as the "poor reusability" one. Also, the

"reusable" system made no use of inheritance, while the "poor reusability" one did. Note however that the sample systems were too small to allow for any statistical testing of significant differences.

The remaining studies all focus on the impact of inheritance on understandability and maintainability.

- In [LC92], students answered comprehension questions and performed debugging and modification tasks on functionally equivalent systems with deep and shallow inheritance hierarchies. The time required to perform the tasks and the correctness and completeness of the students' performance were measured. The answers to comprehension questions showed no difference in understandability between the deep and shallow systems. The debugging tasks were more easily to identify and to correct in the shallow systems, while taking the same time as the deep versions. Modification tasks took shorter for the deep versions, but were carried with lower correctness than on the shallow versions.
- In [WDMR99], students performed modifications on three pairs of functionally equivalent systems. The time they required for the modification was measured. In two experimental runs, systems with 3 levels of inheritance were found to be better maintainable than the equivalent flat versions. In a third experimental using a larger system no difference was found between a system using 5 levels of inheritance and an equivalent flat version.
- In [HCN00], students performed modifications on two pairs of functionally equivalent systems that differ in the use of inheritance (no inheritance, three/five levels of inheritance). The correctness and completeness of their performance was observed, and also the perceived subjective understandability of the systems was measured. Results showed that flat systems were easier to modify than the versions with three or five levels of inheritance. For one pair of systems, the flat system was easier to understand than the inheritance version and no difference was found for the other pair of systems.

It is interesting to observe that the results from controlled experiments mirror well the findings from the correlational studies. Coupling, cohesion, and size appear to consistently affect system quality. For inheritance, inconsistent results are found: use of inheritance can have a beneficial, detrimental, or no effects at all on system quality. Possible explanations were already discussed in Section 4.1.1.

5 Conclusions

Despite a large number of empirical studies and articles, a lot of work remains to be done regarding the construction and application of useful, measurement-based quality models for object-oriented systems.

The difficulty related to developing an empirical body of knowledge stems from:

- A large number of proposed measures, many of them being similar.
- A large number of external quality attributes of interest (all the "ilities" of software development).
- The scarcity of reliable, complete data sets.
- The difficulty to integrate quality prediction models in realistic decision processes where their benefits can be assessed.

Despite such difficulties, reported studies allow us to draw a number of important conclusions that are discussed below.

5.1 Interrelationship between design measures

Many of the coupling, cohesion, and inheritance measures used in this study appear to capture similar dimensions in the data. In fact, the number of dimensions actually captured by the measures is much lower than the number of measures itself. Results from principal component analysis on numerous datasets showed that the measures listed in the Appendix can safely be reduced to a smaller set of about 15 measures, without losing important (i.e., potentially quality-related) design information. This simply reflects the fact that many of the measures proposed in the literature are based on comparable ideas and hypotheses, and are therefore somewhat redundant.

5.2 Indicators of fault-proneness

Measures of size, import coupling, and export coupling appear to be useful predictors of fault-proneness.

- If one intends to build quality models of OO designs, coupling will very likely be an important structural dimension to consider. More specifically, a strong emphasis should be put on method invocation import coupling since it has shown to be a strong, stable indicator of fault- proneness. We also recommend that the following aspects be measured separately since they capture distinct dimensions in our data sets: import versus export coupling, coupling to library classes versus application classes, method invocation versus aggregation coupling.
- As far as cohesion is concerned and measured today, it is very likely not a very good fault-proneness indicator. This is likely to reflect two facts: (1) the weak understanding we currently have of what this attribute is supposed to capture, (2) the difficulty to measure such a concept through static analysis only. One illustration of this problem is that two distinct dimensions are captured by existing cohesion measures: normalized versus non-normalized cohesion measures. As opposed to the various coupling dimensions, these do not look like components of a vector characterizing class cohesion, but rather as two fundamentally different ways of looking at cohesion.
- Inheritance measures appear not to be consistent indicators of class fault-proneness. Their significance as indicators strongly depends on the experience of the system developers and the inheritance strategy in use on the project.
- Size measures are, as expected, consistently good indicators of fault- proneness, and can be considered with fault-proneness models. However, we have observed that the above-mentioned dimensions of coupling and inheritance, when combined with size, help explain fault- proneness further than size alone; models based on all three types of measures clearly outperform models based on size only.

5.3 Indicators of effort

Size seems to be the main effort driver that explains most of the effort variance. While the more sophisticated coupling and inheritance measures also have a univariate relationship to effort, they do not bring substantial gains in terms of goodness of fit and cost estimation accuracy.

5.4 Predictive power of models

Results concerning the predictive power of fault-proneness / effort models are encouraging. When predicting fault-prone classes, using all important fault-proneness indicators mentioned above, the best models have consistently obtained a percentage of correct classifications of about 80% and find more than 80% of faults. Overall, the results suggest that design measurement-based models for fault-proneness predictions of classes may be very effective instruments for quality evaluation and control of OO systems.

