A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems

Lionel C. Briand, John Daly[1], Victor Porter[1], Jürgen Wüst
Fraunhofer IESE, Sauerwiesen 6, D-67661 Kaiserslautern, Germany
{briand,wuest}@iese.fhg.de, Tel. +49 6301 707 251, Fax +49 6301 707 202

This paper aims at empirically exploring the relationships between existing object-oriented coupling, cohesion, and inheritance measures and the probability of fault detection in system classes during testing. 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. Results show that many of the measures capture similar dimensions in the data set, thus reflecting the fact that many of them are based on similar principles and hypotheses. Besides the size of classes, the frequency of method invocations and the depth of inheritance hierarchies seem to be the main driving factors of fault-proneness.

Keywords: coupling, cohesion, inheritance, object-oriented, metrics, measurement, empirical validation

1. Introduction

Many measures have been proposed in the literature to capture the structural quality of object-oriented (OO) code and design [2,7,8,9,13,15,17,18,19,20,22]. Such measurement is aimed at providing ways of assessing the quality of software products, for example, in the context of large scale software acquisition [21]. But how do we know what measures actually capture important quality aspects? Despite numerous theories about what constitutes good OO design, only empirical studies of actual systems' structure and quality can provide tangible answers. Unfortunately, only a few studies have so far investigated the actual impact of these measures on quality attributes such as fault-proneness [1,2,10,18]. In our research [3], we empirically investigate most of the measures proposed in the literature to date, which capture structural aspects of OO designs. As far as we know, this is the first time such a comprehensive investigation is attempted. Based on data collected in a university experiment, we attempt to answer the following questions:

1. Are existing OO design measures capturing different dimensions and structural aspects? If not, what are the underlying structural dimensions they actually capture?
2. How are the measures related to the fault-proneness of classes? Which ones strongly affect fault-proneness?
3. How accurate are the existing measures in predicting faulty classes, and to which extent can they be used to drive code and design inspections.

In this paper, because of space constraints, we investigate questions 1 and 2 only, leaving question 3 to be addressed in [3]. Our investigation is based on one data set only. Such studies should be replicated a large number of times, in different conditions, in order to obtain generalizable results. The results of this paper show that the number of dimensions actually measured is much lower than the number of measures themselves, despite their apparent differences. Some measures, in particular coupling and inheritance ones, are shown to be significantly related to the probability of detecting a fault in a class during testing.

The paper is organized as follows: Section 2 describes the goals of the empirical study we are conducting, 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. We draw our conclusions in Section 5.

2. The Empirical Study

2.1. Dependent Variable

The goal of this study was to empirically assess the object-oriented design measures discussed in a literature review [4,5]. More specifically, we wanted to evaluate whether these measures are useful for predicting the probability of detecting faulty classes. Assuming testing was performed properly and thoroughly, the probability of fault detection[2] in a class during acceptance testing 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 only in 46% of the classes and 77% of the classes contain less than

1. John Daly and Victor Porter were with Fraunhofer IESE when this research was performed. John Daly is now at Hewlett Packard Ltd, QA Dept, Queensferry Microwave Division, South Queensferry, Scotland, UK EH30 9TG. e-mail: john_daly2@hp.com.
   Victor Porter is now at IBM Perth, Level2, Pitheavlis, Perth, Scotland, PH2 0XB. e-mail: victor_porter@uk.ibmmail.com
2. More precisely, the probability of fault detection that is meant here is a conditional probability: the probability to detect at least one fault during testing, depending on the obtained measurement values of the independent variables for that class.

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. The probability of fault detection was therefore the most straightforward and practical measure of fault-proneness we could use, and a suitable dependent variable for our study.

2.2. Independent Variables

The measures of coupling, cohesion and inheritance identified in a literature survey on object-oriented design measures [4,5] are the independent variables used in this study. The measures are described in Tables 1 to 3. We provide informal natural language definitions of the measures. These 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 [4,5]. Column "Source" indicates the literature reference where each measure has originally been proposed.

2.3. Hypotheses

In this study, we want to test the following hypotheses, which relate the design measures to fault-proneness.

H-IC (for all import coupling measures): A class with high import coupling relies on many externally provided services. Understanding such a class requires knowledge of all these services. The more external services a class relies on, the larger the likelihood to misunderstand or misuse some of these services. Therefore, the class is more difficult to understand and develop, and thus likely to be more fault-prone.

H-EC (for export coupling measures): A class with high export coupling has a large influence on the system: many other classes rely on it. Failures occurring in a system are therefore more likely to be traced back to a fault in the class, i.e., the class is more fault-prone.

