Investigating Quality Factors in Object-Oriented Designs: an Industrial Case Study

Lionel C. Briand, Jürgen Wüst
Fraunhofer Institute for
Experimental Software Engineering
Sauerwiesen 6
67661 Kaiserslautern, Germany
+49 6301 707 251
{briand,wuest}@iese.fhg.de

Stefan V. Ikonomovski, Hakim Lounis
Centre de Recherche
Informatique de Montréal
550, Sherbrooke West, Suite 100
Montréal, Qc, Canada H3A 1B9
+1 514 840-1234
{sikonomo, hlounis}@crim.ca

ABSTRACT

This paper aims at empirically exploring the relationships between most of the existing coupling and cohesion measures for object-oriented (OO) systems, and the fault-proneness of OO system classes. The underlying goal of such a study is to better understand the relationship between existing design measurement in OO systems and the quality of the software developed.

The study described here is a replication of an analogous study conducted in an university environment with systems developed by students. In order to draw more general conclusions and to (dis)confirm the results obtained there, we now replicated the study using data collected on an industrial system developed by professionals.

Results show that many of our findings are consistent across systems, despite the very disparate nature of the systems under study. Some of the strong dimensions captured by the measures in each data set are visible in both the university and industrial case study. For example, the frequency of method invocations appears to be the main driving factor of fault-proneness in all systems. However, there are also differences across studies which illustrate the fact that quality does not follow universal laws and that quality models must be developed locally, wherever needed.

Keywords
metrics, measurement, empirical validation, coupling, cohesion, object-oriented.

1 INTRODUCTION

A large number of object-oriented (OO) measures have been proposed in the literature ([2], [8], [9], [10], [13], [15], [17], [18]). A particular emphasis was given to the measurement of design artifacts, in order to help assess quality early on during the development process. However, many of the measures proposed and their relationships to external quality attributes of OO designs, have been the focus of little empirical investigation ([1], [2], [3], [17]). It is therefore difficult to assess whether these measures capture similar dimensions and are indicators of any relevant quality attribute at all.

Recently, some of the authors performed an in-depth, comprehensive analysis of most of the literature OO measures on students' projects [3] of rather small sizes. The goal was to look at the relationship between these measures and the likelihood of detecting a fault in a class during testing, i.e., its fault-proneness. The high cognitive complexity of classes may result in many different types of problems such as low maintainability or high fault-proneness. However, fault-proneness is not only one important quality aspect related to class cognitive complexity but also the easiest one to observe and measure, hence its use in our studies.

In order to draw more general conclusions and (dis)confirm the results obtained in our student experiments, we replicate here this analysis on data we collected on an industrial project which is currently under use and maintenance. By analyzing carefully the results and by comparing them in a systematic way with the results obtained from the students' projects, we identified a number of structural dimensions in OO designs that appear to be related to class fault-proneness across the two data sets. Considering the significant differences between the students' systems and the industrial system studied here (e.g., in terms of size, domain, programmer experience), we hope to draw conclusions that should be robust across many systems. Further replication is of course necessary to build an adequate body of knowledge regarding the use of OO design measures.

The paper is organized as follows. Section 2 describes the goals and setting of the empirical study, and the data collected. Section 3 describes the methodology used to analyze the data. The results of this analysis are then presented in Section 4, where we also compare the results to those obtained for the students' systems in [3]. We draw our conclusions in Section 5.

2 THE EMPIRICAL STUDY

The goal of this study is to empirically assess the object-oriented design measures discussed in a literature review [5][6], and compare the results to those obtained in an analogous study using systems developed by students.

2.1 Dependent Variable

We want to evaluate whether existing measures are useful for predicting the probability that a fault occurs in a class during operation of the system. More precisely, the probability of fault detection that is meant here is a conditional probability: the probability that at least one fault is regis-

ICSE '99 Los Angeles CA

Name	Definition	Src.
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).	[10]
CBO'	Same as CBO, except that inheritance-based coupling is not counted.	[9]
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.	[9]
RFC_1	Same as RFC_∞, except that methods indirectly invoked by methods in M are not included in the response set.	[10]
MPC	Message passing coupling. The number of method invocations in a class.	[17]
DAC	Data abstraction coupling. The number of attributes in a class that have another class as their type.	[17]
DAC'	The number of different classes that are used as types of attributes in a class.	[17]
ICP	Information-flow-based coupling. The number of method invocations in a class, weighted by the number of parameters of the invoked methods.	[18]
IH-ICP	As ICP, but counts invocations of methods of ancestors of classes (i.e., inheritance-based coupling) only.	[18]
NIH-ICP	As ICP, but counts invocations to classes not related through inheritance.	[18]

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.
		[2]

Table 1: Coupling measures

tered during operation in a class, depending on the obtained measurement values of the independent variables for that class. This should be a good indicator of its probability of containing a fault and, therefore, a valid measure of fault-proneness. The construct validity of our dependent variable can thus be considered satisfactory.

Other measures such as class fault density could have been used. However, the variability in terms of number of faults in our data set is small: Faults were detected in 55% of the classes, and 80% of the classes contain less than three faults. Therefore, using a dependent variable with low variability would have affected our ability to identify significant relationships between OO design measures and this dependent variable.