From the results presented in studies predicting development effort, we may conclude that there is a reasonable chance that useful cost estimation models can be built during the analysis and design of object-oriented systems. System effort prediction MREs below 30%, which is an acceptable level of accuracy for cost estimation models [BEMS99], seem realistic to achieve.

5.5 Cross-system application

An important question concerning the usefulness of design measurement-based prediction models is whether they can be viable decision making tools when applied from one object-oriented system to the other, in a given environment. This question has received very little attention in existing empirical studies. The most detailed investigation to date is reported in [BMW01], where the authors applied a fault-proneness model built on one system to another system, developed with a nearly identical development team (a different project manager), using a similar technology (OO analysis and design and Java) but different design strategies and coding standards. We believe that the context of our study represents realistic conditions that are often encountered in practice: change in personnel, learning effects, evolution of technology.

Our results suggest that applying the models across systems is far from straightforward. Even though the systems stem from the same development environment, the distributions of measures change and, more importantly, system factors (e.g., experience, design method) affect the applicability of fault detection predicted probabilities. However, we have shown the prediction model built on a system can be used to rank classes of a second system (within the same environment) according to their predicted fault-proneness. When used in this manner, the model can in fact be helpful at focusing verification effort on faulty classes. Though predicted defect detection probabilities are clearly not realistic based on actual fault data, the fault-proneness class ranking is accurate. The model performs clearly better than chance and also outperforms a simple model based on class size, e.g., number of methods.

It can be argued that, in a more homogeneous environment, this effect might not be as strongly present as in the current study. But we doubt whether such environments exist or are in any case representative. It is likely, however, that the more stable the development practices and the better trained the developers, the more stable the fault-proneness models. For example, in [EMM01] a fault-proneness model built from one version of a system was applied to later versions of the same system, and no shift in predicted probabilities was observed.

5.6 Cost benefit model

To assess the economical viability of fault-proneness models when applied across systems, we proposed a specific cost-benefit model. This model is tailored to a number of specific assumptions regarding the use of the model, but its underlying principles are general and can be reused. The benefit of a fault-proneness model is expressed as a function of a number of parameters: cost of a fault that slipped inspection, defect detection effectiveness during inspections. Continuing the system example we used in [BMW01], with a test system containing 27 faults, the benefit of using a prediction model to select classes for inspection over a simple size-based class selection heuristic showed to be as high as 17.6 fault-cost-units (1 fault-cost-unit = average cost of a fault not detected during inspection), thus demonstrating the usefulness of using measurement-based, fault-proneness models in the environment under study.

5.7 Advanced data analysis techniques

Novel, exploratory analysis techniques (Multivariate Adaptive Regression Splines: MARS, and hybrid regression models with regression trees) have been applied to construct of fault-proneness and effort models. Because we know little about what functional form to expect, such exploratory techniques that help finding optimal model specifications may be very useful.

Initial results support this premise and suggest that the fault-proneness models generated by MARS outperform a logistic regression model where the relationship between the logit and the independent variables is linear (log-linear model).

The combination of Poisson regression and regression trees has helped to improve significantly effort predictions, especially predictions based on size measures only that can be performed early on during the design stages. This can be explained by the fact that regression trees tend to capture complementary structures in the data since they help define new predictors capturing interactions between the original predictors. Though they will not systematically help, such hybrid models are worth trying.

5.8 Exploitation of results

There are a number of typical applications where results from the empirical studies we summarized here can be used to refine decision-making:

- Building quality benchmarks to assess OO software products that are newly developed or under maintenance, e.g., in the context of large- scale software acquisition and outsourcing, but also in-house development. For example, such experiences are reported in [MC96] where the code of acquired products is systematically measured to identify systems or subsystems that strongly depart from previously established measurement benchmarks. Such benchmarks can be built based on existing operational software, which has shown to be of good quality. New software can then be analyzed by comparing, for example, import coupling class distributions with the established benchmark. Any system part that would show a strong departure from the benchmark distribution could, for example, be further inspected to investigate the cause of the deviation. If no acceptable justification can be found then the acquisition manager may decide to require some corrective actions before the final delivery or for future releases of the software. This is particularly important when systems will be maintained over a long period of time and new versions produced on a regular basis.
- Design measurement can be combined with more qualitative analyses of software systems to make the results of a qualitative analysis more objective. For example, [BW01b] describes a study aimed at assessing the modifiability and reusability of a software system. To this end, a set of change scenarios

that the system is likely to undergo in the future was identified, and the impact of change to the system classes assessed for each scenario. The structural properties (size, coupling) of the system classes, which should be indicative of how difficult these changes are to implement, were also measured. The design measures and the scenario evaluation were then integrated by defining a Change Difficulty Index (CDI), which incorporates both the extent of changes to classes, and their associated coupling, complexity, and size. A comparison of the CDIs at the class level then provided insight into the maintainability and reusability of software systems.

When static measurement is performed in isolation, large/high coupling classes are generally considered difficult to maintain or reuse. With the scenario evaluation, we can put this into context. A class with high coupling may cause no problems if it is unlikely to be changed or reused in the future. On the other hand, for a class that is likely to undergo frequent changes, this is not a problem if the class is well designed for it (low size/coupling). Here, the two approaches were used together so that they can address each other's limitations.