H-COH (for cohesion measures): Low cohesive classes indicate inappropriate design, which is likely to be more fault-prone.

H-Depth (measures DIT, AID): The deeper the class in the inheritance hierarchy, the less likely it is to consistently extend or specialize its ancestor classes, and, therefore, the more likely it is to contain a fault.

H-Ancestors (measures NOA, NOP, NMI): The more parents or ancestors a class inherits from, and the more methods a class inherits, the larger the context we need to know in order to understand what an object of that class represents, and therefore the more likely it is to contain a fault.

H-Descendents (measures NOC, NOD, CLD):
A class with many descendents has a large influence on the system, as all descendents rely on it. The class has to serve in many different contexts, and is therefore more likely to be fault-prone.

H-OVR (measures NMO, SIX): The more use of method overriding is being made, the more difficult/complex it is to understand or test the class, the more fault-prone it will be. Also, heavy use of method overriding indicates inappropriate design, which is more fault-prone.

H-NMA: The more methods are added to a class, the more methods have to be developed for that class, the more fault-prone it is likely to be (the number of methods added to a class may be interpreted as a measure of its size).

2.4. Description of the Empirical Study

2.4.1. Setting of the Study

The empirical study uses data collected from a development project which was performed at the University of Maryland over a four month period. Eight different groups of developers, composed of undergraduate and postgraduate students, were asked to develop a medium sized information system. The systems were developed using a Waterfall development model. Requirement specifications and design documents were checked by independent experts to ensure they matched the system requirements. Faults found were reported to the developers thus maximizing the chance that the implementation phase began with an appropriate object-oriented analysis and design. The testing phase was accomplished by an independent group composed of experienced software professionals. This group tested all systems according to similar test plans and used functional testing techniques, spending eight hours testing each system. Full details of the study design can be found in [1].

2.4.2. Data Collection Procedures and Measurement Instruments

The following relevant items were collected:

1. the source code of the C++ programs delivered at the end of the implementation phase,
2. data about faults found during the testing phase and fixes during the repair phase.

A tool developed at Fraunhofer IESE, and based on GEN++ [11], was used to extract the values for the object-oriented design measures directly from the source code of the programs delivered at the end of the implementation phase. To collect item 2, fault report and component origination forms were used.

2.4.3. Data Collected

The eight systems under study consist of a total of 180 classes. Of these 180 classes, 67 were reused verbatim or with minor modifications from class libraries, the other 113 classes were developed from scratch or reused with extensive modifications. For these 113 system classes, the

Name	Definition	Source
CBO (coupling between object classes)	According to the definition of this measures, a class is coupled to another, if methods of one class use methods or attributes of the other, or vice versa. CBO for a class is then defined as the number of other classes to which it is coupled. This includes inheritance-based coupling (coupling between classes related via inheritance).	[9]
CBO'	Same as CBO, except that inheritance-based coupling is not counted.	[8]
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.	[8]
RFC_1	Same as RFC_{∞}, except that methods indirectly invoked by methods in M are not included in the response set this time.	[9]
MPC (message passing coupling)	The number of method invocations in a class.	[18]
DAC (data abstraction coupling)	The number of attributes in a class that have as their type another class.	[18]
DAC´	The number of different classes that are used as types of attributes in a class.	[18]
ICP (information-flow-based coupling)	The number of method invocations in a class, weighted by the number of parameters of the invoked methods.	[20]
IH-ICP (inheritance-based ICP)	As ICP, but counts invocations of methods of ancestors of classes (i.e., inheritance-based coupling) only.	[20]
NIH-ICP (non-inheritance-based ICP)	As ICP, but counts invocations to classes not related through inheritance.	[20]

IFCAIC
ACAIC
OCAIC
FCAEC
DCAEC
OCAEC
IFCMIC
ACMIC
OCMIC
FCMEC
DCMEC
OCMEC
OMMIC
IFMMIC
AMMIC
OMMEC
FMMEC
DMMEC
	These coupling measures are counts of interactions between classes. The measures distinguish the relationship between the classes (friendship, inheritance, none), different types of interactions, and the locus of impact of the interaction (import or export coupling). The acronyms for the measures indicate 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

values for each of the measures presented above were collected.

