An Empirical Study on the Developers' Perception of Software Coupling

Gabriele Bavota[1], Bogdan Dit[2], Rocco Oliveto[3], Massimiliano Di Penta[4], Denys Poshyvanyk[2], Andrea De Lucia[1]
[1]University of Salerno, Fisciano (SA), Italy
[2]The College of William and Mary, Williamsburg, VA, USA
[3]University of Molise, Pesche (IS), Italy
[4]University of Sannio, Benevento, Italy

***Abstract*—Coupling is a fundamental property of software systems, and numerous coupling measures have been proposed to support various development and maintenance activities. However, little is known about how developers actually perceive coupling, what mechanisms constitute coupling, and if existing measures align with this perception. In this paper we bridge this gap, by empirically investigating how class coupling—as captured by structural, dynamic, semantic, and logical coupling measures—aligns with developers' perception of coupling. The study has been conducted on three Java open-source systems—namely ArgoUML, JHotDraw and jEdit—and involved 64 students, academics, and industrial practitioners from around the world, as well as 12 active developers of these three systems. We asked participants to assess the coupling between the given pairs of classes and provide their ratings and some rationale. The results indicate that the peculiarity of the semantic coupling measure allows it to better estimate the mental model of developers than the other coupling measures. This is because, in several cases, the interactions between classes are encapsulated in the source code vocabulary, and cannot be easily derived by only looking at structural relationships, such as method calls.**

***Index Terms*—Software Coupling; Empirical Studies**

I. Introduction

Coupling has been defined by Stevens *et al.* [1] as "*the measure of the strength of association established by a connection from one module to another*". It is considered as one of the fundamental principles of software engineering, with a strong influence on comprehension and maintenance of large software systems. Understanding the coupling between entities is useful for a variety of software development or maintenance activities, such as assessing software quality [2], [3], predicting fault-proneness [4], [5], [6], supporting change impact analysis [7], [8], [9], [10], re-engineering [11], reuse [12], change propagation [13], and clone management [14].

The importance of coupling has motivated researchers to define different types of measures that capture various aspects of software quality. Most of the coupling measures are *structural* and they statically analyze the source code to capture different relations between entities (e.g., classes), such as the number of calls between two entities, variable accesses, or inheritance relations. Over the years, several alternative measures of coupling have been proposed, such as *dynamic* coupling, which takes into account call relationships occurring during program execution [15], *semantic* coupling, which exploits relations captured from the source code lexicon using Information Retrieval techniques [8], or *logical* coupling, which uses historical data to identify co-changing artifacts [16]. The rationale behind these three coupling measures—dynamic, semantic, and logical—is to capture relations between software artifacts, which are not captured by structural coupling.

Some empirical studies evaluated the potential use of many existing coupling measures and how these measures could be used in a complementary manner [17], [8]. However, despite these attempts to characterize different coupling measures, it is still unclear to what extent such coupling measures reflect developers' perception of coupling between classes, and what sources of information align better with their perception. In essence, the following issue still remains completely unaddressed in the literature after several decades of research:

> *To what extent does the coupling level captured by measures reflects the strength of coupling perceived by developers?*

To answer this question, we conducted an empirical study aimed at investigating how different kinds of coupling measures reflect developers' perception of coupling. The study has been conducted on three open source systems, namely ArgoUML[1], JHotDraw[2], and jEdit[3], and involved 64 students, academics and practitioners, as well as 12 professional software developers that have been active contributors to these open-source systems. Firstly, we analyzed the extent to which these coupling measures overlap. Secondly, we selected a number of class pairs from each system—having both high and low coupling according to the studied measures—and asked participants to assess the strength of those couplings. Thirdly, we analyzed if the perceived coupling aligned with the one captured by the structural, dynamic, semantic, and logical coupling. In order to enrich the generalizability of the results achieved and gain a better understanding of these results, we also analyzed to what extent the experimented measures reflect the mental model of the developers grouping together (i.e., in the same package) classes that were originally grouped by the original developers. Specifically, we used these coupling

[1]http://argouml.tigris.org/
[2]http://www.jhotdraw.org/
[3]http://www.jedit.org/

measures to reconstruct the original modularization of the software systems using a state-of-the-art tool, Bunch [18], and using the **Mo**ve **Jo**in e**F**fectiveness **M**easure (MoJoFM) [19] to evaluate the resulting decomposition.

The contribution of our paper is threefold. First, our study and the results provide important insights into better understanding how developers perceive coupling among classes and which measures align with programmers' opinions. It is an important step in understanding the practical implications of software measurement theory and identifying important future directions. Second, we provide concrete results on how different coupling measures can be used for software modularization, and which measures agree with the original software designers. Finally, the study design, the materials, and the data are made publicly available in our online appendix[4] for replication purposes or to support future studies aiming at evaluating other coupling measures. We believe that the results of this work are a stepping stone for current and future approaches on coupling measures, as well as new studies of software metrics that put developers in the loop.

Structure of the paper. Section II discusses the related literature. Sections III and IV describe the design of our empirical study and present the analysis of the achieved results, respectively. Section V discusses the threats to validity that could affected our study, while Section VI concludes the paper and presents the agenda for future work.

II. Related Work

There are many existing coupling metrics that employ different types of information such as structural, dynamic, semantic, or logical. Most of these metrics define coupling at the class-level granularity by determining the degree to which two classes in an object-oriented system depend on one another. On the one hand, the majority of existing coupling metrics capture coupling between classes structurally. Coupling Between Objects (CBO), Response for a Class (RFC) [20] and Information flow-based coupling (ICP) [21] are among the representative structural coupling metrics. Due to space limitations we refer the interested reader to the unified framework for coupling measurement by Briand *et al.* [17], which provides an overview of the structural metrics.

Furthermore, semantic metrics define a coupling for classes based on textual information extracted from source code identifiers and comments [22]. One such semantic metric, CCBC (Conceptual Coupling Between Classes) [22], captures a new dimension of coupling not addressed by structural or dynamic measures. Logical (or evolutionary) coupling [23], [24], [25] utilizes information from repositories to capture the strength of relations among the artifacts (including source code) that are frequently co-changed. Such logical coupling metrics have been used for impact analysis [26], much like structural coupling metrics. Finally, Arisholm *et al.* [15] introduced dynamic import (i.e., object calls) and export (i.e., called by) metrics to capture the coupling between classes and objects at runtime.

While there have been a number of research papers defining and using coupling metrics for various tasks, there has been only one study [10] that investigated if (any) coupling metrics align with developers' opinions. However, the study by Revelle *et al.* [10] defined and evaluated only structural and semantic coupling metrics. Moreover, those metrics were defined at feature level only. Our study aims at bridging this gap and studying which metrics (and sources of information) are useful from a software developer's perspective. Beck and Diehl [27] measured the congruence between different types of coupling (e.g., structural, semantic, evolutionary, fan-out similarities, code clones and code ownership) and the modularity of a system. Their study on 16 Java systems revealed that (i) low coupling and high cohesion and (ii) information hiding are the two of the modularity principles that are used in practice. Our study complements Beck and Diehl's study, by identifying which types of coupling are perceived as more useful by developers. Moreover, in [28], [29] they showed that evolutionary coupling can be successfully used when performing software clustering if substantially evolutionary data is available.

