Modeling Class Cohesion as Mixtures of Latent Topics

Yixun Liu[*], Denys Poshyvanyk[*§], Rudolf Ferenc[†], Tibor Gyimóthy[†], Nikos Chrisochoides[*]

[*]*Computer Science Department*
The College of William and Mary
Williamsburg, VA 23185
{enjoywm, denys, nikos}@cs.wm.edu

[†]*Department of Software Engineering*
University of Szeged
Szeged, Hungary
{ferenc, gyimi}@inf.u-szeged.hu

Abstract

The paper proposes a new measure for the cohesion of classes in Object-Oriented software systems. It is based on the analysis of latent topics embedded in comments and identifiers in source code. The measure, named as Maximal Weighted Entropy, utilizes the Latent Dirichlet Allocation technique and information entropy measures to quantitatively evaluate the cohesion of classes in software. This paper presents the principles and the technology that stand behind the proposed measure. Two case studies on a large open source software system are presented. They compare the new measure with an extensive set of existing metrics and use them to construct models that predict software faults. The case studies indicate that the novel measure captures different aspects of class cohesion compared to the existing cohesion measures and improves fault prediction for most metrics, which are combined with Maximal Weighted Entropy.

1. Introduction

Software cohesion can be defined as a measure of the degree to which elements of a module belong together [6]. The principle of high cohesion for classes is one of the goals of the OO analysis as it facilitates program comprehension, testing, reusability, maintainability, etc. Software cohesion metrics have been shown to support different software maintenance tasks, such as assessment of design quality [4, 10], productivity, design, and reuse efforts [12], prediction of software quality [17, 23, 34, 39], modularization of software [32], and identification of reusable components [18].

Cohesion is usually measured on structural information extracted entirely from the source code (e.g., attribute references in methods and method calls) that captures the degree to which the elements of a class belong together from a structural point of view. These measures provide information about the way a class is built and how its instances work together to address the goals behind their design. Thus, they provide no details as to whether the class is cohesive from a conceptual point of view (for example, whether a class implements one or more domain concepts) nor do they give an indication about the readability or comprehensibility of the source code [34]. Although other types of metrics were proposed by researchers to capture different aspects of cohesion, only a few such metrics address the conceptual and textual aspects of cohesion [19, 34].

We propose a new measure for class cohesion, namely **M**aximal **W**eighted **E**ntropy (MWE), which captures the conceptual aspects of class cohesion based on the analysis of latent topics encoded in source code, expressed in identifiers and comments. We use the Latent Dirichlet Allocation (LDA) technique, which has been recently applied for extracting, representing and analyzing latent topics from the source code [3, 30, 35]. Our measure of cohesion can be interpreted as measuring mixtures of latent topics implemented in software classes within the context of the entire system. The proposed metric is different from existing conceptual cohesion metrics, such as C3 [34], as it measures not only how strongly the methods of a class relate to each other conceptually, but also analyzes the coverage and degree of latent topic distributions in methods of underlying source code.

The paper makes the following contributions:

- We propose a novel application of Latent Dirichlet Allocation technique for discovering latent topics in source code for class cohesion measurement; we also propose a novel measurement mechanism using information entropy based on distributions of latent topics in class methods.
- We thoroughly evaluate the newly proposed metric against a host of existing structural metrics; we also use and compare *MWE* against other metrics and their combinations for predicting faults in classes on a large open-source system.

[§]Corresponding author

2. Related work

Based on the underlying information used to measure class cohesion, we can broadly classify the measures into structural [6, 9, 13, 25, 26, 44], semantic or conceptual [14, 19, 33, 34], information entropy-based [1], slice-based metrics [37], and metrics for specific types of applications such as knowledge-based [27] and aspect-oriented [43] systems. The structural cohesion metrics, which are summarized in the unified framework for cohesion measurement [9], is the most investigated category of cohesion metrics and includes lack of cohesion in methods $LCOM_1$ and $LCOM_2$ [13], $LCOM_3$ and $LCOM_4$ [26], Co (connectivity) [26], $LCOM_5$ [25], Coh [9], TCC (tight class cohesion) and LCC (loose class cohesion) [6], ICH (information-flow-based cohesion) [29], etc. Most structural metrics define and measure relationships among the methods of a class based on class variable referencing and data sharing between methods as contributing to the degree to which the methods of a class belong together. The differences among the structural metrics can also be viewed based on the definition of the relationships among methods, system representation, and counting mechanisms employed. Recently, other structural cohesion metrics have been proposed, improving existing metrics by considering the effects of dependent instance variables [11, 44].

The developers often reason about a class as a set of responsibilities that approximate the concept from the problem domain implemented by the class as opposed to a set of method-attribute interactions. These domain concepts are partially encoded in comments and identifiers in source code. Among existing cohesion metrics, the Logical Relatedness of Methods (LORM) [19], the Lack of Conceptual Cohesion in Methods (LCSM) [33] and Conceptual Cohesion of Classes (C3) [34] are the only ones that use this type of information to measure the conceptual similarity of methods in a class. The idea behind this category of metrics is that a cohesive class is considered to be a crisp implementation of a problem or solution domain concept. Therefore, if the methods of a class are conceptually related to each other, the class is cohesive. For instance, C3 captures the conceptual aspects of class cohesion, as it measures how strongly the methods of a class relate to each other conceptually, whereas the conceptual relation between methods is based on the principle of textual coherence [21]. The conceptual and structural metrics have been recently combined to improve the class cohesion [14].

The metric, which is proposed in this paper, is different from existing structural and conceptual metrics. First of all, the new metric, namely *MWE*, uses advanced information retrieval method, LDA, to extract semantically meaningful topics or concepts implemented in classes. Once topics are gleaned from source code, class cohesion is computed via analysis of topic distributions using information entropy measures. The following section presents details behind adapting LDA and information theory approaches for measuring cohesion of classes in OO systems.