In both of the above examples, it is important that the measures used are carefully selected as they may affect the results and decisions. One important consideration is that these measures have to be consistent indicators of software design problems, and their relationships to external quality attributes, e.g., fault-proneness, should be clearly demonstrated and stable across environments. In the current stage of knowledge, certain measures of import coupling and size appear to be particularly well suited for the abovementioned purposes.

The overall results of the studies tell us that the validity of fault-proneness models may be very context-sensitive. We therefore recommend that fault-proneness models be built locally, in the environment where one intends to use them, and their stability has to be assessed and analyzed across systems in that environment. Existing measurement tools do not support the use of design measures for this particular purpose. Based on the results summarized here, practitioners (quality assurance teams) can verify if the set of measures provided by their measurement tools covers all dimensions of OO designs we identified as important quality indicators. Our data analysis procedure then provides practitioners with detailed guidance on how they can (1) reduce the set of measures provided by their measurement tools to a minimal, non-redundant set, (2) construct prediction models from these measures, (3) use these prediction models for decision making during projects, and (4) evaluate the cost-effectiveness of prediction model usage.

5.9 Future research directions

We have identified a number of open questions that are crucial to the betterment and wider acceptance of measurement-based OO design quality models:

- Most of the design measurement reported is actually based on measurement of source code. Early analysis and design artifacts are, by definition, not complete and only represent early models of the actual system to be developed. Using models on such representations inevitably introduces a measurement error in terms of structural properties such as coupling or inheritance. The use of predictive models based on early artifacts and their capability to predict the quality of the final system still remains to be investigated.
- Future research needs to focus on collecting larger datasets, involving large numbers of systems from a same environment. In order to make the research on fault-prone models of practical relevance, it is crucial that they be usable from project to project. Their applicability depends on their capability to predict accurately and precisely (fine grained predictions) quality attributes (e.g., class fault-proneness) in new systems, based on the development experience accumulated on past systems. Though many studies report the construction of models, few report their actual application in realistic settings. Our understanding has now reached a sufficient level so that such studies can be undertaken.
- Cost-benefit analyses of fault-proneness models used either a simple class selection heuristic to focus verification based on class size as a comparison benchmark or, even worse, compared against the case where no verification is performed at all. Both solutions are not fully satisfactory as experts may very well be accurate at predicting quality attributes such as fault- proneness. But little is known on how prediction models compare with experts' predictions. Furthermore, there might be a way to combine expert opinion with OO design quality models to obtain more reliable statements about system quality.

As it may be expected, more studies are needed to reach a mature body of knowledge and experience. However, though this is a prerequisite it is not enough. Empirical studies in software engineering need to be better performed, analyzed, and reported. To this purpose, standard procedures may be reused and adapted from other empirical fields, such as medicine, and should be used to evaluate empirical articles. We have provided in this chapter a set of procedures that may be used as a starting point for conducting higher quality research and obtaining more fruitful results when investigating measurement-based quality models.

6 Appendix

The following tables describe the coupling, cohesion, inheritance, and size measures mentioned in this chapter. We list the acronym used for each measure, informal definitions of the measures, and literature references where the measures originally have been proposed. The informal natural language definitions of the measures should give the reader a quick insight into the measures. However, such definitions tend to be ambiguous. Formal definitions for most of the measures using a uniform and unambiguous formalism are provided in [BDW99][BDW98], where we also perform a systematic comparison of these measures, and analyze their mathematical properties.

Table 11: Coupling measures

Name	Definition	Source
CBO	Coupling between object classes. According to the definition of this measure, a class is coupled to another, if methods of one class use methods or attributes of the other, or vice versa. CBO is then defined as the number of other classes to which a class is coupled. This includes inheritance-based coupling (coupling between classes related via inheritance).	[CK94]
CBO’	Same as CBO, except that inheritance-based coupling is not counted.	[CK91]
RFC	Response set for class. The response set of a class consists of the set M of methods of the class, and the set of methods directly or indirectly invoked by methods in M. In other words, the response set is the set of methods that can potentially be executed in response to a message received? by an object of that class. RFC is the number of methods in the response set of the class.	[CK91]
RFC-1	Same as RFC, except that methods indirectly invoked by methods in M are not included in the response set.	[CK94]
MPC	Message passing coupling. The number of method invocations in a class.	[LH93]
DAC	Data abstraction coupling. The number of attributes in a class that have another class as their type.	[LH93]
DAC’	The number of different classes that are used as types of attributes in a class.	[LH93]
ICP	Information-flow-based coupling. The number of method invocations in a class, weighted by the number of parameters of the invoked methods.	[LLWW95]
IH-ICP	As ICP, but counts invocations of methods of ancestors of classes (i.e., inheritance- based coupling) only.	[LLWW95]
NIH-ICP	As ICP, but counts invocations to classes not related through inheritance.	[LLWW95]
IFCAIC ACAIC OCAIC FCAEC DCAEC OCAEC IFCMIC ACMIC OCMIC FCMEC DCMEC OCMEC IFMMIC AMMIC OMMIC FMMEC DMMEC OMMEC	These coupling measures are counts of interactions between classes. The measures distinguish the relationship between classes (friendship, inheritance, none), different types of interactions, and the locus of impact of the interaction. The acronyms for the measures indicates what 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 (classes that declare a given class c as their friend), O: Others, i.e., none of the other relationships). The next two letters indicate the type of interaction: 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 (function pointer) to a method of class *c*. The last two letters indicate the locus of impact: IC: Import coupling, the measure counts for a class c all interactions where c is using another class. EC: Export coupling: count interactions where class d is the used class.	[BDM97]
IC	Inheritance Coupling: Number of parent classes to which a class is coupled.	[TKC99]
CBM	Coupling between methods: Number of function dependency relationships between inherited and new/redefined methods (should be similar or the same as AMMIC).	[TKC99]
CC	Class coupling - number of method couplings, i.e., variable references and/or method calls	[RL92]
AMC	Average method coupling - CC divided by number of methods in the class.	[RL92]

