Using Coupling Measurement for Impact Analysis in Object-Oriented Systems

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

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

Abstract

Many coupling measures have been proposed in the context of object-oriented (OO) systems. In addition, due to the numerous dependencies present in OO systems, several studies have highlighted the complexity of using dependency analysis to perform impact analysis. An alternative is to investigate the construction of probabilistic decision models based on coupling measurement to support impact analysis. In addition to providing an ordering of classes where ripple effects are more likely, such an approach is simple and can be automated. In our investigation, we perform a thorough analysis on a commercial C++ system where change data has been collected over several years. We identify the coupling dimensions that seem to be significantly related to ripple effects and use these dimensions to rank classes according to their probability of containing ripple effects. We then assess the expected effectiveness of such decision models.

1 Introduction

A claimed benefit of object-oriented (OO) modeling approaches is that they result in better maintainable and reusable systems. Encapsulation, inheritance, polymorphism are some of the key mechanisms to achieve this improvement, and set the OO paradigm apart from other, more traditional paradigms.

Maintainability is largely determined by how well a system supports typical maintenance tasks, such as impact analysis or regression testing. Interestingly, prevalent OO languages have been reported to be a difficult case for impact analysis [15], [12]. For instance, typical of OO systems is that implementations of class methods tend to be much smaller than subroutines in a procedural system. Often, methods just invoke other methods, thus passing through the request to another class. In addition, OO features like method overriding, polymorphism and dynamic binding further complicate the method dependency graph and cause the number of dependencies quickly to explode.

Therefore, we have good reasons to assume that a good deal of the cognitive complexity of an OO system lies in the way the system objects collaborate, and less in the implementation of individual methods.

In this context, the main goal of this study is to empirically investigate whether coupling measurement, capturing all sorts of class collaborations, can help perform change impact analysis.

Briefly, our strategy is to use a history of changes observed in a commercial system to determine whether existing coupling measures, for object-oriented systems, can be used to identify classes with common changes (i.e., changes related to the same fault report and having the same ID in the configuration management system). This, in turn, should give us indications about how ripple effects are likely to propagate from class to class. Coupling measures that are indicative of ripple effect-prone class pairs could then be used for a first selection of classes to consider when performing the impact analysis of a change. For example, assume that during maintenance a given class C is identified to require modifications. Based on a static source code analysis, a tool could quickly provide the maintainer with a list of other classes to which class C is coupled to, ranked by their strength of coupling with C. This would restrict the scope for a more refined, detailed dependency analysis, and help to contain the explosion of dependencies that we usually observe in OO systems.

Support to restrict or prioritize the scope of dependency analysis would be more particularly useful when the maintainer is unfamiliar with the system, as it would help to focus on parts of the system with a higher probability of ripple effects, with which he or she should get acquainted with.

The paper is organized as follows. Section 2 provides some background on the system and data used in this empirical study. In Section 3 we investigate the relationship between class coupling and ripple changes. We then further investigate in Section 4, if and how class coupling can be used to focus impact analysis. Related work is discussed in Section 5, we draw our conclusions in Section 6.

2 Study setting

This section provides some background on the LALO system, the change data, and the performed design measurement.

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 90 classes, of which 83 were developed manually from scratch, and the remaining seven classes 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. These were excluded from the analysis as well.

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.

Since its first release in 1993, three additional releases have been deployed. Changes were mostly bug fixes and minor functional enhancements. Therefore, they mainly affected the implementation of classes, mostly preserving their interfaces. The system design was stable across all releases, no classes were added or deleted.

Change Data

The change data was obtained from a revision control system that was used to manage the LALO source code. All changes, since the first release in 1993, were logged and associated with a release ID, a change ID, a list of the classes affected by the change, and the amount of change within each class for each change. Using some shell scripts, we generated from this information a list of all class pairs and the number of changes that were common to each class pair, across all releases.