3. Capturing Cohesion using Information Theory and Information Retrieval

Object-oriented analysis and design methods decompose the problem domain addressed by a software system development into classes as an attempt to control complexity. High cohesion for classes and low coupling among classes are design principles aimed at reducing the system complexity. The most desirable type of cohesion for a class is model cohesion [16] such that the class implements a single semantically meaningful concept. This is the type of cohesion that we are measuring in our approach. We use Latent Dirichlet Allocation [7] to discover *latent topics* (i.e., semantically meaningful concepts) and then use weighted information entropy to measure the degree to which methods of a class concentrate on each topic. The maximum weighted entropy (i.e., *MWE*) is a measure of class cohesion as it reflects the degree to which the methods concentrate on a main topic implemented in a class.

3.1. Overview of LDA

LDA is a statistical model, specifically a topic model, originally used in the area of natural language processing for representing text documents. The basic idea behind LDA is that text documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [7].

Given a corpus of documents, LDA attempts to: (a) identify a set of topics; (b) associate a set of words with a topic, and (c) define a specific mixture of these topics for each document in the corpus.

LDA has been previously applied in the context of software engineering for mining software repositories [3, 30, 35] and bug location [31].

3.1.1. LDA model

The terms for describing LDA model are as following:

- A *word* is the basic unit of discrete data, defined to be an item from a vocabulary $V=\{w_1, w_2,..., w_v\}$.
- A *document* is a sequence of n words denoted by $d = (w_1, w_2,..., w_n)$, where w_n is the n^{th} word in the sequence.

- A *corpus* is a collection of m documents denoted by $D = (d_1, d_2, \dots, d_m)$.

Given m documents containing k topics expressed over v unique words, the distribution of i^{th} topic t_i over v words can be represented by φ_i and the distribution of i^{th} document d_i over k topics can be represented by θ_i. The general method to find $\varphi=\{\varphi_1,\varphi_2,\dots,\varphi_k\}$ and $\theta=\{\theta_1, \theta_2,\dots,\theta_m\}$ is to use the expectation-maximization method [15]. However, this approach is susceptible to problems involving local maxima and is slow to converge [7], encouraging the development of models that make assumptions about the source of θ.

Blei et al. [7] proposed a new model, namely Latent Dirichlet Allocation. The LDA based model assumes a prior Dirichlet distribution on θ, thus allowing the estimation of φ without requiring the estimation of θ.

LDA assumes the following generative process for each document $d(w)$ in a corpus D:

1. Choose $N \sim$ Poisson distribution(ξ)
2. Choose $\theta \sim$ Dirichlet distribution(α)
3. For each of the n words w_i:
   (a) Choose a topic $t_i \sim$ Multinomial(θ).
   (b) Choose a word w_i from $p(w_i|z_n,\beta)$, a multinomial probability conditioned on topic t_i.

Viewing documents as mixtures of probabilistic topics makes it possible to formulate the problem of discovering a set of topics that are described in a collection of documents. There are different ways to find the LDA parameters such as variational Bayes [7], expectation propagation [28], and Gibbs sampling [22]. In our research, we employed GibbsLDA++[1], which is a C/C++ Implementation of Gibbs Sampling, to perform LDA analysis on open source software. For further details on LDA, the interested reader is referred to the work of Blei et al. [7].

3.1.2. Applying LDA to source code

To apply LDA on source code, we consider a software system as a collection of documents (i.e., methods) and each document is associated with a set of concepts (i.e., topics). The mapping between LDA model and source code entities is shown in Table 1.

This mapping was used before [35], however, we refine it to account for fine grain source code elements, such as methods and classes.

Once we generate a corpus for a software system, we can apply LDA to extract a set of topics and their distributions among different documents (i.e., methods) in the corpus (i.e., software system). The details on specific settings of applying LDA to source code are summarized in section 4.2.2.

[1] http://gibbslda.sourceforge.net/ (accessed and verified on 02/18/09)

Table 1. Mapping LDA to Source Code

LDA Model	Source Code Entities
word	Identifiers and comments extracted from source code, which comprise the vocabulary set. Further, this set is refined to exclude programming language keywords, stop words and punctuation. Finally, all compound identifiers are split based on the observed naming conventions. $V=\{w_1, w_2,\dots, w_v\}$.
document	A method is treated as a document, which can be expressed as n identifiers and comments from vocabulary, which appear in implementation of a method $m_i=(w_1,w_2,\dots,w_n)$
set of documents	A class corresponds to a collection of documents representing methods of a class $C_i = (m_1, m_2, \dots, m_l)$
corpus	The software system consists of a set of classes $S = (C_1, C_2, \dots, C_z)$ which forms a corpus.

3.2. Information entropy

Our cohesion metric employs information entropy to calculate the degree of distribution for each topic based on the resulting document-topic distributions obtained with LDA.

In information theory, *entropy* (also referred to as self-information) is a measure of the uncertainty associated with a random variable [40]. Information entropy has been previously used to measure software complexity [24], cohesion and coupling [1]. However, information entropy has never been used in conjunction with Information Retrieval to measure software cohesion before.

For a random variable x with n outcomes $\{x_i: i=1,\dots, n\}$ the Shannon information entropy, a measure of uncertainty is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log p(x_i) \qquad (1)$$

where $p(x_i)$ is the probability value of outcome x_i.

The idea of using entropy to measure class cohesion is as the following. We take topic t_i as a random variable with l outcomes $\{m_j: j=1,\dots, l\}$. In this case, $p(m_j)$ is the probability that topic t_i is assigned to method m_j (see Table 1), which is computed using LDA. If topic t_i intersects all the methods evenly, the class should have a high cohesion, because topic t_i appears to be a common underlying theme among all the methods in a class. At the same time the value for the entropy is also high in this case. This is because for a topic to "*cross all the methods evenly*" means that this topic can be assigned to any method with the same probability and implies higher uncertainty (i.e., *higher entropy* from information theory perspective). If this topic occurs with higher probability only in one of the methods, we can determine where this topic should belong to, which implies lower uncertainty and thus lower entropy. Noticeably, classes with higher cohesion must have higher entropy and vice versa. Thus, entropy provides us with a reliable mechanism to measure cohesion of classes. The aforementioned definitions focus on the software cohesion induced by only one topic t_i. Nonetheless, a class may relate to a mixture of topics with varying probabilities, so we must take into account all the topics and relationships among them to properly capture software cohesion. We provide the details behind the proposed metric in the following section.