At this stage, it is pertinent to consider the influence of the library classes. System classes can be coupled to library classes by using some of their functionality. 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 there would be a different effect on our dependent variable, if, for example, a system class is coupled to another system class rather than a library class, i.e., we hypothesized that a

Name	Definition	Source
LCOM1 (lack of cohesion in methods)	The number of pairs of methods in the class using no attribute in common.	[8], [13]
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.	[9]
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 then 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 $Co = 2 \cdot \frac{\lvert E\rvert - (\lvert V\rvert - 1)}{(\lvert V\rvert - 1) \cdot (\lvert V\rvert - 2)}$.	[15] (named "C")
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 $LCOM5 = \frac{\frac{1}{a}\left(\sum_{j=1}^{a} \mu(A_j)\right) - m}{1 - m}$	[13]
Coh	A variation on LCOM5: $Coh = \frac{\sum_{j=1}^{a} \mu(A_j)}{m \cdot a}$	[5]
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 *m'* 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.	[7]
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.	[7]
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.	[20]

Table 2. Cohesion measures

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). 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.

3. Data Analysis Methodology

In this section we describe the methodology used to analyze the coupling, cohesion, and inheritance measure data collected for the 113 system classes.The procedure used to analyze the data collected for each measure comprises an analysis of the descriptive statistics, principal component analysis, and univariate regression analysis against the fault data. We now describe these analyses procedures in some detail.

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.	[8]
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.	[13]
CLD (class-to-leaf depth)	CLD of a class is the maximum number of levels in the hierarchy that are below the class.	[22]
NOC (number of children)	The number of child, parent, ancestor, and descendent classes of a class.	[8]
NOP (number of parents)		[17]
NOD (number of descendents)		[17]
NOA (number of ancestors)		[22]
NMO (number of methods overridden)	The number of methods in a class that override a method inherited from an ancestor class.	[19]
NMI (number of methods inherited)	The number of methods in a class that the class inherits from its ancestors and does not override.	[19]
NMA (number of methods added)	The number of new methods in a class, not inherited, not overriding.	[19]
SIX (specialization index)	SIX is NMO * DIT / (NMO+NMA+NMI)	[19]

Table 3. Inheritance related measures

Descriptive statistics

The data from the eight systems are merged into one data set of 113 classes along with the relevant values for each coupling, cohesion and inheritance 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. Only measures with more than five non-zero data points were considered for principal component and univariate analysis.

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 squares of the coefficients in one 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 ina a PC have large coefficients - also called the loading of the variable - and therefore contribute significantly to its variance. 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 [12] for more details on PCA and rotated components.

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. This analysis is conducted to test the hypotheses in Section 2.3.

The dependent variable we use to validate the design measures is binary, i.e., was a fault detected in a class during testing phases? 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 equation:

$$\pi(X) = \frac{e^{c_0 + c_1 \cdot X}}{1 + e^{c_0 + c_1 \cdot X}}$$

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:

1. When X is not significant, then the curve approximates a horizontal line, i.e., π does not depend on X.
2. 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 ψ(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, ψ(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.

4. Analysis Results

In Sections 4.1 through 4.3, we discuss, for coupling, cohesion, and inheritance measures, respectively, the descriptive statistics, principal component analysis, and the results from univariate analysis. In Section 4.4, we consider threats to the validity of this study.

4.1. Coupling Results

Table 4 presents the descriptive statistics and the results from univariate analysis for the coupling measures. Columns “Max”, “Mean” and “σ” state for each measure the maximum value, mean value, and standard deviation, respectively. From univariate analysis, the regression coefficient c_1 is provided (Column “Coeff.”), its statistical significance (p-value), which is the probability that the coefficient is different from zero by chance, and the $\Delta\psi$ value as defined in Section 3. For univariate analysis, we only show the values for measures with six or more non-zero data points and $p<0.5$.

Concerning the descriptive statistics of the measures, we make the following observations:

- The measures counting coupling between classes related through inheritance (all A**IC and D**EC measures, and NIH-ICP) have relatively low mean values and standard deviations. As we will see in Section 4.3, inheritance has not been used a lot in the systems under study, which explains the low variance of these coupling measures.
- There is evidence of export coupling from non-library classes to library classes, as OCAEC and some of the other export coupling measures have non-zero values. This stems from classes which were reused with minor modifications: in some instances, these classes actually use the non-library classes (which were developed from scratch or reused with extensive modifications).

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 methods from non-library classes.

PCA identified eleven PCs which capture 83% of the data set variance. Below, 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[3], and our interpretation of the dimension that the PC captures.