Name	Definition	Source
CBO	Coupling between object classes. According to the definition of this measure, a class is coupled to another, if methods of one class use methods or attributes of the other, or vice versa. CBO is then defined as the number of other classes to which a class is coupled. This includes inheritance-based coupling (coupling between classes related via inheritance).	[CK94]
NAS	Number of associations - count of the number of association lines emanating from a class in an OMT diagram.	[HCN98]
COF	Coupling Factor - percentage of pairs of classes that are coupled.	[AM96]
CDM	Coupling Dependency Metric. Sum of 1) referential dependency (extent to which a program relies on its declaration dependencies remaining unchanged), 2) structural dependency (extent to which a program relies on its internal organization remaining unchanged, 3) data integrity dependency (vulnerability of data elements in one module to change by other module)	[BS98]
Fan-In	Count of modules (classes) that call a given class, plus the number of global data elements	[BS98]
Fan-Out	Count of modules (classes) called by a given module plus the number of global data elements altered by the module (class)	[BS98]

Table 12: Cohesion measures

Name	Definition	Source
LCOM1	Lack of cohesion in methods. The number of pairs of methods in the class using no attribute in common.	[CK91]
LCOM2	LCOM2 is the number of pairs of methods in the class using no attributes in common, minus the number of pairs of methods that do. If this difference is negative, however, LCOM2 is set to zero.	[CK94]
LCOM3	Consider an undirected graph G, where the vertices are the methods of a class, and there is an edge between two vertices if the corresponding methods use at least an attribute in common. LCOM3 is defined as the number of connected components of G.	[HM95]
LCOM4	Like LCOM3, where graph G additionally has an edge between vertices representing methods *m* and *n*, if *m* invokes *n* or vice versa.	[HM95]
Co	Let V be the number of vertices of graph G from measure LCOM4, and E the number of its edges. Then $Co = 2(\lvert E \rvert - (\lvert V \rvert - 1))/((\lvert V \rvert - 1)(\lvert V \rvert - 2))$.	[HM95]
LCOM5	Consider a set of methods {M_i} (i=1,...,m) accessing a set of attributes {A_j} (j=1,...,a). Let $\mu(A_j)$ be the number of methods which reference attribute A_j. Then $LCOM5 = \frac{1}{a}((\sum_{j=1}^{a} \mu(A_j)) - m)/(1-m)$.	[Hen96]
Coh	A variation on LCOM5: $Coh = (\sum_{j=1}^{a} \mu(A_j))/(m \cdot a)$	[BDW98]
TCC	Tight class cohesion. Besides methods using attributes directly (by referencing them), this measure considers attributes *indirectly* used by a method. Method m uses attribute a indirectly, if m directly or indirectly invokes a method which directly uses attribute a. Two methods are called *connected*, if they directly or indirectly use common attributes. TCC is defined as the percentage of pairs of public methods of the class that are connected, i.e., pairs of methods which directly or indirectly use common attributes.	[BK95]
LCC	Loose class cohesion. Same as TCC, except that this measure also considers pairs of *indirectly connected* methods. If there are methods m_1,..., m_n, such that m_i and m_{i+1} are connected for i=1,...,n-1, then m_1 and m_n are indirectly connected. Measure LCC is the percentage of pairs of public methods of the class which are directly or indirectly connected.	[BK95]
ICH	Information-flow-based cohesion. ICH for a method is defined as the number of invocations of other methods of the same class, weighted by the number of parameters of the invoked method (cf. coupling measure ICP above). The ICH of a class is the sum of the ICH values of its methods.	[LLWW95]
MHF	MHF - method hiding factor, percentage of methods that have external visibility in the system (are not private to a class).	[AM96]
AHF	Attribute hiding factor, percentage of methods that have external visibility in the system.	[AM96]