3.3. Measuring cohesion using LDA and information entropy

Let us consider the following example of a class, which consists of two methods implementing three different topics (see Figure 1).

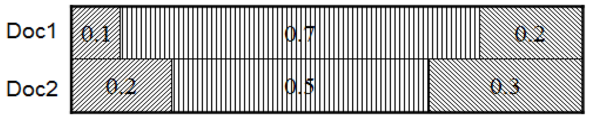

Figure 1. Probability distributions of three topics in two methods (documents) in a class

We can observe that the second topic (vertical filling line pattern in Figure 1), which is relevant to the first document with probability p_t=0.7 and to the second document with probability p_t=0.5, is the dominating topic in the class (as compared to the other two topics). We can summarize the following characteristics, which are pertinent to the second topic:

- *Occupancy (or weight)*: it resides in all the methods (i.e., documents) in the class;
- *Distribution (or entropy)*: it is nearly evenly distributed across the methods in the class.

Intuitively, a highly cohesive class should have one main topic (i.e., refer to a single crisp problem domain concept [16]). We quantify these two characteristics for every class in the software system under analysis to measure class cohesion. We define our new metric *MWE* for class cohesion as the following:

$$MWE(C_z) = \max_{1 \le i \le |t|}(O(t_i) \times D(t_i)) \qquad (2)$$

where, *O(t*$O(t_i)$ is a measure of *occupancy* of topic t_i in methods in a class C_z; $D(t_i)$ is a measure of degree of *distribution* of topic t_i in methods in a class C_z; $|t|$ is the number of topics, extracted from all the classes in a software system.

The class cohesion is measured by taking into account values of occupancy (weight) and distribution (entropy) for the dominating topic, which is addressed in the class. If the maximum value for such a topic is low, this means that the class does not have a distinctive topic or a theme, which can be attributed to reduced class cohesion.

The occupancy of topic t_i in methods of a class C_z is computed as:

$$O(t_i) = \frac{\sum_{d=1}^{n} p_{t_i}^{d}}{n} \qquad (3)$$

where n is the number of methods in a class and $p_{t_i}^{d}$ is the probability of topic t_i in a method d. The occupancy captures the average probability of topic t_i across all methods in a class. In our example, occupancy values for all three topics implemented in a class are *$O(t_1)$=(0.1+0.2)/2=0.15, $O(t_2)$=0.6, $O(t_3)$=0.25.*

In this paper we employ information entropy to measure the degree of topic distributions in methods of a class. Information entropy implies rather high uncertainty and cohesion for topics that are distributed uniformly. In order to compute entropy for a topic t_i we need to transform document-topic probability distributions p_{t_i} to topic-document distributions q_{t_i} as the following:

$$q_{t_i}^{d_j} = \frac{p_{t_i}^{d_j}}{\sum_{d=1}^{n} p_{t_i}^{d}} \qquad (4)$$

In our example, $q_{t_2}^{1}$ *=0.7/(0.7+0.5)=0.583* and $q_{t_2}^{2}$ *=0.5/(0.7+0.5)=0.417.*

Once topic-document distributions q_{t_i} are obtained, distribution $D(t_i)$ is computed using information entropy equation (1). It should be noted that $D(t_i)$ is normalized to account for the number of methods in a class:

$$D(t_i) = \frac{\sum_{d=1}^{n} -q_{t_i}^{d} \times \log(q_{t_i}^{d})}{\log n} \qquad (5)$$

In our example, distribution values for all three topics implemented in a class are *$D(t_1)$=0.92*, *$D(t_2)$*=0.98, *$D(t_3)$=0.97.*

At last our proposed cohesion metric is computed as:

$$MWE(C_z) = \max_{1 \le i \le |t|}(O(t_i) \times D(t_i)) =$$

$$= \max_{1 \le i \le |t|} \left(\frac{\sum_{d=1}^{n} p_{t_i}^{d}}{n} \times \frac{\sum_{d=1}^{n} -q_{t_i}^{d} \times \log(q_{t_i}^{d})}{\log n} \right) \qquad (6)$$

In our example, *MWE = max {0.15×0.92, 0.6×0.98, 0.25×0.97}=0.588*

If $O(t_i)$ and $D(t_i)$ were not normalized, the metric would be susceptible to the cases, where occupancy and distribution values depend of the number of methods in a class. However, our metric definition (see equation 6) is capable of differentiating among various degrees of class cohesion induced by diverse configurations of occupancy and distribution values.

The new metric not only takes into account the occupancy (or weight) of the topic, but also takes into account its degree of distribution (or entropy). The idea behind equation (6) is to find the main topic, which is characterized by the capacities: larger average occupancy and more even distribution for classes with higher cohesion (and vice versa). **M**aximal **W**eighted **E**ntropy (i.e., MWE) adequately quantifies these two characteristics and thus, measures cohesion of classes. The procedure for computing MWE is summarized in the following steps:

```
1.  Run LDA to generate document-topic distribution matrix
    Doc-Top(M, N) for the corpus, which has M documents
    (methods) and N topics
2.  For each class $C_i$ (assume it has m methods)
3.  Extract a sub matrix Doc-Top(m, N) from Doc-Top(M, N)
    which corresponds to m documents in class $C_i$
4.  Calculate topic-document distribution based on Doc-
    Top(m, N) using equation (4)
5.      For each topic $t_j$
6.          calculate occupancy $O(t_j)$ using eq (3)
              based on sub-matrix Doc-Top(m, N);
7.          calculate distribution $D(t_j)$ using
              equation(5) based on Doc-Top(m, N);
8.          save $O(t_j)$* $D(t_{ji})$ to WE(j)
9.      End For
10.     Find max MWE for class $C_i$ in WE using eq (6)
11. End For
```

3.4. Example of measuring class cohesion

To better understand the *MWE* metric, let us consider the class *CircleArea* from source code of *Mozilla* 1.6. The class contains three methods: *IsInside*, *Draw* and *GetRect*. This class has a higher *MWE* value of 0.64. The occupancy and distribution metrics for each topic in this class are plotted in Figure 2.