- PC1 (21%): RFC_1_L, RFC_∞_L, MPC_L, ICP_L, NIH_ICP_L, and OMMIC_L measure the extent of import coupling from library classes through method invocations. The RFC measures additionally count the number of methods of the invoking class, therefore their coefficients are somewhat lower in this PC.
- PC2 (16%): CBO, CBO’, RFC_1, RFC_∞, MPC, NIH-ICP, and OMMIC. Except for CBO and CBO’, these measures count import coupling from non-library classes through method invocation. Again the coefficients of the RFC measures are slightly low. CBO and CBO’ too have low

3. Because of space constraints, we cannot provide the 56x11 matrix of the loadings of each measure in each PC. The interested reader is referred to [3].

coefficients, because these measures count coupling differently: they count the number of classes to which a given class is coupled, and they also count export coupling. It is interesting to observe that import coupling from library classes through method invocation (PC1) and import coupling from non-library classes (PC2) are two orthogonal dimensions. This indicates that a class with high import coupling from library classes does not necessarily have particularly high or low import coupling from non-library classes; the two forms of coupling are unrelated in our data set.

- PC3 (8%): DAC, DAC', and OCAIC measure import coupling from non-library classes through aggregation.
- PC4 (8%): FCAEC, FCMEC, and FMMEC measure export coupling to friend classes.
- PC5 (6%): IFCAIC, IFMMIC, CBO_L, and CBO'_L. This PC cannot easily be interpreted. The fact that these measures show up in the same PC indicates that classes with high import coupling from friend classes also tend to be coupled to a large number of library classes. This is an interesting observation, but may be peculiar to the data set we used.
- PC6 (5%): RFC_1_L, RFC_∞_L, IH-ICP_L, ACMIC_L, and AMMIC_L. The latter three are measures of inheritance-based import coupling from library classes. Why do the RFC_L measures show up in this PC? A possible 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 (cf. definition in Table 1), and thus are counted by the measures. Hence, the RFC_L measures tend to be larger for descendent classes. Because the inheritance-based coupling measures are non-zero for descendent classes only, they have a strong positive correlation to RFC_1_L and RFC_∞_L.
- PC7 (5%): OCAEC, OCMEC, and OCMEC_L measure export coupling to classes not related via friendship or inheritance.
- PC8 (4%): DAC_L, DAC'_L, and OCAIC_L measure import coupling from library classes through aggregation. Again, with PC3 we also have a corresponding dimension which counts aggregation coupling to non-library classes only.
- PC9 (3%): CBO, CBO', FCMEC, OMMEC.
- PC 10 (3%): IH-ICP, OCMIC_L.
- PC 11 (2%): DCMEC.

PC9 and PC10 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.

As we see, most of the PCs are made up either from measures counting coupling to library classes, or from measures counting coupling to non-library classes only. Only a few PCs are constituted by a mix of library and non-library coupling measures.

The 56 coupling measures investigated seem, overall, not to capture more than eleven dimensions in our data set. One of the drawbacks of PCA is that variables with small variance automatically receive negligible coefficients in the main PCs. Therefore, if actual coupling dimensions do not vary much in our data set, we won't be able to identify them. For instance, sixteen of the measures showing low variability are not included in the PCs above. Therefore, more accurately, we can say that the 40 remaining measures capture eleven attributes, seven of which can be readily interpreted.

Because we observe multiple orthogonal dimensions of coupling, we conclude that a thorough analysis of coupling in an OO system requires multiple coupling measures. Defining one measure that would combine the various dimensions of coupling is not likely to be suitable to completely describe class coupling in an OO system.

The results also show that some coupling categories (or combinations of thereof) in the classification suggested by Briand et. al. [2] seem to correspond to actual coupling dimensions, i.e., they match with a PC. However, the original classification contained 18 classes whereas we have identified less orthogonal dimensions. Therefore, several refinements are not supported by empirical evidence, e.g., the different types of interdependency do not appear to capture different dimensions in the context of export coupling to friend classes, since their measures are in the same PC.

Univariate analysis

Looking at the import coupling measures, we note that most measures which display sufficient variance also have a significant relationship to fault-proneness. This provides strong support for hypothesis H-IC that classes with high import coupling are more fault-prone. In particular, the measures that belong to principal components PC1 and PC2 (e.g., CBO*, RFC*, MPC, *ICP) show very strong $\Delta\psi$s, far above the ones of the other measures, for both library and non-library coupling. Since $\Delta\psi$ is corrected for the standard deviation of each measure, we may conclude that method invocation is the main coupling mechanism having an impact on fault-proneness.

The measures IH-ICP, IFCAIC, ACMIC, and IFMMIC for the non-library data, and DAC', ACMIC, and OCMIC for the library data appear not be related to fault-proneness. Although these measures fulfill our minimum variance criterion, their mean values and standard deviations are relatively low as compared to the significant measures. This may explain why we failed to find a significant relationship for these measures.