Table 13: Inheritance Measures

Name	Definition	Source
DIT	Depth of inheritance Tree. The DIT of a class is the length of the longest path from the class to the root in the inheritance hierarchy.	[CK91]
AID	Average inheritance depth of a class. AID of a class without any ancestors is zero. For all other classes, AID of a class is the average AID of its parent classes, increased by one.	[Hen96]
CLD	Class-to-leaf depth. CLD of a class is the maximum number of levels in the hierarchy that are below the class.	[TSM95]
NOC	Number of children. The number of classes that directly inherit from a given class.	[CK91], [CK94]
NOP	Number of parents. The number of classes that a given class directly inherits from.	[LC94], [LK94]
NOD	Number of descendents. The number of classes that directly or indirectly inherit from a class (i.e., its children, 'grand-children', and so on)	[LC94], [TSM95]
NOA	Number of ancestors. The number of classes that a given class directly or indirectly inherits from.	[TSM95]
NMO	Number of methods overridden. The number of methods in a class that override a method inherited from an ancestor class.	[LK94]
NMINH	Number of methods inherited. The number of methods in a class that the class inherits from its ancestors and does not override.	[LK94]
NMA	Number of methods added. The number of new methods in a class, not inherited, not overriding.	[LK94]
SIX	Specialization index. SIX is NMO * DIT / (NMO+NMA+NMINH)	[LK94]
MIF	Method inheritance factor, percentage of methods that are inherited in the system (sum of all inherited methods in all classes, divided by sum of inherited and non-inherited methods in all classes)	[AM96]
AIF	Attribute inheritance factor, equivalent to MIF, for attributes	[AM96]
POF	Polymorphism Factor, percentage of possible opportunities for method overriding that are used.	[AM96]
INHTS	Dummy variable indicating if a class partakes in an inheritance relationship	[CS00]
SPA	Static polymorphism in ancestors	[BM99]
DPA	Dynamic polymorphism in ancestors	[BM99]
SPD	Static polymorphism in descendants	[BM99]
DPD	Dynamic polymorphism in descendants	[BM99]
SP	Static polymorphism in inheritance relations. SP = SPA + SPD	[BM99]
DP	Dynamic polymorphism in inheritance relations. DP = DPA + DPD	[BM99]
NIP	Polymorphism in non-inheritance relations.	[BM99]
OVO	Overloading in stand-alone classes	[BM99]
CHNL	Class Hierarchy Nesting Level (likely identical with DIT)	[BS98]
CACI	class attribute complexity/size, inherited	[NQ98]
CI	class method complexity/size, inherited	[NQ98]
CMICI	class method interface complexity/size, inherited	[NQ98]

Some of the size measures in Table 14 are frequently used in publications and available tools, and no definite source or author can be given for them.

Table 14: Size measures

Name	Definition	Source
NMImp	The number of methods implemented in a class (non-inherited or overriding methods)	
NMInh	The number of inherited methods in a class, not overridden	
NM	The number of all methods (inherited, overriding, and non-inherited) methods of a class. NM = NMImp + NMInh	
NAImp, Totattrib	The number of attributes in a class (excluding inherited ones). Includes attributes of basic types such as strings, integers.	
NAInh	The number of inherited attributes in a class	
NumPar	Number of parameters. The sum of the number of parameters of the methods implemented in a class.	
Stmts	The number of declaration and executable statements in the method of a class.	
NMpub	The number of public methods implemented in a class.	
NMN pub	The number of non-public (i.e., protected or private) methods implemented in a class.	
Attrib	count of attributes per class from the information model	[CS00]
States	count of states per class from the information model	[CS00]
EVNT	count of events per class from the information model	[CS00]
READS	Count of all read accesses by a class (contained in a case tool)	[CS00]
WRITES	Count of all write accesses by a class (contained in a case tool)	[CS00]
DELS	Count of all delete accesses by a class contained tin the case tool	[CS00]
RWD	Count of synchronous accesses per class from the case tool	[CS00]
LOC	Lines of code	[CS00]
LOC_B	C++ body file lines of code per class	[CS00]
LOC_H	C++ header files lines of code per class	[CS00]
NOMA	Number of Object/Memory Allocation: Number of statements that allocate new objects or memory in a class	[TKC99]
AMC-TKC99	Average Method Complexity: Average method size for each class	[TKC99]
CACL	class attribute complexity/size, local	[NQ98]
CL	class method complexity/size, local	[NQ98]
CMICL	class method interface complexity/size, local	[NQ98]

7 Glossary

This glossary provides a list of all abbreviations used throughout the paper. This excludes acronyms used as names of design measures, which are listed in the Appendix above.

ARE	absolute relative error, see Section 3.4.5
AROC	area under receiver-operator curve, Section 3.4.5
C&K	The measures defined by Chidamber and Kemerer [CK91, CK94]
CART	Classification and Regression Trees
CCI	Coupling, cohesion, inheritance Measures, i.e., measures of OO design properties, as opposed to (usually simple) size measures
C-FOOD	Coupling measures for Object-Oriented Designs - the measures defined in [BDM97]
CV	cross validation, see Section 3.5.1
DV	dependent variable
IV	independent variable
LL	log likelihood
LR	logistic regression
LS	least-squares regression
MARS	Multivariate Adaptive Regression Splines
ML	maximum likelihood
MOOD	Metrics for Object-Oriented Designs - the measures introduced in [AGE95]
MRE	magnitude of relative error, Section 3.4.5
OLS	ordinary least-squares regression
OO	object-oriented
PC	Principal component
PCA	Principal component analysis, Section 3.2
ROC	receiver operator curve, Section 3.4.5

8 References

[AGE95] F. Abreu, M. Goulão, R. Esteves, “Toward the Design Quality Evaluation of Object-Oriented Software Systems”, *5th International Conference on Software Quality*, Austin, Texas, USA, October 1995

[AM96] F. Abreu, W. Melo, “Evaluating the Impact of Object-Oriented Design on Software Quality”, Proceedings of Metrics 1996.

