Fraunhofer Einrichtung Experimentelles Software Engineering

A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study

Authors:
Lionel C. Briand
Hakim Lounis
Stefan Ikonomovski
Jürgen Wüst

A short version of this report was submitted to ICSE '99

IESE-Report No. 047.98/E
Version 1.0
November, 1998

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

This paper aims at empirically exploring the relationships between most of the existing design coupling, cohesion, and inheritance measures for object-oriented (OO) systems, and the fault-proneness of OO system classes. The underlying goal of this study is to better understand the relationship between existing design measurement in OO systems and the quality of the software developed. In addition, we aim at assessing whether such relationships, once modeled, can be used to effectively drive and focus inspections or testing.

The study described here is a replication of an analogous study conducted in a 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, although many principles and techniques can be reused, quality does not follow universal laws and quality models must be developed locally, wherever needed.

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

Table of Contents

1 Introduction

A large number of object-oriented (OO) measures have been proposed in the literature ([4], [9], [11], [12], [18], [20], [22], [23], [24], [25]). A particular emphasis has been given to the measurement of design artifacts, in order to help assess quality early on during the development process. Such measures are aimed at providing ways of assessing the quality of software, for example, in the context of large scale software acquisition [27]. Such an assessment of design quality is objective, and the measurement can be automated. Once the necessary measurement instruments are in place, the assessment of large software systems can be thus done very fast, at a low cost, with little human involvement. 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], [4], [5], [13], [15], [22]). It is therefore difficult to assess whether these measures capture complementary dimensions or are indicators of any relevant external quality attributes such as reliability or maintainability. In this context, it is interesting to note that the new ISO/IEC international standard (14598) [31] on software product quality states that "Internal metrics are of little value unless there is evidence that they are related to external quality" and that "Internal metrics should also have predictive validity, that is that they should correlate with some desired external criterion". The general opinion of the practitioners involved in the making of this ISO standard seems to be that empirical investigations regarding product internal measurement and its relationship to external quality measures are, from a practical perspective, a crucial issue for the betterment of quality control early on in the life cycle. This is, however, in sharp contrast to the lack of empirical results published to date.

Recently, some of the authors performed an in-depth, comprehensive analysis of most of the published OO measures on students' projects [5] 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 an 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 in use and under maintenance. By analyzing carefully the results and by comparing them in a system-

atic 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, distribution of the measures, 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 [5]. 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 [6][7], 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 user operation after the shipment 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 results in a failure during user operation and can be traced back to a class, given the design measurement values for that class. In a mature system, which has been tested and under operation for a few years, this should be a good indicator of a class' 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, cohesion, and inheritance identified in a literature survey on object-oriented design measures [6][7] are the independent variables used in this study. We focus on design measurement since we want the measurement-based models investigated in this paper to be usable at early stages of software development. Furthermore, we only use measures defined at the class level since this is also the granularity at which the fault data could realistically be collected.

Tables 1 to 3 describe the 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 in-

sight into the measures. However, such definitions tend to be ambiguous. Formal definitions of the measures using a uniform and unambiguous formalism are provided in [6][7].

Name	Definition	Source
CBO	Coupling between object classes. According to the definition of this measure, a class is coupled to another, if methods of one class use methods or attributes of the other, or vice versa. CBO is then defined as the number of other classes to which a class is coupled. This includes inheritance-based coupling (coupling between classes related via inheritance).	[12]
CBO'	Same as CBO, except that inheritance-based coupling is not counted.	[11]
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.	[11]
RFC_1	Same as RFC_∞, except that methods indirectly invoked by methods in M are not included in the response set.	[12]
MPC	Message passing coupling. The number of method invocations in a class.	[22]
DAC	Data abstraction coupling. The number of attributes in a class that have another class as their type.	[22]
DAC'	The number of different classes that are used as types of attributes in a class.	[22]
ICP	Information-flow-based coupling. The number of method invocations in a class, weighted by the number of parameters of the invoked methods.	[25]
IH-ICP	As ICP, but counts invocations of methods of ancestors of classes (i.e., inheritance-based coupling) only.	[25]
NIH-ICP	As ICP, but counts invocations to classes not related through inheritance.	[25]
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.	[4]

Table 1: Coupling Measures

Name	Definition	Source
LCOM1	Lack of cohesion in methods. The number of pairs of methods in the class using no attribute in common.	[11]
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.	[12]
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.	[20]
LCOM4	Like LCOM3, where graph G additionally has an edge between vertices representing methods *m* and *n*, if *m* invokes *n* or vice versa.	[20]
Co	Connectivity. Let *V* be the number of vertices of graph G from measure LCOM4, and *E* the number of its edges. Then $Co = 2(\|E\| - (\|V\| - 1)) / ((\|V\| - 1)(\|V\| - 2))$.	[20]
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 $LCOM5 = \frac{1}{a}((\sum_{j=1}^{a} \mu(A_j)) - m)/(1-m)$.	[18]
Coh	A variation on LCOM5: $Coh = (\sum_{j=1}^{a} \mu(A_j))/(m \cdot a)$	[7]
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.	[9]
LCC	Loose class cohesion. Same as TCC, except that this measure also considers pairs of *indirectly connected* methods. If there are methods $m_1,\ldots, m_n$, 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.	[9]
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.	[25]

Table 2: Cohesion Measures

Name	Definition	Source
DIT (depth of inheritance tree)	The DIT of a class is the length of the longest path from the class to the root in the inheritance hierarchy.	[11]
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.	[18]
CLD (class-to-leaf depth)	CLD of a class is the maximum number of levels in the hierarchy that are below the class.	[30]
NOC (number of children)	self-explanatory	[11], [12]
NOP (number of parents)		[23], [24]
NOD (number of descendents)		[23], [30]
NOA (number of ancestors)		[30]
NMO (number of methods overridden)	The number of methods in a class that override a method inherited from an ancestor class.	[24]
NMI (number of methods inherited)	The number of methods in a class that the class inherits from its ancestors and does not override.	[24]
NMA (number of methods added)	The number of new methods in a class, not inherited, not overriding.	[24]
SIX (specialization index)	SIX is NMO * DIT / (NMO+NMA+NMI)	[24]

Table 3: Inheritance Measures

2.3 Hypotheses

In this study, we want to test a number of hypotheses, which relate various properties of design to fault-proneness. Our hypotheses are mostly derived from the causal chain depicted in Figure 1: The structural properties of a class, measured by the coupling, cohesion, and inheritance measures, affect the cognitive complexity of the class. By cognitive complexity we mean the mental burden of the persons who have to deal with the class (developers, inspectors, testers, maintainers, etc.). We believe that it is the, sometimes necessary, high cognitive complexity of a class which causes it to display undesirable external qualities, such as increased fault-proneness, or decreased maintainability and testability. The external class quality attributes are therefore indicators of the cognitive complexity.

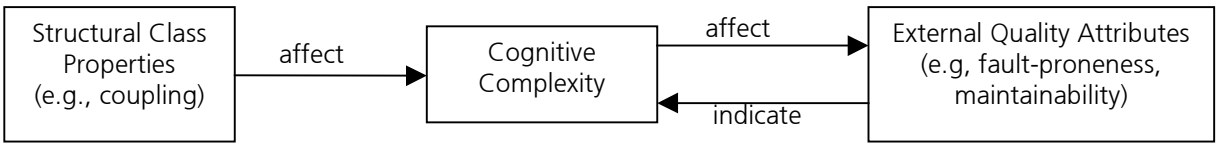

Figure 1: Relationships between structural class properties, cognitive complexity, and external quality

The structural properties of a class are not, however, the only factors which affect a class' cognitive complexity. For instance, the completeness and traceability of the documentation will also have an impact. Cognitive complexity is also dependent on the individuals who deal with the class and is to some degree subjective. In the context of the more general program of research depicted in

Figure 1, we focus, in this study, on an external class quality which can be measured easily and objectively: fault-proneness.

The following hypothesis will be tested in this study:

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 cohesion 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 larger the number of parents or ancestors a class inherits from, and the more methods a class inherits, the larger the context that is needed to know in order to understand what an object of that class represents, and therefore the more fault-prone the class is likely to be.

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

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

2.4.1 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 and maintained since 1993 at CRIM (Centre de Recherche Informatique de Montréal). The LALO system consists of 83 classes that were developed from scratch, seven additional classes that were automatically generated by code generators. These 90 C++ classes have a total of approximately 40K source lines of code (SLOC). In the analysis below, the automatically generated 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. Porting the LALO system to different platforms was an easy task that was not the source of additional faults.

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 experience in system development, and four developers had worked on OO systems before.

2.4.2 Data Collection Procedures and Measurement Instruments

The following relevant items were collected:

- The source code of the LALO system.
- Fault data related to failures detected by world-wide users of LALO, over a period of about one year.