Measure	Coupling to Non-Library Classes						Coupling to Library Classes Only					
	Max.	**Mean**	**σ**	**Coeff.**	**p**	**Δψ**	**Max.**	**Mean**	**σ**	**Coeff.**	**p**	**Δψ**
CBO	12	2.018	2.155	0.325	<.0001	2.012	9	2.027	1.503	0.896	<.0001	3.844
CBO'	12	1.973	2.140	0.338	<.0001	2.062	9	1.805	1.315	0.868	<.0001	3.133
RFC_1	86	16.938	13.716	0.085	<.0001	3.208	130	14.708	21.482	0.048	<.0001	2.797
RFC_∞	138	19.513	20.531	0.102	<.0001	8.168	182	19.265	25.592	0.059	<.0001	4.503
MPC	58	4.434	9.047	0.182	<.0001	5.206	63	11.062	11.254	0.109	<.0001	3.417
ICP	97	7.478	15.128	0.135	<.0001	7.710	166	29.646	30.851	0.041	<.0001	3.511
IH-ICP	12	0.372	1.956	p>0.5			10	0.522	1.643	0.403	0.0491	1.938
NIH-ICP	97	7.106	14.991	0.149	<.0001	9.272	163	29.124	30.227	0.041	<.0001	3.482
DAC	10	0.858	1.569	0.212	0.0329	1.395	4	0.363	0.768	0.464	0.0386	1.428
DAC'	6	0.593	0.960	0.339	0.0389	1.385	3	0.310	0.642	0.345	0.1867	1.248
IFCAIC	2	0.071	0.320	0.748	0.4334	1.270	0	0.000	0.000	<6 non-zero values		
ACAIC	3	0.027	0.282	<6 non-zero values			1	0.044	0.207	<6 non-zero values		
OCAIC	8	0.726	1.390	0.250	0.0307	1.416	3	0.310	0.656	0.511	0.0476	1.398
FCAEC	2	0.071	0.320	0.586	0.3213	1.206	2	0.035	0.229	<6 non-zero values		
DCAEC	3	0.027	0.282	<6 non-zero values			0	0.000	0.000	<6 non-zero values		
OCAEC	27	0.726	2.736	-0.492	0.0030	0.260	6	0.283	0.977	p>0.5		
IFCMIC	9	0.283	1.366	<6 non-zero values			0	0.000	0.000	<6 non-zero values		
ACMIC	13	0.115	1.223	p>0.5			4	0.212	0.773	-0.319	0.1441	0.782
OCMIC	38	2.425	5.540	<6 non-zero values			2	0.354	0.706	p>0.5		
FCMEC	9	0.283	1.392	p>0.5			1	0.009	0.094	<6 non-zero values		
DCMEC	13	0.115	1.223	p>0.5			0	0.000	0.000	<6 non-zero values		
OCMEC	95	2.425	11.600	-0.017	0.2520	0.816	45	1.779	5.390	-0.035	0.2762	0.830
IFMMIC	13	0.558	2.066	0.220	0.0922	1.575	0	0.000	0.000	<6 non-zero values		
AMMIC	7	0.212	1.145	<6 non-zero values			4	0.363	0.897	0.747	0.0051	1.954
OMMIC	56	3.637	8.365	0.191	<.0001	4.937	60	10.699	10.806	0.110	<.0001	3.276
FMMEC	16	0.558	2.142	0.134	0.0891	1.333	8	0.124	0.888	<6 non-zero values		
DMMEC	21	0.212	1.984	<6 non-zero values			5	0.044	0.470	<6 non-zero values		
OMMEC	36	3.637	7.053	0.023	0.1664	1.180	49	4.407	7.412	0.026	0.2616	1.209

Table 4. Descriptive statistics and univariate analysis results for coupling measures

Turning to export coupling measures, we observe that the only measure significant at α=0.05 is OCAEC. However, the negative regression coefficient indicates that classes high with export coupling are likely to be *less* fault-prone than classes with low export coupling. Overall, there is no evidence for hypotheses H-EC that classes with high export coupling are more likely to be fault-prone. Apparently, how much a class is being used by other classes has little effect on the probability for the class to contain a fault. Although CBO and CBO' count both import and export coupling, we gather that the significance of their relationship with fault-proneness is mainly due to the import coupling counting part of the measures.

In [2], more export coupling measures have been found significant. In that study, which uses the same C++ systems and the same fault-data, the distinction between coupling to library classes and non-library classes has not been made: both types of coupling were added up in one measure. This indicates that, at least in the data set used, export coupling to both types of classes has to be counted together to find a relationship to fault-proneness.