[Bar98] Barnard, J., “A new reusability metrics for object-oriented measures”, Software Quality Journal 7, 35-50, 1998.

[BBDD97] L. Briand, C. Bunse, J. Daly, C. Differding, “An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents”, Empirical Software Engineering 2 (3), 291-312, 1997.

[BBD97] L. Briand, C. Bunse, J. Daly, “An Experimental Evaluation of Quality Guidelines on the Maintainability of Object-Oriented Design Documents”, IEEE Transactions on Software Engineering 27(6), 513-530, 2001.

[BBM96] V.R. Basili, L.C. Briand, W.L. Melo, “A Validation of Object-Oriented Design Metrics as Quality Indicators”, IEEE Transactions on Software Engineering, 22 (10), 751-761, 1996.

[BDM97] L. Briand, P. Devanbu, W. Melo, “An Investigation into Coupling Measures for C++”, Proceedings of ICSE ‘97, Boston, USA, 1997.

[BDW98] L. Briand, J. Daly, J. Wüst, “A Unified Framework for Cohesion Measurement in Object-Oriented Systems”, Empirical Software Engineering Journal, 3 (1), 65-117, 1998.

[BDW99] L. Briand, J. Daly, J. Wüst, “A Unified Framework for Coupling Measurement in Object-Oriented Systems”, IEEE Transactions on Software Engineering 25(1), 91-121, 1999.

[BEDL99] Bansiya, J., Etzkorn, L., Davis, C., Li, W., “A Class Cohesion Metric For Object-Oriented Designs”, JOOP, Jan 1999, 47-52

[BEGR00] Benlarbi, S., El Emam, K., Goel, N., Rai, S., “Thresholds for Object-Oriented Measures”, Proceedings of ISSRE2000, 24-37.

[BFOS84] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. Classification and Regression Trees. Wadsworth & Books/Cole Advanced Books & Software (1984).

[BK95] J.M. Bieman, B.-K. Kang, “Cohesion and Reuse in an Object-Oriented System”, in Proc. ACM Symp. Software Reusability (SSR’94), 259-262, 1995.

[BKW80] D. Belsley, E. Kuh, R. Welsch, “Regression Diagnostics: Identifying Influential Data and Sources of Collinearity.” John Wiley & Sons, 1980.

[BM99] Benlarbi, S., Melo, W., “Polymorphism Measures for Early Risk Prediction”, Proceedings of the Proceedings of the 21st International Conference on Software Engineering, ICSE 99, (Los Angeles, USA 1999) 335- 344.

[BMB96] L. Briand, S. Morasca, V. Basili, “Property-Based Software Engineering Measurement”, IEEE Transactions of Software Engineering, 22 (1), 68- 86, 1996.

[BMW01] L. Briand, W. Melo, J. Wüst, “Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects”, to appear in IEEE Transactions on Software Engineering.

[BP95] V. Barnett, T. Price, “Outliers in Statistical Data”, 3rd Ed, John Wiley & Sons, 1995.

[BS96] Binkley, A., Schach, R., “Impediments to the Effective Use of Metrics within the Object-Oriented Paradigm”, TR, Univ. Vanderbilt

[BS98] Binkley, A., Schach, R., “Validation of the Coupling Dependency Metric as a Predictor of Run-Time Failures and Maintenance Measures”, Proc. ICSE98, 452-455

[BWL99] L. Briand, J. Wüst, H. Lounis, “Using Coupling Measurement for Impact Analysis in Object-Oriented Systems“, Proceedings of ICSM’99, 475- 482, 1999.;

[BW01] L. Briand, J. Wüst, “The Impact of Design Properties on Development Cost in Object-Oriented Systems”, to appear in IEEE Transactions on Software Engineering.

[BW01b] L. Briand, J. Wüst , “Integrating scenario-based and measurement-based software product assessment”, Journal of Systems and Software 59(1), p. 3-22, December 2001.

[BWDP00] L. Briand, J. Daly, V. Porter, J. Wüst, “Exploring the relationships between design measures and software quality in object-oriented systems”, Journal of Systems and Software 51, p. 245-273, 2000.

[BWL99] L. Briand, J. Wüst, H. Lounis, “Using Coupling Measurement for Impact Analysis in Object-Oriented System”, IEEE International Conference on Software Maintenance (ICSM), 1999, Oxford, UK.