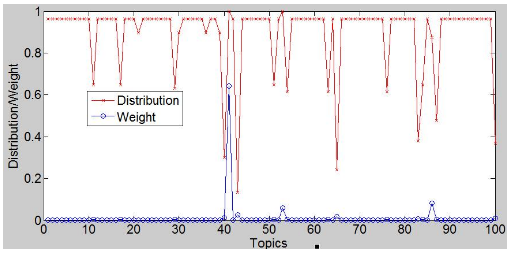

Figure 2. Topic occupancy and distribution values for the class *CircleArea* from *Mozilla*

The occupancy (i.e., weight) curve indicates that the topic# 41 has the highest value, which means that most parts of this class concentrate on this topic.

From the distribution (i.e., entropy) curve we can see that topic# 41 evenly crosses all the methods because its distribution value is relatively high. The weighted entropy, which is the product of these two curves, is shown in Figure 3.

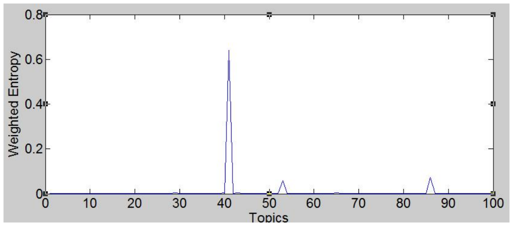

Figure 3. Weighted Normalized Entropy for the topics in *CircleArea* class from *Mozilla*

The weighted entropy (a product of occupancy and distribution values) corresponding to the topic# 41 is much higher than for the other topics. This result indicates that only topic #41 has high occupancy and high distribution values at the same time, which are the main characteristics of a highly cohesive class. While examining words comprising topic# 41 (see Table 2), we observe that this topic closely relates to the *drawing* concept.

Table 2. Top 16 words and their probabilities that comprise the topic #41 *Drawing* in *Mozilla*

Word	p(w)	Word	p(w)
Context	0.0360	Color	0.0122
Rendering	0.0189	float	0.0112
NS	0.0167	Device	0.009
x	0.0161	Draw	0.009
nscoord	0.0158	Pixel	0.008
y	0.0154	Get	0.007
To	0.0141	rect	0.006
ns	0.0117	Units	0.006

While examining the three methods of *CircleArea* (see Table 3), we also discover that all these methods are closely related to the concept captured by topic #41 – *drawing function*.

These methods have no other responsibilities, but drawing function; hence this class has a high cohesion. Other structural cohesion metrics (e.g., $LCOM_1$=10, $LCOM_3$=5, $LCOM_4$=5.) also indicate that *CircleArea* is a highly cohesive class.

Table 3. Partial document-topic distribution matrix for class *CircleArea* from *Mozilla*

		Topics			
		Drawing range	**Drawing**	**Drawing context**	**View**
Docs	**Inside**	0.003	**0.420**	0.062	0.012
	Draw	0.002	**0.671**	0.002	0.002
	GetRect	0.002	**0.462**	0.002	0.202

4. Empirical Assessment of Class Cohesion

In this section, we present the results of two case studies aimed at comparing and combining *MWE* with a set of existing cohesion measures. Sections 4.1 and 4.2 describe the objectives and the design of the case studies. In Sections 4.3 and 4.4 we present quantitative results separately for each case study. The data for our case studies is publicly released for research purposes[2].

4.1. Objectives and methodology

In order to evaluate our measure, we conducted two case studies. The goal of the first case study is to determine whether the *MWE* measure captures additional dimensions of cohesion measurement when compared to existing cohesion measures. Our hypothesis is that, given the novel mechanism employed by *MWE* (e.g., identifying the core latent topic, which is evenly addressed by most methods in a class), it should capture different aspects of class cohesion as compared to the host of existing measures.

In the second case study, *MWE* is compared with existing metrics to assess whether they provide better results in predicting faults in classes. Our conjecture is that combining *MWE* with other cohesion metrics should be a more complete indicator of cohesion given that they capture different aspects of it. In summary, the case studies address the following research questions:

- **RQ1** *Does MWE capture aspects of class cohesion that are not captured by other structural and/or conceptual cohesion metrics*?
- **RQ2** *Do the combinations of cohesion metrics with MWE provide better results in predicting faults in classes than the combinations of the other metrics*?

4.2. Design of the case studies

We designed the studies according to the guidelines presented by Yin [42].

4.2.1. Software system and metrics

We conduct case studies on *Mozilla*[3] version 1.6, which is an open source Web browser ported on almost all known software and hardware platforms.

We selected the following structural cohesion metrics: LCOM$_1$, LCOM$_2$, LCOM$_n$[4], LCOM$_3$, LCOM$_4$, LCOM$_5$, Coh, ICH, and TCC. Moreover, we considered previously published metrics for conceptual cohesion of classes, such as C3 [34] and LCSM [33] to compare with *MWE*. In addition, we computed the LOC metric as it was extensively used in the models for predicting faults in software [23, 36].

Our choice of metrics is guided by the fact that these metrics were extensively studied and compared to each other and to other metrics in previously published studies [4, 9, 10, 33, 34]. We choose these metrics because of availability of the results in the literature in order to facilitate comparison and evaluation with our results. For the definitions, explanations, and further references on these measures, refer to Section 2.

4.2.2. Settings of the case studies

The settings of the case studies are similar to our previous work [34]. All of the structural metrics are computed using Columbus [20] and the conceptual cohesion metrics for C3 were obtained from the previously published study [34]. Overall, we obtained structural, conceptual and *MWE* metrics for 2,068 classes in *Mozilla*. The corpus for *Mozilla* was constructed using the same settings as in the previous work [34]. The extracted corpus resulted in 35,571 documents (i.e., methods) and 110,691 unique terms (i.e., words). However, while indexing the corpus with LDA, the following parameter settings have been applied (see Table 4).