Coupling data

A tool developed at the Fraunhofer IESE, and based on the FAST parser technology of the SEMA group's Concerto2/AUDIT tools [9], was used to compute coupling between all pairs of classes, using a variety of coupling measures described in the next section.

This measurement was performed on the most recent LALO version. However, as explained above, the structural properties of the LALO system that we are measuring were stable across all releases so that measurement of earlier versions would have yielded almost identical results.

3 Investigating the relationship between coupling and ripple Effects

3.1 Coupling measurement

For this study, we used a number of OO coupling measures identified in a survey of the literature [4]. These measures were mostly defined at the class level, meaning that they count, for a given class C, coupling connections from C to all other classes in the system (or to all classes that have a certain relationship to class C, like inheritance or friendship). We therefore altered the definition of these measures to count the coupling between individual pairs of classes only, and then calculated each measure for all such pairs.

In Table 1, we list the acronym used for each measure, provide informal definitions of the measures, and original literature references where the measures have been proposed. The informal natural language definitions of the measures should give the reader a quick insight into the measures. However, such definitions tend to be ambiguous. Formal definitions of the original measures using a uniform and unambiguous formalism are provided in [4].

Among the existing coupling measures, very few measures investigate indirect coupling connections (via the transitive closure). However, such connections may very well create ripple effects and therefore need to be considered in the context of impact analysis. For this reason, we also defined three new measures, counting both direct and indirect method invocations and aggregation relationships: (1) SIMAS for static method invocations, (2) PIMAS to also take polymorphism into account (i.e., polymorphically invoked methods), and INAG for direct and indirect aggregation relationships.

The above measures were calculated for each pair of classes {C,D}, by adding the number of coupling connections (of the particular measure) from C to D, plus the number of connections from D to C. That is, the direction of couplings between C and D were not considered separately but given equal weight. This is because a ripple change can propagate in any direction along a coupling dimension. This direction ultimately is determined by which class happens to be identified to require change first. The order in which classes are changed, for a given fault report, is to a large extent a random process.

Name	Definition	Source
CBO	Coupling between object classes. Two classes C and D are coupled to another, if methods of one class use methods or attributes of the other, or vice versa. CBO is then a binary indicator, yielding 1 if C is coupled to D, else 0.	[7]
CBO'	Same as CBO, except that coupling is not considered when there is an inheritance relationship between C and D..	[6]
MPC	Message passing coupling. The number of static invocations in class C of methods in class D.	[13]
DAC	Data abstraction coupling. The number of attributes in a class C that have class D as their type.	[13]
DAC'	Binary indicator: 1 if class C has an attribute of type class D, else zero.	[13]
ICP	Information-flow-based coupling. The number of method invocations in a class C, of methods in class D, weighted by the number of parameters of the invoked methods. Takes polymorphism into account.	[14]
IH-ICP	As ICP, but counts invocations of methods of ancestor classes (i.e., inheritance-based coupling) only.	[14]
NIH-ICP	As ICP, but counts invocations to classes not related through inheritance only.	[14]
SIMAS	The number of method invocations in C, which directly or indirectly invoke methods of D. Takes static invocations into account only.	new
PIM	The number of method invocations in C of methods in D. Takes polymorphism into account.	new
PIMAS	The number of method invocations in C, which directly or indirectly invoke methods of D, taking polymorphism into account.	new
INAG	'Indirect aggregation coupling'. Takes the transitive closure of the 'C has an attribute of type D' relation into account. E.g. C has attribute of type D, D has an attribute of type E => C and D are indirectly coupled. The measure yields one if a class pair is directly or indirectly coupled in that manner, otherwise zero.	new
ACAIC OCAIC ACMIC OCMIC AMMIC OMMIC	These coupling measures are counts of interactions between two classes. The measures distinguish the relationship between classes (inheritance, yes or no), and different types of interactions. The acronyms for the measures indicates what interactions are counted: • The first letter indicates the relationship (A: coupling to ancestor classes,, O: coupling to other classes, i.e., not related via inheritance). • 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 IC stand for import coupling.	[1]

Table 1: Coupling measures

3.2 Descriptive statistics

Table 2 summarizes the descriptive statistics for the coupling measures. We provide, for each measure, maximum, minimum, interquartiles, mean (μ) and standard deviation (σ). The values are based on the 83*82/2=3403 class pairs formed by the 83 classes of the LALO system.