The study by Counsell *et al.* [30] approached the issue of investigating the developers' perception of cohesion, similarly to what we have done for coupling, i.e., by asking 24 experienced and novice IT professionals to evaluate the cohesion of 10 classes from a Java system. They found that the perceived cohesion (i) is not correlated with the class size, (ii) is correlated to comments density, and (iii) does not depend on the developers' experience. The correlation they found with comments somehow reflect the findings of our paper concerning the capability of coupling to capture developers' perception.

III. Empirical Study Definition and Design

The *goal* of this study is to analyze different types of coupling measures— structural, dynamic, semantic, and logical coupling— and investigate to what extent these measures reflect developers' perception of coupling.

The *context* consists of (i) *objects*, i.e., source code, execution traces, and change history of three Java open-source systems, namely ArgoUML, JHotDraw, and jEdit; and (ii) *subjects* providing their perception of the strength of coupling between entities, identified by the different coupling measures. We recruited both original developers working on these projects, as well as external developers: students, faculty members, and industrial programmers. As for the subject software systems, ArgoUML is an open-source UML modeling tool with advanced features, such as reverse engineering and code generation. JHotDraw is a Java framework for drawing 2D graphics, initially developed to illustrate the applicability of object-oriented design patterns. jEdit is an open-source textual editor developed in Java and supports source code editing. The choice of these three systems is motivated by the need to have: (i) systems of different sizes that are not too small nor too

[4]http://www.distat.unimol.it/reports/coupling

TABLE I
CHARACTERISTICS OF THE COUPLING MEASURES OF THE THREE OBJECT SYSTEMS AND ANALYZED DATA

Measure	Characteristic	ArgoUML 0.34	JHotDraw 6.0b1	jEdit 2.4.1
Structural	KLOC	280	29	28
	# of classes	1,889	289	245
	Exploited metric	Information-flow-based Coupling (ICP)		
Dynamic	Stmt. coverage reached	66%	67%	86%
	Exploited metric	Import Coupling Class Dynamic Message (IC_CD)		
Semantic	Vocabulary Size	7,961	2,834	2,418
	Exploited metric	Conceptual Coupling Between Classes (CCBC)		
Logical	Period of analysis	1998-2006	2000-2005	2000-2012
	Exploited metric	Association rule-based Change Coupling		

large, to allow developers to assess coupling among classes in the context of an entire system; (ii) systems belonging to different problem domains, (iii) availability of sufficient historical data and (iv) the possibility to collect execution traces by manually invoking specific features, which capture the functionality that is exposed to the user through GUI. In addition, in this way we avoid the need to have an accurate test suite to derive execution scenarios. Table I presents the characteristics of three systems involved in our study, relevant for the four sources of information we used to obtain the coupling measures.

A. Research Questions

The study answers the following research questions:

- RQ_1: *To what extent are the sources of information for coupling measures complementary?* This research question is preliminary to the other two, and aims at determining whether the four coupling measures capture the same relations or whether, instead, there are links captured by some measures and not by others.
- RQ_2: *To what extent are the sources of information for coupling measures useful for capturing the strength of coupling as perceived by developers?* This research question is the core of our study, and aims at investigating how developers of these three projects and external ones, perceive the strength of coupling among classes as identified by our four measures.
- RQ_3: *To what extent do coupling measures reflect the mental model of developers when grouping together classes?* Finally, this research question considers the original system modularization as an "oracle" for assessing the coupling measures and determines to what extent a state-of-the-art modularization tool, Bunch [18], is able to produce—using the four measures—class clusters similar to the original ones.

B. Variables and Procedure

Independent Variables. The independent variables of our study are four representative and commonly used coupling measures, each one exploiting a different source of information. The first measure, the *structural coupling* is quantified through the information-flow-based coupling (*ICP*) [21], by statically analyzing the software source code[5]. *ICP* measures the amount of information flowing into and out of a class via parameters through method invocation (i.e., the measure sums the number of parameters passed at each method invocation). Like the majority of coupling measures in literature, this measure is defined at the system level (i.e., for a given class c all method calls between c and all other classes in the system are taken into account). For our study we need to redefine *ICP* to take into account the coupling between a pair of classes. We used the pairwise *ICP* metric as redefined in [8], i.e., the *ICP* between a pair of classes c_i and c_j is measured as the number of method invocations in the class c_i to methods in the class c_j, weighted by the number of parameters of the invoked methods. Then, the overall information-flow-based coupling between the classes c_i and c_j is computed as the maximum between $ICP(c_i, c_j)$ and $ICP(c_j, c_i)$.

We collected execution traces by running the software on an instrumented Java Virtual Machine[6]. For each software system we collected a set of execution traces $T = \{t_1, t_2, \ldots, t_N\}$. Each trace t_i was collected by manually exercising a subset of features of the system. For example, one feature associated with t_i was opening, saving, exporting and closing a file, or creating a UML diagram (for ArgoUML), drawing a shape (for JHotDraw) or editing some text (for jEdit). The dynamic coupling between two classes c_i and c_j was computed by slightly adapting the formula **I**mport **C**oupling **C**lass **D**ynamic Message (*IC_CD*) introduced by Arisholm *et al.* [15]. In its original form, the $IC\_CD(c)$ metric counts the total number of distinct messages (i.e., method calls) sent from class c to all the other classes in the system. However, similarly to the *ICP* metric, for our purpose, we adapted the metric to quantify the dynamic coupling between pairs of classes. In other words, the adapted metric $IC\_CD(c_i, c_j)$ counts all the distinct method calls from objects of type c_i to objects of type c_j. Due to space limitations, we refer the reader to the original formula defined by Arisholm *et al.* [15] in Table 2. The only difference is that our formula replaces $IC\_CD(c_1)$ with $IC\_CD(c_1, c_2)$.

The third measure, the *semantic coupling*, is represented by the Conceptual Coupling Between Classes (CCBC) metric [8]. The CCBC is based on the lexical information derived from comments and identifiers. Two classes are semantically related if the terms present in their comments and identifiers are similar. The conjecture is that if two classes contain similar terms, it is likely that developers used them to describe similar responsibilities. The CCBC between two classes c_i and c_j is computed as the average textual similarity between all unordered pairs of methods from class c_i and class c_j. The textual similarity is computed using Latent Semantic Indexing (LSI) [31]. The CCBC values are in $[0 \ldots 1]$, where 0 represents two classes having a totally different lexicon, while 1 represents two classes containing exactly the same text.