Table 4. Parameter setting for GibbsLDA++

Params	Value	Params	Value
Alpha	0.25	#docs	35,571
Beta	0.10	#words	110,691
#topics	100	#iterations	1,000

The reader is referred to the manual of GibbsLDA[5] for detailed information on how to set these parameters.

4.3. RQ1 - principal component analysis of metric data

In order to understand the underlying orthogonal dimensions captured by the cohesion measures, we performed the **P**rincipal **C**omponent **A**nalysis (PCA) on all the metrics computed for *Mozilla*. PCA is a technique that has been used in a number of previous case studies to identify important underlying dimensions captured by a set of metrics. We performed PCA with the same settings as in prior studies [33, 34, 38], with a goal to identify groups of variables (that is, metrics), which likely measure the same underlying dimension (that is, the mechanism that defines cohesion) of the object to be measured (that is, the cohesion of a class). In order to identify these variables and interpret the principal components (PCs), we consider the rotated components, which is a technique where PCs are subjected to an orthogonal rotation. Thus, the resulting

[2] http://www.cs.wm.edu/~denys/data/icsm09-mwe (posted 03/19/09)
[3] http://www.mozilla.org/ (accessed and verified on 02/18/09)
[4] LCOM$_n$ is a version of LCOM$_2$ metric, except its value is not set to zero when the subtraction (the number of pairs of methods without shared attributes minus the number of pairs of methods with shared attributes) is negative.

[5] http://gibbslda.sourceforge.net/ (accessed and verified on 02/18/09)

Table 5. Results of PCA analysis

	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6
Prop	42.83	19.43	12.05	7.17	4.99	3.61
Cumul	42.83	62.26	74.31	81.48	86.5	90.1
C3	-0.15	-0.17	0.26	**0.88**	-0.02	0.28
LCSM	-0.50	-0.26	0.01	-0.17	**0.78**	0.08
LOC	0.65	0.61	0.07	-0.21	-0.06	0.10
$LCOM_1$	**0.91**	0.28	-0.03	0.09	0.04	-0.10
$LCOM_2$	**0.96**	0.07	0.01	0.07	0.10	-0.09
$LCOM_n$	**0.96**	0.06	0.01	0.06	0.11	-0.10
$LCOM_3$	**0.94**	-0.11	0.09	0.07	0.07	-0.11
$LCOM_4$	**0.81**	-0.34	0.01	0.17	0.09	-0.30
$LCOM_5$	0.66	0.17	-0.23	0.22	0.27	0.21
ICH	0.58	0.56	0.16	-0.22	-0.01	**0.42**
TCC	-0.38	**0.83**	-0.27	0.16	0.07	-0.12
LCC	-0.26	**0.87**	-0.28	0.13	0.07	-0.07
Coh	-0.65	0.45	-0.31	0.22	0.08	-0.21
MWE	-0.26	0.32	**0.84**	0.04	0.08	-0.13

rotated components show clearer patterns of loading for the variables. The PCA was performed on the data set consisting of metric values for 2068 classes from *Mozilla*. The PCA results are summarized in Table 5. We interpret the loadings determined for every PC as follows:

PC_1 (42.83%): $LCOM_1$, $LCOM_2$, $LCOM_n$, $LCOM_3$, $LCOM_4$. These metrics count the number of pairs of methods that share instance variables. Another commonality among $LCOM_1$-$LCOM_4$ is that these measures are not normalized and they do not have upper bounds.

PC_2 (19.43%): TCC and LCC. These are among the measures that are computed as the ratio of method pairs with shared instance variables, also considering indirect sharing of instance variables by method invocations. Noticeably, the measures are also normalized.

PC_3 (12.05%): *MWE*. This is our conceptual cohesion metric that measures the cohesion of a class in the context of the complete software system based on the maximum weighted entropy. Note that it captures even more data variance than another conceptual cohesion metric, C3 (see PC_4).

PC_4 (7.18%): C3. This is a conceptual cohesion metric that measures the cohesion of a class in the context of the complete software system based on the usage of the terms shared between pairs of methods in a class, assuming that there is an underlying or latent structure in word usage for the software system for which a document set (that is, corpus) is constructed.

PC_5 (4.96%): LCSM. This is another conceptual cohesion metric that inversely measures cohesion. A higher value for LCSM indicates lower cohesion.

PC_6 (3.61%): ICH. This is information flow-based cohesion measure based on the information strength (i.e., method invocations weighted by the number of parameters invoked) among the methods of a class.

The PCA results indicate that *MWE* defines a dimension of its own − *MWE* is the only major factor in PC_3. These results *statistically* support our hypothesis that the *MWE* cohesion measure captures different aspects of what is considered to be a cohesion measurement of the class (**RQ1**), as defined by all the metrics computed in the case study. The PCA results also demonstrate that the dimension defined by *MWE* captures more variance than the dimension defined by another conceptual metric C3. The low correlation value between *MWE* and C3 also reinforces this conclusion.

4.4. RQ2 – predicting faults in *Mozilla* classes

The first case study confirmed that *MWE* captures different aspects as compared to other cohesion metrics. Given our interpretation of cohesion, we conjecture that combining *MWE* with other cohesion measures should result in a more complete cohesion indicator. One use of cohesion metrics in software engineering is to predict faults in classes [17, 23, 34, 39]. The focus of the second research question is to analyze the extent to which each of the cohesion measures used in the case study can be used to predict faults and to compare combinations of *MWE* and other metrics for identifying fault-prone classes.

The second case study aimed at answering **RQ2** is performed in a similar fashion as in previous work [23, 34]. We used Bugzilla[6] to collect bugs between two versions of Mozilla (that is, 1.6 and 1.7), and correlated each bug with specific classes. Details on how we mined the bugs can be found in previous work [23]. This data set has been used in prior studies [34], which allows direct comparison of the results.