2.2 Independent Variables

The measures of coupling and cohesion identified in a literature survey on object-oriented design measures [5][6] are the independent variables used in this study. We only use measures defined at the class level, because this is also the granularity at which the fault data was collected.

Tables 1 and 2 describe the coupling and cohesion measures used in this study. 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 of the measures using a uniform and unambiguous formalism are provided in [5][6], where we also perform a systematic comparison of these measures, and analyze their mathematical properties.

In order to make possible the use of these measures at the design stage, we adapted some of the measures involving counts of method invocations as follows. Measures that are based on counts of multiple invocations of pairs of methods (say methods *m'* and *m*) were changed to be solely sensitive to the fact that a given method invokes another one at least once. The rationale for this decision is that the precise number of times a given method *m'* invokes *m* is an information which is available only after implementation is completed, whereas the information that *m'* invokes *m* is usually available earlier during the design phase. The measures affected by this simplification are MPC, the ICP measures, the method-method interaction measures by Briand et al [2], and ICH.

2.3 Description of the empirical study

This subsection provides details about the LALO system and the fault data and design measurement data collected.

2.3.1 Setting of the Study

The data were collected from an open multi-agent system development environment, called LALO (Langage d'Agents Logiciel Objet). This system has been developed

Name	Definition	Src.
LCOM1	Lack of cohesion in methods. The number of pairs of methods in the class using no attribute in common.	[9]
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.	[10]
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.	[15]
LCOM4	Like LCOM3, where graph G additionally has an edge between vertices representing methods *m* and *n*, if *m* invokes *n* or vice versa.	[15]
Co	Connectivity. Let V be the number of vertices of graph G from measure LCOM4, and E the number of its edges. Then $Co = 2(\lvert E\rvert - (\lvert V\rvert - 1))/((\lvert V\rvert - 1)(\lvert V\rvert - 2))$.	[15]
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 $LCOM5 = \frac{1}{a}((\sum_{j=1}^{a}\mu(A_j)) - m)/(1-m)$.	[15]
Coh	A variation on LCOM5: $Coh = (\sum_{j=1}^{a}\mu(A_j))/(m \cdot a)$	[2]
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 which are connected, i.e., pairs of methods which directly or indirectly use common attributes.	[8]
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.	[8]
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.	[18]

Table 2: Cohesion Measures

and maintained since 1993 at CRIM (Centre de Recherche Informatique de Montréal); it includes 90 C++ classes with approximately 40K source lines of code (SLOC). Classes automatically generated by software tools, e.g., OO lex/yacc are included in this amount. Therefore, in the analysis below, these classes were not investigated since they are much less likely to contain faults than classes implemented manually. In fact, the use of these classes could have biased the results.

In addition to the 90 application-specific classes, a number of standard library classes for IO, threading, socket communication, etc., are used in the LALO system.

LALO was mostly developed under Windows NT using Visual C++ and then ported to Sun OS and Solaris. We have investigated only the Sun OS version.

Six developers have worked on the LALO system over its lifetime, with at most three developers working on the system in parallel. All developers had several years of previous experience in system development, and four developers have worked on OO systems before.

2.3.2 Data Collection Procedures and Measurement Instruments

The following relevant items were collected:

- the source code of the LALO system.
- data about faults found by world-wide users of LALO, over a period of about one year.

A tool developed at the Fraunhofer IESE, and based on the FAST parser technology of the SEMA group's Concerto2/AUDIT tools [12], was used to extract the values for the object-oriented measures directly from the source code of LALO. To collect the fault data, change report forms (CRF) were used to document the nature of each problem reported by a LALO user, the names of the faulty C++ classes, and the type and location of the maintenance change. The CRFs were registered in a revision control system, which could then be used to generate statistics about the number of faults traced back to each individual class.

2.3.3 Data Collected

LALO consists of a total of 90 classes. Of these 90 classes, seven were automatically generated by code generators. The other 83 classes were developed from scratch or reused with extensive modifications. For these 83 system classes, the values for each of the design measures were collected.

At this stage, it is pertinent to consider the influence of the library classes and the automatically generated classes (for simplicity, we collectively refer to these classes as 'library classes' thereafter). For coupling measures, a decision regarding whether or not to count coupling to library classes will have an impact on the computed measures' values. We hypothesized that a class is more likely to be fault prone if it is coupled to a system class than if it is coupled to a library class (although this may be dependent on the experience of the developer with the class library being used, or the quality of the library documentation). Consequently, the results for each coupling measure were calculated twice for each system class: counting coupling to other system classes only, and counting coupling to library classes only.

Analysis was then performed on both resulting data sets.

2.4 Comparison to students' systems

The results of the current study will be compared to those obtained in a previous study [3], where we used data collected from a development project performed at the University of Maryland (UMD) over a four-month period. Eight different groups of developers, composed of undergraduate and postgraduate students, were asked to develop an information system each. The systems were implemented in C++, and ranged in size from 4 to 15 KSLOC. The eight systems contained a total of 113 non-library classes.

The independent variables were the same as described in Section 2.2. The fault data used in that study stemmed from a thorough acceptance test performed on each system by an independent group composed of experienced software professionals. See [3] for more details on the setting of that study.