A tool developed at the Fraunhofer IESE, and based on GEN++ [16], was used to extract the values for the object-oriented measures directly from the source code of LALO version 1.3. 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.

2.4.3 Data Collected

The values for each of the design measures were collected for each of the 83 LALO classes that were developed from scratch. 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' hereafter). 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 rather than 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.

Similarly, for inheritance we could hypothesize that deriving a class from a library class has a different effect on the derived class than deriving it from a system-specific non-library class. However, in the LALO system there are inheritance relationships only between non-library classes, but not between library- and non-library classes. Therefore, it was not necessary to measure inheritance with library- and non-library classes separately.

2.5 Comparison to students' systems

The results of the current study will be compared to those obtained in a previous study [5], 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 each asked to develop an information system. 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. However, the fault data used in that study were of different nature since they came from some acceptance testing activity performed on each system by an independent group composed of experienced software professionals. See [5] 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 and multivariate logistic regression analysis against the fault data, and correlation to class size. We now describe these techniques in some detail.

3.1 Descriptive statistics and outlier analysis

The data set consists of 83 classes along with the relevant values for each coupling and cohesion and inheritance measure. The distribution (mean, median, and interquartile ranges) and variance (standard deviation) of each measure is examined. Low variance measures do not differentiate classes very well and therefore are not likely to be useful predictors in our data set. Measures with less than five non-zero data points were not considered for subsequent analysis.

Outliers are data points which are located in an empty part of the sample space. Inclusion or exclusion of outliers can have a large influence on the analysis results and prediction models. It is important that conclusions drawn are not solely dependent on a few outlying observations, otherwise, the resulting prediction models are unstable and cannot be reliably used. For this reason it is important to identify outliers, test their influence, and possibly remove them to obtain stable results. We distinguish univariate and multivariate outliers:

- Univariate outliers: A class that has an outlying value in the distribution of any one of the measures used in the study. The influence of the identified data point is tested: an outlier is influential, if the significance of the relationship between the measure and fault-proneness depends on the absence or presence of the outlier. Such influential outliers are not considered in the univariate analysis results.

- Multivariate outliers: Our set of n independent variables spans an n-dimensional sample space. To identify multivariate outliers in this sample-space, we calculate, for each data point, the Mahalanobis Jackknife distance from the sample space centroid. The Mahalanobis Distance is a measure that takes correlations between measures into account. Multivariate outliers are data points with a large distance from the sample space cen-

troid. Again, a multivariate outlier may be over-influential and therefore removed, if the significance of any of the n variables in the model depends on the absence or presence of the outlier. More detailed information on outlier analysis can be found in [10].

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 [28].

Principal components (PCs) are linear combinations of the standardized variables, i.e., design measures. The sum of the square of the weights in each linear combination is equal to one. PCs are calculated as follows. The first PC is the linear combination of all standardized variables that explain a maximum amount of variance in the data set. The second and subsequent PCs are linear combinations of all standardized variables, where each new PC is orthogonal to all previously calculated PCs and captures a maximum variance under these conditions. Usually, only a subset of all variables shows large weights and therefore contributes significantly to the variance of each PC. To better identify these variables, the *loadings* of the variables in a given PC can be considered. The loading of a variable is its correlation with the PC. The variables with high loadings help identify the dimension the PC is capturing but this usually requires some degree of interpretation.

In order to further ease interpretation of the PCs, we consider the rotated components. This is a technique where the PCs are subjected to an orthogonal rotation. 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 [17] for more details on PCA and rotated components.

We do not consider the PCs for use as independent variables in the prediction model. This is often done in linear least-square regression. In the context of logistic regression, this has shown to provide sub-optimal prediction models and is not current practice. However, PCs are very useful to better understand the actual dimension or concept that a design measure actually captures and therefore will help the interpretation of observed relationships with class fault-proneness.

3.3 Univariate Regression analysis

Univariate logistic regression is performed for each individual measure (referred to as independent variable in the context of regression analysis) against the dependent variable, i.e., no fault/fault detection, in order to determine if the measure is a useful predictor of fault-proneness. An introduction to logistic regression is given in Section 3.6. Univariate regression analysis is conducted for two purposes:

1.) to test the hypotheses in Section 2.3, and

2.) to screen out measures not significantly related to fault-proneness. Only measures that are significant at significance level α=0.25 are considered for the subsequent multivariate analysis, as described in Section 3.5.

3.4 Correlation to size of class design

For each measure, we analyze its relationship to the size of the class, as measured based on design information. This is to determine empirically whether the measure, even though it is assumed to be a coupling, cohesion, or inheritance 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: larger 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 design in terms of the number of methods implemented in the class, which corresponds to the simpler version of the well-known WMC measure [12]. We then calculate Spearman's Rho coefficient between each design measure and size.

3.5 Multivariate Regression analysis

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

1.) Minimizes the number of independent variables in the model. Using too many independent variables can have the effect of increasing the esti-

mated standard error of the model's prediction, making the model more dependent on the data set, i.e., less generalizable. A rule of thumb is to have at least ten data points per independent variable in the model.

2.) Reduces multicollinearity, i.e., independent variables which are highly correlated. This makes the model more interpretable.

This study is exploratory in nature, that is, we do not have a strong theory that tells us which variables should be included in the prediction model and which not. In this situation, a stepwise selection process can be used, where prediction models are built in a stepwise manner, each time one variable enters or leaves the model.

The two major stepwise selection processes used for regression model fitting are forward selection and backward elimination [19]. The general forward selection procedure starts with a model that includes the intercept only. Based on certain statistical criteria, variables are selected one at a time for inclusion in the model, until a stopping criteria is fulfilled. Similarly, the general backward elimination procedure starts with a model that includes all independent variables. Variables are selected one at a time to be deleted from the model, until a stopping criteria is fulfilled.

Because of the large number of independent variables used in this study, the initial model in a backward selection process would contain too many variables and could not be interpreted in a meaningful way. Therefore, we opted for the forward selection procedure to build the prediction models. In each step, all variables not already in the model are tested: the most significant variable is selected for inclusion in the model. If this causes a variable already in the model to become not significant (at α_{Exit}=0.10), it is deleted from the model. The process stops when adding the best variable no longer improves the model significantly (at α_{Enter}=0.05).

The significance levels to enter and exit the model (0.05 and 0.10, respectively) are stricter than those suggested in [19]. We made this choice because it is an indirect means to control the number of variables in the final model. A less stringent choice (e.g. 0.25 to enter the model as suggested in [19]) resulted in models which violated the rule of thumb to have at least ten data points per independent variable.

To evaluate the model's goodness of fit, we apply the prediction model to the classes of our data set from which we derived the model. A class is classified fault-prone, if its predicted probability to contain a fault is higher than a certain threshold p_0. We select the threshold p_0 such that the percentage of classes being classified fault-prone is roughly the same as the percentage of classes that actually are fault-prone. We then compare the predicted fault-proneness

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

- Completeness:
  To explain the way we define completeness and justify its practical meaning, it is better to take an example. Assume we use the prediction model to select classes that are classified fault-prone so that they can undergo inspection. Completeness is then defined as the number of faults in classes that are classified fault-prone, divided by the total number of faults in the system. It is a measure of the maximum percentage of faults that could possibly be found if we used the prediction model in the stated manner. Low completeness indicates that many faults cannot be detected when relying on the model's prediction. These faults would then slip to subsequent development phases, where they are likely to be more expensive to correct.

  Counting the percentage of faults found is more precise than counting the percentage of fault-prone classes found. It ensures that detecting a class containing many faults contributes more to completeness than detecting a class with only one fault. Of course, faults can be more or less severe (e.g., measured by the effort required to fix a fault). It would be desirable to also account for the severity of faults when measuring completeness. However, for this study we had no reliable data available concerning the severity of the faults.

- Correctness:
  We can always increase the completeness of our prediction model by lowering the threshold p_0 used to classify classes as fault-prone ($\pi > p_0$). This causes more classes to be classified as fault-prone, thus completeness increases. However, the number of classes that are incorrectly being classified as fault-prone also increases. It is therefore important to consider the correctness of the prediction model. Correctness is the number of classes correctly classified as fault-prone, divided by the total number of classes classified as fault-prone. Low correctness means that a high percentage of the classes being classified as fault-prone do not actually contain a fault. We want correctness to be high, as inspections of classes that do not contain faults is a waste of resources.

- Kappa:
  Kappa [14] is a measure of the degree of agreement of two categorical variables. Kappa values range between -1 and 1, the higher the value, the better the agreement. A Kappa of zero indicates that the agreement is no better than what can be expected from chance. Kappa can have a negative value if agreement is weaker than expected by chance, but this is rare. In our case, we use Kappa to measure the agreement between predicted and actual fault-proneness of a class. Applying Kappa in this context requires

that the number of classes classified fault-prone, and the number of faulty classes are roughly the same. This is ensured by our choice of threshold p_0.

