Abstract — Coupling metrics capture the degree of interaction and relationships among source code elements in software systems. A vast majority of existing coupling metrics rely on structural information, which captures interactions such as usage relations between classes and methods or execute after associations. However, these metrics lack the ability to identify conceptual dependencies, which, for instance, specify underlying relationships encoded by developers in identifiers and comments of source code classes. We propose a new coupling metric for object-oriented software systems, namely Relational Topic based Coupling (RTC) of classes, which uses Relational Topic Models (RTM), generative probabilistic model, to capture latent topics in source code classes and relationships among them. The case study on thirteen open source software systems is performed to compare the new measure with existing structural and conceptual coupling metrics. The case study demonstrates that proposed metric not only captures new dimensions of coupling, which are not covered by the existing coupling metrics, but also can be used to effectively support impact analysis.

I. INTRODUCTION

Coupling measures capture the degree of interaction and relationships among source code elements, such as classes, in software systems. Coupling is one of the fundamental properties of software with most influence on software maintenance and evolution. A large host of coupling measures are used in tasks such as impact analysis [8, 37], assessment of the fault-proneness of classes [16, 21, 38] and identification of design patterns [1].

A large majority of coupling metrics presented in the literature relies on structural information, which captures relations, such as method calls or attributes usages. However, these metrics lack the ability to identify conceptual associations, which, for example, specify implicit relationships encoded by developers in identifiers and comments of source code.

In this paper we propose a novel coupling measure, namely Relational Topic based Coupling among classes, which is based on Relational Topic Model [14], an unsupervised probabilistic topic modeling technique. RTM identifies latent topics associated with documents (e.g., source code elements) as well as links between documents in a large corpus of text. RTM is based on the Latent Dirichlet Allocation (LDA) [7] technique, which has been recently applied for extracting, representing and analyzing latent topics from the source code [4, 27, 28, 30, 31, 36]. Our measure of coupling can be interpreted through deducing mixtures and relationships of latent topics implemented in software classes within the context of an entire system. The proposed metric is different from existing conceptual coupling metrics [34], as it captures not only how strongly the classes relate to each other conceptually, but also analyzes the relationships among latent topic distributions in underlying identifiers and comments in source code classes.

In order to evaluate a proposed metric, we compare RTC metric against a large host of existing structural and conceptual coupling metrics on the source code of thirteen open-source software systems to examine if the proposed measure captures new dimensions in coupling measurement. We also compare the performance of RTC metric against a host of existing structural and conceptual coupling metrics for impact analysis, an important software maintenance task. Impact analysis entails detecting source code elements impacted by a change to a given source code element. While structural coupling metrics have been successfully used for impact analysis in the literature we conjecture that the proposed RTC metric not only provides good accuracy but also identifies relevant source code elements not captured by structural and/or conceptual coupling metrics.

This paper makes the following research contributions:

• We define a novel coupling metric based on Relational Topic Models for discovering latent topics and their relationships in source code for class coupling measurement.
• We empirically evaluate the newly proposed metric against a host of existing structural and conceptual metrics; we also use and compare RTC against other metrics and their combinations for impact analysis on large open-source systems.
• We publicly release the metrics data from the case studies and ensure their reproducibility.

Section II outlines the related work for object-oriented coupling metrics. Section III describes our approach and the proposed measure. This section also
describes implementation details of the tool that we developed to compute our metrics as well as mathematical properties of the measures. In section IV we provide empirical study to assess the newly proposed metrics. Section V concludes the paper and discusses the future work.

II. RELATED WORK

Measuring coupling among classes is an active research area, which has resulted in many different approaches to capture coupling among classes based on structural static information [11, 15, 25], dynamic information [2], evolutionary data [20, 40], and textual information [34, 35]. Other coupling metrics have been defined to capture coupling metrics for specific types of software applications, such as knowledge-based [23] and aspect-oriented software systems [39].

The structural coupling metrics is the most developed family of coupling metrics, which has received noteworthy attention in the research literature. These coupling metrics have been summarized and classified within the unified framework for coupling measurement [9]. Some of the metrics, widely used and applied within software maintenance tasks, are coupling between objects (CBO) [15], response for class (RFC) [15], message passing coupling (MPC) [26], data abstraction coupling (DAC), information-flow-based coupling (ICP) [25], the suite of coupling measures formulated by Briand et al. [11] (that is, IFCAIC, ACAIC, OCAIC, FCAEC) and efferent coupling (C_e) and afferent coupling (C_a) and coupling factor (COF) [9]. All these and some other coupling metrics have been implemented by tools, such as Columbus [19], which is publicly available to academic community.

The vast majority of existing static structural coupling measures listed above is based on method invocations, attributes references and static or dynamic execute after relationships [6]. For instance, RFC, MPC, and ICP metrics are based on method invocations, whereas CBO and COF measures are based on counting method invocations and references to both methods and attributes. The suite of measures defined by Briand et al. [11] captures several types of relationships among classes, such as class-attribute, class-method, and method-method relations. The measures from the suite also discriminate between import and export coupling and other types of interactions such as friends, ancestors, descendants etc.