3 DATA ANALYSIS METHODOLOGY

In this section we describe the methodology used to analyze the coupling and cohesion measurement data collected for the 83 system classes. The analysis procedure comprises an analysis of the descriptive statistics, principal component analysis, univariate regression analysis against the fault data, and correlation to size. We now describe these techniques in some detail.

3.1 Descriptive statistics

The data set consists of 83 classes along with the relevant values for each coupling and cohesion measure. The distribution and variance of each measure is examined to select those with enough variance for further analysis. Low variance measures do not differentiate classes very well and therefore are not likely to be useful predictors in our data set.

3.2 Principal component analysis

If a group of variables in a data set are strongly correlated, these variables are likely to measure the same underlying dimension (i.e., class property) of the object to be measured. Principal component analysis (PCA) is a standard technique to identify the underlying, orthogonal dimensions that explain relations between the variables in the data set.

Principal components (PCs) are linear combinations of the standardized independent variables. The sum of the square of the coefficients in each linear combination is equal to one. PCs are calculated as follows. The first PC is the linear combination of all standardized variables which 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 have large coefficients - also called the loading of the variable - and therefore contribute significantly to the variance of each PC. The variables with high loadings help identify the dimension the PC is capturing but this usually requires some degree of interpretation.

In order to identify these variables, and interpret the PCs, we consider the rotated components. This is a technique where the PCs are subjected to an orthogonal rotation. As a result, the 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. We used the *varimax* rotation, which is the most frequently used strategy in the literature. See [11] for more details on PCA and rotated components.

3.3 Univariate regression analysis

Univariate regression analysis is performed for each individual measure (independent variable) against the dependent variable, i.e., no fault/fault detection, in order to determine if the measure is a useful predictor of fault-proneness.

The dependent variable we use to validate the design measures is binary, i.e., was a fault reported by a user traced back to a class during the maintenance phase? Therefore, we use logistic regression, a standard technique based on maximum likelihood estimation, for the regression analysis. In the following, we give a short introduction to logistic regression, full details can be found in [14] or [16].

The logistic regression model is based on the following relationship equation:

$$\pi(X) = \frac{e^{(c_0+c_1X)}}{1+e^{(c_0+c_1X)}}$$

π is the probability that a fault was found in a class during the validation phase, and X is the design measure. The curve between π and X takes a flexible S shape which ranges between two extreme cases:

- When X is not significant, then the curve approximates a horizontal line, i.e., π does not depend on X.
- When X entirely differentiates fault-prone software parts, then the curve approximates a step function.

The coefficients c_0 and c_1 are estimated through the maximization of a likelihood function L, built in the usual fashion, i.e., as the product of the probabilities of the single observations, which are functions of the covariates (whose values are known in the observations) and the coefficients (which are the unknowns). For mathematical convenience, $l=\ln[L]$, the loglikelihood, is usually the function to be maximized. This procedure assumes that all observations are statistically independent. In our context, an observation is the (non) detection of a fault in a C++ class. Each (non) detection of a fault is assumed to be an event independent from other fault (non) detections. Each data vector in the data set describes an observation and has the following components: an event category (fault, no fault) and a set of OO design measures (described in Section 2.2).

$\Delta\psi$, which is based on the notion of the odds ratio [14], provides an evaluation of the impact of the measure on the dependent variable. More specifically, the odds ratio $\psi(X)$ represents the ratio between the probability of having a fault and the probability of not having a fault when the value of the measure is X. As an example, if, for a given value X, $\psi(X)$ is 2, then it is twice as likely that the class does contain a fault than that it does not contain a fault. The value of $\Delta\psi$ is computed by means of the following formula:

$$\Delta\psi = \frac{\psi(X+\sigma)}{\psi(X)}$$

σ is the standard deviation of the measure. Therefore, $\Delta\psi$ represents the reduction/increase in the odds ratio when the value X increases by one standard deviation. This is designed to provide an intuitive insight into the impact of independent variables. However, as we will see in Section 4, some measures display very extreme outliers which inflate the standard deviation of those measures. The $\Delta\psi$s then can no longer be reasonably interpreted. Therefore, outliers were excluded for the calculation of the $\Delta\psi$s.

3.4 Correlation to size

For each measure, we analyze its relationship to the size of the class. This is to determine empirically whether the measure, even though it is assumed to be a coupling or cohesion measure, is essentially measuring size. This is important for several reasons. First, if a measure is strongly related to size, then it might shed light on its relationship with fault-proneness: bigger classes are more likely to contain faults. Recall that we are interested in increasing our understanding of OO code and design quality, independently of its size. Second, a model that systematically identifies bigger classes as more fault-prone is a priori less useful: the predicted fault-prone classes are likely to cover a larger part of the system, the model thus could not help to focus inspection and testing efforts very well.

In this study, we measure the size of a class in terms of the number of methods implemented in the class. We then calculate Spearman's Rho coefficient between each design measure and size.

3.5 Comparison to previous study

One focus of this paper to compare the results obtained from LALO to those obtained to from the systems analyzed in [3] (referred to thereafter as the "UMD systems"). Therefore, in the analyses below, we include a systematic comparison of the results with this previous study and try to explain differences and common observation. Since the systems studied are very different in nature, this should allow us to identify what results are more likely to be generalizable.