A fourth measure of the goodness of fit is the R^2 statistic. Unlike completeness, correctness, and Kappa, the definition of R^2 used here is specific to the technique of logistic regression, and will be explained in Section 3.6.

3.6 Logistic regression

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 [19] or [21].

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

$$\pi(X) = \frac{e^{(c_0 + c_1 X_1 + \cdots + c_n X_n)}}{1 + e^{(c_0 + c_1 X_1 + \cdots + c_n X_n)}}$$

π is the probability that a fault was traced back to a class after a failure during operation, and the X_is are the design measures. The curve between π and a single X_i – assuming that all other X_js are constant - takes a flexible S shape which ranges between two extreme cases:

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

Such a S shape is perfectly suitable as long as the relationship between X_is and π is monotonic, an assumption consistent with the empirical hypotheses to be tested in this study. Otherwise, higher degree terms have to be introduced in the regression equation.

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 [19], 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.

To assess the statistical significance of each independent variable in the model, a likelihood ratio chi-square test is used. Let l = ln[L] be the loglikelihood of the model given by our regression equation, and l_i be the loglikelihood of the model without variable X_i. Assuming the null hypothesis that the true coefficient of X_i is zero, the statistic G=-2(l-li) follows a chi-square distribution with one degree of freedom (denoted by $\chi^2(l)$). We test $p=P(\chi^2(l)>G)$. If p is larger than some level of significance α (typically, α=0.05), the observed change in the loglikelihood may well be due to chance, and X_i is not considered significant. If $p<=\alpha$, the observed change in the loglikelihood is unlikely to be due to chance, and X_i is considered significant.

The global measure of goodness of fit we will use for such a model is assessed via R^2—not to be confused with the least-square regression R^2—they are built upon very different formulae, even though they both range between 0 and 1 and are similar from an intuitive perspective. The higher R^2, the higher the effect of the model's explanatory variables, the more accurate the model. However, as opposed to the R^2 of least-square regression, high R^2s are rare for logistic regression. For this reason, the reader should not interpret logistic regres-

sion R^2s using the usual heuristics for least-square regression R^2s. Logistic regression R^2 is defined by the following ratio:

$$R^2 = \frac{LL_S - LL}{LL_S}$$

where:

LL is the loglikelihood obtained by Maximum Likelihood Estimation of the model described in formula (*)

LL_S is the loglikelihood obtained by Maximum Likelihood Estimation of a model without any variables, i.e., with only C0. By carrying out all the calculations, it can be shown that LL_S is given by

$$LL_S = m_0 \ln\left(\frac{m_0}{m_0 + m_1}\right) + m_1 \ln\left(\frac{m_1}{m_0 + m_1}\right)$$

where m_0 (resp., m_1) represents the number of observations for which there are no faults (resp., there is a fault). Looking at the above formula, $LL_S/(m_0+ m_1)$ may be interpreted as the uncertainty associated with the distribution of the dependent variable Y, according to Information Theory concepts. It is the uncertainty left when the variable-less model is used. Likewise, $LL/(m_0+ m_1)$ may be interpreted as the uncertainty left when the model with the covariates is used. As a consequence, $(LL_S - LL)/(m_0+ m_1)$ may be interpreted as the part of uncertainty that is explained by the model. Therefore, the ratio $(LL_S - LL)/ LL_S$ may be interpreted as the proportion of uncertainty explained by the model.

3.7 Comparison to previous study

In order to build on previous experience and attempt to generalize further our results, we compare the results obtained from LALO with those obtained from the systems analyzed in [5] (referred to hereafter 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 similarities. Since the systems studied are very different in nature, this should allow us to identify what results are more likely to be generalizable. One difference between the two studies should however be recalled. The UMD systems' dataset contains fault data collected during acceptance testing whereas the LALO fault data comes from operation. Although it is difficult to assess the extent to which this could blur the comparison results, it is important that this fact be stated.

4 Analysis Results

The analysis results are partitioned into four subsections. In Section 4.1 through Section 4.3, we discuss, for coupling, cohesion, and inheritance measures, respectively, the results of the data distribution and outlier analyses, the dimensions identified by principal component analysis, the univariate analysis, and degree to which the measures are correlated to size. Section 4.4 then presents and evaluates a number of multivariate prediction models built from the design measures. In Section 4.5, we consider threats to the validity of this study.

4.1 Coupling Results

This section describes the analysis results for all coupling measures. The structure of the section follows the analysis procedure described in Section 3, i.e., descriptive statistics, principal component analysis, univariate analysis, and correlation with size measures.

4.1.1 Descriptive Statistics

Table 4 presents the descriptive statistics for the coupling measures. Columns "Max", "75%", "Med.", "25%", "Min.", "Mean" and "Std Dev" state for each measure the maximum value, 75% quartile, median, 25% quartile, minimum, mean value, and standard deviation, respectively.

From this table, we can make the following observations:

- 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. These measures count instances where a class has an attribute whose type is an ancestor class. That is, there is both an "is-a" and a "has-a" relationship between two classes. Of course we expect this to occur infrequently (this was also observed in the UMD systems).
- 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.
- The measures OCAEC, OCMEC, and OMMEC counting export coupling to library classes are non-zero for some classes. This indicates that some library classes use non-library classes, something one normally would not expect to happen. In our case, the automatically generated classes in the LALO system, which we treat as library classes, implement a parser and use some of the 83 non-library classes.

	Coupling to non-library classes only						Coupling to library classes only						
Measure	Max	75%	Med	25%	Min	Mean	Std Dev	Max	75%	Med	25%	Min	Mean	Std Dev
CBO	31	9	5	3	0	7,181	6,659	2	1	0	0	0	0,422	0,587
CBO'	31	8	4	3	0	6,614	6,648	2	1	0	0	0	0,422	0,587
RFC_1	358	46	33	17	3	48,301	57,839	27	2	0	0	0	2,241	4,825
RFC_oo	669	142	39	21	3	106,036	143,229	119	89	5	0	0	32,277	41,602
MPC	274	14	5	1	0	16,205	37,448	73	2	0	0	0	3,47	10,233
ICP	769	31	12	3	0	49,241	113,13	201	7	0	0	0	9,446	27,88
IH-ICP	190	6	2	0	0	7,458	22,317	0	0	0	0	0	0	0
NIH-ICP	579	24	9	1	0	41,783	98,346	201	7	0	0	0	9,446	27,88
DAC	8	2	1	0	0	1,193	1,477	0	0	0	0	0	0	0
DAC'	7	1	1	0	0	1	1,22	0	0	0	0	0	0	0
IFCAIC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ACAIC	2	0	0	0	0	0,024	0,22	0	0	0	0	0	0	0
OCAIC	8	2	1	0	0	1,169	1,48	0	0	0	0	0	0	0
FCAEC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DCAEC	2	0	0	0	0	0,024	0,22	0	0	0	0	0	0	0
OCAEC	16	1	0	0	0	1,169	2,603	6	0	0	0	0	0,084	0,666
IFCMIC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ACMIC	8	1	0	0	0	0,807	1,477	0	0	0	0	0	0	0
OCMIC	205	8	4	2	0	9,386	23,644	2	0	0	0	0	0,217	0,443
FCMEC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DCMEC	38	0	0	0	0	0,807	4,432	0	0	0	0	0	0	0
OCMEC	135	4	1	0	0	9,386	23,247	62	1	0	0	0	1,735	8,412
IFMMIC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AMMIC	24	3	1	0	0	2,108	4,024	0	0	0	0	0	0	0
OMMIC	250	9	4	0	0	14,096	35,343	73	2	0	0	0	3,47	10,233
FMMEC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DMMEC	69	0	0	0	0	2,108	8,528	0	0	0	0	0	0	0
OMMEC	124	15	4	1	0	14,096	23,705	26	0	0	0	0	0,542	2,91