In contract to static coupling measures, dynamic coupling measures [2] were defined as the refinement to existing coupling measures to address some of the shortcomings of existing structural metrics while dealing with polymorphism, dynamic binding, and the presence of unused code.

Another rapidly evolving family of coupling measures stems from the evolution of a software system in contrast to static coupling, which is determined by a single-version static program analysis, or dynamic coupling, obtained via program execution. These multi-version metrics are coined as evolutionary [40] or logical [20] couplings among parts of the systems and are determined via analysis of co-changes among source code artifacts using advanced data mining techniques.

Existing work on software clustering [24] uses a notion of conceptual similarity between elements of source code [29], which stands at the foundation of the conceptual coupling [34]. However, the coupling metric, which is proposed in this paper, is different from existing structural and conceptual coupling metrics. First of all, the novel metric, that is RTC, uses an advanced information retrieval method, RTM, which is based on LDA, to extract semantically meaningful topics or concepts implemented in classes. Once topics are gleaned from source code classes, coupling among classes is computed via analysis of relationships among topics using relational topic analysis model. The following section presents details behind adapting RTM and LDA for measuring coupling among classes in OO systems.

III. USING RELATIONAL TOPIC MODELS FOR COUPLING MEASUREMENT IN SOFTWARE

In this work we utilize RTM to capture conceptual relationships between classes in order to determine the degree of coupling among them. RTM is a probabilistic topic model, which models both documents (i.e., classes) and links amid documents (i.e., couplings) within a software corpus. Applications of RTM include assisting social network users in identifying potential friends, locating relevant citations for scientific papers, and pinpointing related web pages of a particular web page [14].

The model extends Latent Dirichlet Allocation to allow for the prediction of links between documents based on underlying topics and textual relationships amongst them. In this section we provide details behind LDA followed by how RTM extends this model to capture links among documents. While LDA has been previously applied in the context of software engineering for measuring conceptual cohesion of classes [27], recovering traceability links [3, 31], mining software repositories [4, 30, 36] and bug location [28], RTM has not been utilized for software measurement tasks before.

A. Latent Dirichlet Allocation

LDA [7], a probabilistic topic model, identifies underlying topics within a corpus and models documents as probabilistic mixtures over those latent topics. More specifically, the topics extracted by LDA correspond to likelihood distributions, which indicate how likely a word is to be assigned for a specific topic. Additionally, each document is modeled as a probability distribution indicating how likely it is that the document expresses each topic. More specifically, given a corpus of documents, that is classes from the software system, LDA attempts to identify a set of topics and associate a set of words with a topic, and define a specific mixture of these topics for each document (i.e., class) in the corpus (i.e., software system).
In order to apply LDA on the source code, we represent a software system as a collection of documents (i.e., classes) where each document is associated with a set of concepts (i.e., topics). More specifically, LDA model consists of the following building blocks (previously defined and applied to cohesion measurement [27]):

- A word is the basic unit of discrete data, defined to be an item from a software vocabulary \(V = \{w_1, w_2, \ldots, w_v\}\), such as an identifier or a word from a comment.
- A document\(^1\), which corresponds to a class, is a sequence of \(n\) words denoted by \(d = (w_1, w_2, \ldots, w_n)\), where \(w_n\) is the \(n^\text{th}\) word in the sequence.
- A corpus is a collection of \(m\) documents (that is, classes) denoted by \(D = (d_1, d_2, \ldots, d_m)\).

Given \(m\) documents containing \(k\) topics expressed over \(v\) unique words, the distribution of \(k^\text{th}\) topic \(t_i\) over \(v\) words can be represented by \(\varphi\), and the distribution of \(k^\text{th}\) document \(d_i\) over \(k\) topics can be represented by \(\theta\). The LDA-based model assumes a prior Dirichlet distribution on \(\theta\), thus allowing the estimation of \(\varphi\) without requiring the estimation of \(\theta\). More specifically, LDA assumes the following generative process for each document \(d_i\) in a corpus \(D\):

1. Select \(N\) – Poisson distribution \(\zeta\)
2. Select \(\theta\) – Dirichlet distribution \(\alpha\)
3. For each of the \(N\) words \(w_j\):
   (a) Select a topic \(t_{i,j} \sim \text{Multinomial}(\theta)\).
   (b) Select a word \(w_j\) from \(p(w_j | z_{i,j}, \beta)\), a multinomial probability conditioned on topic \(t_{i,j}\).

By using LDA it is possible to formulate the problem of discovering a set of topics describing a set of source code classes by viewing these classes as mixtures of probabilistic topics. For further details on LDA, the interested reader is referred to the original work of Blei et al. [7].

B. Relational Topic Model

RTM is a model capable of predicting links between documents based on the context of documents (i.e., underlying latent topics) in a corpus [14]. In RTM, prediction of links, which are modeled as binary random variables, is dependent on the information modeled by LDA (e.g., probabilistic topic assignments to the documents in a corpus).