4 ANALYSIS RESULTS

In this section, we discuss for the coupling and cohesion measures separately, the descriptive statistics, principal component analysis, univariate analysis, and correlation to size, and compare the results to those obtained with the UMD systems. In Section 4.3, we briefly summarize the results from building and evaluating a multivariate prediction model.

4.1 Coupling Results

Table 3 summarizes the descriptive statistics, univariate analysis, and correlation to size for the coupling measures. The left half of the table provides the data for the measures counting coupling to non-library classes only, the right half for the measures counting coupling to library classes only. Columns "Max", "Mean" and "σ" state, for each measure, the maximum value, mean value, and standard deviation. From univariate analysis, the regression coefficient and standard error is provided (Columns „Coeff." and „S.E. "), the $\Delta\psi$ value as defined in Section 3.3, and the statistical significance (p-value) of the regression coefficient. For measures which also have a significant relationship to size (at α=0.05), Spearman's Rho coefficient with size is given in column "Rho".

4.1.1 Descriptive statistics

- The measures that count coupling to friend classes (the F***C and IF***C measures) are all zero. That is, there are no friendship relationships in the system.
- There is little inheritance coupling, as can be seen by the low mean and standard deviation of the measures which count this type of coupling: NIH-ICP, the A***C and D***C measures. Measures ACAIC and DCAIC have only one class with a non-zero value. There is no inheritance coupling to library classes. Therefore, ICP and NIH-ICP yield identical values, as do MPC and OMMIC.
- Overall, there is only very little coupling to library classes. 60% of the LALO classes only interact with other LALO classes.
- There is evidence of export coupling to library classes. One of the automatically generated classes in the LALO system uses some of the 83 non-library classes.

Comparison to UMD systems

For the UMD study, we investigated eight independent, smaller systems, whereas in the current study, we have one large system. Therefore, overall there is more coupling present in the LALO system, especially the measures which involve method invocations have higher means and standard deviations than in the UMD systems. However, there are two exceptions to this:

- There is less aggregation coupling in the LALO system (in this paper by aggregation we mean instances where a class has an attribute whose type is another class). In particular, there is no aggregation coupling to library classes, which is to be expected for the kinds of libraries used (IO, threading).
- Unlike in the UMD systems, there is no friendship coupling in the LALO system. This was considered bad practice and was avoided, e.g., by introducing access methods to set and retrieve values of class attributes, whenever this was required.

4.1.2 Principal Component Analysis

Since each coupling measure has been measured twice (once counting coupling to library classes only, once counting coupling to non-library classes only), we consider the two versions of each measure to be distinct measures. To distinguish them, we denote the version counting coupling to library classes by appending an „_L" to its name. For example, MPC_L denotes the measure that counts invocations of methods from library classes, whereas MPC denotes the measure that counts invocations of method from non-library classes.

For the PCA, measures that did not vary were discarded. From pairs of measures with identical values, one redundant measure was removed. PCA with the remaining measures identified seven PCs which capture 82% of the data set variance.

	Coupling to non-library classes only								Coupling to library classes only							
	Descriptive Stats.			Univariate Analysis				Size	Descriptive Stats.			Univariate Analysis				Size
Measure	Max	Mean	σ	Coef.	S.E.	Δψ	p	Rho	Max	Mean	σ	Coef.	S.E	Δψ	p	Rho
CBO	31	7.18	6.66	0.404	0.084	5.5	<.0001	0.39	2	0.422	0.587	2.108	0.47	3.45	<.0001	0.32
CBO'	31	6.61	6.65	0.447	0.091	4.44	<.0001	0.39	2	0.422	0.587	2.108	0.47	3.45	<.0001	0.32
RFC1	367	46.31	53.41	0.024	0.007	2.14	0.0005	0.52	27	2.458	5.61	0.535	0.165	2.01	0.0013	0.31
RFC∞	638	101.0	124.8	0.0051	0.001	1.57	0.0070	0.32	118	31.94	41.10	0.016	0.005	1.65	0.0006	0.20
MPC	274	16.21	37.45	0.099	0.03	1.73	0.0008	0.40	73	3.47	10.23	0.687	0.231	3.34	0.0029	0.36
ICP	676	49.37	105.4	0.051	0.012	2.57	0.0001	0.42	201	9.446	27.88	0.257	0.087	3.61	0.0031	0.36
IH-ICP	50	4.916	8.857	0.080	0.030	1.17	0.0078	n. s.	0	0	0	All zero				
NIH-ICP	626	44.46	101.3	0.071	0.017	2.74	<.0001	0.46	201	9.446	27.88	0.257	0.087	3.61	0.0031	0.36
DAC	8	1.19	1.48	0.257	0.124	1.46	0.0386	0.52	0	0	0	All zero				
DAC'	7	1.00	1.22	0.304	0.147	1.45	0.0391	0.51	0	0	0	All zero				
IFCAIC	0	0.00	0.00	All zero					0	0	0	All zero				
ACAIC	2	0.02	0.22	Only one non-zero data point					0	0	0	All zero				
OCAIC	8	1.17	1.48	0.241	0.121	0	0.0467	0.49	0	0	0	All zero				
FCAEC	0	0.00	0.00	All zero					0	0	0	All zero				
DCAEC	2	0.02	0.22	Only one non-zero data point					0	0	0	All zero				
OCAEC	16	1.17	2.60	0.588	0.242	1.46	0.0152	n. s.	6	0.084	0.666	Only two non-zero data points				
IFCMIC	0	0.00	0.00	All zero					0	0	0	All zero				
ACMIC	8	0.81	1.48	0.339	0.152	1.23	0.0261	0.28	0	0	0	All zero				
OCMIC	205	9.39	23.64	0.1015	0.036	2.01	0.0052	0.52	2	0.217	0.443	3.179	1.019	4.09	0.0018	0.42
FCMEC	0	0.00	0.00	All zero					0	0	0	All zero				
DCMEC	38	0.81	4.43	0.035	0.066	1.17	0.6019	n. s.	0	0	0	All zero				
OCMEC	135	9.39	23.25	0.093	0.035	1.37	0.0086	n. s.	62	0	8.412	1.82	0.458	2.40	<.0001	0.35
IFMMIC	0	0.00	0.00	All zero					0	0	0	All zero				
AMMIC	24	2.11	4.02	0.19	0.076	1.40	0.0121	n. s.	0	0	0	All zero				
OMMIC	250	14.10	35.34	0.096	0.03	1.56	0.0014	0.36	73	3.47	10.23	0.687	0.231	3.34	0.0029	0.36
FMMEC	0	0.00	0.00	All zero					0	0	0	All zero				
DMMEC	69	2.11	8.53	0.138	0.092	1.20	0.1349	n. s.	0	0	0	All zero				
OMMEC	124	14.10	23.71	0.051	0.017	1.54	0.0025	0.47	26	0.542	2.91	0.826	0.377	1.45	0.0282	0.23