The fourth and last measure is the *logical coupling*. To compute it we first extract change sets from the version control

[5]Static analysis is performed by using Eclipse's AST parser.
[6]We used Eclipse's TPTP (http://www.eclipse.org/tptp/)

system. For ArgoUML and jEdit we rely on SVN change sets. JHotDraw uses CVS, therefore, we grouped together file changes if the commit note and the committer name matched, and if the time interval between them was less than 200 seconds [32]. Then, we applied association rule discovery, similarly to the work by Ying *et al.* [24] and by Zimmermann *et al.* [32] (using the *R* package *arule* and, specifically, the Apriori [33] algorithm) to detect frequent itemsets in co-changes (i.e., groups of files frequently changing together). The rules are ranked in terms of (i) *support* of each itemset X, $supp(X)$ (i.e., the proportion of change sets that contain the learned itemset), and (ii) *confidence* of a rule $X \rightarrow Y$, defined as $supp(X \rightarrow Y)/supp(X)$ (i.e., the fraction of change sets containing X where Y also appears). We set the *confidence* threshold to 0.8 and the *support* threshold to 0.02. The rationale was that (i) the learned rules must be precise enough (hence, the high confidence threshold) and (ii) the itemsets may appear in a limited percentage of the change sets (hence, the low support threshold). Using these thresholds, we assumed that there is a logical coupling between two files if they appeared in a frequent itemset. Finally, we mapped files to classes by analyzing each file's source code.

Dependent Variables. To address these three research questions, we relied on different dependent variables collected through the procedures described below.

For $\mathbf{RQ}_1$, given a pair of measures M_i and M_j, we compared the overlap between coupling links (i.e., pairs of coupled classes) by computing the intersection of the sets of coupling links identified by the two metrics. Note that in this research question we were not interested in analyzing the "value" of the identified coupling links (true or false), but only in estimating the complementarity of these four measures.

For $\mathbf{RQ}_2$, we asked software developers to look into pairs of classes and determine the extent to which they were coupled. We selected a sample of 16 class pairs for each subject system adhering to the following process:

1) we computed the coupling metrics (structural, dynamic, semantic, and logical) between all pairs of classes in the system;
2) for each coupling measure M_i, we selected (i) two pairs of classes having the highest coupling (as stated by M_i) and (ii) two pairs of classes having the lowest coupling (as stated by M_i).

Thus, for each of the four coupling measures we selected four pairs of classes from each system (hence the total of 16 pairs of classes per system). Using this sample we wanted to investigate if the developers perception of high (low) coupling aligns with the values of our underlying software metrics.

We used the following procedure to recruit participants for our study:

1) *Developers working on the three open-source systems*. We sent an invitation only to the topmost committers for all the investigated classes[7]. In total, we invited 19 developers from ArgoUML, 7 from JHotDraw, and 15 from jEdit. We received responses from 6, 3, and 3 developers, respectively. In the following, we will refer to them as *original developers*.
2) *Undergraduate and graduate students, academics, and practitioners from around the world*. Note that while the original developers of the three systems were invited to evaluate only the pairs of classes extracted from the system that they were working on, academics and practitioners were invited to evaluate pairs of classes on all the three systems. Of the 96 invited, 64 (3 undergraduate students, 33 MS students, 16 PhD students, 7 faculty, and 5 people with industrial experience) participated in our study, evaluating the strength of coupling between the pairs of classes from all the systems. In the following discussion, we will refer to them as *external developers*.

[7]To identify such developers, we analyzed the history of software changes.

Each developer received an email (available in our replication package) with instructions on how to perform the task and a link to the website where each developer could log in to visualize and rate the class pairs. More specifically, for each system, each of the 16 webpages provided to the developers contained (i) a class pair to analyze (class names only), (ii) hyper-links to the source code of these classes, (iii) a hyper-link to the source code of the complete project, (iv) an input form containing five radio buttons representing a Likert scale [34] ranging from 1 (two classes are *not coupled*) to 5 (two classes are *strongly coupled*), and (v) a text field where the participant could add explanations for their assessment. Developers were allowed up to four weeks to complete this survey. The study was designed to last no more than 2 hours.

The results of $\mathbf{RQ}_2$ were analyzed through box-plots and statistical analysis using the Mann-Whitney test [35] (the test was performed only on data collected from the external developers, due to the limited number of original developers). We grouped the pairs of classes evaluated by the developers into eight different groups: pairs having $\{low, high\} \times \{semantic, structural, logical, dynamic\}$ coupling. Then, considering two groups at a time (e.g., *high semantic coupling vs. high structural coupling*), we used the Mann-Whitney test to analyze the statistical significance of the difference between the coupling perceived by the developers in pairs of classes classified as *high* or *low* coupled by the two different coupling measures. The results were intended as statistically significant at $\alpha = 0.05$. However, since we performed multiple tests, we adjusted our p-values using the Holm's correction procedure [36]. This procedure sorts the p-values resulting from n tests in ascending order, multiplying the smallest by n, the next by $n - 1$, and so on. We also estimated the magnitude of the difference between the employed metrics. We used Cliff's Delta (or d), a non-parametric effect size measure [37] for ordinal data. The effect size is small for $d < 0.33$ (positive as well as negative values), medium for $0.33 \leq d < 0.474$ and large for $d \geq 0.474$ [37]. Clearly, our statistical analysis was applicable only to the data provided by external developers, since the number of data points obtained from the original developers was too few to run statistical tests. However, we

TABLE II
OVERLAP BETWEEN THE COUPLING LINKS IDENTIFIED BY THE FOUR MEASURES.

Project	Measure	#Links	#Exclusive Links	∩ Str.	∩ Sem.	∩ Log.	∩ Dyn.
ArgoUML	Structural	124,346	48,471	–	61%	<1%	2%
	Semantic	1,078,663	1,002,358	7%	–	<1%	<1%
	Logical	180	0	24%	88%	–	13%
	Dynamic	3,138	89	89%	81%	1%	–
JHotDraw	Structural	5,029	3,042	–	37%	<1%	5%
	Semantic	9,613	7,665	20%	–	<1%	2%
	Logical	52	6	37%	39%	–	29%
	Dynamic	358	31	85%	64%	4%	–
jEdit	Structural	1,924	822	-	54%	<1%	13%
	Semantic	21,152	19,953	5%	–	<1%	2%
	Logical	87	5	52%	40%	–	14%
	Dynamic	453	50	84%	75%	2%	–

used the data from the original developers to corroborate the findings and analyze the results qualitatively.

Finally, for $\mathbf{RQ}_3$, we used each of the coupling measures to reconstruct the modularization of the software systems. More specifically, we simulated the original developers' perception of coupling, by assuming that they decomposed the system trying to group together classes having similar responsibilities (and thus, coupled). As starting point for the modularization algorithm, we flattened the software structure, by putting all the classes in the same package; then, we used the coupling information to re-modularize the software. To this aim, we relied on Bunch [18], which uses a hill-climbing search-based optimization procedure to find a (near) optimal modularization, which minimizes inter-module dependencies (i.e., coupling) and maximizes intra-module dependencies (i.e., cohesion). In our case, the dependencies between classes were provided using the four studied coupling measures, one at a time. The hill-climbing algorithm has been executed 30 times with each measure and on each system to account for its inherent randomness.