Table 4: Descriptive Statistics for Coupling Measures

Comparison to UMD systems

For the UMD study, we investigated eight independent, smaller systems, whereas in the current study, we have one larger system. Thus, because the LALO classes are embedded in a larger context, there is overall 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 was 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 „_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.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
EigenValue:	7,686	4,345	3,358	2,675	2,185	1,400	1,251
Percent:	27,445	15,518	11,993	9,554	7,802	5,001	4,467
CumPerc.	27,445	42,963	54,956	64,510	72,312	77,313	81,781
CBO	0,411	0,368	-0,048	0,241	-0,344	0,294	**-0,586**
CBO'	0,434	0,331	-0,023	0,240	-0,295	0,314	**-0,604**
RFC_1	0,289	0,087	0,021	0,395	**-0,762**	-0,044	0,035
RFC_oo	0,243	-0,166	0,229	0,195	**-0,768**	0,039	0,099
MPC	**0,962**	0,032	0,033	0,137	-0,134	-0,008	-0,062
ICP	**0,963**	0,076	0,070	0,076	-0,177	-0,006	-0,071
IH-ICP	0,083	0,072	0,402	-0,013	**-0,734**	0,009	0,001
NIH-ICP	**0,976**	0,069	0,023	0,079	-0,094	-0,007	-0,073
DAC	0,054	0,163	0,062	**0,943**	-0,100	-0,006	-0,099
DAC'	0,100	-0,072	0,066	**0,884**	-0,260	-0,055	-0,089
OCAIC	0,058	0,133	0,065	**0,958**	-0,072	0,016	-0,090
OCAEC	0,148	**0,837**	-0,000	0,116	-0,040	0,110	-0,136
ACMIC	-0,037	**0,727**	0,007	-0,128	-0,230	-0,197	-0,113
OCMIC	0,387	-0,084	-0,036	**0,673**	0,095	-0,074	0,066
DCMEC	-0,063	0,062	-0,094	-0,199	0,053	**0,826**	0,011
OCMEC	0,043	**0,856**	0,109	0,186	0,057	0,055	-0,181
AMMIC	0,097	**0,576**	-0,119	-0,027	**-0,619**	-0,214	-0,074
OMMIC	**0,965**	-0,045	0,050	0,143	-0,053	0,021	-0,053
DMMEC	0,146	-0,233	0,473	0,254	-0,135	**0,573**	-0,009
OMMEC	0,031	0,409	0,060	0,198	0,163	-0,047	**-0,790**
CBO_L	-0,043	**0,641**	0,469	-0,004	0,078	-0,015	-0,070
RFC_1_L	0,013	0,076	**0,821**	0,111	-0,240	-0,080	0,010
RFC_oo_L	-0,059	0,470	0,193	-0,039	**-0,545**	0,172	0,049
MPC_L	0,035	0,018	**0,961**	-0,012	-0,111	-0,019	-0,056
ICP_L	0,036	0,028	**0,961**	-0,017	-0,110	-0,020	-0,061
OCMIC_L	-0,005	**0,590**	**0,618**	-0,104	-0,126	0,042	-0,196
OCMEC_L	0,042	0,026	0,108	-0,058	0,107	-0,090	**-0,908**
OMMEC_L	0,079	0,387	**0,599**	0,166	0,006	0,202	0,040

Table 5: Rotated Components of all Coupling Measures

For the PCA, measures that did not vary were discarded. For pairs of measures with identical values, one redundant measure was removed. PCA using the remaining measures identified seven PCs which capture 82% of the data set variance. Table 5 shows for each rotated component the loadings of the measure, with loadings larger than 0.5 set in boldface. The eigenvalue, the percentage of the data set variance each PC captures, and the cumulative variance percentage are also provided. Based on the analysis of the loadings associated with each coupling measure within each of the seven rotated components, the PCs are interpreted as follows: PC1 (27%): MPC, ICP, NIH-ICP, and OMMIC: measure the extent of import coupling through method invocations to non-library classes.

- PC2 (16%): OCAEC, ACMIC, OCMEC, AMMIC, CBO_L, and OCMIC_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.

- PC3 (12%): RFC_1_L, MPC_L, ICP_L, OCMIC_L, and OMMEC_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. OMMEC_L is an export coupling measure, and has the weakest loading of the five measures.

- PC4 (10%): 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_∞, ICH-ICP, AMMIC, RFC_∞_L. Measures ICH-ICP and AMMIC count import coupling through method invocations to 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 coupling measures are non-zero for descendent classes only, they have a positive correlation to RFC_1 and RFC_∞.

- PC6 (5%): DCMEC, DMMEC count export coupling to (non-library) descendent classes. DCAEC, which would also fit in this PC, is missing here because it has too little variance.

- PC7 (4%): 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

The results of the univariate analysis are summarized in Table 6. For each measure, the regression coefficient and standard error is provided (Columns "Coeff." and "Std Err"), the R^2 and $\Delta\psi$ value (as defined in Section 3.6), and the statistical significance (p-value), which is the probability that the coefficient is different from zero by chance. Measures with little or no variance are not considered in the table.

Measure	Coupling to non-library classes only					Coupling to library classes only				
	R^2	Coef.	Std Err	$\Delta\psi$	p-value	R^2	Coef.	Std Err	$\Delta\psi$	p-value
CBO	0,3171	0,404	0,084	5.493	**<,0001**	0,1688	2,108	0,47	3.445	**<,0001**
CBO'	0,3421	0,447	0,091	4.439	**<,0001**	0,1688	2,108	0,47	3.445	**<,0001**
RFC_1	0,1046	0,018	0,006	1.368	**0,0019**	0,2037	0,535	0,165	1.879	**0,0012**
RFC_oo	0,0622	0,004	0,002	1.296	**0,0098**	0,0801	0,016	0,005	1.966	**0,0006**
MPC	0,2159	0,099	0,03	1.73	**0,0008**	0,2349	0,687	0,231	3.34	**0,0029**
ICP	0,2571	0,043	0,013	2.215	**0,0014**	0,2377	0,257	0,087	3.612	**0,0031**
IH-ICP	0,0848	0,062	0,025	1.409	**0,0129**	All zero				
NIH-ICP	0,2759	0,066	0,02	2.386	**0,0011**	0,2377	0,257	0,087	3.612	**0,0031**
DAC	0,0308	0,257	0,124	1.461	**0,0386**	All zero				
DAC'	0,0312	0,304	0,147	1.448	**0,0391**	All zero				
OCAIC	0,0281	0,241	0,121	1,429	**0,0467**	All zero				
OCAEC	0,0925	0,588	0,242	1.462	**0,0152**	All zero				
ACMIC	0,0376	0,339	0,152	1.226	**0,0261**	All zero				
OCMIC	0,0618	0,1015	0,036	2.009	**0,0052**	0,1689	3,179	1,019	4,089	**0,0018**
DCMEC	not significant				0,6019	All zero				
OCMEC	0,1320	0,093	0,035	1.365	**0,0086**	0,1355	1,82	0,458	2.395	**<,0001**
AMMIC	0,0758	0,19	0,076	1.395	**0,0121**	All zero				
OMMIC	0,1938	0,096	0,03	1.562	**0,0014**	0,2349	0,687	0,231	3.34	**0,0029**
DMMEC	0,0192	0,138	0,092	1.204	0,1349	All zero				
OMMEC	0,1034	0,051	0,017	1.542	**0,0025**	0,0732	0,826	0,377	1.454	**0,0282**

Table 6: Univariate Analysis Results for Coupling Measures

As we can see in Table 6, 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 confirms hypotheses H-IC and H-EC stated in Section 2.3 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 that 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 were 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 variation, we failed to find a statistically significant relationship to fault-proneness in those systems.

4.1.4 Correlation to size

In this section we analyze the correlation of the coupling measures to the size of the class. We measure the size of the class design as the number of methods that are implemented in the class. In Table 7, we indicate for each coupling measure its Spearman Rho coefficient with size, and the corresponding p-value.

	Coupling to non-library classes only		Coupling to library classes only	
Measure	Rho	p-value	Rho	p-value
CBO	0,3937	**0,0002**	0,3152	**0,0037**
CBO'	0,3939	**0,0002**	0,3152	**0,0037**
RFC_1	0,4788	**<,0001**	0,3091	**0,0045**
RFC_oo	0,2937	**0,0070**	0,2053	**0,0626**
MPC	0,3969	**0,0002**	0,3569	**0,0009**
ICP	0,3850	**0,0003**	0,3601	**0,0008**
IH-ICP	0,1174	0,2906	All zero	
NIH-ICP	0,4268	**<,0001**	0,3601	**0,0008**
DAC	0,5208	**<,0001**	All zero	
DAC'	0,5083	**<,0001**	All zero	
OCAIC	0,4866	**<,0001**	All zero	
OCAEC	0,1331	0,2303	All zero	
ACMIC	0,2826	**0,0096**	All zero	
OCMIC	0,5167	**<,0001**	0,4201	**<,0001**
DCMEC	-0,1664	0,1326	All zero	
OCMEC	0,1984	**0,0722**	0,3455	**0,0014**
AMMIC	0,0996	0,3701	All zero	
OMMIC	0,3591	**0,0009**	0,3569	**0,0009**
DMMEC	0,0498	0,6548	All zero	
OMMEC	0,4688	**<,0001**	0,2343	**0,0330**

Table 7: Correlation of Coupling Measures to Size

The measures DAC, DAC', OCAIC, and OCMIC of PC4 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). However, the Rho coefficients for the measures in PC4 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 statistically significant correlation to size is present, but it is weak.

4.2 Cohesion Results

This section presents the analysis results for all cohesion measures. The discussion is structured in the same manner as Section 4.1.

4.2.1 Descriptive Statistics

Table 8 presents the descriptive statistics for the cohesion measures. We make the following observations:

- 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.