Generating a model consists of two main steps: modeling the documents in a corpus and modeling the links between all the pairs of documents. The first step is identical to the generative process described in the previous section. The second step is outlined as the following:

For each pair of documents \(d_i, d_j\):

(a) Draw a binary link indicator:
\[ y_{d_i, d_j} \sim \text{Bernoulli}(\psi_e(t_i, t_j)) \]

where \(t_i = \{t_{i,1}, t_{i,2}, \ldots, t_{i,n}\}\)

The link probability function \(\psi_e\) is defined as:
\[ \psi_e(y = 1) = \exp(\eta^T (\mathbf{t}_{d_i} \odot \mathbf{t}_{d_j}) + \nu) \]

where the links between documents are modeled by the means of logistic regression. The \(\odot\) notation corresponds to the Hadamard product, \(\mathbf{t}_{d_i} = \frac{1}{N_d} \sum z_{i,x}\), and \(\exp()\) is an exponential mean function. It is parameterized by coefficients \(\eta\) and intercept \(\nu\). While the complete mathematical details are rather complex and lengthy to be explained in the context of this paper, the reader is referred to the original work on RTM [14] for comprehensive details.

C. Measuring Coupling using Relational Topic Model

Our use of Relational Topic Model to measure coupling among source code classes is motivated by the fact that RTM provides a comprehensive model for describing documents (i.e., classes represented as words from identifiers and comments) and the existence of links between documents based on underlying textual information and other knowledge of the document network. In the context of our application, the binary variable, which indicates whether a link exists between two documents (i.e., classes), is used as an indicator of coupling amid any pair of classes. That is, if the model identifies a link between two classes in the corpus with a high probability, we consider these classes to be coupled. One main benefit of the relational topic model is that it does not require knowledge of any existing links to make these predictions. So, RTM is capable of identifying coupling between classes without any preliminary input related to a priori known class couplings.

Establishing a model of a software system using RTM first requires a term-by-document co-occurrence matrix representation of a software system. With these input documents, which represent source code classes, are modeled as distributions of topics within the corpus as described in the previous section. This model provides necessary underlying information to indentify relationships among the classes in a software system. Links are modeled using the properties of the underlying textual information, captured by LDA and RTM.

Given two classes \(C_1\) and \(C_2\) the degree of coupling (i.e., pair-wise RTC\(_{C}\)) between these is defined as follows:

\[ \text{RTC}_{C} = \text{RTM}(C_1, C_2) \]

where the function RTM() returns the probability that a link exists between the classes \(C_1\) and \(C_2\). The coupling of a class within the context of an entire software system, or a degree to which a class is coupled to the other classes in the system (i.e., system-level RTC\(_{S}\)) is defined as:  

\[ \text{RTC}_{S} = \text{RTM}(C, \text{system}) \]
The definition of the RTC builds on our previous work on measuring cohesion of classes using LDA. However, in this work, we use an extension of LDA, which is based on Relational Topic Models and we do not utilize information theory approach, as in prior work. Computing relational topic coupling among classes in source code indicates whether the classes are conceptually related. Considering the relational topic similarities of a class with all the other classes in the software system, one can measure the degree to which this class relates to the rest of the classes within the context of the entire software system, based on which latent topics they implement and how they are related. These relationships, based on probabilistic topics and their likelihood interactions, delineate a new form of coupling, coined as Relational Topic Coupling.

D. An Example computing RTC using RTM

To provide more insight on how relational topic model is used to capture coupling among software classes in the context of a specific software maintenance task, such as an impact analysis, let us consider a commit from the software system Rhino, which addresses the bug# 204576. Previous applications of coupling measures indicated that a change in a class may trigger ripple changes in other classes, which are highly coupled to the changed class [8]. In this example we illustrate how RTMC measure can capture coupling among related classes in the context of an impact analysis task guided by the RTMC metric.

Assume that while addressing this bug, the developer has located a starting point of a change in one of the classes. Following modification of that class the developer can use RTC for impact analysis to identify remaining classes which also need to be modified, since they are highly coupled to the changed class.

To fix bug #204576, which states "1.5R4 regression: java.lang.String can not be used when argument type is java.lang.CharSequence" the following classes were modified \{NativeJavaClass, NativeJavaMethod, NativeJavaObject\}. Relational topic model represents NativeJavaMethod as a probability distribution over latent topics with the topic #47 (top ranked words consist of "method", "member", "type", etc.) having a relatively high probability. This underlying information leads the model to identify the classes NativeJavaObject, NativeJavaClass, NativeJavaConstructor, and JavaMembers as related to NativeJavaMethod, since the topic distributions which model the classes also show topic #47 with a relatively high probability. In particular, all these classes have a high probability of being associated with topic #47. Thus, using RTC we are able to identify the remaining classes (that is, NativeJavaClass and NativeJavaObject) which also require modification to address the bug #204576. Note that relational topic model considered all topics of a document as well as other underlying information and not just a single topic. We focus on a single topic to simplify the example and its explanation.

IV. Case Study

In this section we present the design of an empirical case study aimed at comparing RTC with other structural and conceptual coupling metrics and analyzing whether the combination of RTC with existing conceptual coupling metrics improves the accuracy of ranking source code classes during impact analysis. The description of the study follows the Goal-Question-Metrics paradigm outlined by Basili et al. [5]. All the data used and generated in this section have been posted online to ensure reproducibility of our results.