To assess the quality of the resulting re-modularization, we compared it to the original decomposition, by computing the MoJoFM [19] between the original modularization of the subject systems (authoritative partitions, considered as oracle) and the one proposed by Bunch. The MoJoFM is computed as follows:

$$MoJoFM(A,B) = 1 - \left(\frac{mno(A,B)}{max(mno(\forall A,B))} \right)$$

where $mno(A,B)$ is the minimum number of *Move* or *Join* operations one needs to perform in order to transform the partition A into B, and $max(mno(\forall\ A,B))$ is the maximum possible distance of any partition A from the partition B of the oracle. Thus, $MoJoFM$ returns 0 if a clustering algorithm produces the farthest partition away from the oracle and it returns 1 if a clustering algorithm produces exactly the oracle.

IV. ANALYSIS OF THE RESULTS

In this section we report the results aiming at answering the three research questions formulated in Section III-A.

A. To what extent are the sources of information for coupling measures complementary?

Table II shows for each system (i) the number of coupling links (i.e., pairs of coupled classes) identified by each coupling measure, (ii) the number of coupling links identified by each measure but not by the other three (column #Exclusive Links), and (iii) the percentage of links that overlap with each measure (on the rows) and the other measures (on the column). For example, for JHotDraw the *semantic* measure captures 37% of the links identified by the *structural* measure, which in turns captures the 20% of links identified by the *semantic* measure. From Table II we can easily observe that the number of links captured by four measures differ by an order of magnitude. As expected, the *semantic* measure is the one capturing more coupling relationships between pairs of classes. This happens because *semantic* coupling is based on LSI (see Section III), and thus, due to the clustering effect of LSI, captures the degree of textual similarity between all class pairs for each system. To alleviate this effect, we filtered out spurious relationships using a threshold (i.e., we discarded class pairs that had textual similarity lower than 0.1, similarly to our previous work on exploiting *semantic* measures [38]). Despite such a pruning, *semantic* measure still captures a high number of coupling links, as shown in Table II. However, a large subset of these coupled pairs of classes exhibit only low semantic coupling (e.g., for ArgoUML, out of the 1,078,663 coupled pairs of classes identified, $\sim 350,000$ have a CCBC value higher than 0.5 and only $\sim 2,000$ higher than 0.9).

The *structural* measure generally captures more coupling links than the *dynamic* one, for two reasons. First, *dynamic* coupling depends on achieved coverage (which in our case is below 100% because we exercised features from the perspective of a developer, hence we did not cover everything). Second, *structural* measures computed statically tend to overestimate the relations (i.e., method calls) between classes. In other words, while the *dynamic* measure captures coupling between two classes only if one class effectively invokes the other at runtime, the *structural* one captures coupling between two classes when there is a call between them in the source code, which may or may not be triggered at run-time. Finally, the relatively low number of *logical* dependencies identified between classes is mostly due to the fact that change sets generally included a limited number of classes. The mean (median) size of a change set is 5 (1) for ArgoUML, 2 (1) for JHotDraw, and 3 (2) for jEdit.

Concerning the complementarity of our four coupling measures, Table II shows how *logical* coupling captures very few links that are not identified by at least one of the other three measures (see column #Exclusive Links). Conversely, both *structural* and *semantic* measures exclusively identify a high number of coupling links and, as expected, the *semantic* measure captures a high percentage of the links identified by the other measures (due to the high number of links identified by it). For example, for ArgoUML the *semantic* measure captures 61%, 88% and 81% of the links identified by the *structural*, *logical*, and *dynamic* measures respectively. A strong overlap is also observed between the *structural* and *dynamic* measures: for the three systems, the *structural* measure captures on average 86% of the links identified by the *dynamic* measure.

On the complementarity of coupling measures. The *semantic* and *structural* measures identify a high number of coupling links that other measures do not capture, also partially complementing each other. Links identified by the *logical* and *dynamic* measures are, for the most part, captured by the other two measures.

The results indicate that *logical* and *dynamic* coupling measures may not be very useful. However, until now we only analyzed the complementarity of these four measures, without looking how such coupling links are perceived by the developers. For example, it might be possible that the only *semantic* coupling links recognized by developers are those that were also identified by the *logical* measure. This analysis is the core of our work and is investigated in the next two research questions.

B. To what extent are the sources of information for coupling measures useful for capturing the strength of coupling as perceived by developers?

Figs. 1, 2, and 3 show boxplots of the Likert values which represent the coupling perceived by the (original and external) developers for ArgoUML, JHotDraw, and jEdit respectively, while Table III shows the results of the Mann-Whitney test (p-value) together with the descriptive statistics of the differences, and Cliff's d effect size (i.e., positive if the left-hand side (lhs) measure is higher than the right-hand side (rhs)). Due to the small number of original developers that responded to our survey, the statistical analysis was performed only considering external developers. Note that the results for high coupling should be interpreted in favor of the lhs if the mean/median/Cliff's d are positive (because the higher the score attributed by developers, the better), whereas for low coupling they should be interpreted in favor of the lhs if mean/median/Cliff's d are negative (because the lower the score attributed by developers, the better).

Concerning **ArgoUML** (see Fig. 1), both original and external developers perceived a *strong* coupling between pairs of classes, indicated by the *semantic* measure as highly coupled (median 5). Also note that for external developers, the *semantic* measure appears to be the one that better aligns with their perceived coupling, since it is the only one achieving a median of 5 for the pairs with "highly coupled classes". Moreover, the Mann-Whitney test highlights a statistically significant difference between the coupling perceived by external developers when looking at pairs of classes having high *semantic* coupling and pairs of classes identified as highly coupled using the remaining three measures (p-values are always < 0.01 - see Table III, with a medium/high effect size). In addition, the *structural* and *dynamic* measures are able to capture the high coupling perceived by developers (both original and external) relatively well, with a median ≥ 4. However, the *logical* measure does not seem to align well with developers' opinions on this system.

To understand the reasons why the *semantic* measure aligns better with developers' opinions in this context,

TABLE III
PERCEIVED COUPLING OF EXTERNAL DEVELOPERS: MANN-WHITNEY TEST (P-VALUE), DESCRIPTIVE STATISTICS OF DIFFERENCES, AND CLIFF'S EFFECT SIZE (d). THE MEASURE THAT BETTER REFLECTS THE DEVELOPER PERCEPTION IN EACH COMPARISON IS HIGHLIGHT IN **BOLD** FACE.