4.4.1. Analyses

We employed regression analysis methods to discover the possible relationships between values of collected metrics and the fault proneness of those classes. These methods have been widely used to study the relationships between the metrics and fault or change proneness of classes [2, 5, 8, 10, 23, 34, 41]. In order to analyze our data, we employed *univariate* and *multivariate logistic regression* analysis methods for predicting if a class is faulty or not.

[6] http://bugzilla.mozilla.org/ (accessed and verified on 02/18/09)

Table 6. Results of the Univariate Logistic Regression (sorted by R^2)

Metric	Prec	Prec Rank	Corr	Corr Rank	Compl	Compl Rank	R^2	C_0	C_1
LOC	64.26	1	71.74	5	65.96	5	0.131	-0.740	0.002
$LCOM_1$	61.99	5	74.55	3	60.46	8	0.109	-0.423	0.001
$LCOM_3$	62.72	2	70.30	6	64.01	7	0.107	-0.724	0.061
$LCOM_2$	62.19	3	76.19	2	58.59	9	0.106	-0.393	0.001
$LCOM_4$	59.86	9	65.99	7	54.72	12	0.079	-0.621	0.073
C3	62.14	4	61.44	8	71.73	4	0.075	2.230	-4.155
ICH	60.88	8	73.25	4	52.99	13	0.069	-0.324	0.008
$LCOM_n$	61.56	7	78.99	1	55.08	11	0.06	-0.267	0.001
Coh	61.79	6	60.17	9	78.18	2	0.034	0.350	-1.982
MWE	*57.21*	*11*	*55.47*	*10*	*65.67*	*6*	*0.03*	*0.603*	*-3.033*
LCSM	56.29	12	53.11	12	91.69	1	0.024	0.082	-0.191
TCC	51.98	13	50.44	13	56.51	10	0.01	0.126	-0.797
$LCOM_5$	57.30	10	55.28	11	75.21	3	0.007	-0.402	0.488
LCC	50.68	14	48.72	14	32.74	14	0.002	0.029	-0.291

Since logistic regression is a commonly used statistical method, we are not providing any details. For more information on regression analyses and their applications on using metrics to detect fault prone classes, the reader is referred to the literature [5, 10, 23, 41]. We applied logistic regression analyses using the same settings as in our earlier work [34].

In the second case study, the univariate regression analysis is used to analyze the effect of each metric separately, whereas multivariate regression is used to analyze the effect of the combination of metrics on the final results to see whether combinations of *MWE* with other cohesion metrics can improve detecting fault-prone classes as compared to other combinations. For the logistic multivariate analysis, we build models for predicting faults in classes based on all possible combinations of pairs of cohesion metrics used in this case study (that is, 91 different pairs of metrics for 14 unique cohesion metrics). The parameters (constant C_0 and coefficients C_1 and C_2) for the instantiated models are provided in Table 6 and Table 7.

4.4.2. Results

First, we performed univariate logistic regression (see the results in Table 6). The R^2 coefficient is defined as the proportion of the total variation in the dependant variable (i.e., the fault proneness of a class) that is explained by the regression model. The bigger the value of R^2, the larger the portion of the total variance in the dependent variable that is explained by the regression model and the better the dependent variable is explained by the explanatory variables.

In order to evaluate logistic regression models based on the studied metrics and their combinations, we utilize the following quantitative characteristics: *precision*, *correctness*, and *completeness*. We use these measures to be consistent with previously published results [23, 34]. Precision is used to evaluate how well the model classifies classes as faulty or non-faulty. Correctness is used to capture the percentage of the faulty predicted classes that are really faulty. Completeness is the other characteristic used to evaluate the percentage of the total number of faults that can be captured by the model. For definitions and examples for computing precision, correctness and completeness refer to our previous work [34].

The results of the univariate logistic regression (see Table 6) allow us to draw the conclusions that, if we use each of the 14 metrics as separate indicators of fault proneness, *MWE* has the 11th largest precision, 10th largest correctness, 6th largest completeness and 10th largest R^2 value. These results are not surprising as *MWE* captures only certain aspects of cohesion, whereas faults may be caused by other issues affecting cohesion which are not captured by *MWE* alone.

Our assumption is that *MWE* complements existing metrics, so, in order to investigate whether combining *MWE* with other cohesion measures can improve the detection of fault-prone classes, we applied multivariate logistic regression analysis (see Table 7). We built 91 models based on all combinations of the pairs of cohesion metrics. Table 7 presents the top ten models based on the largest R^2 values. Based on the results, the model *MWE* combined with LOC appears to be in the first position.

The model, which is based on the combination of *MWE* and LOC has the highest precision (i.e., 67.31%) and R^2 values (i.e., 0.166) among all the 91 possible pairs of metrics. While the Table 7 present only the top 10 pairs of metrics according to the R^2 values, the combinations of *MWE* with $LCOM_1$, $LCOM_2$ and $LCOM_3$ are ranked in positions 19, 20 and 21 according to the R^2 values.

Table 7. Results of the multivariate logistic regression for the top ten pairs (out of 91) of cohesion metrics with the largest R^2 values (sorted by R^2 Values)

Model	Prec	Prec Rank	Corr	Corr Rank	Compl	Compl Rank	R^2	C_0	C_1	C_2
MWE+LOC	*67.31*	*1*	*73.02*	*30*	*73.97*	*18*	*0.166*	*0.063*	*-3.884*	*0.002*
LOC+$LCOM_4$	66.68	2	74.19	20	72.70	22	0.165	-1.049	0.002	0.048
C3+LOC	66.68	3	69.79	50	74.98	12	0.164	1.054	-3.106	0.002
C3+$LCOM_3$	66.19	5	68.32	55	75.77	10	0.161	1.397	-3.831	0.061
LOC+$LCOM_3$	66.39	4	74.40	16	71.63	25	0.157	-0.968	0.001	0.035
C3+$LCOM_1$	65.38	10	68.19	56	73.79	20	0.154	1.552	-3.520	0.001
C3+ $LCOM_2$	64.65	16	67.09	59	72.44	23	0.152	1.585	-3.532	0.001
LOC+ $LCOM_2$	66.15	6	75.00	10	70.46	28	0.148	-0.753	0.001	0.001
LOC+$LCOM_1$	66.10	7	74.55	14	70.59	27	0.145	-0.747	0.001	0.001

Table 8 provides the comparison of the univariate and the multivariate models, in which one variable is *MWE*. If the multivariate model has lower values for precision, correctness or completeness as compared to the univariate model, we mark that case in boldface (see Table 8).