A. Definitions and Context

Our primary goals include comparing RTC against existing coupling metrics and determining whether combining RTC with other metrics can improve the performance of coupling metrics when applied to the task of impact analysis. In this study the quality focus was on establishing orthogonality among RTC and existing coupling metrics and improving on accuracy during an impact analysis task, while the perspective was of a software developer performing a modification task on a software system and conducting impact analysis, given a starting point of the change, which requires retrieving the other relevant source code entities that need to be inspected or modified.

The context of this case study consists of 11 C/C++ and two Java software systems. It should also be noted that one of the software systems, that is Mozilla, is implemented in a mix of programming languages including C/C++, Java, IDL, XML, HTML and JavaScript. In case of Mozilla, we analyzed only C++ source code and computed RTC measure among object-oriented classes implemented in C++ only.

1) Coupling metrics

In order to determine whether the newly proposed metrics capture new dimensions in coupling measurement, we selected nine existing structural metrics for comparison: CBO, RFC, MPC, DAC, ICP, ACAIC, OCAIC, ACMIC, and OCMIC (for more details on these metrics, refer to Section II). We considered a subset of these structural coupling metrics relying on the results of previous empirical studies [34], which identified great amount of redundancy among a larger set of existing coupling metrics [13]. In addition to structural coupling metrics, we also considered two system-level conceptual coupling metrics, that is CoCC and CoCCm (and their

Gethers

2 https://bugzilla.mozilla.org/show_bug.cgi?id=204576

pair-wise versions, CCBC and CCBCm) [34]. Other guiding criteria that we used to choose the metrics is availability of the results reported for these metrics elsewhere in the literature [13, 34, 35] to facilitate systematic comparison and evaluation of the results.

2) Subject software systems

For our case study we have chosen 13 various sized open-source software systems from different domains. The source code for these systems, but Mozilla, Eclipse and Rhino, is available at http://sourceforge.net. It should be noted that the software systems 1-10 have been used in the previous studies on evaluating CoCC and CoCCm measures.

ANote (/projects/a-note) is the system that lets the user organize sticky notes on the desktop. TortoiseCVS (/projects/tortoisecvs) is an extension for Microsoft Windows Explorer that makes using CVS convenient and easy. WinMerge (/projects/winmerge) is a tool for visual differencing and merging for both files and directories. Doxygen (/projects/doxygen) is a javadoc like documentation system for C++, C, Java, and IDL. Kalpa (/projects/kalpa) is a multi-user client-server accounting, management, CRM, EPR, and MRP system. K-Meleon (/projects/kmeleon) is a customizable Win32 web-browswer, which uses the same rendering engine as Firefox Mozilla. VoodooUML (/projects/voodoo) is a UML class diagram editor. EMule (/projects/emule) is a file-sharing client; one of the most popular downloads on sourceforge.net. KeePass (/projects/keepass) is a lightweight Win32 password manager, which allows storing the passwords in a highly-encrypted database. Umbrello (/projects/uml) is a system for creating and maintaining UML diagrams. Mozilla is a popular open-source web browser ported on almost all known software and hardware platforms (www.mozilla.org). Eclipse is a popular open-source integrated development environment (www.eclipse.org). Finally, Rhino is an open-source implementation of JavaScript written entirely in Java (www.mozilla.org/rhino).

3) Building and indexing software corpora

In order to capture relational topic based coupling among classes in a software system we need first to generate a corresponding corpus for the system. To build a corpus we extracted the textual information, i.e., identifiers and comments, from the source code using class level granularity level, where each document in the corpus represents a class in the software system (that is, a sequence of identifiers and comments implementing corresponding class).

Once a corpus is built, we model it using Relational Topic Model using the term-by-document co-occurrence matrix corresponding to the corpus. This relational topic model captures important conceptual relationships (i.e., couplings) among classes within the corpus. After modeling the corpus using RTM, coupling between classes can be computed (for the details on how RTM is used to compute RTC refer to the section III). The next section describes specific settings for researchers who wish to reproduce the results of our case study.

4) Setting of the case study

All the structural coupling measures were computed using Columbus [19]. The CoCC and CoCCm measures were computed with the IRC2M tool [34], whereas RTC measure was computed using the R-project’s package of the open-source R-project.

We used a class level granularity, to construct corpora for software systems in the case study. We extracted all types of methods from classes in the source code, including constructors, destructors, and accessors. Comments and identifiers were extracted from each class as well. The resulting text from the source code is preprocessed using the following settings: some of the tokens are eliminated (e.g., operators, special symbols, some numbers, keywords of the C++ programming language, standard library function names including standard template library); the identifier names in the source code are split into parts based on observed coding standards and naming conventions. For instance, all the following identifiers are broken into separate words ‘relational’ and ‘topics’: ‘relational_topics’, ‘RelationalTopics’, etc. Since n-grams were not considered, the order of words in source code is of no particular significance. In this process, RTM does not use a predefined vocabulary, or a predefined grammar, therefore no morphological analysis or transformations are necessary, such as stemming or abbreviation expansion. We evaluate the performance of various numbers of topics and identify values which best suite our application as done in the related work performed by Baldi et al. [4]. We use 75 topics for systems 1 through 10, 225 topics for systems 11 and 12, and 125 topics for system 13. Automatic identification of application specific topic parameters is beyond the scope of this work as we focus primary on demonstrating that, once the appropriate parameters are selected, applying the model as a means of measuring coupling provides useful results.