Measure	Max	75%	Med	25%	Min	μ	σ
CBO	2	0	0	0	0	0.18	0.57
CBO'	2	0	0	0	0	0.16	0.53
MPC	61	0	0	0	0	0.40	2.38
ICP	140	0	0	0	0	1.20	5.40
IHICP	54	0	0	0	0	0.12	1.59
NIHICP	140	0	0	0	0	1.08	5.18
DAC	6	0	0	0	0	0.03	0.21
DAC'	1	0	0	0	0	0.02	0.15
ACAIC	2	0	0	0	0	0.00	0.03
OCAIC	6	0	0	0	0	0.03	0.21
ACMIC	5	0	0	0	0	0.02	0.23
OCMIC	46	0	0	0	0	0.23	1.72
AMMIC	24	0	0	0	0	0.05	0.70
OMMIC	61	0	0	0	0	0.34	2.28
SIMAS	173	0	0	0	0	1.45	7.84
PIM	64	0	0	0	0	0.76	3.26
PIMAS	374	3	0	0	0	4.12	16.7
INAG	2	0	0	0	0	0.05	0.22

Table 2: Descriptive Statistics for Coupling Measures

The 75th percentile is mostly zero, except for measure PIMAS. The low mean values and higher standard deviation values indicate very skewed distributions of the measures. This makes sense, since we would expect a class to be strongly coupled to only few other classes. Therefore, there is no coupling between the majority of class pairs. More than 25% of all class pairs are connected according to measure PIMAS, which also has the highest mean and standard deviation. This measure takes indirect method invocations and polymorphism into account. From all the measures considered here, it is the least restrictive with respect to the types of coupling connections.

3.3 Principal component analysis

Given the similarity of the definitions of some of the coupling measures, we expect them to be somewhat correlated. In order to better understand the underlying, orthogonal dimensions captured by the coupling measures, and help interpret the subsequent results, we performed principal component analysis (PCA) on our data set.

PCA is a standard technique to identify the underlying, orthogonal dimensions that explain relations between the variables in a data set, i.e., coupling measures in our data set. Principal components (PCs) are linear combinations of the standardized variables. The sum of the square of the coefficients in each linear combination is equal to one. PCs are calculated as follows. The first PC is the linear combination of all standardized variables 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 have large coefficients - also called the loading of the variable - and therefore contribute significantly to the variance of each PC. The variables with high loadings help identify the dimension the PC is capturing, but this usually requires some degree of interpretation.

In order to identify these variables, and interpret the PCs, we consider the rotated components. This is a technique where the PCs are subjected to an orthogonal rotation. As a result, the rotated components show a clearer pattern of loadings, where the variables either have a very low or high loading, thus showing either a negligible or a significant impact on the PC. There exist several strategies to perform such a rotation. We used the *varimax* rotation, which is the most frequently used strategy in the literature. See [8] for more details on PCA and rotated components.

Results from PCA

PCA on the coupling measures for the 3403 pairs of classes identified five principal components that capture 82% of the variance in the data set (we used the "Eigenvalue>1" stopping criterion). To ease the interpretation of the PCs, we subjected them to an orthogonal rotation (varimax). Table 3 provides the loadings of the measures in the rotated components. We also provide the eigenvalue and percentage of variance (%Var.) in the data set explained by each PC, as well as the cumulative variance (Cum %).