- The low median and 75% quartile of ICH indicate that there are relatively few method invocations within classes: only one third of all classes have a non-zero value for ICH. This explains why LCOM3 and LCOM4 have similar distributions: in addition to counting pairs of methods which use attributes in common like LCOM3, LCOM4 also takes method invocations into account. Because for most classes there are no method invocations among methods, LCOM4 produces similar values to LCOM3. This was also observed in the UMD systems.

Measure	Max	75%	Median	25%	Min	Mean	Std Dev
LCOM1	5437	79	45	17	0	139,5904	598,2496
LCOM2	4988	46	22	0	0	99,5904	547,7370
LCOM3	58	7	5	3	1	5,91566	6,51491
LCOM4	51	7	5	3	1	5,72289	5,78752
LCOM5	1,2500	0,7500	0,6202	0,4886	0	0,62071	0,25716
Coh	1	0,5555	0,4375	0,2813	0	0,42850	0,23098
Co	0,5000	0,2338	0,0972	0	-1	0,11246	0,20116
LCC	1	0,7108	0,5238	0,2222	0	0,48434	0,30022
TCC	1	0,5735	0,3333	0,1818	0	0,38255	0,25424
ICH	343	2	0	0	0	9,32530	42,54384

Table 8: Descriptive Statistics for Cohesion Measures

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. Unlike the coupling measures, the cohesion measures are concerned with the internal structure of each individual class and are not strongly affected by the overall size of the systems. Overall, 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 four PCs that describe 91% of the variance in the data set. Table 9 shows, for each rotated component, the loadings of the measure in the same form as Table 5. We interpret the PCs as follows:

- PC1 (57%): 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). It should also be noted that LCOM1 and ICH have, of all measures considered here, the strongest correlation to size. Possibly, this is the common denominator that links the measures together in this PC.

- PC3 (9%): LCOM3, LCOM4, LCC. 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 [7], based on a comparison of the definitions of the measures, and is confirmed by empirical evidence in this study.

	PC1	PC2	PC3
EigenValue:	5,6551	2,5674	0,9231
Percent:	56,5508	25,6739	9,2314
CumPercent:	56,5508	82,2247	91,4561
LCOM1	0,078	**-0,970**	0,102
LCOM2	0,186	**-0,896**	0,197
LCOM3	0,270	-0,438	**0,841**
LCOM4	0,269	-0,317	**0,886**
LCOM5	**0,941**	-0,198	0,216
Coh	**-0,884**	0,297	-0,243
Co	**-0,797**	-0,131	-0,423
LCC	-0,496	-0,211	**-0,807**
TCC	**-0,712**	0,004	**-0,663**
ICH	-0,005	**-0,873**	0,028

Table 9: Rotated Components with Cohesion Measures

Among the cohesion measures considered here are variants of earlier measures capturing the same concept. These variants were mostly defined with the in-

tention to improve existing measures by eliminating problems that were identified based on theoretical considerations (see [7] for a summary of these discussions). From a practical perspective, these differences in the definitions do not seem to matter much, because the variants lie within the same PCs as the original measure: LCOM5 and Coh in PC1, LCOM1 and LCOM2 in PC2, LCOM3 and LCOM4 in PC3. An exception is TCC and LCC: the two measures are related, but TCC is stronger in PC1, LCC is stronger in PC3.

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 more 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 in terms of fault-proneness. TCC and LCC, however, do not show a strong relationship in the LALO system, in contrast to the UMD systems.

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.

4.2.3 Univariate logistic regression

The results from univariate analysis with cohesion measures are summarized in Table 10.

Measure	R^2	Coef.	Std Err	$\Delta\psi$	p-value
LCOM1	0,2078	0,02	0,006	3.942	**0,0004**
LCOM2	0,0949	0,011	0,005	1.613	**0,0249**
LCOM3	0,0525	0,141	0,063	1.529	**0,0257**
LCOM4	0,0355	0,108	0,065	1.365	0,0966
LCOM5	Not significant				0,8061
Coh	Not significant				0,3786
Co	Not significant				0,2619
LCC	0,0357	1,712	0,696	1.672	**0,0140**
TCC	Not significant				0,3726
ICH	0,2494	0,674	0,271	6.179	**0,0128**

Table 10: Univariate Analysis with Cohesion Measures

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 our Hypothesis H-Coh that low cohesion is bad design.

- As was discussed in [7], 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 [8], but which appears not to be effective from a prediction point of view. In PC3 (connectivity), only LCOM3 is significant at $\alpha = 0.05$ (in the expected direction).

Variants of the same measure that lie in the same PC also have similar results for univariate analysis. Both LCOM5 and Coh of PC1 are not significant. LCOM1 and LCOM2 in PC2 are significant with similar coefficients and p-values. For LCOM3 and LCOM4 in PC3, the difference of the p-values too is small, however, LCOM3 is below the 0.05 threshold, LCOM4 above. For TCC and LCC, the results are different, but these measures are also in different PCs.

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

Table 11 shows, for each cohesion measure, the Spearman's Rho coefficient with size (the number of methods implemented in the class).

Measure	Rho	p-value
LCOM1	0,8069	**<,0001**
LCOM2	0,3087	**0,0045**
LCOM3	0,2555	**0,0197**
LCOM4	0,2294	**0,0369**
LCOM5	0,0870	0,4373
Coh	-0,1259	0,2569
Co	0,3491	**0,0012**
LCC	0,4093	**0,0001**
TCC	0,2013	0,0680
ICH	0,5426	**<,0001**

Table 11: Correlation of Cohesion Measures 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. As discussed in [7], access methods typically reference one class attribute only. Thus, a large number of pairs of methods that use no attributes in common can be formed with these access methods. The more methods a class has, the larger the number of such pairs is likely to be, and therefore the stronger is the correlation of LCOM1 to size.

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 (especially Co, TCC, and LCC). ICH displays the strongest correlation 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 Inheritance results

This section presents the results from the analysis performed with the inheritance measures. The discussion is structured in the same manner as Sections 4.1 and 2.4.

4.3.1 Descriptive statistics

Table 12 summarizes the descriptive statistics of all inheritance measures. We make the following observations:

- The maximum value of NOP is 1, no class had two or more parents, i.e., multiple inheritance was not used. Therefore the measures DIT, AID, and NOA produce identical values, and we removed AID and NOA from all subsequent analyses.

- NMA and NMI show an extreme outlier (104 and 127 respectively). The two classes involved are central to the architecture of LALO. The class with NMI=127 inherits from the class with NMA=104.

- Overall, little use of inheritance was made. This is observed in most data sets reported ([12], [13], [15]).

Measure	Max	75%	Median	25%	Min	Mean	Std Dev
DIT	3	1	1	0	0	0,83133	0,85282
AID	3	1	1	0	0	0,83133	0,85282
CLD	3	0	0	0	0	0,27711	0,61114
NOC	9	0	0	0	0	0,59036	1,41453
NOP	1	1	1	0	0	0,59036	0,49476
NOD	12	0	0	0	0	0,83133	2,23508
NOA	3	1	1	0	0	0,83133	0,85282
NMO	13	4	0	0	0	2,46988	3,58635
NMI	127	8	3	0	0	6,54217	15,98269
NMA	104	15	9	8	1	12,81928	12,16366
SIX	1,1818	0,3333	0	0	0	0,16626	0,25566

Table 12: Descriptive Statistics for Inheritance Measures

Comparison to UMD systems

Overall, the distributions are comparable between the two systems, most measures have similar means and standard deviations. Notable differences:

- NOC and NOD do vary more in the LALO systems.

- Method overriding is much more used in the LALO system. In the UMD study, the average NMO was 0.61, which is one fourth of the value of the LALO system. This may be explained by the larger experience of the LALO developers with OO and the inheritance mechanism: they are better able to design class inheritance hierarchies that exploit the opportunities for polymorphism to a larger degree.

4.3.2 Principal component analysis

After performing PCA with the inheritance measures, for PCs are identified, as shown in Table 13. The PCs capture 90% of the data set variance. Based on the analysis of the loadings associated with each measure within each of the rotated components, we provide the following interpretations:

- PC1: DIT, NOP, NMO, SIX: This PC represents the depth of the class in the inheritance hierarchy. A non-zero DIT is a prerequisite for NOP, NMO, and SIX to be non-zero. The deeper the class, the more likely it is to override methods. The numerator in the definition of SIX contains the product of DIT and NMO, therefore the measure has a strong weight in this PC.

- PC2: CLD, NOC, and NOD measure, for a class, the depth of the class hierarchy below the class. A class with a non-zero CLD has at least one child class (NOC) and is likely to have more descendents.

- PC3: This PC is determined by NMI, the number of methods inherited.

- PC4: This PC is determined by NMA, the number of methods added to the class. This measure is not purely measuring inheritance-related aspects, it is more an indicator of the size of the class (see Section 4.3.4).