5) Impact Analysis using Coupling Metrics

The structural [8] and conceptual [35] coupling measures have been shown to facilitate ordering (i.e., ranking) classes in software systems, based on different types of dependencies among classes, that is, structural or conceptual. Such coupling measures and derived ranks of classes, as demonstrated in the previous work can be derived automatically. Here we provide some details on how we compare RTC with other structural and conceptual coupling metrics for the impact analysis task.

For a given class C (considered to be a starting point of a change task, which may be identified by a programmer via a feature location technique, e.g.,

---

4 http://cran.r-project.org/web/packages/lda/
5 http://www.r-project.org/
PROMESIR [33]), the remaining classes in the system are ranked according to their strength of coupling to the class C, based on a coupling measure or a combination of such measures (see some previous work on the details [8, 35]). The list of ranked classes is presented to the developer for further inspection (for instance, a ranked list of classes as shown in an existing tool for impact analysis, namely JRipples [32]). Since software systems may contain thousands of classes, e.g., Mozilla or Eclipse, focusing impact analysis on classes, which are strongly coupled to a starting point, may provide valuable automated support to the developer.

In the research literature, structural and conceptual coupling measures are defined and used at the system level (classic definitions of coupling measures), meaning that they count, for a given class C, all dependencies from C to the other classes in the system. In order to use the coupling measures for impact analysis, they need to be modified to account for coupling between pairs of classes only (see previous work on defining and using pair-wise conceptual [35] and structural [8] coupling measures).

Our evaluation strategy is based on the history of changes, observed in Mozilla, Eclipse and Rhino to compare RTC to existing structural and conceptual coupling measures to identify classes with common changes (i.e., changes in classes related to the same bug report and having the same bug identification number in the configuration management system). The history of changes in this case can be used to evaluate rankings of classes produced with different coupling measures against actual changes in the software system. We conjecture that the RTC measure will be at least as effective as the nine existing coupling measures and two conceptual coupling metrics in ranking classes during impact analysis.

The evaluation methodology can be summarized in the following steps (note that we use the same evaluation methodology as used in related case studies on impact analysis [8, 35]):

1. For a given software system, a set of bug reports Bugs = \{bug_1, bug_2, ..., bug_n\} is mined from the bug tracking system, such as Bugzilla. The set of classes, which had been changed to fix each bug (e.g., c(bug_i) = \{c_1, c_2, ..., c_n\}) are mined from the configuration management system. Specific details on how the bug reports and changed classes are identified are described in [35].

2. For each class in c(bug_i), pair-wise RTC, structural and conceptual coupling metrics are computed. The values of each metric are used to compute ranks of the remaining classes in the software system.

3. Using a specific cut point criteria (which ranges anywhere from 5 to 500 classes), defined as \(\mu\), select top \(n\) classes in each ranked list of results generated by every metric. For every class in c(bug_i), which is used in the evaluation, we assess the accuracy of identifying relevant classes (i.e., the other classes in c(bug_i)) via rankings of specific coupling metrics.

In order to evaluate each coupling measure and compare all the coupling measures, the suggested ranked lists of classes are compared against classes that were actually modified. Average precision and recall measures for each class in c(bug_i) for each bug report are computed for every metric. For each measure, a higher value is more beneficial.

B. Research Questions

We address the following research questions (RQ) within the context of our case study.

- **RQ1**: Is RTC\(^6\) metric orthogonal as compared to existing structural and conceptual coupling metrics?
- **RQ2**: Does RTC\(_C\) outperform existing structural metrics for the task of impact analysis?
- **RQ3**: Does RTC\(_C\) or its combinations with conceptual coupling metrics outperform existing conceptual coupling metrics for the task of impact analysis?

To respond to our research questions we compare RTC with other coupling metrics as well as explore the impact of combining coupling metrics.

C. Metrics and statistical analyses

1) **Precision and Recall**

Precision and recall are two widely used information retrieval metrics, are employed to measure performance of coupling measures for impact analysis. In this context, precision is the percentage of classes correctly identified using our metric out of the total number of classes returned by our metric. Whereas recall indicates the percentage of classes in the set that are correctly identified using a coupling metric. Formal definitions of these metrics are as follows:

\[
\text{precision} = \frac{|S \cap R_n|}{R_n} \times 100\%
\]

\[
\text{recall} = \frac{|S \cap R_n|}{|S|} \times 100\%
\]

where \(S\) represents a set of actually changed classes and \(R\) is the set of highest ranked \(\mu\) entities returned by a coupling metric.

2) **Testing Statistical Significant - Wilcoxon Test**

To demonstrate that our results are unlikely to be obtained by chance we performed statistical testing. To test for statistical significance we utilized Wilcoxon's signed-rank test, a non-parametric paired samples test. The goal of this test within the context of our study was to confirm that the improvement accuracy obtained is statistically significant when compared to a baseline technique.