[BWL01] L. Briand, J. Wüst, H. Lounis, “Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs, Empirical Software Engineering: An International Journal, Vol 6, No 1, 11-58, 2001.

[CDK98] S. Chidamber, D. Darcy, C. Kemerer, “Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis”, IEEE Transactions on Software Engineering, 24 (8), 629-639, 1998.

[CK91] S.R. Chidamber, C.F. Kemerer, “Towards a Metrics Suite for Object Oriented design”, in A. Paepcke, (ed.) Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA’91), October 1991. Published in SIGPLAN Notices, 26 (11), 197-211, 1991.

[CK94] S.R. Chidamber, C.F. Kemerer, “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, 20 (6), 476-493, 1994.

[CL93] J-Y. Chen, J-F Lu, “A New Metric for Object-Oriented Design”, Information and Technology Vol 35, No 5, 232-240.

[CS00] M. Cartwright, M. Shepperd, “An Empirical Investigation of an Object- Oriented Software System”, IEEE TSE 2000

[CN00] S. Counsell, P. Newson, “Use of friends in C++ Software: an empirical investigation”, Journal of Systems and Software 53, 15-21, 2000.

[Dun89] Dunteman, G., 1989. Principal Component Analysis. Sage University Paper 07-69, Thousand Oaks, CA.

[EBD99] L. Etzkorn, J. Bansiya, C. Davis, “Design and Code Complexity Metrics for OO Classes”, JOOP March/April 1999, 35-40

[EDW98] Etzkorn, L., Davis, C., Li, W., “A Practical Look at the Lack of Cohesion in Methods Metric”, JOOP September 1998, 27-34.

[EBGR01] El Emam, K., Benlarbi, S., Goel N., Rai, S., “The Confounding Effect of Class Size on The Validity of Object-Oriented Metrics”, IEEE Transactions on Software Engineering Vol 27. No 7, 630-650, 2001.

[EBML00] El Emam, K., Benlarbi, S., Melo, W., Lounis, H., Rai, S., “The Optimal Class Size for Object-Oriented Software: A Replicated Case Study”, Technical Report ERB-1074, NRC, 2000. Available at www.object-oriented.org

[EBGR99] El Emam, K., Benlarbi, S., Goel, N., Rai, S., “A Validation of Object-Oriented Metrics”, Technical Report ERB-1063, NRC, 1999. Available at www.object-oriented.org

[EKS94] J. Eder, G. Kappel, M. Schrefl, “Coupling and Cohesion in Object-Oriented Systems”, Technical Report, University of Klagenfurt, 1994.

[EMM01] El Emam, K., Melo, W., Machado, J., “The Prediction of Faulty Classes Using Object-Oriented Design Metrics”, Journal of Systems and Software 56, 63-75, 2001.

[Eve93] Everitt, “Cluster Analysis”, Edward Arnold, 1993.

[Fri91] J. Friedman, “Multivariate Adaptive Regression Splines”, The Annals of Statistics, Vol. 19, 1-141, 1991

[GEMM00] Glasberg, D., El Emam, K., Melo, W., Madhavji, N., “Validating Object- Oriented Design Metrics on a commercial Java application”, TR ERB- 1080, NRC, Sep. 2000.

[GS74] Green, D., Swets, J., “Signal Detection Theory and Psychophysics”, rev. ed. Huntington, NY: Krieger, 1974.

[Hal77] Halstead, H.M., “Elements of Software Science”, Elsevier North Holland, 1977.

[Hay94] Hayes W. Statistics. Fifth Edition, Hartcourt Brace College Publishers (1994).

[HC98] Harrison, R., Counsell, S., “The role of inheritance in the maintainability of object-oriented systems”, Proceedings of ESCOM ‘98, p. 449-457, 1998.

[HCN00] Harrison, R., Counsell, S., Nithi, R., “Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems”, Journal of Systems and Software 52, 173-179, 2000.

[HCN98] Harrison, R., Counsell, S., Nithi, R., “Coupling Metrics for Object-Oriented Design”, Proceedings of the 5th International Software Metrics Symposium, Bethesda, MD, 150-157, 1998.

[Hen96] B. Henderson-Sellers, “Software Metrics”, Prentice Hall, Hemel Hempstaed, U.K., 1996.

[HL89] D.W. Hosmer, S. Lemeshow, “Applied Logistic Regression”, John Wiley & Sons, 1989.

[HM95] M. Hitz, B. Montazeri, “Measuring Coupling and Cohesion in Object-Oriented Systems”, in Proc. Int. Symposium on Applied Corporate Computing, Monterrey, Mexico, October 1995.

[HN96] R. Harrison, R. Nithi, TR, “An Empirical Evaluation of Object-Oriented Design Metrics”, TR, 1996

[HSDL96] R. Harrison, L.G. Samaraweera, M.R. Dobie, P.H. Lewis, “An evaluation of Code Metrics for Object-Oriented Programs”, TR, 1996

[MHM99] Moser, Henderson-Sellers, Misic, “Cost Estimation Based on Business Models”, JSS 49, 1999, 33-42

[MT98] Misic, V., Tesic, D., “Estimation of effort and complexity: An object-oriented case study”, JSS 41, 1998, 133-143

[ISO14598] ISO/IEC DIS 14598-1, “Information Technology – Product Evaluation”, Part 1: General Overview.

