ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

Measuring Class Cohesion using Topic Modeling

Rohinee Deshmukh [1], Manjiri Karande [2], Ankush Narkhede [3]

P.G. Student, Department of Computer Engineering, VBKCOE, Malkapur, Maharashtra, India[1]

Assistant Professor, Department of Computer Engineering, VBKCOE, Malkapur, Maharashtra, India[2]

Assistant Professor, Department of Computer Engineering, VBKCOE, Malkapur, Maharashtra, India[3]

ABSTRACT: High cohesion for classes is one of the most desire properties in Object Oriented (OO) analysis as it supports program comprehension, testing, maintainability, reusability. Cohesion metrics have been used for quality valuation, software modularization, and fault prediction and so on. Existing approaches of class cohesion metrics are largely based on the structural information of the source code, like attribute references in class methods. Only looking at particular aspect of cohesion is not sufficient for completely and accurately specifying class cohesion and there is a need to other aspects of cohesion like conceptual aspect. The two conceptual cohesion metrics ACSM+ and LCSM2# use more efficient document modeling technique, known as LDA, used for defining similarity between methods. Maximum Weighted Entropy (MWE) has two issues, first does not take into account prominent object oriented concepts like inheritance and second MWE does not make difference between access methods and other method stereotypes which can artificially increase or decrease cohesion. Extended the MWE by considering the concept of inheritance and the effect of getter and setter methods and called it MWE+. Two case studies on an open source java software system called Rhino are performed. First case study specifies that the new cohesion metrics capture different aspects of class cohesion compared to the exiting cohesion metrics. Second case study specifies new dimensions that MWE+ adds to MWE.

KEYWORDS: Metric, Class Cohesion, Topic Modeling, LDA, MWE.

I. INTRODUCTION

A measurement of the degree to which elements of a module belong together is known as software cohesion. One of the goals of the Object Oriented (OO) analysis and design as it facilitates program comprehension, testing, maintainability, reusability and so on. Software cohesion metrics have to support different software maintenance tasks.

A. *MEASURING CLASS COHESION*

The comparative number of joined methods of class is known as class cohesion .A subclass inherits methods and instance variables from its super class. Two different forms of reusing a class, reuse via instantiation and reuse via inheritance (both direct and indirect subclass).

i. Class Cohesion from Structural Point of View

Cohesion is measured on structural information extract completely from the source code such as attribute references in methods and method calls of a class. In structural point of view the extracted information captures the scale to which the elements of a class collectively. These cohesion metrics give information about the technique of building a class and how its instances work together to address the goals behind their design. Among all structural cohesion metrics, most widely used cohesion metrics are LCOM1-5 (Lack of cohesion of methods). All LCOM metrics defined cohesion of a class as degree of similarity of its methods that is modeled by sharing of references of attributes among methods of a class.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

ii. Class Cohesion from Conceptual Point of View

Apart from structural point of view, cohesion is also considered from a conceptual point of view. In this view, a cohesive module is a crisp abstraction of a concept or feature from problem domain that express in the specifications or requirements. Class is cohesive from a conceptual point of view whether a class implements one or more domain conception. Although other types of metrics proposed to capture different aspects of cohesion, there is a few such metrics like C3, LCSM, LORM and MWE address these conceptual and textual aspects of cohesion [1, 2].

B. *CONCEPTUAL COHESION METRICS*

The LCSM along with ACSM and C3 metrics similarity between methods of a class is defined Information retrieval using Vector Space Model technique. Vector space model is a very old technique for modeling text copra that means collection of documents and it has several limitations like it provides a relatively small amount of deduction in the collection and reveals little in way of inter- or intra documents statistical structure. These shortcomings were overcome by other dimensionality reduction techniques like Latent semantic indexing (LSI), probability LSI (PLSI). The latest technique in that series is Latent Drichlet Allocation (LDA).

i. LDA

Latent Dirichlet allocation (LDA) is a probabilistic generative model, in which everything in collection is modeled as a finite mixture on to underlying set of topics [5]. Every topic is, modeled as an unlimited mixture on to underlying topic probabilities set. In topic modeling, a topic is often co-occurring words collection in the corpus. Given a corpus of n documents m1,... ,mn, topic modeling techniques automatically discover a set Z of topics, Z = {z1, ..., zK}, as well as the mapping θ between topics and documents as in [1].

ii. INFORMATION ENTROPY

The 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 is a measure of the uncertainty associated with a random variable [20]. Information entropy has been previously used to measure software complexity and cohesion [1]. For a random variable x with n outcomes {xi : i=1,…, 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. In this case, p(mj) is the probability that topic t_i is assigned to method mj which is computed using LDA. A class may relate to a mixture of topics with varying probabilities, so must take into account all the topics and relationships among them to properly capture software cohesion.