3) **Principal component analysis**

\(\mu\) It should be noted that we use system-level RTCs measure for PCA and a pair-wise version of an RTC\(_C\) measure for impact analysis.
In order to understand the underlying, orthogonal dimensions captured by the coupling measures (both conceptual and structural) we performed Principal Component Analysis (PCA) on the measured coupling metrics. Applying PCA to metrics data consist of the following steps: collecting the metrics data, identifying outliers, and performing PCA. We applied PCA in the similar manner as in our previous work [29, 31, 36], including procedures on identifying outliers and rotating principal components. Overall, by performing PCA we can identify groups of variables (i.e., coupling metrics), which are likely to measure the same underlying dimension (i.e., specific mechanism that defines coupling) of the object to be measured (i.e., coupling of a classes).

D. Case study results

1) **RQ1** – Results of principal component analysis of the metrics data

In this work we present a new coupling metric, namely RTC. An initial step towards justifying its usefulness is in determining if it captures a unique dimension unexplained by existing metrics. We performed PCA on 979 classes from 10 different open source software systems to answer RQ1. All twelve measures were subjected to an orthogonal rotation. We identified six orthogonal dimensions spanned by 12 coupling measures. The six principal components (PCs) capture 90.97% of the variance in the data set, which is significant enough to support our conclusions for the RQ1.

The loadings of each measure in each rotated component are presented in Table II. Values higher than 0.5 are highlighted as the corresponding measures are the ones we look into while interpreting the PCs. For every PC, we provide the variance of the data set explained by the PC and the cumulative variance in the Table II.

Our results suggest that RTC captures a unique dimension in the data. In this case RTC is the only coupling metric highly correlated with PC6 which explains 8.11% of the variance in the data. The results indicate that structural metrics and conceptual metrics do not capture the same dimensions in the data. For instance, in the case of PC3 and PC6, the principal components are both explained primarily by the conceptual coupling metrics. These results clearly indicate that our coupling measure, that is RTC, captures different types of coupling between classes, than those captured by the structural or even existing conceptual metrics. We believe that this unique result derives from the fact that RTC is a coupling measure that is based on a different underlying mechanism to extract and analyze conceptual information (i.e., RTM is used to compute RTC, whereas, LSI used to compute CoCC).

In addition, the results of the PCA in this work can be compared with those reported in the literature [10, 13, 34]. Although the PCs and component loadings obtained in this case study and those reported in the research literature do not entirely overlap, they are similar. This can be explained by the fact that we used a slightly different set of coupling metrics in our analysis as well as a new metric, that is, RTC.

2) **RQ2** – Comparing results of RTC with structural and conceptual coupling metrics for impact analysis

While answering the previous RQ1 we demonstrated that RTC captures a unique dimension in the data. Our findings demonstrate that the new conceptual coupling metric is capable of explaining a dimension in the data that is overlooked by existing structural and conceptual coupling metrics. Our second research question focuses on justifying its usefulness in practice and detailed comparison with a large body of existing structural coupling metrics, that have been previously utilized for a task of impact analysis in source code [8]. Our goal is to provide insight on whether RTC outperforms structural metrics for the task of impact analysis. We performed impact analysis on Mozilla using some historical information extracted from its version control system to determine which classes had been modified together (for the details on how we apply the metrics for impact analysis and evaluate the results refer to section A.5). We apply both structural and conceptual coupling metrics and obtain the results, which are summarized in the Table III.

Only three metrics are normalized, that is RTC, CCBC and CCBCm. The other coupling metrics are not normalized as they count the total number of coupling connections of a class with the other classes in the system (i.e., the larger the metric value, the stronger the coupling between two classes). The only outlier in our set of metrics is CBO, which corresponds to a binary value indicating if two classes have a coupling connection or not. In case of CBO, we based our evaluation on choosing n coupled classes to a given class instead of using actual metric values as in prior work [35].
In case of each coupling measurement we varied a cut point from 10 to 500 classes respectively. For example, in case of using RTC metric (see Table III), with a cut point of 30 classes, obtained precision was 14.7%, recall was 27.3%. Increasing a cut point to 50 classes provides more candidate classes, thus decreasing precision to 13.1%, but increasing recall values to 36.1%. It should be noted that for this particular cut point RTC outperforms all other coupling metrics (including conceptual and all structural ones) in terms of recall. Overall, based on the results, it is clear that RTC and CCBCm are the top coupling metrics, which significantly outperform all the other structural metrics. While RTC has somewhat lower precision as compared to CCBCm at lower cut points, it has comparable or better recall at cut point 50 or higher (see highlighted RTCC values in Table III). Since we demonstrated in the RQ1 that RTC captures a unique dimension, and performance of RTC is better than a majority of coupling metrics, we conclude that RTC is a useful increment in the field of coupling measurement. Furthermore, we explore if combining RTC with other best performing conceptual metric CCBCm brings any additional improvements in accuracy of coupling-based techniques for the task of impact analysis.