	PC1	PC2	PC3	PC4	PC5
Eigen V.	6.91	3.14	2.30	1.29	1.15
% Var.	38.38	17.47	12.80	7.18	6.38
Cum %	38.38	55.85	68.64	75.82	82.20
CBO	0.32	-0.20	0.17	**0.89**	0.05
CBO'	0.34	-0.21	0.04	**0.89**	-0.01
MPC	**0.87**	-0.07	0.14	0.28	0.06
ICP	**0.88**	-0.06	0.12	0.26	0.05
IHICP	0.11	0.01	**0.93**	0.07	0.05
NIHICP	**0.88**	-0.07	-0.16	0.25	0.04
DAC	0.09	**-0.95**	-0.02	0.09	0.09
DACd	0.10	**-0.93**	-0.02	0.11	0.05
ACAIC	0.01	-0.08	-0.04	-0.06	**0.88**
OCAIC	0.08	**-0.95**	-0.01	0.10	-0.05
ACMIC	-0.02	0.01	0.45	0.15	**0.70**
OCMIC	**0.72**	-0.37	-0.09	-0.07	0.01
AMMIC	0.10	0.02	**0.95**	0.08	0.12
OMMIC	**0.87**	-0.08	-0.15	0.27	0.03
SIMAS	**0.72**	-0.06	0.34	0.01	-0.12
PIM	**0.89**	-0.05	0.08	0.27	0.06
PIMAS	**0.67**	-0.01	0.29	-0.10	-0.13
INAG	0.10	**-0.76**	0.02	0.09	0.01

Table 3: Rotated components

Based on these loadings and the definitions of the associated coupling measures, we interpret the PCs as follows:

- PC1: Method invocations to 'other classes'
- PC2: Aggregation coupling
- PC3: Method invocations to ancestor classes
- PC4: CBO.
- PC5: Coupling other than method invocations to ancestor classes.

The distinction between direct and indirect coupling appears not to make a practical difference, as the indirect coupling measures show high loadings (boldface) in the same PCs as their direct coupling counterparts. PIMAS and SIMAS are in PC1 with other direct invocation coupling measures. INAG is in PC2 with aggregation coupling measures. Rather the type of coupling (method invocation, aggregation), and the relationship between classes (inheritance or not) appear to be the distinguishing factors for the coupling dimensions in our data set.

3.4 Univariate regression analysis

Modeling

To determine which of the measures are useful indicators of ripple effects, we perform univariate regression analysis for each individual measure (independent variable) against the dependent variable, i.e., a binary variable indicating the presence of a common change for a class pair (yes or no).

Such a binary variable can be satisfactorily analyzed with logistic regression, a standard technique based on maximum likelihood estimation. In the following, we give a short introduction to logistic regression and full details can be found in [10] or [11].

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

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

π is the probability that a common change was shared by a pair of classes, and X is the coupling measure. The curve between π and X takes a flexible S shape which ranges between two extreme cases:

- When X is not significant, then the curve approximates a horizontal line, i.e., π does not depend on X.
- When X entirely differentiates class pairs with and without common changes, then the curve approximates a step function.

The coefficients c_0 and c_1 are estimated through the maximization of a likelihood function L, built in the usual fashion, i.e., as the product of the probabilities of the single observations, which are functions of the covariates (whose values are known in the observations) and the coefficients (which are the unknowns). For mathematical convenience, l=ln[L], the loglikelihood, is usually the function to be maximized. This procedure assumes that all observations are statistically independent. In our context, an observation is the occurrence of a common change in a pair of classes. Each (non) occurence of a common change is assumed to be an event independent from other such (non) occurences. Each data vector in the data set describes an observation and has the following components: an event category (common change, no common change) and a set of OO coupling measures (described in Section 3.1).

Analysis procedure

The skewed distribution we observed for the independent variables is also present in the dependent variable. 87% of all class pairs are not affected by common changes. This is a problem for the ML estimation used in logistic regression, which then optimizes the model to correctly predict for the prevalent zero outcomes, neglecting the correct prediction of positive outcomes.