Table 8. Comparison between univariate and multivariate models with *MWE* as a variable

Metric	Prec.	Prec MWE	Corr.	Corr MWE	Compl	Compl MWE
C3	62.13	63.05	61.44	61.87	71.72	73.94
LCSM	56.28	59.62	53.10	57.43	91.69	**72.80**
LOC	64.26	67.31	71.73	73.02	65.96	73.97
$LCOM_2$	62.18	63.58	76.19	**71.35**	58.59	65.01
$LCOM_n$	61.55	61.70	78.99	**63.53**	55.08	68.69
$LCOM_1$	61.99	64.65	74.54	**72.24**	60.45	67.81
$LCOM_3$	62.71	63.49	70.29	**67.97**	64.00	68.53
$LCOM_4$	59.86	61.12	65.98	**63.52**	54.72	62.86
$LCOM_5$	57.30	57.64	55.27	55.88	75.21	**67.03**
ICH	60.88	62.57	73.25	**63.83**	52.99	72.01
TCC	51.98	57.15	50.43	55.39	56.51	67.16
LCC	50.67	56.91	48.71	55.17	32.73	66.05
Coh	61.79	**61.60**	60.17	**59.71**	78.17	**74.62**

While comparing precision values for univariate and multivariate models, we can see that only Coh has reduced values. As for the correctness values, we observe that almost half of the multivariate models have improved correctness. Most of the completeness values have been improved as well. Since all of the R^2 values have been improved, we do not list them in Table 8.

Overall, the results of both case studies indicate that *MWE* is a useful indicator of an external property of classes in OO systems, that is, the fault proneness of classes. Based on the results of the regression analyses, we can conclude that *MWE* is a valuable complement in a number of combinations with other cohesion metrics, especially with LOC. More importantly, the results support our assumption that the combination of *MWE* with other cohesion metrics allows us to build superior models for detecting fault prone classes based on cohesion metrics.

4.5. Threats to validity

Several issues affect the results of the case studies and limit our interpretations and generalizations of the results. The first case study showed that our metric captures new dimensions in cohesion measurement; however, we obtained these results by analyzing classes from only one open source application written mainly in C++, even though *Mozilla* represents a real-life application. In order to generalize the results, a large-scale study is required which should take into account software systems from different domains written in different programming languages and of varying class sizes. One issue that may affect internal validity of the second case study is that cohesion is not the only factor affecting the fault proneness of classes. To build complete models for fault prediction, other factors would have to be considered (e.g., other metrics, such as coupling). However, this is out of the scope of this paper as the purpose of analyzing fault proneness of classes was to see if the combination of *MWE* with other metrics conveys any improvements.

5. Conclusions

Classes in object-oriented systems, written in different programming languages, contain identifiers and comments which reflect concepts from the domain of the software system. This information can be used to measure the cohesion of software. Latent Dirichlet Allocation is used to extract latent topics used for cohesion measurement and then information entropy is used to measure the degree of the distribution of the topics. This paper defines the *MWE* cohesion metric, which captures new, yet complementary dimensions of cohesion as compared to a host of existing metrics. Principal component analysis of measurement results on an open source software system statistically supports this fact. The combination of LOC with *MWE* appears to be the best model (in terms of R^2 and precision) for the prediction of faults in classes among the other 91 possible combinations of metrics. Moreover, *MWE* improves fault prediction for most existing metrics if combined with *MWE*.

6. Acknowledgements

We acknowledge Andrian Marcus for contributions to previous versions of the research on conceptual cohesion of classes (C3). We would like to thank Zheng “Eddy” Zhang for her help with LDA tool, Tibor Bakota for verifying the statistical results and comments. This research was supported in part by the United States Air Force Office of Scientific Research under grant number FA9550-07-1-0030, Hungarian national grants RET-07/2005, OTKA K-73688, TECH_08-A2/2-2008-0089 (SZOMIN08) and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

7. References

[1] Allen, E. B., Khoshgoftaar, T. M., and Chen, Y., "Measuring coupling and cohesion of software modules: an information-theory approach", in Proc. of METRICS'01, April 4-6 2001, pp. 124-134.

[2] Arisholm, E., Briand, L. C., and Foyen, A., "Dynamic coupling measurement for OO software", *IEEE TSE*, Aug'04, pp. 491-506.

[3] Baldi, P., Linstead, E., Lopes, C., and Bajracharya, S., "A Theory of Aspects as Latent Topics", in Proc. of OOPSLA'08, pp. 543-562.

[4] Bansiya, J. and Davis, C. G., "A hierarchical model for object-oriented design quality assessment", *IEEE TSE*, Jan'02, pp. 4-17.

[5] Basili, V. R., Briand, L. C., and Melo, W. L., "A Validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE TSE*, vol. 22, no. 10, October 1996, pp. 751-761.

[6] Bieman, J. and Kang, B.-K., "Cohesion and reuse in an object-oriented system", in Proc. of SSR'95, April 1995, pp. 259-262.

[7] Blei, D. M., Ng, A. Y., and Jordan, M. I., "Latent Dirichlet Allocation", *Journal of Machine Learning Research*, vol. 3, 2003.

[8] Briand, L., Melo, W., and Wust, J., "Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects", *IEEE TSE*, vol. 28, no. 7, 2002, pp. 706-720.

[9] Briand, L. C., Daly, J. W., and Wüst, J., "A Unified Framework for Cohesion Measurement in OO Systems", *ESE'98*, 3/1, pp. 65-117.