	PC1	PC2	PC3	PC4
EigenValue	3,7304	2,1511	1,2020	0,9989
Percent:	41,4492	23,9009	13,3560	11,0990
CumPercent:	41,4492	65,3501	78,7062	89,8052
DIT	**-0,816**	-0,093	-0,397	-0,179
CLD	0,120	**0,940**	0,039	0,134
NOC	0,114	**0,945**	0,052	0,014
NOP	**-0,751**	-0,119	-0,467	-0,077
NOD	0,112	**0,972**	0,052	-0,034
NMO	**-0,890**	-0,135	0,104	0,170
NMI	-0,035	-0,067	**-0,934**	0,084
NMA	0,049	0,069	-0,064	**0,982**
SIX	**-0,934**	-0,111	0,156	-0,095

Table 13: Rotated Components with Inheritance Measures

Comparison to UMD systems

In the UMD systems, three PCs were identified: PC1' with measures DIT, NOP, and NMI, PC2' with exactly the same measures as PC2 above, and PC3' with measures NMO, NMA, and SIX.

So, measures NMO and SIX are more correlated to DIT (PC1) in the LALO system, and NMI is less correlated to DIT (PC1). For NMO and SIX, this might be

explained by the fact that method overriding was used more extensively in the LALO system.

4.3.3 Univariate analysis

The results from univariate analysis for the inheritance measures are summarized in Table 14. We make the following observations:

- Only three measures are significant indicators of fault-proneness: DIT, NMO, and NMA.

- The negative coefficicient of DIT indicates that fault-proneness decreases with the depth of the class. That is, classes located deeper in the inheritance hierarchy are less fault-prone. An explanation for this is that, in the LALO system, classes situated deeper in the inheritance hierarchy provide only implementations for a few specialized methods, and are therefore less likely to contain faults than classes at the root. This stems from a deliberate strategy, from the LALO development team, to place as much functionality as possible near the root of the inheritance hierarchies.

- For NMO and NMA, the positive coefficient indicates that fault-proneness increases with the number of overriding or added methods. This is consistent with our hypotheses H-OVR and H-NMA.

- Of PC1, only DIT is significant. None of the measures in PC2 is significant. PC3 'is' NMI which is not significant, since in the way faults are "assigned" to classes, the subclass does not "inherit" the faults of its ancestors. PC4 'is' NMA and is significant.

Measure	R^2	Coef.	Std Err	$\Delta\psi$	p-value
DIT	0,0414	-0,656	0,245	0,572	**0,0074**
CLD	0,0085	0,366	0,314	1,251	0,2431
NOC	Not significant				0,7064
NOP	Not significant				0,5587
NOD	Not significant				0,8280
NMO	0,0479	0,152	0,057	1,724	**0,0082**
NMI	Not significant				0,3254
NMA	0,1075	0,112	0,037	3,925	**0,0021**
SIX	Not significant				0,7766

Table 14: Univariate Analysis with Inheritance Measures

Comparison to UMD systems

Measures NMO and NMA appear to be consistently good indicators of fault-proneness in both systems.

In the UMD-Systems, all inheritance measures were significant indicators of fault-proneness. Besides NMO and NMA, only DIT is a significant predictor in LALO. However, its relationship to fault-proneness is in the opposite direction. In the UMD systems, classes deeper down in the inheritance hierarchy were likely to be more fault-prone, as stated in hypothesis H-Depth. The student developers of the UMD systems had no previous experience with object-orientation, most of the LALO developers have worked on OO systems before. This could indicate that the added cognitive complexity of using inheritance may cause critical problems to beginners (such as the UMD systems' student developers) but is not an issue for an experienced team using a well defined inheritance strategy, such as the LALO developers.

A related result is that NOC was found significant in the UMD systems, but not in the LALO system, even though the measure has higher variance in the LALO system.

To summarize, the inheritance measures do not appear to be stable quality indicators of OO design, but seem to rather indicate how well the development team can deal with inheritance. In conjunction with fault data, the inheritance measures help to determine if the way inheritance is used in the system causes the developers problems. On their own, however, the existant inheritance measures are not able to distinguish, in an objective manner, proper use of inheritance from improper use.

4.3.4 Correlation to size

Table 15 shows Spearman's Rho correlation coefficients with size for all inheritance measures. Only NMA, the number of methods added to a class, is strongly correlated to size (which is to be expected). All other measures are at most weakly correlated to size. This is identical to what was observed in the UMD systems.

Measure	Rho	p-value
DIT	0,0005	0,9966
AID	0,0005	0,9966
CLD	-0,0346	0,7563
NOC	-0,0434	0,6972
NOP	0,0200	0,8575
NOD	-0,0410	0,7130
NOA	0,0005	0,9966
NMO	0,3748	**0,0005**
NMI	-0,1488	0,1793
NMA	0,7740	**<,0001**
SIX	0,2445	0,0259

Table 15: Correlation of Inheritance Measures with Size

4.4 Multivariate Regression analysis

In this section we investigate a number of multivariate prediction models, built from different subsets of the measures we have analyzed so far. We will then further evaluate the accuracy of the best prediction model we found using a 10-cross-validation process (Section 4.4.2), and discuss possible applications of such a model in Section 4.4.3.

4.4.1 Comparing Multivariate Models

In this section, we compare models built from coupling, cohesion, and inheritance design measures only, and one allowing all measures (design coupling, cohesion, inheritance, and size) to enter the model. With these models, we seek to find answers to the following questions:

Are coupling, cohesion, and inheritance design measures fault-proneness predictors that are complementary to design size measures?

How much more accurate is a model including the more difficult to collect coupling, cohesion, and inheritance measures? If it is not significantly better, then the additional effort of calculating these more expensive measures instead of some easily collected size measures would not be justified.

Size Model

We considered a number of design size measures that measure the size of classes at the design level:

- The number of methods. This is the size measure that coupling, cohesion, and inheritance measures were compared to in the previous sections.
- The number of method parameters parameters in a class (numpara).
- The number of all attributes (noattr).
- The number of aggregate objects.
- The number of referenced objects.

Based on univariate analysis, all these measure are significantly related to fault-proneness at α=0.05 and in the expected direction (i.e., the larger the class size, the more likely it is to contain a fault). Because the number of size measures considered is small, a backward stepwise selection process was used to build a prediction model based solely on size measures:

Measure	Coefficient	Std. Err.	p-value
numpara	.0577699	.0267288	0.031
noattr	.2654983	.1153678	0.021
Intercept	-.301569	.3846904	0.433

Table 16: Model I based on design size measures only

We will refer to this model as "Model I". Because there are strong correlations present among the size measures we considered, only two measures ended up in this model. Thus, the goodness of fit of the model cannot be expected to be very high: the loglikelihood of the model is -74.053746, and R^2=0.1639. We applied this model to the 83 classes of the LALO system in order to compare the predicted and actual fault-proneness of the classes. A class was classified 'predicted fault-prone', if its predicted probability to contain a fault is higher than 0.65. This threshold was selected to roughly balance the number of actual and predicted fault-prone classes. The contingency table below summarizes these results:

		Predicted		Σ
		π<=0.65	π>0.65	
actual	No fault	19 classes	18 classes	37 classes
	Fault	16 classes, 25 faults	30 classes, 106 faults	46 classes 131 faults
Σ		35 classes	48 classes	83 classes

Table 17: Goodness of fit of Model I

The model identifies 48 classes as fault-prone, 30 of which actually are fault-prone (62% correctness) and contain a total of 106 of all 131 faults (80% completeness). The Kappa value for the degree of agreement between actual and predicted fault-proneness is 0.30, which ranks 'fair' on standard Kappa interpretation scales [14]. If we use the predictions of this model to select classes for inspections, the completeness figure seems satisfactory: by inspecting 55% of the classes, we can expect to find a maximum of 80% of the faults. However, the low correctness of the model implies that a large number of classes (18) that do not contain any fault would have been inspected in vain.

Model built from coupling, cohesion, and inheritance measures

Next, we build a model allowing all coupling, cohesion, and inheritance measures significant at α=0.25 to enter the model, following a forward selection process as described in Section 3.5. Here it is:

Measure	Coeff.	Std. Err	p-value
CBO'	.4122264	.1626507	0.011
ICP_L	.342327	.1298004	0.008
OCAIC	-1.965132	.6337208	0.002
ICH	.4388982	.2303712	0.057
NIH-ICP	.0649752	.0256848	0.011
OCMIC	.1393342	.0550869	0.011
Intercept	-2.613054	.7571831	0.001

Table 18: Model II – based on coupling, cohesion, and inheritance measures

This model (Model II) consists of six measures: five import coupling measures, and the "cohesion" measure ICH, which, as discussed, rather has properties of a class complexity measure.None of the other cohesion and inheritance measures are included, reflecting the results found in univariate analysis that many of these measures are not significant indicators of fault-proneness in the LALO system. The five coupling measures cover four of the seven dimensions we identified through principal component analysis in Section 4.1.2: PC1 (NIH_ICP), PC3 (ICP_L), PC4 (OCAIC and OCMIC), and PC7 (CBO'). There are two measures from PC4 present, and they are somewhat correlated (Spearman's Rho between OCAIC and OCMIC=0.7). This may explain the negative sign of OCAIC, which was positive in the univariate analysis. Changes in coefficient magnitude or sign are common in multivariate regression analysis and can be due to multicolinearity and adjustment effects between covariates.