Therefore, we used the following approach: we randomly selected 100 class pairs with a common change and 100 class pairs with no common change. We thus have a sample of 200 class pairs with a balanced number of positive and negative outcomes. We then perform univariate analysis with all measures on this sample.

Measure	PC	Number of times significant
AMMIC	PC3	5
MPC	PC1	198
CBO	PC4	200
PIMAS	PC1	198
IHICP	PC3	6
NIHICP	PC1	200
DACD	PC2	93
DAC	PC2	72
SIMAS	PC1	192
OCMIC	PC1	168
PIM	PC1	200
ICP	PC1	200
INAG	PC2	153
OMMIC	PC1	194
CBO’	PC4	200
ACMIC	PC5	6
OCAIC	PC2	67

Table 4: Results from univariate analysis

This procedure was repeated 200 times, each time sampling or resampling 200 classes. We then counted, for each measure, a score of the number of times it appeared as a significant predictor (α=0.05) in univariate logistic regression[1]. This helps determine which of the measures are consistent indicators of common changes and can therefore be used to predict ripple changes. The results are summarized in Table 4, which shows how many times each measure was found to be significant.

We see that measures counting method invocations are frequently significant. These are the measures from PC1. Also, the CBO measures (PC4) are systematically significant, and measure INAG is significant three quarter of the time. Measures capturing other types of coupling (aggregation, PC2, PC5) or coupling between ancestor classes (PC3, PC5) are only seldom significant.

We will rely on these results in Section 4 to select measures and decision models to focus impact analysis.

4 Using coupling to focus impact analysis

For a given class C, we investigate here a way to rank the other classes according to their strength of coupling to C, and we assess how many of the ripple changes we are likely to cover when using such a ranking. This number will give an indication of the usefulness of coupling measurement for the purpose of focusing impact analysis.

4.1 Computation of ranks

Ideally, we would have liked to use logistic regression to build a multivariate prediction model for the probability that a given class pair has a change in common. For a given class C, the classes whose predicted probability to have a change in common with C exceeds a certain threshold (e.g., 50%) would be selected as candidates to be inspected for impact analysis. However, because of the skewed distributions of dependent and independent variables, this approach does not work. Therefore, a different approach was devised.

First, we selected a subset of the measures that

1. are consistently good indicators of common changes in univariate analysis, and
2. cover different dimensions as identified in PCA.

Based on these criteria, we chose measures PIM, CBO’, and INAG to derive the ranking of classes.

Next, for each class C, we separately ranked the 82 class pairs in which C participates by their PIM, CBO’, and INAG values. Then, the separate ranks were averaged into a unique score to derive a final ranking, thus granting equal weight to each of the three significant coupling dimensions.

4.2 Relationship between ranks and ripple effects

In this subsection, we look closely at the relationship between class ranks and ripple effects, as measured by common changes between a class and the corresponding ranked classes. Figure 1 shows the distribution of

- the number of other classes that a given class is coupled to by any one of the mechanism considered by PIM, CBO’ and INAG (bars, the 83 classes are sorted by this number, increasing from left to right)

[1] Many comparisons are performed, on many samples, and it is to be expected that, even for very significant variables, a score of 200 may not be achieved (e.g., MPC) due to normal limitations in statistical power.