3) **RQ 3** – Results of combining RTC with another conceptual coupling metric for impact analysis

To answer RQ3, we performed impact analysis on two open-source systems, Eclipse and Rhino, using combinations of RTC and CCBC m. In this case the combination is done using a simple affine transformation combination using equal weights, which was previously applied in the context of feature location [33]. We compare combinations of RTC and CCBCm against CCBCm, the existing state of the art conceptual coupling metric, used as a baseline in this case. Our results, which appear in Table IV, indicate that standalone coupling metrics, RTC and CCBCm, provide equivalent performances, which are also confirmed in the RQ2. However, our new results on Eclipse and Rhino corroborate that the combination of RTC and CCBCm provide superior accuracy than either standalone technique. For example, Table IV shows, when performing impact analysis on both Eclipse and Rhino we are able to attain an average of approximately 5% absolute improvement in recall. In many cases, the relative gain in recall is even higher. In particular, while using a combination of RTC and CCBCm metrics with a cut point of 50 classes, obtained absolute gain in recall was 5%, while the relative gain was 11% as compared to CCBCm standalone technique. Overall, the results indicate that, for Eclipse, the acquired gain in precision and recall increases in concert with increasing cut points, whereas the results for Rhino pinpoint that the gain decreases as the cut point increases. This, in part, can be accredited to the differences in sizes of the two systems.

We also performed the Wilcoxon's signed-rank test to examine if these results are statistically significant. We formulate the following null hypotheses:

- **H_{NULL-Precision}:** The combination of RTC and CCBCm does not significantly improve precision results of impact analysis compared to either standalone coupling metric.
- **H_{NULL-Recall}:** The combination of RTC and CCBCm does not significantly improve recall results of impact.

We generate alternative hypotheses for the cases where the null hypotheses can be rejected with relatively high confidence:

- **H_{ALT-Precision}:** The combination of RTC and CCBCm significantly improves precision results of impact analysis compared to conceptual coupling.
- **H_{ALT-Recall}:** The combination of RTC and CCBCm significantly improves recall results of impact analysis compared to conceptual coupling.

Table V presents the results of our test. The test is performed for each of the null hypotheses presented. Our findings signify that the results obtained are statistically significant for alpha 0.05 for both Eclipse and Rhino.

Overall, the results of the case study indicate that RTC is a new useful coupling metric, which can serve as an effective indicator of an external property of classes in OO systems, that is, the change proneness of classes.

Table II. The results for precision (P) and recall (R) of applying RTC and existing coupling metrics for impact analysis in Mozilla

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>R</th>
<th></th>
<th>P</th>
<th>R</th>
<th></th>
<th>P</th>
<th>R</th>
<th></th>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC</td>
<td>17.3</td>
<td>14.0</td>
<td>15.8</td>
<td>22.0</td>
<td>14.7</td>
<td>27.3</td>
<td>13.8</td>
<td>31.6</td>
<td>13.1</td>
<td>36.1</td>
<td>9.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCBCm</td>
<td>27.8</td>
<td>14.6</td>
<td>24.7</td>
<td>22.1</td>
<td>18.4</td>
<td>34.5</td>
<td>18.4</td>
<td>34.5</td>
<td>12.6</td>
<td>43.1</td>
<td></td>
</tr>
<tr>
<td>ICP</td>
<td>11.9</td>
<td>6.9</td>
<td>10.1</td>
<td>9.7</td>
<td>8.6</td>
<td>16.5</td>
<td>8.6</td>
<td>16.5</td>
<td>6.5</td>
<td>22.8</td>
<td>4.13</td>
</tr>
<tr>
<td>PIM</td>
<td>11.3</td>
<td>6.60</td>
<td>9.84</td>
<td>9.56</td>
<td>8.52</td>
<td>16.3</td>
<td>8.52</td>
<td>16.3</td>
<td>6.52</td>
<td>22.6</td>
<td>4.12</td>
</tr>
<tr>
<td>CCBC</td>
<td>10.8</td>
<td>5.6</td>
<td>9.5</td>
<td>8.9</td>
<td>6.7</td>
<td>14.1</td>
<td>6.7</td>
<td>14.1</td>
<td>5.2</td>
<td>19.8</td>
<td>3.99</td>
</tr>
<tr>
<td>CBO</td>
<td>7.2</td>
<td>6.2</td>
<td>5.4</td>
<td>9.4</td>
<td>2.8</td>
<td>11.3</td>
<td>2.8</td>
<td>11.3</td>
<td>1.6</td>
<td>12.0</td>
<td>1.05</td>
</tr>
<tr>
<td>MPC</td>
<td>6.6</td>
<td>5.7</td>
<td>3.9</td>
<td>6.7</td>
<td>1.7</td>
<td>7.0</td>
<td>1.7</td>
<td>7.0</td>
<td>0.9</td>
<td>7.2</td>
<td>0.72</td>
</tr>
<tr>
<td>OCMIC</td>
<td>2.0</td>
<td>2.1</td>
<td>1.1</td>
<td>2.2</td>
<td>0.5</td>
<td>2.3</td>
<td>0.5</td>
<td>2.3</td>
<td>0.3</td>
<td>2.5</td>
<td>0.47</td>
</tr>
<tr>
<td>OCAIC</td>
<td>1.7</td>
<td>2.0</td>
<td>1.0</td>
<td>2.1</td>
<td>0.4</td>
<td>2.1</td>
<td>0.4</td>
<td>2.1</td>
<td>0.2</td>
<td>2.3</td>
<td>0.45</td>
</tr>
<tr>
<td>DAC</td>
<td>1.8</td>
<td>2.0</td>
<td>1.0</td>
<td>2.1</td>
<td>0.4</td>
<td>2.1</td>
<td>0.4</td>
<td>2.1</td>
<td>0.2</td>
<td>2.3</td>
<td>0.19</td>
</tr>
<tr>
<td>ACMIC</td>
<td>0.9</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.1</td>
<td>0.6</td>
<td>0.40</td>
</tr>
<tr>
<td>ACAIC</td>
<td>0.8</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.5</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Gethers 8
impact analysis and can be effectively combined to gain superior results with other conceptual metrics, such as RTCC. More importantly, the statistical tests support our hypothesis that the combination of RTCC with other conceptual coupling metric allows us to build useful impact analysis techniques for detecting class change ripple effects based on coupling metrics.