The model has a high goodness of fit (loglikelihood = -28.94, R2=0.68). Again, we applied the model to the 83 classes of LALO to compare the predicted and actual fault-proneness:

		Predicted		Σ
		π<=0.5	π>0.5	
Actual	no fault	33 classes	4 classes	37 classes
	Fault	3 classes, 4 faults	43 classes, 128 faults	46 classes 131 faults
Σ		36 classes	47 classes	83 classes

Table 19: Goodness of fit of Model II

We selected a threshold of 0.5 to balance the number of actual and predicted fault-prone classes. Model II performs very well: Off the 47 classes predicted fault-prone, 43 actually are fault-prone (91% correctness), which contain all but four faults (97.7% completeness). The Kappa value of 0.87 is rated 'almost perfect' on the standard interpretation scale. The high goodness of fit indicates that the coupling measures and ICH capture structural dimensions with a strong relationship to fault-proneness, which go beyond the size of classes. A model based on structural class properties can outperform models based on class size measures alone, which justifies the extra effort to collect these measures.

Model built from coupling, cohesion, inheritance, and design size measures

Finally, we will consider a model built using the forward selection process, allowing all coupling, cohesion, inheritance, and design size measures (significant at 0.25) to enter the model:

Measure	Coefficient	Std Err.	p-value
CBO'	.6936107	.1723732	0.000
ICP_L	.3978622	.1281559	0.002
OCAIC	-3.956819	1.121695	0.000
Numpara	.1904476	.0568858	0.001
Noattr	1.627108	.6579819	0.013
ICH	.4669395	.2378392	0.050
Intercept	-5.444157	1.478333	0.000

Table 20: Model III – allowing all measures to enter the model

This model (Model III) consists of six covariates: three import coupling measures, two size measures, and ICH. Compared to Model II, the two coupling measures OCMIC and NIH-ICP were replaced by two size measures: numpara and noattr (which happen to be exactly the size measures that were selected in Model I). The three coupling measures now cover three of the coupling dimensions: PC1, PC4, and PC7.

The results above could indicate that some of the coupling measures in Model II were included because of their moderate relationship to size, which is related to fault-proneness. When size measures are allowed to enter the model, those coupling measures are then not selected as significant covariates anymore.

The loglikelihood of this model is = -25.257995,and the R2 = 0.7148, i.e., slightly higher than the model without size measures. The classification results of this model actually are, however, a bit worse than Model II, but they are still excellent:

		Predicted		Σ
		π<=0.6	π>0.6	
Actual	No fault	31 classes	6 classes	37 classes
	Fault	5 classes, 5 faults	41 classes, 126 faults	46 classes 131 faults
Σ		36 classes	47 classes	83 classes

Table 21: Goodness of fit of Model III

A threshold of 0.6 was selected to better balance the number of actual and predicted fault-prone classes. 47 classes are predicted fault-prone, which contain 126 faults (96% completeness), and 6 of these classes are not actually fault-prone (87% correctness). Kappa=0.81 still is rated 'almost perfect' on the standard interpretation scale.

4.4.2 Model evaluation

Let us further investigate Model II which yielded the best classification for class fault-proneness. The accuracy of this model looks very promising. However, it is somewhat optimistic since the prediction model was applied to the same data set it was derived from, i.e., we were assessing the goodness of fit of the model. To get an impression of how well the model performs when applied to different data sets, i.e., its prediction accuracy, we performed a 10-cross validation [29] of model II. Then more realistic values for completeness, correctness and Kappa can be devised.

For the 10-cross validation, the 83 data points were randomly split into ten partitions of roughly equal size (seven partitions of 8 data points each, three partitions of 9 data points). For each partition, we re-fitted the model using all data points *not* included in the partition, and then applied the model to the data points in the partition. We thus again obtain for all 83 classes a predicted probability of their fault-proneness. Classes with predicted probability $\pi <= 0.5$ were classified 'not predicted fault-prone', the others 'predicted fault-prone'. The threshold 0.5 was good enough to precisely match the number of actual and predicted fault-prone classes.

		Predicted		Σ
		$\pi <= 0.5$	$\pi > 0.5$	
Actual	No fault	31 classes	6 classes	37 classes
	Fault	6 classes, 7 faults	40 classes, 124 faults	46 classes 131 faults
Σ		37 classes	46 classes	83 classes

Table 22: Results from 10-cross validation of Model II

The 46 classes predicted fault-prone contain 124 faults (94.7% completeness), 6 of these classes do not actually contain a fault (87% correctness). The Kappa value of 0.78 indicates 'substantial' agreement between actual and predicted fault-proneness.

4.4.3 Model applications

They are many ways in which the prediction models above can be used. However, before one can do so, it is important to demonstrate that, in a given application domain and environment, such models can be stable. Thus, from project to project, the models can be applied with a reasonable level of confidence. Such a stability can only be obtained if enough project data have been collected to obtained stable distributions on the design measures' ranges. As discussed above, many of the differences across regression results between the UMD systems and the LALO system came from significant differences in data distributions, some of the design measures being more or less variable in one

data set or the other. However, despite the sharp differences with respect to almost every project factor, e.g., application domain, experience, it is interesting to note that some strong commonalties have been clearly identified.

One important application is to use the fault-proneness models to help focus inspections or testing. Taking the example of inspection planning during the software development process, we would like to be able to make a trade-off between the resources spent on inspections on the one hand, and the effectiveness of inspections on the other hand. To optimize this trade-off, we can use a graph such as the one in Figure 2 to determine how many percent of the faults in the system we can expect to find by inspecting a certain percentage of the system code, and which classes should be inspected. On the X-axis, we plotted the 83 system classes, sorted in decreasing order of their predicted fault-proneness (the predicted probabilities were taken from the 10-cross validation described above). The lower curve is the accumulated size of the classes, in percent (measured in terms of the number of methods in a class). At X=30 the value of the curve is 50%, which means that the 30 classes with highest predicted fault-proneness contain 50% of the methods of the system. The upper curve is the accumulated number of actual faults in the classes, in percent. At X=30, this curve is at Y≈75%, which means the first 30 classes contain 75% of the actual faults.

As an example of an application of this graph, assume we have resources available to inspect 40% of all methods. From Figure 2 we can tell that the first 12 classes with highest predicted fault-proneness roughly contain 40% of all methods. If we select these classes for inspection, then we can expect to find a maximum of about 57% of all faults in them. The graph also tells us that by increasing the resources by 5 points to inspect 45% of all methods, the maximum percentage of faults we can find grows to about 67%.

A practical problem with the application of the graph in Figure 2 is that the upper curve, the accumulated percentage of actual faults, is a priori unknown. However, if the shape of the upper curve proves to be stable across projects within a development environment, the graph could be used as a reference when planning inspections. Initial results indicate that the shape of the graph may indeed be similar in different projects. In the UMD study, the analogous graph for that data set was very close to the one presented here, which also indicates the models in the UMD study and the present study have about the same predictive power. But in general, whether the shape of the upper curve is stable across projects in a given environment needs to be verified individually for each environment.