4.2. Cohesion results

Table 5 presents the descriptive statistics and the results from univariate analysis for the cohesion measures. The columns are the same as in Table 4. Regarding the descrip-

tive statistics of the measures, we make the following observations:

- All measures have six or more non-zero values and sufficient variance to be considered for further analysis.
- Measures with similar definitions also have similar distributions, and, as will become evident in PCA, are strongly correlated. For instance, LCOM3 and LCOM4 both count common attribute references. However, LCOM4 also counts method invocations between methods in a class (cf. definitions in Section 2.2). The values calculated for LCOM3 and LCOM4 are very similar, at least in the data set we used. This indicates that, when method invocations within a class do occur, the methods are also connected through other means such as common attribute usage.

Measr.	**Max.**	**Mean**	**σ**	**Coeff**	**p**	**Δψ**
LCOM1	819	62.10	115.20	0.00	0.2135	1
LCOM2	818	43.10	105.80	p>0.5		
LCOM3	40	5.20	4.80	0.15	0.0164	2.054
LCOM4	40	5.10	4.90	0.09	0.0914	1.554
LCOM5	2	0.78	0.28	0.98	0.1240	1.315
Coh	0.82	0.34	0.19	-2.23	0.0045	0.654
Co	1	0.36	0.28	-0.54	0.4005	0.859
LCC	1	0.50	0.38	-0.27	0.4870	0.902
TCC	1	0.40	0.35	p>0.5		
ICH	72	6.36	11.35	0.11	0.0002	3.485

Table 5. Descriptive statistics and univariate analysis results for cohesion measures

Principal component analysis

PCA identified four PCs which describe 92.7% of the variance in the data set. Based on the loadings of the cohesion measures, the PCs are interpreted as follows:

- PC1 (49%): LCOM1, LCOM2, LCOM3, and LCOM4 count *common attribute usage* within a class. Also, these measures are not normalized, i.e., they have no upper bounds. As noted above, the fact that, e.g., LCOM4 additionally accounts for method invocations does not significantly affect the distribution of the measure, and therefore the interpretation of this PC.
- PC2 (26%): Co, LCC, and TCC count both *common attribute usage* and *method invocations* within a class. These measures are normalized, i.e., they have upper and lower bounds.
- PC3 (10%): ICH. This measure is a count of *method invocations* within a class. ICH is the only measure that does not account for references to attributes at all, which distinguishes it from the other measures. ICH is not normalized.
- PC4 (8%): LCOM5 and Coh count individual *references to attributes* by methods. Both measures are normalized.

The results show that normalized and non-normalized measures clearly measure different dimensions: each PC is dominated either by normalized or non-normalized measures only. No dimension includes both normalized and non-normalized measures. Although it is still unclear whether cohesion measures should be normalized to reflect variations in size across classes [6], it is definitely a decision which has an impact on the structural property captured by the measures.

The results also indicate that the types of connections within a class that are counted by the measures make a difference. The measures in PC4 count individual *references to attributes*, the measures in PC2 count *common attribute usage*. All measures of these PCs are normalized.

ICH appears to define a dimension of its own. Besides its unique type of intra-class-connection, it also possesses a theoretical property that sets it apart from the other measures: ICH is additive. If two *unrelated*, but *highly cohesive* classes *c* and *d* are merged into a single class *e*, the cohesion of the class *e* would be the sum of the cohesion of the separate classes *c* and *d*. That is, class *e* has an even higher cohesion than any of the separate classes. This is counter-intuitive, as an object of class *e* represents two separate, semantic concepts and therefore should be less cohesive. It is therefore unlikely that ICH is a measure of cohesion. However, since it has been proposed as a cohesion measure in [20], we do consider it here.

Univariate analysis

The coefficients for the inverse cohesion measures LCOM1 to LCOM5 are positive, those for the straight cohesion measures Coh, Co, and LCC are negative. In each case, this indicates an increase in the predicted probability of fault-detection as the cohesion of the class (as measured by the measures) decreases. The only exception to this is ICH, where the positive coefficient suggests that fault-proneness increases with cohesion. This is another indicator that ICH is unlikely to be a cohesion measure. In fact, ICH can be shown to fulfill the properties for complexity measures defined in [6]. We can then interpret this result that the higher the complexity of a class, the more fault-prone it is.

Besides ICH, the only measures significant at α=0.05 are LCOM3 and Coh. Overall, these results provide only weak support for hypothesis H-COH that the cohesion of a class has a causal relationship to fault-proneness.

Measure LCOM2 is the least significant measure. LCOM2 has been criticized not to discriminate classes very well [1,13]. As a result, it is unlikely to be a useful predictor. Our findings provide some empirical evidence that supports this.