iii. MWE

A conceptual cohesion metric called Maximum Weighted Entropy (MWE) was proposed by Y. Liu et. al. [1] based on the analysis of latent topics encoded in source code, expressed in identifiers and comments. It also uses the Latent Dirichlet Allocation (LDA) technique. MWE 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. Quantify characteristics like occupancy and distribution for every class in the software system under analysis to measure class cohesion. Define metric MWE for class cohesion as the following:

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

Where, $O(t_i)$ is a measure of occupancy of topic ti in methods in a classC_z; $D(t_i)$ is a measure of degree of distribution of topict_i in methods in a classC_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. The occupancy of topic t_i in methods of a class C_z is computed as:

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

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

Where n is the number of methods in a class and P_{ti}^{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 order to compute entropy for a topic t_i need to transform document-topic probability distributions $p_{t_i}$to topic-document distributions q_{t_i} as the following:

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

Once topic-document distributions 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:

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

At last the MWE cohesion metric is computed as:

$$MWE(C_z) = \max_{1\le i\le |t|}\left(O(t_i) * D(t_i)\right) = \max_{1\le i\le |t|}\left(\frac{\sum_{d=1}^{n} P_{t_i}^{d}}{n} * \frac{\sum_{d=1}^{n} -q_{t_i}^{d} * \log q_{t_i}^{d}}{\log n}\right) \qquad (6)$$

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. MWE 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). However, MWE also has some of the important issues described as:

- MWE does not consider prominent object oriented concepts like inheritance.
- MWE do not make distinction between access methods and other method stereotypes. Some of these methods can artificially increase or decrease cohesion. So effect of access methods like getter and setter methods should take into account while calculating MWE.

In this work, proposed a set of cohesion metrics as LCSM+ (Lack of Conceptual Similarity between Methods). Among proposed set of cohesion metrics, LCSM+ 1-5, each of them is conceptual version of corresponding LCOM metric[6]. The basic idea of specifying similarity of methods of a class in our proposed metrics is number of topics those are common in methods of a class. To approximate these topics, use a very popular topic modeling technique known as LDA (Latent Drichlet Allocation) [5]. In this work, also extended the work of Marcus and Poshyvanyk [2] and proposed two additional conceptual cohesion metrics as ACSM+ and LCSM2# which use a new technique of document modeling, LDA [5] to define similarity between methods, instead of vector space model.

In this work, I also extended the MWE (called it MWE+) by considering the concept of inheritance and the effect of getter and setter methods. Proposed MWE+ metric consider all those methods which a class implements or overloads or inherits from a super class. Here excluded getter and setter methods in MWE+ calculation. Also conduct a case study that compares the results of MWE and MWE+ and shows the significance of considering effect of inheritance and getter-setter methods. From implementation point of view, all of the available conceptual metrics have been tested on only C/C++ open source software. In order to generalize their results, a large scale study is necessary which should take into account software systems from dissimilar domains written in different programming languages and varying class sizes. To fulfil this purpose, I apply, analyze and test this proposed metrics on Rhino an open source implementation of JavaScript that written in Java [24].

II. RELATED WORK

There are numerous different approaches to measure cohesion in Object Oriented systems. Based on the underlying mechanisms used to measure the cohesion of a class one can distinguish: structural metrics [4, 6, 9, 14, 15, 23], the most popular class of cohesion metrics, semantic or conceptual metrics [1, 2, 10, 18], metrics based on data mining and metrics for specific types of applications, information entropy-based metrics [1] etc. In object oriented systems some methods are used for topic modeling [4, 5, 12, 19].

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

A. *STRUCTURAL COHESION METRICS*

Chidamber and Kemerer developed a set of software metrics for OO design these are based in measurement theory and reflect viewpoints of OO software developers. They develop a LCOM metric which measures the attributes of an object class. LCOM1-2 is proposed by Chidamber and Kemerer [9]. Hitz and Montazeri extended the work of Chidamber and Kemerer and proposed LCOM3, LCOM4 and CO (connectivity) cohesion metrics [15]. For improving a class cohesion they define Lack of Cohesion in Methods (LCOM) is defined as the number of connected components of graph GX ($1 \leq LCOM(X) \leq |MX|$) where X denotes class and Mx denotes methods of a class. In addition to this mere formal improvement of the definition of LCOM, they would like to get rid of more semantic flaws in the definition of LCOM. LCOM5 was proposed by Bieman and Kang which was further extension of Chidamber and Kemerer [14] work

Chae et. al., invented an approach for improving the cohesion metrics by considering the characteristics of the dependent instance variables in an object-oriented program [8]. Zhou analyzes the limitations of Chae typical cohesion measures for classes in detail, and then proposes an improved cohesion measure ICBMC. This used to be a guideline about quality evaluation and restructure badly designed classes. Therefore, the ICBMC is a well-defined class cohesion metric, which overcomes the limitations of CBMC and extends its application [23].