Table 3: Analysis results for coupling measures

In the following, we provide for each PC the percentage of the data set variance the PC describes, a list of the measures with high loadings in the PC, and our interpretation of the dimension that the PC captures.

- PC1 (27%): MPC, ICP, NIH-ICP, and OMMIC: measure the extent of import coupling through method invocations to non-library classes.
- PC2 (15%): OCAEC, ACMIC, OCMEC, and CBO_L. When considering all variables, this PC is difficult to interpret. However, the two variables with the highest loadings, OCAEC and OCMEC, capture export coupling to non-library classes. This will be our tentative interpretation for this PC.
- PC3 (12%): RFC_1_L, MPC_L, ICP_L, and OCMIC_L: MPC_L and ICP_L count import coupling through method invocations to library classes. RFC_1_L captures the number of methods invoked plus the local methods, and is therefore expected here. OCMIC_L also counts import coupling to library classes through CM interactions, but its loading is comparatively low. We therefore interpret this PC as capturing import coupling through method invocations to library classes.
- PC4 (11%): DAC, DAC', OCAIC, OCMIC: The first three measures are the strongest and count import coupling through aggregation relationships to non-library classes.
- PC5 (8%): RFC_1, RFC_∞, IH-ICP, AMMIC. Measures IH-ICP and AMMIC count import coupling through method invocations to non-library ancestor classes. The correlation of these measures to the RFC measures was also observed in the UMD systems. The explanation is that classes which import from ancestors also inherit methods from their ancestors. These inherited methods are part of the "response set" of the class (see the definition of RFC in Table 1) and thus counted by the measures. Hence, the RFC measures tend to be larger for descendent classes. Because the inheritance-based import coupling measures are non-zero for descendent classes only, they have a positive correlation to RFC_1 and RFC_∞.
- PC6 (5%): This PC is determined by the measure DCMEC only.

- PC7 (5%): CBO, CBO', OMMEC, OCMEC_L. This PCs cannot be reasonably interpreted. It is common in principal component analysis that the weaker PCs explaining a small amount of variance are difficult to interpret.

Comparison to UMD systems
There are a number of orthogonal coupling dimensions common to both systems: the dimensions represented by PC1 (method invocations to non-library classes), PC3 (method invocations to library classes), and PC4 (import aggregation coupling) are also present in the UMD systems.

Some dimensions identified in the UMD systems could not be observed here, because the corresponding measures had little or no variation in the LALO system, e.g., import aggregation coupling to library classes, import and export coupling to friend classes.

4.1.3 Univariate logistic regression
As we can see in Table 3, most of the measures have a significant relationship to fault-proneness (at α=0.05). The exceptions are DCMEC and DMMEC of PC6, which count export coupling to descendent classes.

For all significant measures, the regression coefficients are positive. This is consistent with the common notion that classes with higher import or export coupling are more likely to be fault-prone.

The impact of export coupling on fault-proneness is weaker than that of import coupling: the export coupling measures mostly have lower coefficients and $\Delta\psi$s than their import coupling counterparts.

The CBO measures are the only measures which count both import and export coupling. Their relationship to fault-proneness is particularly strong (high coefficients and $\Delta\psi$s).

Comparison to UMD systems
Similar to the results obtained with the UMD systems, all import coupling measures with sufficient variation were found to be significant predictors of fault-proneness.