4.3. Inheritance Results

Table 6 presents the descriptive statistics and results from univariate analysis for the inheritance measures. From the descriptive statistics, the following observations can be made:

- The low mean values for DIT and NOC show that inheritance has been used sparingly within the eight systems. Similar results have also been found in other studies ([9], [10]).
- There is the presence of multiple inheritance indicated by the different values returned for DIT, AID, and NOA, which would otherwise be identical. On further inspection of the systems, there is just one library class with two parents, i.e., NOP=2. As it is a library class, this class is not considered in Table 6, but affects the measurement values for some of the system classes. The class with the maximum NMI of 104 is derived from that library class with two parents, which explains the high value.

Measr.	**Max**	**Mean**	**σ**	**Coeff.**	**p**	**Δψ**
DIT	4	0.850	1.197	0.7	0.0001	2.311
AID	4	0.845	1.193	0.693	0.0001	2.285
CLD	2	0.079	0.331	-2.276	0.0045	0.470
NOC	5	0.177	0.770	-1.468	0.0276	0.322
NOP	1	0.416	0.495	1.572	0.0001	2.177
NOD	9	0.230	1.102	-1.393	0.0429	0.215
NOA	4	0.867	1.228	0.681	0.0001	2.307
NMO	10	0.610	1.566	0.515	0.0243	1.948
NMI	104	6.97	16.13	0.025	0.0373	1.496
NMA	41	11.38	7.89	0.068	0.0021	1.710
SIX	0.52	0.047	0.095	6.886	0.0089	1.337

Table 6. Descriptive statistics and univariate analysis results for inheritance measures

Principal component analysis

PCA identified three PCs, which capture 82% of the data set variance. Based on the analysis of loadings of each measure within each of the three PCs, the PCs are interpreted as follows.

- PC1 (46%): DIT, AID, NOP, NOA, and NMI. We interpret this PC to be the *depth of the class in the hierarchy* where it is located. A class deeper down in the hierarchy (high DIT, AID) has at least one parent and ancestor class (high NOP and NOA), and is likely to inherit many methods (high NMI).
- PC2 (25%): CLD, NOC, and NOD measure, for a class *the depth of the hierarchy* which is below the class. A class with a non-zero CLD has at least one child class (NOC) and is likely to have more descendents (NOD).
- PC3 (11%): NMO, NMA, and SIX. These measures are concerned with method overriding and adding of methods. We interpret this PC to measure the *change to a class* through specialization.

Recall that the principal components are orthogonal. This means that a class that is nested deep in the inheritance hierarchy (PC1) is not bound to have a low CLD and/or few descendents (PC2).

Univariate analysis

First, we note that all inheritance measures are significant at α=0.05, indicating that the inheritance relationships of a class affect its fault-proneness. In the following, we investigate the hypotheses established in Section 2.3 for inheritance measures.

Hypothesis H-Depth for measures DIT and AID is supported. The deeper the class is located in the inheritance hierarchy, the more fault-prone it is.

Hypothesis H-Ancestors for measures NOA, NOP, NMI is supported. The more parents or ancestors a class inherits from, and the more methods a class inherits, the more fault-prone the class is. Because these measures together with DIT and AID are all in PC1, H-Depth and H-Ancestors are difficult to distinguish.

Hypothesis H-Descendents for measures NOC, NOD, CLD (all in PC2) is not supported. The results indicate that classes with a high number of children or descendents are less fault-prone. Perhaps, a greater attention (e.g., through inspections) is given to them as they are developed since many other classes depend on them. As a result, fewer defects are found during testing. However, the relatively low $\Delta\psi$ shows that the impact on fault-proneness is not so strong for these measures

Hypothesis H-OVR for measures NMO and SIX is supported. The more use of method overriding is being made, the more fault-prone it will be.

Hypothesis H-NMA is empirically supported. The more methods are added to a class, the more fault-prone it is.

The results also show that the measures belonging to PC1 and PC3 have the strongest impact on fault-proneness (high values for $\Delta\psi$). The interpretation is that what matters the most in terms of class fault-proneness is its depth in the hierarchy and the extent of change from its parent class(es).

4.4. Threats to validity

We distinguish between three types of threats to validity for an empirical study:

- Construct validity: The degree to which the independent and dependent variables accurately measure the concepts they purport to measure.

- Internal validity: The degree to which conclusions can be drawn about the causal effect of the independent variables on the dependent variables.
- External validity: The degree to which the results of the research can be generalized to the population under study and other research setting.

We now apply these criteria to assess the results described above.

4.4.1. Construct validity

The construct validity of the dependent variable has been argued in Section 2.1. We still have to address the question to which degree the coupling, cohesion, and inheritance measures used in this study measure the concepts they purport to measures. In [4] and [5], the coupling and cohesion measures were theoretically validated against properties for coupling and cohesion measures proposed in [6]. These properties are one of the more recent proposals to characterize coupling and cohesion in a reasonably intuitive manner. The properties are considered necessary but not sufficient, as a measure that fulfills all properties is not guaranteed to make sense or be useful.