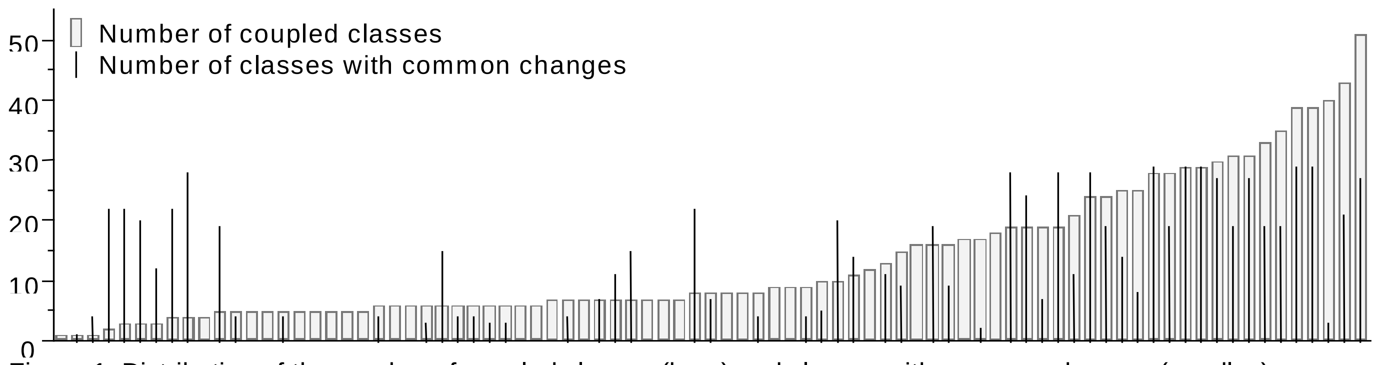

Figure 1: Distribution of the number of coupled classes (bars) and classes with common changes (needles). On the horizontal axis, the 83 classes of LALO are sorted by increasing number of coupled classes.

- the number of other classes a given class has common changes with (needles).

The number of coupled classes (i.e., at least one of the measures PIM, CBO', and INAG is different from zero) ranges from 1 to 51, the mean and standard deviation are 13.5 and 11.5 classes, respectively.

The number of classes a given class has common changes with ranges from 0 to 29, with a mean of 10.2 and standard deviation of 10.5. There are 26 classes having no common changes with other classes.

Figure 1 shows that common changes tend to occur more often in classes that are coupled to many other classes: the needles are more frequent on the right side of the distribution. However, we cannot tell from Figure 1 whether the classes that a given class is coupled to are the same as those it has common changes with.

To further investigate this question, we use the ranking described in Section 4.1 by selecting, for a given class C, the N highest ranked (i.e., N most strongly coupled) classes as candidates to consider for impact analysis. Several questions arise: What is the percentage of common changes that are covered by these N classes, across the 83 classes of the system? How does this percentage increase as N increases?

These questions are addressed in Figure 2. It shows how the percentage of common changes found is distributed, over the 83 classes, depending on the number N of classes selected for impact analysis. The 26 classes with no ripple changes were not considered in this figure, since the 'percentage of common changes found' is not an applicable measurement for these classes. The dashed, straight line is the percentage of common changes that can be expected to be found by chance alone (this percentage is proportional to the percentage of classes selected from all 82 possible classes).

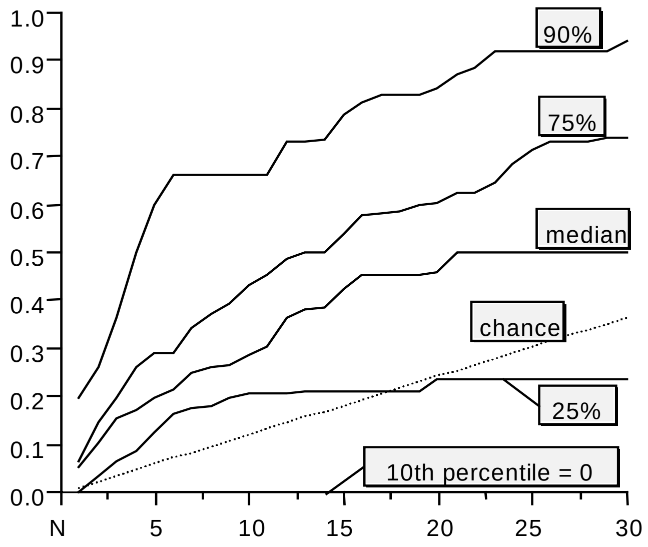

Figure 2: Cumulative distribution of the percentage of common changes found.