However, in the UMD systems, none of the export coupling measures was found to be significantly related to fault-proneness in the expected direction. In the LALO system, most of the export coupling measures that do vary also are indicators of fault-proneness. Maybe, because of the weaker impact of export coupling, and because there was overall less coupling in the UMD systems, thus resulting in less statistical power, we failed to find a statistically significant relationship to fault-proneness in those systems.

4.1.4 Correlation to size
The non-library versions of measures DAC, DAC', OCAIC, and OCMIC (all in PC4), and RFC_1, have the strongest relationship to size. For OCMIC this may be explained because the more methods a class has, the more method parameters there are, the higher OCMIC is likely to be (which counts the number of method parameters that have an "other" class as their type). RFC_1 includes a count of the number of methods; therefore a correlation to size is expected here. However, the Rho coefficients for all these measures are only barely above 0.5, i.e., the relationship to size is not very strong. For all other coupling measures, significant Rho coefficients are below 0.5, that is, there is at most a moderate correlation to size for these measures, if any.

Comparison to UMD systems
Common to the LALO and UMD systems is that overall a correlation to size is present, but it is weak.

4.2 Cohesion Results
Table 4 presents the descriptive statistics, univariate analysis, and correlation to size for all cohesion measures. The meaning of the columns is the same as in Table 3.

4.2.1 Descriptive statistics
ICH, LCOM1 and LCOM2 have extreme outliers. For the LCOM measures, this is due to the presence of access methods. These methods usually only reference one attribute, and therefore increase the number of pairs of methods in the class that do not use attributes in common.

Comparison to UMD systems
Overall, the distributions of the individual measures in terms of their mean and standard deviations are very similar to the UMD systems. This is to be expected, as the cohesion measures are concerned with the internal structure of each individual class. Unlike the coupling measures, they are not strongly affected by the overall size of the systems.

It is interesting to see that both studies' results show normalized cohesion measures (i.e., Coh, Co, LCOM5, LCC, TCC) with mean and median values far below 1. In light of the relatively high experience of the LALO developers, the question is now whether achieving values near 1 is a realistic expectation for such measures and, consequently, how should we interpret them?

4.2.2 Principal Component Analysis
PCA identified three PCs which describe 91% of the variance in the data set. Below, we provide for each PC the percentage of the data set variance the PC describes, the list of the measures with high loadings in the PC, and our interpretation of the dimension that the PC captures:

- PC1 (55%): LCOM5, Coh, Co, and TCC: These are normalized cohesion measures which are based on attribute usage by methods. Also common to these measures is that they do not take the transitive closure of the attribute-usage relationship between methods into account.
- PC2 (26%): LCOM1, LCOM2, and ICH. This PC is difficult to interpret. LCOM1 and LCOM2 are non-normalized measures based on common attribute usage between methods. ICH is a count of method invocations in a class. Since these measures are in the same PC indicates that the more the methods of a class invoke each other (high ICH), the less likely they are to use attributes in common (high LCOM1 and LCOM2). An explanation for this may be the presence of access methods in many classes. If the methods of a class do not reference the attributes of their class directly, but via access methods, ICH is artificially increased (invocation of access methods), and both LCOM1 and LCOM2 are also artificially increased (fewer direct references to attributes).

Measure	Descriptive Statistics			Univariate Analysis				Size
	Max	Mean	σ	Coef.	S.E.	p	Δψ	Rho
LCOM1	5437	139.6	598.3	0.02	0.006	0.0004	3.94	0.81
LCOM2	4988	99.59	547.7	0.011	0.005	0.0249	1.61	0.31
LCOM3	56	5.867	6.320	0.135	0.063	0.0316	1.50	0.24
LCOM4	12	4.952	2.837	Not significant				n. sig.
LCOM5	1.25	0.621	0.257	Not significant				n. sig.
Coh	1	0.429	0.231	Not significant				n. sig.
Co	0.5	0.112	0.201	Not significant				0.35
LCC	1	0.484	0.300	1.712	0.696	0.0140	1.67	0.41
TCC	1	0.383	0.254	Not significant				n. sig.
ICH	343	9.615	42.55	0.700	0.214	0.0010	3.53	0.54

Table 4: Analysis results for cohesion measures

- PC3 (10%): LCOM3, LCOM4, LCC, TCC. These measures operate on graphs whose vertices represent the methods of a class, and edges between nodes represent common-attribute usage relationships (the precise definitions of the graphs differ slightly between measures, see their definitions in Table 2). This PC captures the degree of connectivity in these graphs. LCOM3 and LCOM4 count the connected components of their respective graphs, LCC is a (normalized) count of the edges of a graph that takes the transitive closure of common attribute-usage relationships into account. Whenever the graphs consist of few connected components (low values of LCOM3 and LCOM4), the number of edges of the transitive closure of the graph is large (high values of LCC). This similarity was observed in [6], based on a comparison of the definitions of the measures, and is confirmed by empirical evidence in this study. The above interpretation does not apply to TCC, which is also present in this PC, but its loading is comparatively low here.