Of the coupling measures, the following measures fulfill all coupling properties: MPC, the ICP measures, and the measures of the suite by Briand et al. [2]. Of the remaining measures, some measures do not have a null value (the RFC measures, DAC and DAC'), some are not additive when two unconnected classes are merged (CBO, CBO', the RFC measures, DAC').

Of the cohesion measures, only Coh, TCC and LCC fulfill all cohesion properties. The other measures are not normalized (LCOM1-LCOM4, ICH), or not properly normalized as they make assumptions which may not be fulfilled for all classes (LCOM5, Co). In addition, LCOM2 is not monotonic, and, as already discussed, ICH is additive when unconnected classes are merged.

For the inheritance measures, the concepts they purport to measure are not clear. No empirical relation systems for these measures have been proposed; doing so would be desirable but it is beyond the scope of this paper.

The function of the measures as *design measures* is emphasized, i.e., these measures are applicable in early stages of the development process. If these measures are found to be useful, they can be used to detect problems in the design before implementation starts, thus potentially saving time and effort for rework of the design. However, since we needed testing fault data to perform the analysis, measurement was performed only after implementation was completed. If measurement had taken place, say, before implementation started, different measurement data could have been obtained because of the uncertainty inherent to early design information (e.g., the information about actual method invocations may be uncertain before implementation and method invocations are counted by CBO, RFC, MPC, OMMIC, IFMMIC, AMMIC, OMMEC, FMMEC, and DMMEC). This in turn could have led to different results in the statistical analyses.

4.4.2. Internal validity

The analysis performed here is correlational in nature. We have demonstrated that several of the measures investigated had a statistically and practically significant relationship with fault-proneness during testing. Such statistical relationships do not demonstrate per se a causal relationship. They only provide empirical evidence of it. Only controlled experiments, where the measures would be varied in a controlled manner and all other factors would be held constant, could really demonstrate causality. However, such a controlled experiment would be difficult to run since varying coupling, cohesion, and inheritance in a system, while preserving its functionality, is difficult in practice. On the other hand, it is difficult to imagine what could be alternative explanations for our results besides a relationship between coupling, inheritance depth, and the cognitive complexity of classes.

4.4.3. External validity

The following facts may restrict the generalizability of our results.

- The systems developed are rather small: they lie between 5000 and 14000 source lines of C++ code.
- The systems developed have a limited conceptual complexity.
- The developers are not as well trained and experienced as average professional programmers.

5. Conclusions

Our goal was to perform a comprehensive empirical validation of all the object-oriented (OO) design measures found in the literature. We wanted to understand the underlying structural dimensions captured by these measures, and their individual impact on one important aspect of class fault-proneness: cognitive complexity.

Many of the coupling, cohesion, and inheritance measures studied in this paper 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. 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.

Univariate analysis results have shown that many coupling and inheritance measures are strongly related to the probability of fault detection in a class. In particular, coupling induced by method invocations, the rate of change in a class due to specialization, and the depth of a class in its inheritance hierarchy appear to be important quality factors. On the other hand, cohesion, as currently captured by

existing measures, does not seem to have a significant impact on fault-proneness. 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 syntactical analysis only.

The study performed in this paper should be replicated across many environments and systems in order for our community to gather a body of knowledge about what OO measurement can do to help assess the quality of early designs and systems, and also about the suitability of the data analysis methodologies we use. The results above still support the idea that measurement of OO products can shed light on their quality. However, as long as empirical studies remain as scarce as they are today, product quality measurement for OO development is likely to remain an elusive target.

Acknowledgments

The authors would like to thank Michael Ochs for developing the M-System, by which our measurement was performed. Special thanks also go to Drs. Basili and Melo who were, with Lionel Briand, involved in the original studies from which the defect data used here come from.

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

[5] 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 available as *Technical Report* ISERN-97-05.

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

[7] 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.

[8] 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.

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

[10] M. Cartwright, M. Shepperd, "An Empirical Investigation of an Object-Oriented Software System", Technical Report, Department of Computing, Bournemouth University, UK, 1997.

[11] P. Devanbu, "A Language and Front-end Independent Source Code Analyzer", Proceedings of ICSE '92, Melbourne, Australia, 1992.

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

[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] 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.

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

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

[20] 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.

[21] J. Mayrand, F. Coallier, "System Acquisition Based on Software Product Assessment", *Proceedings of ICSE'96*, Berlin, Germany, 210-219,1996.

[22] 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.