E. Threats to validity

We identify several issues that affected the results of our case study and limit our interpretations. We have demonstrated that our metrics capture new dimensions in coupling measurement; however, we obtained these results by analyzing classes from ten C++ open-source systems. In order to permit generalization of the results, large-scale evaluation is necessary, which needs to take into account software from different domains, developed using different programming languages.

In the case study we consider structural (that is, static) metrics that are based on the static information obtained from the source code. The results may be rather different if we considered dynamic coupling metrics [2].

The conceptual coupling measures (that is, CoCC, CCBCin, RTCs and RTCC) depend on coherent naming conventions for identifiers and comments. When these are missing, the emphasis for capturing coupling should be placed on static or dynamic coupling metrics.

RTC measure, as currently defined, does not take into account polymorphism and inheritance. The measure only considers methods of a class that are implemented or overloaded in the class. One of the existing solutions in the research literature, which accounts for inheritance, consists of extending the conceptual measures to include the source code of inherited methods into the documents of derived classes [24]. We are planning to incorporate this solution in our future work.

In our case study on impact analysis we used two large (i.e., Eclipse and Mozilla) and one medium-sized (i.e., Rhino) software systems, however, to allow for generalization of these results, again, large-scale evaluation is necessary, which should take into account several releases of software systems from different domains, implemented in different programming languages (that is, not only C++ and Java as covered in our case study).

Also, our evaluation on Mozilla, Eclipse and Rhino is based on the changed classes extracted from patches in corresponding bug reports. This could have somewhat impacted evaluation procedure as these patches may contain incomplete information about actually changed classes or these changes could have introduced other bugs. We alleviate this issue by considering only patches which are officially approved by module owners in Mozilla, Eclipse and Rhino. Moreover, this threat to validity is minimized by the fact that the bug data that we utilize in this case study have been used and verified by other researchers [17, 35].

V. Conclusions and Future Work

The paper defines a novel operational measure for the relational topic based coupling of classes, which is theoretically and empirically validated. An extensive case study indicates that RTC captures new dimensions in coupling measurement as compared to existing structural and conceptual metrics. Likewise, RTC has been shown useful during impact analysis on large software systems.

The paper lays the basis for the future work that makes use of the relational topic models for conceptual coupling measurement. The proposed metric could be further extended and refined, for example by taking into account inheritance. Another direction is to improve the quality of underlying textual information by applying advanced source code pre-processing techniques for splitting [18] and expanding [22] identifiers and comments in source code. Since RTC relies on textual information, we are considering including external documentation in the corpus. This should allow extending the context in which words are used in the software and identifying inconsistencies in latent topics generated from source code and external documentation.

REFERENCES


Table III. Precision (P) and recall (R) percentages results for impact analysis in Eclipse and Rhino using various cut points

<table>
<thead>
<tr>
<th>System</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>P</td>
<td>R</td>
<td>P</td>
<td>R</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Eclipse</td>
<td>RTC+C+CCBCin</td>
<td>21</td>
<td>25</td>
<td>17</td>
<td>35</td>
<td>15</td>
<td>40</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>CCBCin</td>
<td>19</td>
<td>23</td>
<td>15</td>
<td>31</td>
<td>13</td>
<td>36</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Absolute gain</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Relative gain</td>
<td>11</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>15</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Rhino</td>
<td>RTC+C+CCBCin</td>
<td>14</td>
<td>38</td>
<td>11</td>
<td>57</td>
<td>9</td>
<td>68</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>CCBCin</td>
<td>13</td>
<td>32</td>
<td>11</td>
<td>52</td>
<td>9</td>
<td>64</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Absolute gain</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Relative gain</td>
<td>8</td>
<td>19</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Table IV. Results of Wilcoxon signed-rank test

<table>
<thead>
<tr>
<th>System</th>
<th>H0P</th>
<th>H0R</th>
<th>Null Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>&lt; 0.0001</td>
<td>&lt; 0.0001</td>
<td>Reject</td>
</tr>
<tr>
<td>Rhino</td>
<td>0.012</td>
<td>0.009</td>
<td>Reject</td>
</tr>
</tbody>
</table>
Languages and Applications (OOPSLA'08), Nashville, Tennessee, 2008, pp. 543-562.