Among the cohesion measures considered here are variants of the same concept. These variants were mostly defined with the intention to improve existing measures by eliminating problems that were identified based on theoretical considerations (see [6] for a summary of these discussions). From a practical perspective, these differences in the definitions do not seem to matter much, because the related measures lie within the same PCs: LCOM5 and Coh in PC1, LCOM1 and LCOM2 in PC2, LCOM3 and LCOM4 in PC3, TCC and LCC in PC3, even though TCC tends more towards PC1.

There appears to be a separation between normalized and non-normalized measures. PC1 consists of normalized measures only, PC2 only of non-normalized measures. PC3, however, contains a mix of one normalized and non-normalized measures, for reasons explained above.

Comparison to UMD systems

The separation between normalized and non-normalized measures observed in the LALO system was even stronger visible in the UMD systems.

The fact that measures which are variants of the same concept show up in the same PC was also observed in the UMD systems. From a practical perspective, it means that although those variant measures were deemed important from a conceptual perspective, they do not seem to make a tangible difference.

Notable differences between the systems: LCOM1,2,3 and 4 all were in one PC in the UMD systems, and ICH defined a dimension of its own. The reason is that, as explained above, these measures are affected by access methods. It is important to note that access methods are present in the LALO system, but not in the UMD systems.

4.2.3 Univariate logistic regression

Five of the ten cohesion measures are significant at α=0.05:

- The significant measures LCOM1, LCOM2, and LCOM3 show positive correlation coefficients. This indicates that the higher the values of these measures, the more fault-prone the class is likely to be. This is consistent with the common belief that low cohesion is bad design.
- As was discussed in [3], the mathematical properties of ICH make it unlikely to be measuring cohesion. ICH possesses properties of a complexity measure. With this information, the positive coefficient of ICH is reasonable: the higher ICH (and thus class complexity), the more fault-prone the class.
- For LCC, the positive coefficient indicates that fault-proneness increases with class cohesion, which is counter-intuitive. As we will see below, LCC is also positively correlated to size, which may explain this unexpected relationship to fault-proneness.

All other measures (LCOM4, LCOM5, Coh, Co, TCC) are not significant predictors.

All measures of PC2 are significant: LCOM1, LCOM2, and ICH. Two of these measures, LCOM1 and ICH are correlated to size. None of the measures in PC1 is significant. This PC contains normalized measures, which we consider a necessary property of a cohesion measure [7], but which appears not to be effective. In PC3 (connectivity), only LCOM3 is significant at α=0.05 (in the expected direction).

We would expect that variants of the same measure that lie in the same PC also have similar results for univariate analysis. This is confirmed by LCOM5 and Coh of PC1, which both are not significant, and LCOM1 and LCOM2 in PC2, both significant with similar coefficients and p-values. For LCOM3 and LCOM4 in PC3, however, despite

the similar distributions and strong correlation between the measures (Spearman's Rho 0.93), the difference of the p-values is substantial. LCOM3 is significant at α=0.05, LCOM4 is not significant at all. Note that an outlier of LCOM3 (the maximum value of 56) is not influential and has no impact on the significance of LCOM3. Likewise, LCC is significant, but not TCC, despite their strong correlation (Spearman's Rho 0.90).

Comparison to UMD systems

From the cohesion measures found significant in the LALO system, LCOM3 and ICH were also significant in the UMD systems. Coh was significant in the UMD systems, but not in the LALO system. All other measures, including LCOM1, LCOM2, and LCC, were not significant in the UMD systems.

4.2.4 Correlation to size

LCOM1 is very strongly correlated to size, which may explain its significant relationship to fault-proneness. The correlation to size is due to the presence of access methods in some classes, which artificially increases the number of pairs of methods that do not use attributes in common.

We saw that the presence of access methods has strong impact on some of the measures in PCA, univariate analysis and correlation to size. Several authors suggested modifications to the definition of these measures to better account for the presence of access methods (see [6] for a summary of these discussions). However, for a completely automated static analysis of the systems, there must be a way to automatically recognize access methods (e.g., a consistently applied naming scheme that uniquely identifies access methods by "set_" and "get_" name prefixes, which must not be used for any other methods). If there is no way to safely identify access methods, as this is the case in LALO, these modified measures are difficult to implement and use in practice.

ICH and LCC have a moderate positive correlation to size. For ICH, which is a count of method invocations within classes, this is understandable: the more methods a class has, the more method invocations are likely to occur within the class (even though this could not be observed in the UMD systems). For LCC, however, we have no explanation for the correlation to size. Most of the other measures have a significant, but weak correlation to size.

Comparison to UMD systems

In the UMD systems, we also found that many measures had a significant, but weak correlation to size. In particular, ICH displayed one of the strongest correlations among all cohesion measures in both the UMD and LALO systems. Such a relationship is explained, as discussed above, by the way the measure is defined. Such correlations just confirm the argument that ICH cannot probably be considered as a cohesion measure.