B. *CONCEPTUAL OR SEMANTIC COHESION METRICS*

Modeling class cohesion as mixtures of latent topics by Y. Liu, D. Poshyvanyk proposed the Maximal Weighted Entropy which uses the latent Drichlet Allocation and information entropy for measuring the cohesion of class in software system. It is based on analysis of latent topics in source code such as comments and identifiers [1].

De Lucia et. al. developed the metric that measures structural information of class but they consider the graph based representation and class refactoring extraction. They developed the extended structural LCOM metric which shows the structural similarity between methods [10]. Marcus, A. and Poshyvanyk, D., propose a new measure named the Conceptual Cohesion of Classes (C3) for the cohesion of classes in OO software systems based on the analysis of the unstructured information embedded in the source code, such as comments and identifiers. They consider the open source software Mozilla and WinMerge for case study that defines the C3 captures new and complementary dimensions of cohesion compared to a host of existing structural metrics. To extract the information that contains identifiers and comments for cohesion measurement they use Latent Semantic Indexing [18].

III. PROPOSED SYSTEM

The proposed new class cohesion metrics are: LCSM+ metrics set, CCo, ACSM+, LCSM2# and MWE+. All LCSM+ Metrics use number of topics those are common among methods of a class to define the similarity between methods of a class. These topics are identified by LDA, a topic modeling technique that approximates topics among a collection of documents. Same counting mechanism as used by the corresponding LCOM metric is used by LCSM+. Just as LCOM metrics, LCSM+ metrics are inverse cohesion metric of cohesion, which means that a higher value for LCSM+ metrics indicates lower cohesion. Proposed ACSM+ and LCSM2# are enhanced form of ACSM and LCSM metric proposed by Marcus and Poshyvanyk [2]. These metrics use a new technique of document modeling, LDA [5] instead of vector space model that used by Marcus and Poshyvanyk for their metrics, to define similarity between methods. Extended the MWE (called it MWE+) by considering the concept of inheritance and the effect of getter and setter methods. Proposed MWE+ metric consider all those methods which a class implements or overloads or inherits from a super class. Here excluded getter and setter methods in MWE+ calculation.

A. *PROPOSED MAXIMUM WEIGHTED ENTROPY METRIC (MWE+)*

Proposed Maximum Weighted Entropy (MWE+) is extended MWE. It also considers the concept of inheritance and the effect of getter and setter methods. Proposed MWE+ metric consider all those methods which a class implements or overloaded or inherited from a super class. I excluded getter and setter methods in MWE+ calculation.

To obtain proposed MWE+ metric, I slightly modify the approach used by MWE as follow:

- While calculating MWE+ for a class not only consider documents of methods implemented or overloaded by the class but also documents of methods of direct and indirect super classes of that class. As per java language

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

rules, all public and protected methods of super class are available in child class, so include documents of all public and protected methods of all direct and indirect super class but private methods of super class is not available in child class so does not include documents of all private methods of all direct and indirect super class.

- MWE do not make distinction between access methods and other method stereotypes. Some of these methods can artificially increase or decrease cohesion. An access method provides read or write access to an attribute of the class. Access methods typically reference only one attribute, namely the one they provide access to. So access methods have only one topic to which that attribute belongs. While calculating MWE, access methods affect the value of both occupancy and distribution of a topic as described follows:
  i) From equation 3, that is used to calculate occupancy of each topic, see that occupancy of a topic is simple mean of topic's membership value for all methods. So occupancy depends on number of methods in the class. So inclusion of access method unnecessary affects occupancy of the topics.
  ii) Distribution of a topic depends on sum of measure of uncertainty of that topic to appear in each method of class. Inclusion of an access method affect this sum because this method has only one topic (i.e. that means no uncertainty), hence low value of information entropy that employs low contribution to sum value. But Inclusion of access method increase numerator log n value. Consequently it reduces distribution value of topics. So while calculating MWE+ exclude getter and setter methods.

So in MWE metric calculation, inclusion of access methods affect the value of occupancy and distribution of a topic consequently affect the final vale of MWE. Thus exclude access method in MWE+ metrics calculation.