We consider some examples in order to illustrate how Figure 2 can be interpreted. For N=10, the median is 0.29. That is, in more than 50% of the cases we find more than 29% of the ripple changes within the 10 classes with the highest rank. This number may seem low, but still it is clearly above what can be expected by chance, namely 10/82=12%. The 90th percentile is at 66.7%, i.e., in 10% of all cases we can expect to find more than two third of the ripple changes within 10 classes.

Note that selecting, for example, N=10 does not imply that for each individual class C, exactly 10 other classes have to checked for impact analysis. Many classes actually are coupled to less than other 10 classes. In Table 5, the row labeled "act. pairs" indicates, for selected values of N, how many class pairs would have to be actually verified for impact analysis, on average, across all 83 classes. Clearly, this number does not increase proportionally with N. As N increases, fewer and fewer class pairs are added to the count, since fewer classes actually are coupled to larger numbers of other classes (see Figure 1).

N	5	10	15	20	25	30
act. Pairs	3.8	6.1	8.2	9.5	11.9	12.7
hits (%)	47.0	47.6	47.7	47.4	41.7	40.8

Table 5: Average number of actually coupled pairs per class, percentage of pairs actually containing ripple changes

From a practical perspective, it is also important to look at the correctness of the model, i.e., how many of the class pairs that our model predicts to have a change in common actually do have a change in common. This is addressed in row "Hits (%)" in Table 5, which indicates, for selected values of N, the percentage of class pairs that actually have a common change. For N<=20, this percentage is constantly around 47%, and then decreases to 40%. In practice, this means that 50% to 60% of the class pairs selected for impact analysis would be investigated in vain.

In Figure 2, similarly to other quantiles, the median reaches its maximum (50%) at N=21. Although the results are significant and much better than one would get by chance, we can see that a significant number of ripple effects would not be covered by selecting highly coupled classes. These classes may be coupled via mechanisms that were not considered here (e.g., in our model, inheritance relationships are only accounted for by measure PIM if invocations of ancestors' methods occur). Also, there exist dependencies between the affected classes that cannot even be detected by static analysis alone (e.g., design decisions affecting several classes, such as the format of inputs and outputs produced). This latter problem is however not specific to our study but concerns any impact analysis based on code dependency analysis.

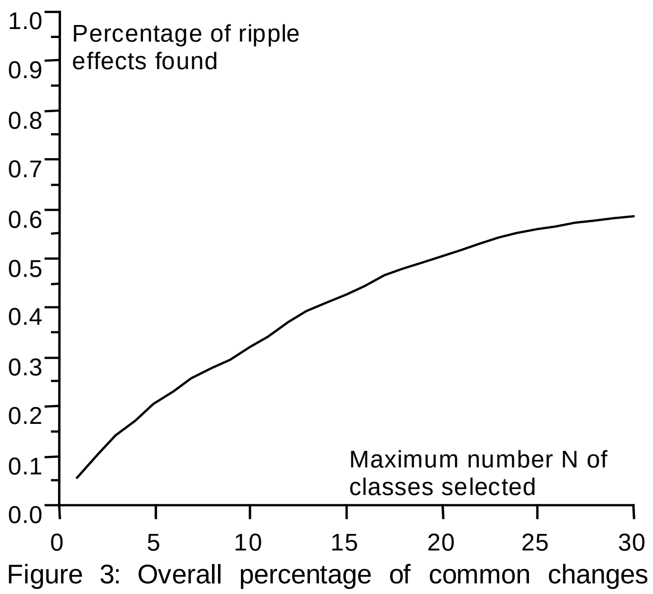

Figure 3: Overall percentage of common changes covered across 83 classes

Figure 3 provides a complementary perspective to Figure 2 and shows the percentage of ripple changes we can expect to find across all 83 (or less) classes for each value of N, the maximum number of classes selected for impact analysis. We can clearly see that the curve asymptote is around 60%.