4.3 Building Predictive Models

In this paper, we have focused on empirically investigating the bivariate relationships between OO coupling and cohesion measures and fault-proneness. Our goals were to better understand the dimensions captured by existing measures and quantitatively investigate their impact on one important aspect of quality. However, in addition to the results presented above, it is interesting to note that by combining coupling and cohesion measures into multivariate prediction models, it is possible to obtain very accurate prediction models of fault-prone classes based on design information. Such models can then be used to focus inspection or testing activities on specific parts of the system [3]. Because of space constraints, we cannot report here the full details of our multivariate analysis results [4]. However, we think it is important to summarize those results in order to illustrate the level of predictive accuracy that can be obtained through design measurement.

We built a prediction model using multivariate logistic regression and a forward selection procedure. The significant coupling and cohesion measures, plus a number of inheritance measures that we additionally investigated in [4] were allowed to enter the model. The resulting model consisted of six measures: five coupling measures (CBO', ICP_L, NIH-ICP, OCAIC, OCMIC), and one 'cohesion' measure (ICH, which, as discussed, is unlikely to measure cohesion).

To evaluate the prediction model, we performed a 10-cross validation. To this end, the 83 data points were randomly split into ten partitions of roughly equal size. For each partition, we re-fitted the model with the six covariates listed above, using only data points not included in the partition, and then applied the model to those data points in the partition. Classes with a predicted probability π>0.5 were classified fault-prone. The threshold 0.5 was good enough to balance the number of actually faulty and predicted fault-prone classes. We then compared, across all ten partitions, the predicted and actual fault-proneness of the classes. The results of this comparison are summarized in Table 5.

		Predicted		Σ
		π<0.5	π>0.5	
Actual	No fault	31	6	37
	Fault	6 (7 faults)	40 (124 faults)	46 (131 faults)
Σ		37	46	83

Table 5: Prediction model evaluation

From Table 5 we see that 46 classes were predicted fault-prone. 40 of these 46 classes actually contain a fault (87% correctness). The classes predicted fault-prone contain 45 classes contained 124 out of all 131 faults in the system (95% completeness).

The 46 classes predicted fault-prone contain 66% of the methods. The 46 actually fault-prone classes contain 65% of all methods. That is, the portion of the system that is predicted fault-prone is close to the theoretical minimum, the portion of the system that is actually fault-prone.

Comparison to UMD systems

In terms of its predictive power, the multivariate model derived from the UMD data set performs almost as well as the LALO (~90% completeness, ~80% correctness). The model itself differs from the LALO model: it contains a larger number of variables (due to the larger data set), and a different selection of variables. However, the dimensions that were found to have a strong relationship to fault-proneness in both systems (e.g., import coupling through

method invocations to library/non-library classes) are also represented in both models; see [4] for a more detailed comparison of the models.

5 CONCLUSIONS

Based on the comparison of the results from the two studies performed so far, we provide a number of recommendations. 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 to be a very good fault-proneness indicator. This stems mainly from the current difficulty to define clearly the concept and measure it. 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. Which one is actually measuring the concept of cohesion, if any?

When using design measures to build predictive models of fault-prone classes, we have consistently obtained, across two studies, high levels of classification accuracy (i.e., around 90% correctness and completeness). This suggests that design measurement-based models may be very effective instruments for quality evaluation and control. However, the overall results of our studies also tell us that the validity of fault-proneness models may be very context-sensitive. In a given environment, their stability has to be assessed and analyzed across systems so that conditions (e.g., application domain) under which models are stable can be identified. Although some of the patterns and relationships presented above seem stable across very different study settings and systems, replication of such studies is necessary in order to build over time a credible body of empirical knowledge on which to base the quality assessment of OO designs and systems.

ACKNOWLEDGMENTS

We would like to thank Michael Ochs for developing the analyzers used in this study, Michel Lavallée for making the LALO study possible, and the LALO developers without whom such quality data could not have been collected. The Concerto2/AUDIT tools and the FAST technology are marketed by SEMA Group, France. We also want to thank SEMA group for providing us with this tool suite.

REFERENCES

All ISERN technical reports below are available from http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.

[1] 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.

[2] L. Briand, P. Devanbu, W. Melo, "An Investigation into Coupling Measures for C++", Proceedings of ICSE '97, Boston, USA, 1997.

[3] L. Briand, J. Daly, V. Porter, J. Wüst, "A Comprehensive Empirical Validation of Product Measures for Object-Oriented Systems", Technical Report ISERN-98-07, 1998.

[4] L. Briand, S. Ikonomovski, H. Lounis, J. Wüst, "A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study", Technical Report ISERN-98-29, 1998.

[5] L. Briand, J. Daly, J. Wüst, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE Transactions on Software Engineering: to be published, 1998. Also Technical Report ISERN-96-14.

[6] 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. Also Technical Report ISERN-97-05 .

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

[8] 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.

[9] 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.

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

[11] G. Dunteman, "Principal Component Analysis", SAGE Publications, 1989.

[12] "FAST Programmer's Manual", SEMA Group, France, 1997.

[13] B. Henderson-Sellers, "Software Metrics", Prentice Hall, Hemel Hempstaed, U.K., 1996.

[14] D.W. Hosmer, S. Lemeshow, "Applied Logistic Regression", John Wiley & Sons, 1989.

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

[16] T. Khoshgoftaar, E. Allen, "Logistic Regression Modeling of Software Quality", TR-CSE-97-24, Florida Atlantic University, March 1997.

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

[18] 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.