Test	ArgoUML				
	p-value	mean	median	st. dev.	d
semantic high *vs* structural high	<**0.01**	0.56	1.00	1.81	**0.26**
semantic high *vs* dynamic high	<**0.01**	0.50	0.00	1.65	**0.26**
semantic high *vs* logical high	<**0.01**	1.05	1.00	1.43	**0.47**
structural high *vs* dynamic high	0.74	-0.06	0.00	1.48	-0.02
structural high *vs* logical high	**0.01**	0.48	1.00	1.80	**0.21**
dynamic high *vs* logical high	<**0.01**	0.55	0.00	1.59	**0.24**
semantic low *vs* structural low	<**0.01**	-0.17	0.00	0.59	**-0.10**
semantic low *vs* dynamic low	<**0.01**	-0.28	0.00	0.66	**-0.16**
semantic low *vs* logical low	**0.03**	-0.10	0.00	0.43	**-0.07**
structural low *vs* dynamic low	0.11	-0.11	0.00	0.74	-0.06
structural low *vs* logical low	0.25	0.07	0.00	0.63	0.03
dynamic low *vs* **logical** low	**0.02**	0.18	0.00	0.68	**0.08**

Test	JHotDraw				
	p-value	mean	median	st. dev.	d
semantic high *vs* structural high	<**0.01**	1.10	1.00	1.65	**0.45**
semantic high *vs* dynamic high	<**0.01**	0.94	1.00	1.79	**0.37**
semantic high *vs* logical high	<**0.01**	0.70	1.00	1.88	**0.25**
structural high *vs* dynamic high	0.45	-0.16	0.00	1.64	-0.05
structural high *vs* logical high	0.06	-0.40	0.00	2.04	-0.14
dynamic high *vs* logical high	0.45	-0.24	0.00	2.31	-0.09
semantic low *vs* structural low	0.08	-0.24	0.00	1.18	-0.08
semantic low *vs* dynamic low	0.08	0.21	0.00	0.98	0.14
semantic low *vs* logical low	0.42	0.07	0.00	1.18	0.09
structural low *vs* **dynamic** low	<**0.01**	0.45	0.00	1.29	**0.21**
structural low *vs* logical low	0.06	0.31	0.00	1.41	0.16
dynamic low *vs* logical low	0.23	-0.14	0.00	1.09	-0.04

Test	jEdit				
	p-value	mean	median	st. dev.	d
semantic high *vs* structural high	<**0.01**	1.53	1.00	1.49	**0.52**
semantic high *vs* dynamic high	<**0.01**	0.93	1.00	1.85	**0.33**
semantic high *vs* logical high	<**0.01**	0.78	1.00	1.60	**0.30**
structural high *vs* **dynamic** high	<**0.01**	-0.60	0.00	1.51	**-0.21**
structural high *vs* **logical** high	<**0.01**	-0.75	-0.50	1.16	**-0.33**
dynamic high *vs* logical high	0.22	-0.15	0.00	1.35	-0.08
semantic low *vs* structural low	<**0.01**	-0.60	0.00	1.16	**-0.31**
semantic low *vs* dynamic low	<**0.01**	-1.14	-1.00	1.40	**-0.56**
semantic low *vs* logical low	<**0.01**	-0.60	0.00	1.08	**-0.32**
structural low *vs* dynamic low	<**0.01**	-0.54	0.00	1.28	**-0.26**
structural low *vs* logical low	0.97	0.00	0.00	0.71	-0.01
dynamic low *vs* **logical** low	<**0.01**	0.54	0.00	1.28	**0.26**

we analyzed the feedback provided by six original developers of ArgoUML. One developer, when evaluating a pair of classes (i.e., `ActionVisibilityPrivate` and `ActionVisibilityProtected`) with high *semantic* coupling stated: "*these classes have a very high coupling even if there is no cooperation between them*". This comment highlights one important peculiarity of the *semantic* measure (which is shared by the *logical* one): it does not require explicit relations between two classes (e.g., method calls or inheritance relations) to capture coupling between them. Indeed, the two classes mentioned above implement very similar responsibilities. Moreover, the *semantic* measure is the only one capturing coupling between them.

Regarding the pairs of classes exhibiting low coupling, developers showed a coupling perception inline with the coupling measures, except for the *dynamic* coupling, where even if the median is 1, there is a wider distribution of values as compared to the other measures (see Fig. 1).

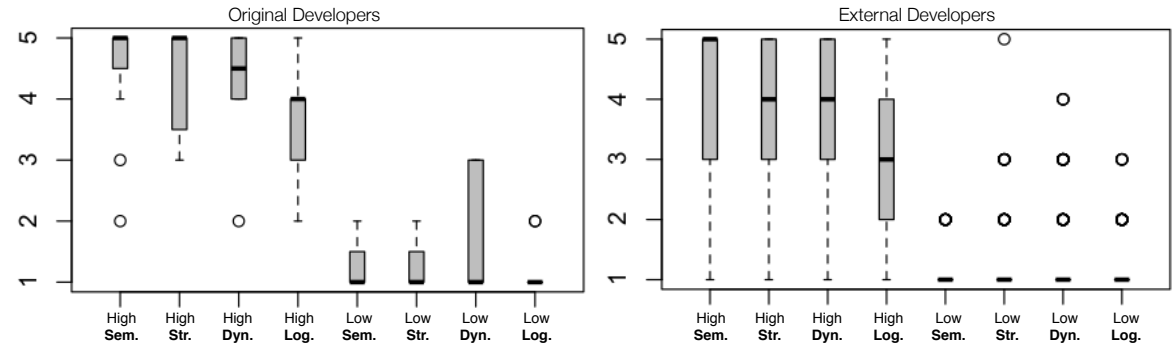

Fig. 1. Boxplots of developers' evaluation on pairs of ArgoUML classes.

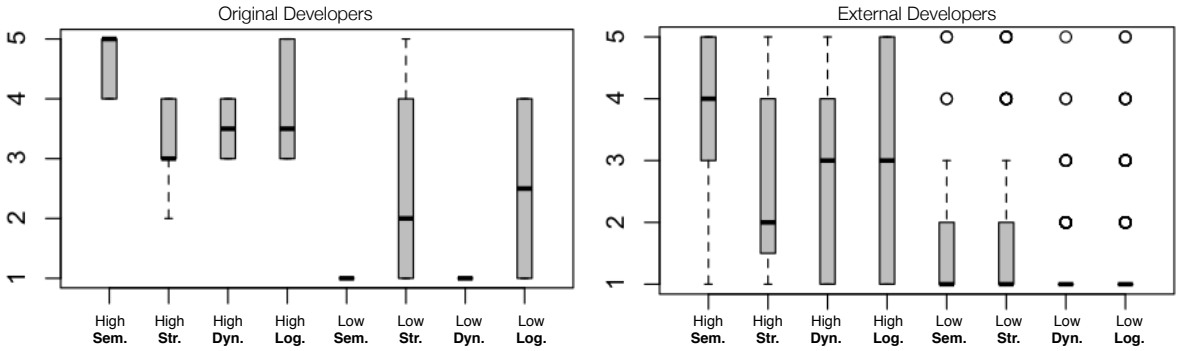

Fig. 2. Boxplots of developers' evaluation on pairs of JHotDraw classes.