Proposed Algorithm of MWE+

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 Ci (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 Ci
4. Find all getter and setter methods of class Ci. Let no. of such methods is x.
5. Exclude these x documents from m documents and Extract a sub matrix Doc-Top(y, N) from Doc-Top (m, N) where y is corresponds to all documents in class Ci except x documents.
6. Find methods of all direct and indirect super class of Ci.Let no. of such methods is z.
7. Include these z documents to y documents and form matrix Doc-Top (m', N) from Doc-Top(y, N) where m' is corresponds to all y documents in class Ci plus z documents.
8. Calculate topic-document distribution based on Doc-Top (m', N) using equation (4)
9. For each topic tj
10. Calculate occupancy O(tj) using equation (3) Based on sub-matrix Doc-Top(m', N);
11. Calculate distribution D(tj) using Equation (5) based on Doc-Top(m', N);
12. Save O(tj)* D(tji) to WE(j)
13. End For
14. Find max MWE+ for class Ci in MWE using equation (6)
15. End For

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

B. *SYSTEM FRAMEWORK*

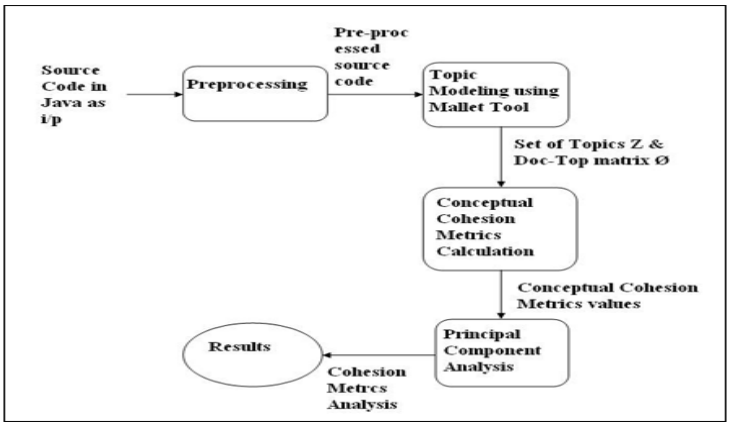

Fig. 1 Block diagram of system

Figure 1 shows the system block diagram. In this give the target Rhino software system as input to pre-processing module. Pre-processing module consist of four steps: fact extraction, identifier splitting, stemming and filtering. In fact extraction step, extract methods from each class and create one document for each method. In identifier splitting, split all identifiers and comments into meaningful sub words. In stemming step, stem each words of splitting step into their common root words. In filtering step, filter all those words which do not indicate any business concept.

After pre-processing of input software system, get that a corpus of text documents one for each method. Each document contains list of pre-processed words of that particular methods. This corpus is given input to Mallet and set parameter as shown in table 1, an LDA tool. Mallet group related words into topics and produces two things: set of topics (Z) and topic –documents membership Matrix (Ø). Using Z and Ø, calculate LCSM1+, LCSM2+, LCSM3+, LCSM4+, LCSM5+, ACSM+, LCSM2#, MWE and MWE+ metrics. Also calculate structural cohesion metrics such as LCOM1, LCOM2, LCOM3 and LCOM4 for Rhino. For this purpose used Simple code Metric plugin of Netbeans [25]. After that Principal Component Analysis (PCA) performs on these metrics in order to understand the underlying, orthogonal dimensions captured by the coupling measures. PCA results shows that these conceptual cohesion metrics capture different dimension than existing cohesion metrics. Also analyze the results of MWE and MWE+ metrics in order to indicate the significance of MWE+ over MWE.

Table 1: Parameter settings for Mallet

Parameters	Value
Alpha (α)	0.05
Beta (β)	0.01
No of Topics	100
No of Documents	347
No of iterations	1000
Threshold (δ)	1%

IV. EXPERIMENTAL RESULTS

In order to understand the underlying orthogonal dimensions captured by proposed and considered existing cohesion metrics, perform Principal Component Analysis (PCA) on all these metrics values measured for the Rhino software systems by using same settings as in prior studies [1, 18].