5 Related work

The only other publication we are aware of that *empirically* investigates coupling and ripple effects in object-oriented systems is [16]. This work focuses on one coupling measure: CBO[2]. The authors investigate if classes with high CBO are more likely to be affected by ripple changes. Although this hypothesis is not supported, CBO is found to be an indicator of change-proneness in general. They also investigate, for ripple changes affecting three or more classes, if each of the affected classes is directly coupled to at least one other class in that group. CBO was found to be insufficient to account for all changes. Other dependencies (inheritance, indirect coupling) needed to be considered to explain the remaining ripple effects.

The focus of their investigation is different from ours, as we try to use existing coupling measurement to identify pairs of classes with common changes, in order to support impact analysis directly. A full, direct comparison of the results is therefore not possible. However, it is clear that changes may ripple through a large variety of dependencies and that no coupling measure is likely to do the job by itself.

Dependency analysis in OO systems received a theoretical treatment in [15], [12], where a detailed investigation into inter- and intra class dependencies and types of changes at the class-, method-, attribute-, and statement level is provided. Our work covers a subset of the dependencies described in there. No empirical data concerning the completeness and correctness of models using these dependencies is given. Future research should investigate if these dependencies can be used to refine and complement the model investigated here, which is mostly based on published coupling measures to date.

6 Conclusions

This study has investigated the use of coupling measurement, and derived decision models, for identifying classes likely to contain ripple changes when another class is being changed. A commercial system, which has been under maintenance, was used to investigate this question. The study shows that a number of coupling measures, related to aggregation and invocation coupling, are related to a higher probability of common changes, that is changes having the same configuration ID and related to the same fault report. This indicates that these coupling measures should be good

[2] CBO is, however, measured differently, and is roughly the sum of DAC and OCMIC as defined in this paper. Such inconsistencies are common in software measurement (see [4] for solutions).

indicators of ripple effects and are used as such in a decision model for ranking classes according to their probability to contain ripple effects. Results show that such a coupling-based model indeed can indicate class pairs with higher ripple effect probability. However, it is also clear that a substantial number of ripple effects are not covered by the selected highly coupled classes. Thus, such models can be used to focus dependency analysis and help reduce impact analysis effort. But other important dependencies are clearly not measured or accounted for, and may not be measurable from code alone, e.g., traceability to common requirements that translates into several methods making common assumptions. It is also very likely that the current set of coupling measures, as defined in the literature, does not fully capture all the code-visible dependencies that are important for impact analysis, e.g., inherited aggregation relationships. Future research should focus on completing the existing coupling measure set and building models derived not only from code, but all sorts of requirement and design documentations, thus providing additional information for coupling measurement. However, more empirical studies are needed in this domain to better understand the most common mechanisms through which changes ripple across object-oriented systems.

Acknowledgments

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

References

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

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

[2] L. Briand, J. Daly, V. Porter, J. Wüst, "A Comprehensive Empirical Validation of Product Measures for Object-Oriented Systems", to be published in the Journal of Systems and Software. Also published as Technical Report ISERN-98-07, 1998.

[3] L. Briand, J. Wüst, S. Ikonomovski, H. Lounis, “A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study”, Proceedings of ICSE’99, LA, USA, 345-354. Also published as Technical Report ISERN-98-29, 1998.

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

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

[6] 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.

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

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

[9] “FAST Programmer’s Manual”, SEMA Group, France, 1997.

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

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

[12] D. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, C. Chen, “Change Impact Identification in Object-Oriented Software Maintenance”, Proc. Conf. Software Maintenance, pp. 202-211, 1994.

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

[14] 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.

[15] N. Wild, R. Huitt, “Maintenance Support for Object-Oriented Programs”, IEEE Transactions on Software Engineering 18 (2), pp. 1038-1044, December 1992.

[16] F. G. Wilkie, B. A. Kitchenham, “Coupling Measures and Change Ripples in C++ Application Software”, published in the proceedings of EASE’99, University of Keele, UK, 1998.