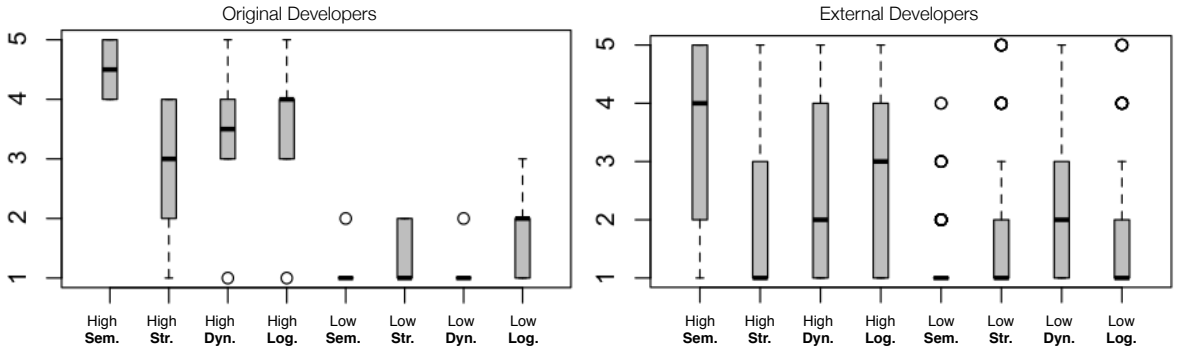

Fig. 3. Boxplots of developers' evaluation on pairs of jEdit classes.

However, there is a statistically significant difference between the coupling perceived by external developers on classes having a low *semantic* coupling with those having low *structural*, *dynamic*, and *logical* coupling (p-value < 0.05 - see Table III, although the effect size is low). This confirms that the *semantic* measure provides a better approximation of the coupling perceived by the developers. Also, *logical* coupling exhibits good results in approximating coupling as perceived by developers for low coupled classes, achieving significantly better results than *dynamic* coupling (p-value 0.02).

We also analyzed the feedback from the original developers to understand the limitations of the *dynamic* coupling. The problem was mainly due to a pair of classes exhibiting low *dynamic* coupling, but evaluated by almost all (original and external) developers as relatively strongly coupled (median is 3). The two classes are `AboutBox` and `FigCompartmentBox`, and an ArgoUML developer mentioned that "*even if indirectly, the two classes exhibit some form of coupling since they implement similar jobs and changes to the GUI are likely to involve both classes*". For these two classes, the only measure capturing some coupling is the *semantic* one, with CCBC=0.48. This further confirms the quality of the *semantic* coupling.

ArgoUML findings. For pairs of classes exhibiting high coupling, the scores for both original and external developers highlight the *semantic* measure as the one providing a better approximation of their perceived coupling. However, also *structural* and *dynamic* measures achieve good results. Similar results are obtained for pairs of classes exhibiting low coupling, but here the *dynamic* measure has been highlighted as the least suitable one.

For **JHotDraw** (see Fig. 2) it can be noticed how, similarly

to ArgoUML, the *semantic* measure is the one that better approximates (for highly coupled classes) the coupling perceived by both original and external developers (median=5 for original developers, and median=4 for the external ones). Indeed, the Mann-Whitney test reports a statistically significant difference between the coupling perceived by external developers on pairs of classes having high *semantic* coupling and the pairs of classes reported as highly coupled by the other three measures (p-value always < 0.01, medium/high effect size). As for the other three measures, the *structural* measure is the most unsuited for capturing coupling between highly coupled classes, even if there is no statistically significant difference between *structural*, *dynamic*, and *logical* coupling. The *dynamic* and the *semantic* coupling are the only ones that properly capture low coupling, having median=1 for both original and external developers (see Fig. 2). On the other hand, the *structural* measure seems to be completely out of sync with developers' perception of low coupling. This is also confirmed by the original developers' feedback, which perceived the pair of classes (`ZoomTool`, `ZoomUpdateStrategy`) as coupled (median=3) and one of them explained that "*these classes are peer features participating in the same context*". Also in this case, the *semantic* measure is the only one capturing some form of coupling between these two classes (i.e., CCBC=0.52). Nevertheless, we need to point out that the *semantic* measure does not represent a silver bullet. We also observed cases like the one for the class pair (`AWTCursor`, `Locator`) classified as not coupled by the *logical* measure, and confirmed as such by developers (median=1), where the *semantic* measure reports a coupling of 0.41 (not very high, nor as low as perceived by developers).

JHotDraw findings. The results confirm our previous findings obtained for ArgoUML: the *semantic* measure is the one that better approximates the coupling perceived by developers on pairs of highly coupled classes, while there is no big difference in performances of the other three measures. The *structural* measure is the least suited for approximating the perceived coupling for low coupled classes.

Finally, for **jEdit** most of the results observed on the other two systems are confirmed (see Figure 3). In particular, the *semantic* measure is still the one that better aligns with developers' perceptions for highly-coupled classes (median=4.5 for original developers and 4 for external), achieving also statistically significant differences when compared to the other three measures (p-value always < 0.01, medium/high effect size). The *logical* coupling is the second best measure for this system, with statistically significant differences with respect to the *structural* one (p-value < 0.01, with a medium effect size), which turned out to be the most inefficient for capturing high coupling. On this system, the *semantic* measure also works particularly well for capturing low coupling, as its performances are significantly better than the other measures (p-value < 0.01 in all cases, with a medium effect size).

jEdit findings. The *semantic* measure is, by far, the most aligned with developers' perception of coupling. The *structural* measure, as observed on JHotDraw, does not properly capture cases of highly perceived coupling, while the *logical* coupling achieves good results.

Coupling captured by the experimented measures and perceived by developers. The *semantic* measure is the closest measure to the developers' perception of coupling. This result can be explained by the fact that developers embed their knowledge and design rationale in the comments and identifiers in code, information that is captured by the *semantic* measure. Surprisingly, the *structural* coupling is the one exhibiting more inconsistencies, especially when capturing low coupling, even more than the *semantic* measure.

C. To what extent do coupling measures reflect the mental model of developers when grouping together classes?

Table IV reports the descriptive statistics for the MoJoFM achieved by each of the four coupling measures when reconstructing the original software modularization of the three software systems. The *semantic* measure is the one achieving better reconstruction accuracy with respect to the original design (average MoJoFM=0.60), followed by *structural* (0.53), *dynamic* (0.43), and *logical* (0.04) measures. These results confirm what we observed in $\mathbf{RQ}_2$: the *semantic* measure is the best one in capturing the developers' perceived coupling even if, overall, none of the measures achieve very good results in this evaluation. In addition, they suggest that semantic coupling should be considered as a useful source of information when performing modularization, although the research performed in the past mainly used static [18] or dynamic [39] information. It is also important to highlight the relatively low performance achieved by the *logical* measure, which may be not due to the low quality of coupling relationships, but rather to the incomplete information (i.e., coupling links) provided by this measure to the clustering algorithm. For example, for the 1,782,272 possible pairs of classes in ArgoUML, the *logical* coupling only provides information for about 180 pairs, making it impossible for the clustering algorithm to even approximate the original decomposition. In essence, this indicates that *logical* coupling could be used for some purpose (e.g., to identify change impacts [24], [32]), but may not be suitable for modularization purposes.