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

A. ANALYSIS PROCEDURE

Table 2: Descriptive statistics for all proposed and subjected existing cohesion metrics values for Rhino software system

	N	Minimum	Maximum	Mean	Std. Deviation	Variance
LCSM1+	281	0	4197	125.55	417.743	174509.034
ACSM+	281	0E-12	.500000000000	.05403755718801	.076890750058764	.006
LCSM2+	281	0	3038	80.66	290.568	84429.632
LCSM2#	281	0	456	20.94	56.685	3213.221
LCSM3+	281	1	13	3.01	2.029	4.118
LCSM4+	281	1	9	2.05	1.357	1.840
LCSM5+	281	0E-12	1.000000000000	.83157902173390	.260304271964128	.068
LCOM1	281	0	2769	65.85	252.127	63568.192
LCOM2	281	0	1	.60	.490	.240
LCOM3	281	0	2	.65	.493	.243
LCOM4	281	1	52	5.38	6.538	42.743
MWE	281	0E-12	.960051352728	.25989847688498	.229957568738074	.053
MWE+	281	0E-12	1.014695268018	.24651497872151	.211174743290976	.045
Valid N (list wise)	281					

Briand et al. [7] proposed a methodology to analyze software engineering data in order to make an experiment repeatable and the results comparable. The methodology consists of the following three steps: collecting the data, identifying outliers, and performing PCA.

i. Collecting data

Earlier calculate all proposed and considered existing cohesion metrics and collected their resulted data for all classes of subjected Rhino software system. The descriptive statistics of collected data are shown in table 2 which obtained using IBM SPSS tool [29]. From that descriptive statistics we get all the values like maximum, minimum, mean, standard deviation and variances for each cohesion metrics. So this is easy way to analyze that metrics. This statistics used on collecting resulted data.

ii. Identifying and Eliminating Outliers

As results analysis impacted by the outliers, they were removed. For identify outliers in the data, utilized the T2max procedure based on the Mahalanobis distance [26]. For performing outlier analysis using IBM SPSS tool [29].

iii. Performing PCA

After outliers were eliminated, performed PCA using Tanagra software tool [28], which was used in our case to identify groups of variables (i.e., metrics), which are likely to measure the same underlying dimension (i.e., mechanism that defines cohesion) of the object to be measured (i.e., cohesion of a class). The resulting rotated components show clearer patterns of loading for the variables [27].

B. *PCA RESULTS*

The PCA performed on the set of 308 classes of Rhino software systems. All 13 cohesion metrics were subjected to an orthogonal rotation. I identified seven orthogonal dimensions spanned by 13 cohesion measures. The seven principal components (PCs) capture 93% of the variance in the data set, which is significant enough to support our findings.

Table 3: Results of PCA performed on all proposed and subjected existing cohesion metrics values for Rhino software system

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

Attribute	PC_1		PC_2		PC_3		PC_4		PC5_5		PC_6		PC_7	
-	Corr.	% (Tot.%)	Corr.	% (Tot.%)	Corr.	% (Tot.%)	Corr.	% (Tot.%)	Corr.	% (Tot.%)	Corr.	% (Tot.%)	Corr.	% (Tot.%)
LCSM2+	0.95263	91 % (91 %)	0.05612	0 % (91 %)	0.13546	2 % (93%)	0.03617	0 % (93%)	0.02486	0 % (93%)	0.01688	0 % (93%)	0.01762	0 % (93%)
LCSM1+	0.95019	90 % (90 %)	0.04798	0 % (91 %)	0.09325	1 % (91%)	0.04548	0 % (92%)	0.04052	0 % (92%)	-0.01508	0 % (92%)	0.03409	0 % (92%)
LCOM1	0.93205	87 % (87 %)	0.05072	0 % (87 %)	-0.01355	0 % (87%)	0.06147	0 % (88%)	-0.00694	0 % (88%)	0.08025	1 % (88%)	-0.00015	0 % (88%)
LCOM4	0.84262	71 % (71 %)	0.04864	0 % (71 %)	0.03278	0 % (71%)	0.23087	5 % (77%)	0.12273	2 % (78%)	0.08610	1 % (79%)	0.11180	1 % (80%)
EMWE	-0.07564	1 %(1%)	-0.94717	90 % (90 %)	-0.02418	0 % (90%)	0.07757	1 % (91%)	0.03211	0 % (91%)	-0.08878	1 % (92%)	-0.14659	2 % (94%)
MWE	-0.07033	0 % (0%)	-0.93873	88 % (89 %)	-0.03954	0 % (89%)	0.06459	0 % (89%)	0.09105	1 % (90%)	-0.07484	1 % (91%)	-0.17093	3 % (93%)
LCSM2#	0.13518	2 % (2%)	0.04052	0 % (2%)	0.94370	89 % (91 %)	0.08566	1 % (92%)	0.02764	0 % (92%)	0.23121	5 % (97%)	0.03020	0 % (97%)
LCOM2	0.14555	2 % (2%)	-0.05087	0 % (2%)	0.07337	1 % (3%)	0.95219	91 % (94 %)	0.11694	1 % (95%)	0.09482	1 % (96%)	0.01138	0 % (96%)
LCOM3	0.11727	1 % (1 %)	-0.09299	1 % (2%)	0.04808	0 % (2%)	0.95092	90 % (93 %)	0.12434	2 % (94%)	0.10021	1 % (95%)	-0.01897	0 % (95%)
LCSM5+	0.10654	1 % (1%)	-0.12141	1 % (3%)	0.05693	0 % (3%)	0.23085	5 % (8%)	0.93318	87 % (95 %)	0.19794	4 % (99%)	0.03926	0 % (99 %)
LCSM4+	0.02486	0 % (0%)	0.11076	1 % (1 %)	0.11412	1 % (3%)	0.11551	1 % (4%)	0.09214	1 % (5%)	0.95507	91 % (96 %)	0.04248	0 % (96%)
LCSM3+	0.15060	2 % (2%)	0.11116	1 % (4%)	0.48405	23 % (27 %)	0.14179	2 % (29%)	0.24559	6 % (35%)	0.70047	49 % (84 %)	0.20542	4 % (88%)
ACSM	-0.11053	1 % (1%)	-0.42942	18 % (20 %)	-0.06641	0 % (20%)	0.01329	0 % (20%)	-0.04809	0 % (20%)	-0.15046	2 % (23%)	-0.87542	77 % (99%)
Var. Expl.	3.49985	27 % (27 %)	2.02537	16 % (43 %)	1.18369	9 % (52%)	1.97580	15 % (67 %)	0.99853	8 % (74%)	1.56498	12 % (87 %)	0.87799	7 % (93%)