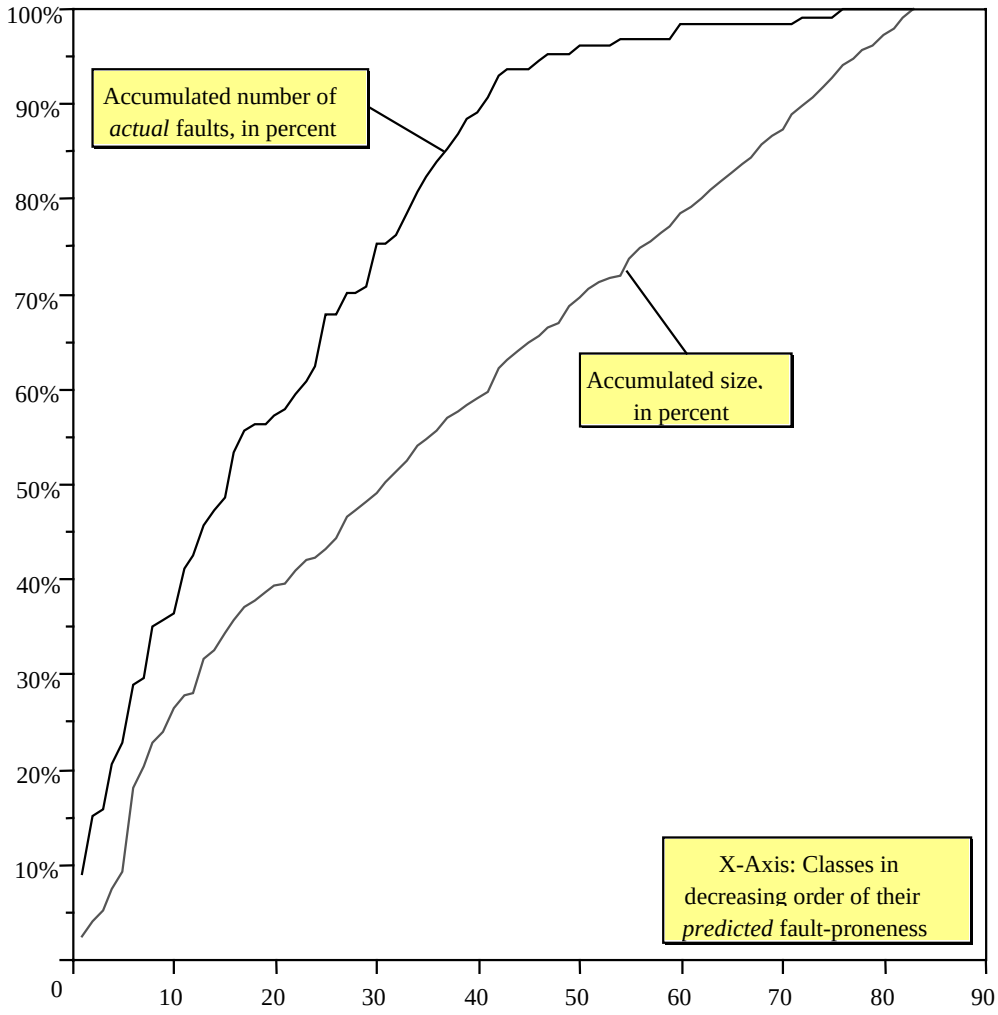

Figure 2: Application of the prediction model

In Figure 2, the upper curve (percent of actual faults) shows a steeper slope than the lower curve (percent of system size), before reaching a plateau. The lower curve shows an approximately linear growth, except that the slope for the first ten classes with highest predicted fault-proneness is steeper than for the remaining classes (which means that these ten classes have an above-average size). But overall, the discrepancy between the upper and lower curve indicates that our model does not simply assign higher fault-proneness to larger classes, but that the design measures capture an additional effect beyond size that has an impact on the fault-proneness of classes. This confirms the results obtained from multivariate analysis which showed that the design measures can help model an effect on fault-proneness which is complementary to the one of class size.

Last, as another example of application, once one knows the design properties that matter with respect to OO software quality in a given organization and application domain, then such measurement can be used as quality benchmark to assess and compare products. For example, such experiences are reported in [27] where the code of acquired products is systematically measured to identify systems or subsystems that strongly depart from previously established measurement benchmarks. Such benchmarks can be built based on existing operational software, which has shown to be of good quality. New software can then be analyzed by comparing, for example, import coupling class distributions with the established benchmark. Any system part that would show a strong departure from the benchmark distribution could, for example, be further inspected to investigate the cause of the deviation. If no acceptable justification can be found then the acquisition manager may decide to require some corrective actions before the final delivery or for future releases of the software. This is particularly important when systems will be maintained over a long period of time and new versions produced on a regular basis.

Comparison to multivariate model from the UMD study

In the following table, we state the covariates of the best multivariate models found in the UMD study and the present study, and match them by the PCs they lie in.

LALO	UMD	Interpretation of PC
CBO'		- not interpreted -
OCMIC		- not interpreted -
ICH		'Complexity'
NIH-ICP	**RFC_oo**	**Import coupling to non-library classes via method invocations**
ICP_L	**NIH_ICP_L**	**Import coupling to library via method invocations**
OCAIC	**DAC'**	**Import aggregation coupling**
	OCAEC	Export coupling to "others"
	FMMEC	Export coupling to friend classes
	RFC_1_L	Inheritance-based import coupling from library classes
	ACMIC_L	
	NMI	Depth of class in inheritance hierarchy
	NOP	
	SIX	Changes to class
	NOC	Depth of inheritance hierarchy below class

Table 23: Comparison of covariates in prediction models from UMD and LALO

As we see, three orthogonal dimensions that could be observed and interpreted in both systems also are represented in the best prediction models of both systems: Import coupling via method invocations to library/non-library classes, and aggregation import coupling to non-library classes.

Differences in the models are mostly due to the following reasons:

- Some measures that were found significant indicators of fault-proneness in the UMD study, and, consequently, were included in the multivariate model, were not significant in the LALO system. This is the case for all inheritance measures in the table (NMI, SIX, NOC, NOP), and can be partly explained by differences in the distributions of the measures.

- Some dimensions that were observed in the UMD systems could not be observed in the LALO system, for instance, friend coupling, inheritance-based coupling to library classes.

- In the UMD study, we had more data points and therefore more independent variables to be included in the multivariate model.

As was mentioned in the previous section, the predictive power of the two models is about the same, despite their differences. The UMD model achieved 95% completeness and 85% correctness, which is slightly worse than the 98% completeness and 91% correctness of Model II in this study.

As discussed above, in a given environment, we expect trends and relationships to become more stable once project data are collected to a sufficient extent to "cover" the ranges of the design measures, so that their data distributions does not change dramatically from project to project. This would be a major cause of variation in the observed relationships between design measures and fault-proneness.

4.5 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 settings.

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

4.5.1 Construct validity

The construct validity of the dependent variable has been demonstrated 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 [6] and [7], the coupling and cohesion measures were theoretically validated against properties for coupling and cohesion measures proposed in [8]. These properties are one of the more recent proposals to characterize coupling and cohesion in a reasonably intuitive but rigorous 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. [4]. 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.

4.5.2 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 operation. 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. Some attempts to do so are reported in [2][3]. 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.

4.5.3 External validity

The present study is a case study, therefore, we have the usual threats to the external validity: The models we built are validated for the LALO system and development environment only. However, LALO is a midsize system developed by experienced, professional developers and is therefore somewhat representative of some of the OO development in industry.

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 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? Similarly, inheritance measures appear not to be consistent indicators of class fault-proneness. Their significance as indicators strongly depends on the experience of the system developers and the inheritance strategy in use on the project. The combined use of inheritance measures and fault data to assess the use of inheritance in a project/organization is an important topic of further research.

As discussed, one important application of OO design measures is to build quality benchmarks to assess OO software products that are newly developed or under maintenance, e.g., in the context of large scale software acquisition and outsourcing [27]. Ideally, the measures used for this purpose should be consistent indicators of software design problems, and their relationship to external quality attributes, e.g., fault-proneness, should not strongly depend on the characteristics of the environment where the software was developed. We therefore conclude that, in the current stage of knowledge, measures of import coupling appear to be particularly well suited to develop simple and practical quality benchmarks.

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 of OO systems by aggregating fault-proneness predictions for classes. 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, the 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.

In our future work we will also investigate alternative modeling techniques aimed at predicting counts of defects in classes (as opposed to the predicted probability of fault detection in logistic regression) and thereby facilitate the use of design measurement-based prediction models. Techniques such as Poisson or negative-binomial regression analysis [26] should be investigated, depending on the defect distributions at hand.

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.

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, C. Bunse, J. Daly, C. Differding, "An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents", Empirical Software Engineering 2 (3), 1997. Also available as Technical Report ISERN-96-14.

[3] L. Briand, C. Bunse, and J. Daly, "An Experimental Evaluation of Quality Guidelines on the Maintainability of Object-Oriented Design Documents", Proceedings of Empirical Studies of Programmers: Seventh Workshop (ESP 7), 1997. Also available as Technical Report ISERN-97-02.

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

[5] 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.

[6] 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.

[7] 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 .

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

[9] 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.

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

[11] 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.

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

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

[14] J. Cohen, “A Coefficient of Agreement for Nominal Scales”, Educational and Psychological Measurement, Vol. XX, No. 1, 1960.

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

[16] P. Devanbu, “A Language and Front-end Independent Source Code Analyzer”, Proceedings of ICSE `92, Melbourne, Australia, 1992.

[17] G. Dunteman, “Principal Component Analysis”, SAGE Publications, 1989.

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

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

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

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

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

[23] 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.

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

[25] 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.

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

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

[28] J. Munson, T. Khoshgoftaar, "The Dimensionality of Program Complexity", Proceedings of ICSE'89, Pittsburgh, 245-254, 1989.

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

[30] 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.

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

Document Information

Title:	A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study
Date:	November, 1998
Report:	IESE-047.98/E
Status:	Final
Distribution:	Public
	also published as ISERN-98-29