Ability to reconstruct the original modularization. The *semantic* measure is confirmed as the most suitable for capturing the coupling perceived by developers—in terms of the original modularization—followed by the *structural* and the *dynamic* measures.

V. Threats to Validity

Threats to *construct validity* concern the relationship between theory and observation. They are primarily related to the way we measured *structural*, *dynamic*, *logical* and *semantic* coupling. For *structural* [21] and *conceptual* [8] coupling, we used measures that are available in literature. For *dynamic* coupling, we used a measure derived by the one defined by Arisholm *et al.* [15] that we adapted to capture the coupling

TABLE IV
MOJOFM ACHIEVED WITH BUNCH PROVIDING AS INPUT THE FOUR EXPERIMENTED COUPLING MEASURES.

Project	Measure	MoJo FM		
		Mean	Median	St. Dev.
ArgoUML	Structural	0.57	0.62	0.11
	Semantic	0.65	0.69	0.12
	Logical	0.03	0.03	0.01
	Dynamic	0.26	0.29	0.10
JHotDraw	Structural	0.42	0.43	0.16
	Semantic	0.46	0.49	0.10
	Logical	0.03	0.03	0.00
	Dynamic	0.25	0.26	0.08
jEdit	Structural	0.59	0.59	0.09
	Semantic	0.68	0.73	0.11
	Logical	0.05	0.06	0.02
	Dynamic	0.58	0.59	0.16

between pairs of classes, as opposed to capturing the coupling between a class and all the other classes of a system. We are aware they are based on a incomplete (66%-86%) code coverage, however we preferred to use scenarios reflecting users' interaction rather than targeting full coverage. For *logical* coupling, we relied on the approach by Zimmermann *et al.* [32]. Concerning *structural* coupling, due to polymorphism is possible that some call relationships were lost. However, this does not invalidate the results of our study.

For $\mathbf{RQ}_2$ we only sampled 16 class pairs for each system (eight having low cohesion and eight having high cohesion). We are aware that these classes may not be fully representative of the whole system, however we opted for such a choice to allow developers enough time to assess each pair.

Another construct threat is that, for $\mathbf{RQ}_3$, we assumed that the original modularization reflects the developers' perception of coupling. We understand that this may not be fully accurate because developers could have decided to group classes based on other reasons than coupling level. Nevertheless, the use of original modularization as an oracle for clustering approaches is a well-established practice [40], [18].

Threats to *internal validity* concern co-factors that could influence our results, and they are mainly related to various measurement settings used in our study. For $\mathbf{RQ}_3$, we choose the Bunch settings used by other approaches in the literature [18], although we are aware that our results for $\mathbf{RQ}_3$ could have changed if we used different settings. Also, to limit variability of the results due to randomness, we iterated the hill-climbing based modularization 30 times.

Threats to *conclusion validity* concern the relationship between treatment and outcome. Where appropriate we used non-parametric statistical tests (Mann-Whitney) and effect size measures (Cliff's delta) to show statistical significance for the obtained results. Clearly, we could perform statistical tests only on the results collected from external developers. Due to the limited size of the sample, for the original developers we discussed results from a qualitative standpoint.

Threats to *external validity* concern generalization of the obtained results. The main threat could be related to recruiting students and professional—in addition to original developers—to assess the coupling between classes. All participants had previous experience in developing complex Java systems, and in performing program comprehension tasks; however we are aware that knowledge and perception about the systems could not be compared to those of the original developers. Nevertheless, their indications are pretty consistent with the ones provided by the original developers.

Another threat to external validity concerns the choice of coupling measures. Although we used measures available in the literature, we are aware that other kinds of measures—based on the same sources of information—could have exhibited different results. However, our aim was to investigate the coupling contribution provided by the different sources of information, rather than by the specific measure.

Last, but not the least, we limited this study to three Java open source systems. This was mainly due to the fact that we could provide only a limited number of class pairs to be evaluated in a reasonable amount of time.

VI. CONCLUSION AND FUTURE WORK

This paper reported an empirical study aimed at investigating to what extent coupling measures based on *structural*, *dynamic*, *semantic* and *logical* information capture the developers' perception of coupling. The study has been conducted on three Java open-source projects, ArgoUML, JHotDraw, and jEdit, and consisted of (i) analyzing the complementarity of coupling measures using four sources of information; (ii) investigating how developers of these three projects, as well as a larger population of students, academics, and industry professionals rate the coupling links identified by the measures, and (iii) investigating if these four measures are able to reproduce a modularization close to that one of the original system, which indirectly reflects the coupling perception among the classes by original developers.

The results indicate that a large percentage of the links is captured by *semantic* and *structural* measures, which seems to complement each other. In addition, the *semantic* measure aligns well with developers' perceptions of coupling between classes, better than other measures and seems to be the most suitable to reconstruct the original modularization of a software system.

In summary, our results suggest that coupling is not a trivial quality attribute of a software system that could be captured and measured using only *structural* information, such as method calls. More sophisticated approaches, and different source of information, need to be analyzed in order to provide a better evaluation of the coupling perceived by developers. To this end, the semantic coupling seems to reflect the developers' mental model when identifying interaction between entities. This is because, in some cases, there are latent coupling relationships—that do not manifest in terms of structural dependencies—that could be identified only by analyzing source code lexicon and comments.

Future work will aim at extending the study using different kinds of metrics and sources of information. Also, we would like to thoroughly investigate how different coupling measures

can be combined to provide better support for software engineering tasks. Finally, we are planning to replicate this study on other software projects.

ACKNOWLEDGMENT

The authors would like to thank the developers of ArgoUML, JHotDraw and jEdit systems, as well as the other developers that participated in our study. This work is supported in part by NSF CCF-1016868 and NSF CCF-1218129 awards. Any opinions, findings and conclusions expressed here are the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] W. Stevens, G. Myers, and L. Constantine, "Structured design," *IBM Systems Journal*, vol. 13, no. 2, pp. 115–139, 1974.

[2] L. C. Briand, J. Wüst, J. W. Daly, and V. D. Porter, "Exploring the relationship between design measures and software quality in object-oriented systems," *Journal of Systems and Software (JSS)*, vol. 51, no. 3, pp. 245–273, 2000.

[3] H. Olague, L. Etzkorn, S. Gholston, and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes," *IEEE Transactions on Software Engineering (TSE)*, vol. 33, no. 6, pp. 402–419, 2007.

[4] T. Gyimóthy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering (TSE)*, vol. 31, no. 10, pp. 897–910, 2005.