The loadings on each measure in each rotated component in presented in Table 3. Values higher than 0.5 are highlighted as the corresponding measures are the ones look into while interpreting the PCs. For every PC, provide the variance of the data set explained by the PC and the cumulative variance in Table 3. Based on this analysis of the coefficients associated with every cohesion metrics within each of the rotated components, interpret PCs as the following.

PC1 (27%): LCSM1+, LCSM2+, LCOM1 and LCOM4. Proposed conceptual cohesion metrics LCSM1+ and LCSM2+ capture the same dimension captured by structural cohesion metrics LCOM1 and LCOM4.

PC2 (16%): MWE and MWE+. These metrics measure conceptual cohesion of a class based on the analysis of latent topics encoded in source code, expressed in identifiers and comments which modeled by using LDA, a topic modeling tool. These metrics use information entropy for measuring coverage and degree of latent topic distributions in methods of class. Proposed Maximum Weighted Entropy plus (MWE+) is extended form of MWE. Proposed MWE+ metric consider all those methods which a class implements or overloaded or inherited from a super class.

PC3 (9%): LCSM2#. This is proposed conceptual cohesion metrics. It is enhanced form of LCSM metric proposed by Marcus and Poshyvanyk [2]. LCSM2# metrics use a new technique of document modeling, LDA [5] instead of vector space model that used by existing LCSM to define similarity between methods and use same counting mechanism adopted by LCSM.

PC4 (15%): LCOM2 and LCOM3. These Structural metrics count the number of pairs of methods that share instance variables.

PC5 (8%): LCSM5+. This is proposed cohesion metric that is conceptual version of LCOM5.

PC6 (12%): LCSM3+ and LCSM4+. These are proposed cohesion metrics those are conceptual version of LCOM3 and LCOM4 respectively.

PC7 (7%): ACSM+. This is proposed conceptual cohesion metrics. It is modified form of ACSM metric proposed by Marcus and Poshyvanyk [2]. ACSM+ metrics use a new technique of document modeling, LDA [5] instead of vector space model that used by existing LCSM to define similarity between methods.

The PCA results indicate that proposed conceptual cohesion metrics defines a dimension of their own – LCSM3+ and LCSM4+ are the only major factor in PC6. LCSM2# is the only major factor in PC3. LCSM5+ is the only major factor in PC5. ACSM+ is the only major factor in PC7. Although proposed LCSM1+ and LCSM2+ metric capture the

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

same dimension captured by structural cohesion metrics LCOM1 and LCOM4, but rest of proposed conceptual metrics capture new dimensions. So PCA results statistically support hypothesis that the proposed LCSM3+ LCSM4+, ACSM+ and LCSM2# cohesion measure captures different aspects of class cohesion than those capture by all the metrics computed in case study.