[10] Briand, L. C., Wüst, J., Daly, J. W., and Porter, V. D., "Exploring the relationship between design measures and software quality in object-oriented systems", *JSS*, 51/3, May'00, pp. 245-273.

[11] Chae, H. S., Kwon, Y. R., and Bae, D. H., "Improving Cohesion Metrics for Classes by Considering Dependent Instance Variables", *IEEE TSE*, vol. 30, no. 11, November 2004, pp. 826-832.

[12] Chidamber, S., Darcy, D., and Kemerer, C., "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis", *IEEE TSE*, vol. 24, no. 8, August 1998, pp. 629-639.

[13] Chidamber, S. R. and Kemerer, C. F., "A Metrics Suite for OO Design", *IEEE TSE*, vol. 20, no. 6, 1994, pp. 476-493.

[14] De Lucia, A., Oliveto, R., and Vorraro, L., "Using structural and semantic metrics to improve class cohesion", in ICSM'08, pp 27-36..

[15] Dempster, A., Laird, N., and Rubin, D., "Likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society*, vol. 39, no. 1, 1977, pp. 1-38.

[16] Eder, J., Kappel, G., and Schreft, M., "Coupling and Cohesion in Object-Oriented Systems", Univ. of Klagenfurt, Tech. Report 1994.

[17] El-Emam, K. and Melo, K., "The Prediction of Faulty Classes Using OO Design Metrics", *NRC/ERB-1064*, NRC 43609, Nov'99.

[18] Etzkorn, L. H. and Davis, C. G., "Automatically Identifying Reusable OO Legacy Code", *IEEE Computer*, Oct'97, pp. 66-72.

[19] Etzkorn, L. H., Gholston, S., and Hughes, W. E., "A Semantic Entropy Metric", *JSME*, vol. 14, no. 5, July/Aug. 2002, pp. 293-310.

[20] Ferenc, R., Beszédes, Á., Tarkiainen, M., and Gyimóthy, T., "Columbus - Reverse Engineering Tool and Schema for C++", in Proc. of ICSM'02, Montréal, Canada, October 3-6 2002, pp. 172-181.

[21] Foltz, P. W., Kintsch, W., and Landauer, T. K., "The Measurement of Textual Coherence with Latent Semantic Analysis", *Discourse Processes*, vol. 25, no. 2, 1998, pp. 285-307.

[22] Griffiths, T. and Steyvers, M., "Finding scientific topics", *Proc. of the National Academy of Sciences* 2004.

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

[24] Harrison, W., "An Entropy-Based Measure of Software Complexity", *IEEE TSE*, vol. 18, no. 11, Nov'92, pp. 1025 - 1029.

[25] Henderson-Sellers, B., *Software Metrics*, Prentice Hall, 1996.

[26] Hitz, M. and Montazeri, B., "Measuring Coupling and Cohesion in Object-Oriented Systems", in Proc. of International Symposium on Applied Corporate Computing, Monterrey, Mexico, October 1995.

[27] Kramer, S. and Kaindl, H., "Coupling and cohesion metrics for knowledge-based systems using frames and rules", *ACM TOSEM*, vol. 13, no. 3, July 2004, pp. 332-358.

[28] Lafferty, J. and Minka, T., "Expectation-propagation for the generative aspect model", in Proc. of UAI'02.

[29] Lee, Y. S., Liang, B. S., Wu, S. F., and Wang, F. J., "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow", in Proc. of ICSQ'95, Maribor, Slovenia, 1995.

[30] Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., and Baldi, P., "Mining concepts from code with probabilistic topic models", in Proc. of ASE'07, Atlanta, Georgia, 2007, pp. 461-464.

[31] Lukins, S., Kraft, N., and Etzkorn, L., "Source Code Retrieval for Bug Location Using Latent Dirichlet Allocation", in Proc. of WCRE'08, Antwerp, Belgium, 2008, pp. 155-164.

[32] Maletic, J. I. and Marcus, A., "Supporting Program Comprehension Using Semantic and Structural Information", in Proc. of ICSE'01, Toronto, Ontario, Canada, May 12-19 2001, pp. 103-112.

[33] Marcus, A. and Poshyvanyk, D., "The Conceptual Cohesion of Classes", in Proc. of ICSM'05, Hungary, Sept. 2005, pp. 133-142.

[34] Marcus, A., Poshyvanyk, D., and Ferenc, R., "Using the Conceptual Cohesion of Classes for Fault Prediction in Object Oriented Systems", *IEEE TSE*, vol. 34, no. 2, 2008, pp. 287-300.

[35] Maskeri, G., Sarkar, S., and Heafield, K., "Mining Business Topics in Source Code using LDA", in ISEC'08, India, pp. 113-120.

[36] Menzies, T., Greenwald, J., and Frank, A., "Data Mining Static Code Attributes to Learn Defect Predictors", *TSE'07*, 30/1, pp. 2-13.

[37] Meyers, T. M. and Binkley, D., "An Empirical Study of Slice-based Cohesion and Coupling Metrics", *ACM TOSEM,* 17/1, 2007.

[38] Poshyvanyk, D. and Marcus, A., "The Conceptual Coupling Metrics for Object-Oriented Systems", in ICSM'06, pp. 469-478.

[39] Quah, T.-S. and Thwin, M. M. T., "Application of neural networks for software quality prediction using object-oriented metrics", in Proc. of ICSM'2003, September 2003, pp. 116-125.

[40] Shannon, C. E., "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, July 1948, pp. 379–423

[41] Subramanyam, R. and Krishnan, M. S., "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects", *IEEE TSE*, 29/4, April 2003, pp. 297-310.

[42] Yin, R. K., *Applications of Case Study Research*, 2003.

[43] Zhao, J. and Xu, B., "Measuring Aspect Cohesion", in FASE'04.

[44] Zhou, Y., Xu, B., Zhao, J., and Yang, H., "ICBMC: an improved cohesion measure for classes", in ICSM'02, pp. 44-53.