[KPF95] B. Kitchenham, S. Pfleeger, N. Fenton, “Towards a framework for software measurement validation: A measurement theory perspective”, IEEE Transactions on Software Engineering 12, 929-944. 1995.

[LC92] A. Lake, C. Cook, “A Software complexity metric for C++”, Technical Report 92-60-03, Department of Computer Science, Oregon State University, 1992.”

[LC94] A. Lake, C. Cook, “Use of factor analysis to develop OOP software complexity metrics”, Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.

[LH93] W. Li, S. Henry, “Object-Oriented Metrics that Predict Maintainability”, J. Systems and Software, 23 (2), 111-122, 1993.

[LHKS95] Li, W. Henry, S., Kafura, D., Schulman, R., “Measuring Object-Oriented Design”, JOOP Vol 8, No 4, July/August 1995, 48-55

[Li98] W. Li, “Another metric suite for object-oriented programming”, Journal of Systems and Software 44 (1998), 155-162, 1998.

[LK94] M. Lorenz, J. Kidd, “Object-Oriented Software Metrics”, Prentice Hall Object-Oriented Series, Englewood Cliffs, N.J., 1994.

[LLWW95] Y.-S. Lee, B.-S. Liang, S.-F. Wu, F.-J. Wang, “Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow”, in Proc. International Conference on Software Quality, Maribor, Slovenia, 1995.

[Lon97] S. Long, “Regression Models for Categorical and Limited Dependent Variables”, Advanced Quantitative Techniques in the Social Sciences Series, Sage Publications, 1997.

[Lew80] M. Lewis-Beck, “Applied Regression: An Introduction”, Sage publications, 1980

[McC76] McCabe, T.J., A complexity measures, IEEE Transactions on Software Engineering 16 (5), 510-522, 1976.

[MB91] Meyer, M., Booker, J., “Eliciting and Analyzing Expert Judgement. A Practical Guide”, London: Academic Press, 1991.

[MC96] J. Mayrand, F. Coallier, “System Acquisition Based on Software Product Assessment”, Proceedings of ICSE’96, Berlin, Germany, 210-219,1996.

[Mil81] Miller, R. Jr., “Simultaneous Statistical Inference”, 2nd. ed., Springer Verlag, 1981.

[NQ98] P. Nesi, T. Querci, “Effort estimation and prediction of object-oriented systems”, Journal of Systems and Software 42, p. 89-102, 1998.

[Pre81] Pregibon, D., “Logistic Regression Diagnostics”, Annals of Statistics 9, 705-724, 1981.

[RL92] Rajaraman, Lyu, “Reliability and Maintainability Related Software Coupling Metrics in C++ Programs”, Proceedings of ISSRE 1992.

[Ros97] Rosenberg, J., “Some Misconceptions About Lines of Code”, Proceedings of the 4th International Software Metrics Symposium (Metrics ‘97), 137-142, 1997.

[SC93] Sharble, R., Cohen, S., “The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods”, Software Engineering Notes 18 (2), 60-73, 1993.

[SC99] D. Steinberg, N. Cardell, “The Hybrid CART-Logit Model in Classification and Data Mining”, Salford Systems, http://www.salford-systems.com, 1999.

[SM88] Shlaer, S., Mellor, S., “Object-Oriented Systems Analysis: Modeling the World in Data”, Yourdon Press, 1988.

[Spe81] Spector, P. “Research Design”, Newbury Park: Sage Publications, Quantitative Applications in the Social Sciences, 1981.

[Sto74] M. Stone, “Cross-validatory choice and assessment of statistical predictions”, J. Royal Stat. Soc., Ser. B 36, 111-147, 1974.

[TKC99] Tang, Kao, Chen, “An Empirical Study on Object-Oriented Metrics”, Proceedings of Metrics 1999, 242-249

[TSM95] D.P. Tegarden, S.D. Sheetz, D.E. Monarchi, “A Software Complexity Model of Object-Oriented Systems”, Decision Support Systems, 13(3-4), 241-262, 1995.

[WDMR99] Wood, M., Daly, J., Miller, J., Roper, M., “Multi-method research: An empirical investigation of object-oriented technology”, Journal of Systems and Software 48, 13-26, 1999.

[Whi97] Whitmire, S., “Object-Oriented Design Measurement”, John Wiley & Sons, 1997.

[WH98] Wilkie, Hylands, “Measuring Complexity in C++ Application Software”, Software Practice and Experience Vol 28 No 5, 1998, 513-546.

[WK00] Wilkie, G., Kitchenham, B., “Coupling Measures and Change Ripples in C++ Applications”, JSS 52, 2000, 157-264

[WWW90] Wirfs-Brock, R., Wilkersion, B., Wiener, L., “Designing Object-Oriented Software”, Prentice Hall, 1990.

[Zus98] Zuse, H., “A Framework of Software Measurement”, Walter de Gruyter, 1998.