C. *MWE V/S MWE+*

Table 4: Descriptive Statistics for MWE and MWE+

	N	Minimum	Maximum	Mean	Std. Deviation	Variance
MWE	307	0E-12	.996488429746	.26786309179170	.241343374094172	.058
MWE$^+$	307	0E-12	1.014695268018	.24650129431479	.213869667631371	.046
Valid N (list wise)	307					

The table 4 shows the descriptive statistics for MWE and Extended MWE (MWE+) metric for 307 numbers of classes. From that descriptive statistics we get all the values like maximum, minimum, mean, standard deviation and variances for MWE and MWE+ cohesion metrics for 307 classes of Rhino software. That Rhino software is the java script application which is implemented in java. This software consider as a input source code.

Proposed Maximum Weighted Entropy plus (MWE+) considers the concept of inheritance and the effect of getter and setter methods. Proposed MWE+ metric consider all methods those a class implements or overloads or inherits from all direct and indirect super classes. I excluded getter and setter methods in MWE+ calculation.

Table 5: Top 15 Classes of Rhino having maximum MWE and MWE+ difference

	CLASSNAME	MWE	MWE+	DIFFERENCE
2	rhino1_7R4/src/org/mozilla/javascript/ast/EmptyExpression.java	0.823318139	0.105264332	0.718053807
3	rhino1_7R4/deprecatedsrc/org/mozilla/javascript/xml/impl/xmlbeans/XMLWithScope.java	0.818973282	0.103351494	0.715621787
4	rhino1_7R4/xmlimplsrc/org/mozilla/javascript/xmlimpl/XMLWithScope.java	0.791416275	0.103351494	0.688064781
5	rhino1_7R4/src/org/mozilla/javascript/ast/EmptyStatement.java	0.719790949	0.127726921	0.592064027
6	rhino1_7R4/src/org/mozilla/javascript/TopLevel.java	0.657925713	0.152033301	0.505892412
7	rhino1_7R4/src/org/mozilla/javascript/JavaMembers.java	0.652595968	0.167373792	0.485222176
8	rhino1_7R4/src/org/mozilla/javascript/commonjs/module/ModuleScope.java	0.647428398	0.168229339	0.479199059
9	rhino1_7R4/testsrc/org/mozilla/javascript/tests/JsTestsTest.java	0.960051353	0.495231109	0.464820244
10	rhino1_7R4/toolsrc/org/mozilla/javascript/tools/shell/SecurityProxy.java	0	0.451342878	0.451342878
11	rhino1_7R4/src/org/mozilla/javascript/ast/ParenthesizedExpression.java	0.592312544	0.156300323	0.436012221
12	rhino1_7R4/src/org/mozilla/javascript/EvaluatorException.java	0.99648843	0.57266154	0.423826889
13	rhino1_7R4/src/org/mozilla/javascript/ast/Yield.java	0.560849927	0.152451191	0.408398737
14	rhino1_7R4/src/org/mozilla/javascript/ast/DoLoop.java	0.512911405	0.152756794	0.360154611
15	rhino1_7R4/src/org/mozilla/javascript/NativeDate.java	0.536486228	0.185724443	0.350761785

To understand what new dimension MWE+ adds to MWE performed a case study that has following steps:

1. First, calculated MWE and MWE+ metrics for all 308 classes of target Rhino software system. Table 4 provides Descriptive Statistics for calculated MWE and MWE+ metrics values. These descriptive statistics were obtained using IBM SPSS tool [29].
2. Then calculated the difference between MWE value and MWE+ values for each class of the Rhino software system. This difference shows the deviation of MWE value due to inclusion of all those methods which a class inherited from all direct and indirect super classes and exclusion of getter setter methods.
3. Arranged these difference values in descending order and analyzed the difference values and derived following conclusion :

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

i. All classes those doesn't inherit any class have same MWE and MWE+ values, hence difference value is 0. There are 160 such classes in Rhino software system.
ii. 148 classes of Rhino Software system inherit other class. even most of 148 classes have a inheritance hierarchy, due to this there is deviation between values of MWE and MWE+.
iii. The difference between MWE and MWE+ values for classes of Rhino software system ranges from 0.001310746 to 0.718053807. MWE and MWE+ values itself range from 0 to 1, maximum deviation of 0.718053807 against maximum standard value of MWE considering the effect of inheritance and access methods while calculating the MWE for object oriented software system.
iv. Table 5 shows the top 15 classes of rhino software system having maximum difference between MWE and MWE+ values.