[5] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings of the 30th IEEE/ACM International Conference on Software Engineering (ICSE'08)*, 2008, pp. 531–540.

[6] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Transactions on Software Engineering (TSE)*, vol. 35, no. 6, pp. 864–878, 2009.

[7] L. Briand, J. Wust, and H. Louinis, "Using coupling measurement for impact analysis in object-oriented systems," in *Proceedings of the 15th IEEE International Conference on Software Maintenance (ICSM'99)*, 1999, pp. 475–482.

[8] D. Poshyvanyk, A. Marcus, R. Ferenc, and T. Gyimóthy, "Using information retrieval based coupling measures for impact analysis," *Empirical Software Engineering (EMSE)*, vol. 14, no. 1, pp. 5–32, 2009.

[9] M. Gethers, B. Dit, H. Kagdi, and D. Poshyvanyk, "Integrated impact analysis for managing software changes," in *Proceedings of the 34th IEEE/ACM International Conference on Software Engineering (ICSE'12)*, 2012, pp. 430–440.

[10] M. Revelle, M. Gethers, and D. Poshyvanyk, "Using structural and textual information to capture feature coupling in object-oriented software," *Empirical Software Engineering (EMSE)*, vol. 16, no. 6, pp. 773–811, 2011.

[11] F. Abreu, G. Pereira, and P. Sousa, "A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems," in *Proceedings of the 4th European Conference on Software Maintenance and Reengineering (CSMR'00)*, 2000, pp. 13–22.

[12] S. Chidamber, D. Darcy, and C. Kemerer, "Managerial use of metrics for object-oriented software: An exploratory analysis," *IEEE Transactions on Software Engineering (TSE)*, vol. 24, no. 8, pp. 629–639, 1998.

[13] M. Petrenko and V. Rajlich, "Variable granularity for improving precision of impact analysis," in *Proceedings of the 17th IEEE International Conference on Program Comprehension (ICPC'09)*, 2009, pp. 10–19.

[14] R. Geiger, B. Fluri, H. Gall, and M. Pinzger, "Relation of code clones and change couplings," in *Proceedings of the 9th International Conference of Fundamental Approaches to Software Engineering (FASE'06)*, 2006, pp. 411–425.

[15] E. Arisholm, L. C. Briand, and A. Foyen, "Dynamic coupling measurement for object-oriented software," *IEEE Transactions on Software Engineering (TSE)*, vol. 30, no. 8, pp. 491–506, 2004.

[16] H. Gall, K. Hajek, and M. Jazayeri, "Detection of logical coupling based on product release history," in *Proceedings of 14th IEEE International Conference on Software Maintenance*, 1998, pp. 190–198.

[17] L. C. Briand, J. Daly, and J. Wüst, "A unified framework for coupling measurement in object oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[18] B. S. Mitchell and S. Mancoridis, "On the automatic modularization of software systems using the bunch tool," *IEEE Transactions on Software Engineering*, vol. 32, no. 3, pp. 193–208, 2006.

[19] Z. Wen and V. Tzerpos, "An effectiveness measure for software clustering algorithms," in *Proceedings of the 12th IEEE International Workshop on Program Comprehension*, 2004, pp. 194–203.

[20] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering (TSE)*, vol. 20, no. 6, pp. 476–493, June 1994.

[21] Y. Lee, B. Liang, S. Wu, and F. Wang, "Measuring the coupling and cohesion of an object-oriented program based on information flow," in *Proc. of International Conference on Software Quality*, 1995, pp. 81–90.

[22] D. Poshyvanyk and A. Marcus, "The conceptual coupling metrics for object-oriented systems," in *Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM'06)*, 2006, pp. 469 – 478.

[23] H. Gall, M. Jazayeri, and J. Krajewski, "CVS release history data for detecting logical couplings," in *Proceedings of 6th International Workshop on Principles of Software Evolution (IWPSE'03)*, 2003, pp. 13–23.

[24] A. T. T. Ying, G. C. Murphy, R. T. Ng, and M. Chu-Carroll, "Predicting source code changes by mining change history," *IEEE Transactions on Software Engineering (TSE)*, vol. 30, no. 9, pp. 574–586, 2004.

[25] T. Zimmermann, A. Zeller, P. Weigerber, and S. Diehl, "Mining version histories to guide software changes," *IEEE Transactions on Software Engineering (TSE)*, vol. 31, no. 6, pp. 429–445, 2005.

[26] M. Sherriff and L. Williams, "Empirical software change impact analysis using singular value decomposition," in *Proceedings of the 1st International Conference on Software Testing, Verification and Validation (ICST'08)*. IEEE, 2008, pp. 268–277.

[27] F. Beck and S. Diehl, "On the congruence of modularity and code coupling," in *Proceedings of the 8th joint meeting of the European Software Engineering Conference and the 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'11)*. ACM, 2011, pp. 354–364.

[28] ——, "Evaluating the impact of software evolution on software clustering," in *Proceedings of the 17th Working Conference on Reverse Engineering (WCRE'10)*, 2010, pp. 99–108.

[29] ——, "On the impact of software evolution on software clustering," *Empirical Software Engineering (ESE)*, pp. to appear, doi: 10.1007/s10 664–012–9225–9, 2012.

[30] S. Counsell, S. Swift, A. Tucker, and E. Mendes, "Object-oriented cohesion subjectivity amongst experienced and novice developers: an empirical study," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 5, pp. 1–10, 2006.

[31] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.

[32] T. Zimmermann, P. Weißgerber, S. Diehl, and A. Zeller, "Mining version histories to guide software changes," in *Proceedings of the 26th IEEE/ACM International Conference on Software Engineering (ICSE'04)*, 2004, pp. 563–572.

[33] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993, pp. 207–216.

[34] A. N. Oppenheim, *Questionnaire Design, Interviewing and Attitude Measurement*. London: Pinter, 1992.

[35] W. J. Conover, *Practical Nonparametric Statistics*, 3rd ed. Wiley, 1998.

[36] S. Holm, "A simple sequentially rejective Bonferroni test procedure," *Scandinavian Journal on Statistics*, vol. 6, pp. 65–70, 1979.

[37] R. J. Grissom and J. J. Kim, *Effect sizes for research: A broad practical approach*, 2nd ed. Lawrence Earlbaum Associates, 2005.

[38] G. Bavota, A. De Lucia, and R. Oliveto, "Identifying extract class refactoring opportunities using structural and semantic cohesion measures," *Journal of Systems and Software*, vol. 84, pp. 397–414, March 2011.

[39] J. Gargiulo and S. Mancoridis, "Gadget: A tool for extracting the dynamic structure of java programs," in *Proceedings of the Thirteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2001)*, 2001, pp. 244–251.

[40] R. Koschke and T. Eisenbarth, "A framework for experimental evaluation of clustering techniques," in *8th International Workshop on Program Comprehension (IWPC)*, 2000, pp. 201–210.