The figure 2 shows the graph of table 5 details that mean the MWE Vs extended MWE (MWE+). This graph shows the Weighted Entropy values for top 15 classes of Rhino software which are having the maximum difference in Extended MWE (MWE+) and previous MWE. In that the MWE+ values are minimum than MWE therefore it get high cohesion that support to object oriented analysis to facilitate to program comprehension, testing, reusability and maintainability.

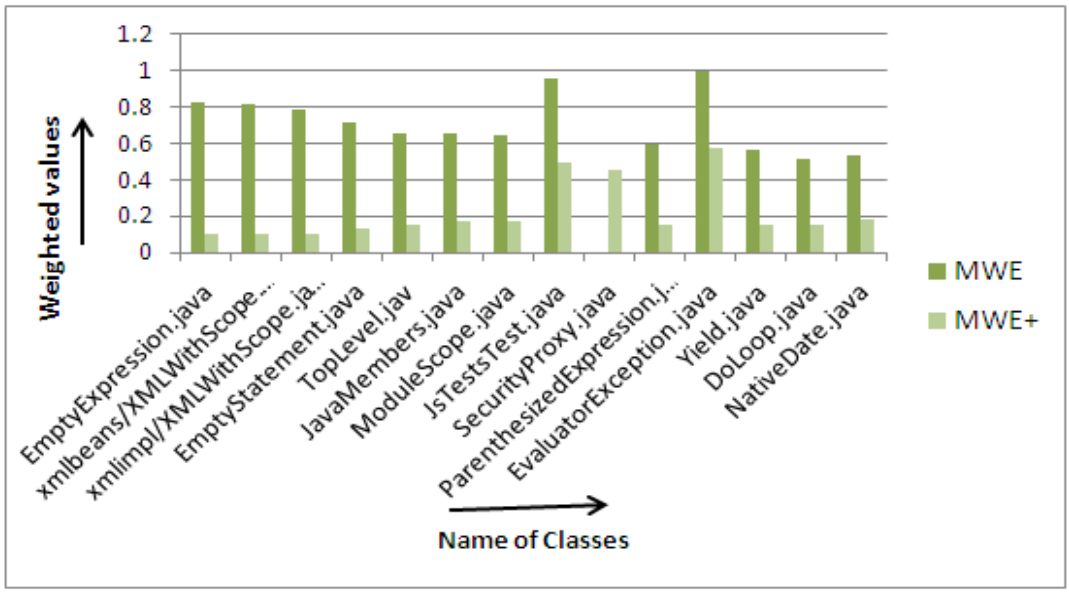

Fig. 2 MWE Vs MWE+ for top 15 classes of Rhino

V. CONCLUSION

Classes contain identifiers and comments in object-oriented systems can be used to measure the cohesion of software. Latent Dirichlet Allocation is used to extract latent topics for cohesion measurement. By using this topic modeling technique (LDA), set of LCSM+ cohesion metrics, ACSM+ and LCSM2# metrics capture new, yet complementary dimensions of cohesion as compared to existing metrics. Principal component analysis of measurement results on an open source software system, called Rhino, statistically supports. PCA results statistically support hypothesis that the LCSM3+ LCSM4+, ACSM+ and LCSM2# cohesion measure captures different aspects of class cohesion than those capture by all the metrics computed in case study. The second case study indicate that Maximum Weighted Entropy (MWE+) metric, a extended form of MWE, that also consider all those methods which a class inherits from all direct and indirect super classes along with all those methods which the class implements or overloads and excludes getter and setter methods, has significant improvements over existing MWE metrics. In object oriented software development, most of the classes access functionality of other classes by inheriting them. Special methods

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 11, November 2016

such as getter and setter methods may artificially increase or decrease class cohesion values calculated by MWE. So considering effect of inheritance, access methods like getter and setter methods improve the accuracy of MWE metric. Due to support of reusability, information hiding and code comprehending concept of inheritance is widely used in software system. So these results are quite encouraging.

.

REFERENCES

[1] Y. Liu, D. Poshyvanyk, R. Ferenc, T. Gyimothy, and N. Chrisochoides, "Modeling class cohesion as mixtures of latent topics", in Proc. of the 25th ICSM, IEEE, pp. 233 –242, 2009.
[2] Marcus, A. and Poshyvanyk, D., "The Conceptual Cohesion of Classes", in Proc. of ICSM'05, Hungary, Sept., IEEE, pp. 133-142, 2005.
[3] Bansiya, J. and Davis, C. G., "A hierarchical model for object oriented design quality assessment", IEEE TSE Jan'02, pp. 4-17, 2002.
[4] H. Wallach, D. Mimno, and A. McCallum, "Rethinking LDA: Why priors matter", Proceedings of NIPS-09, Vancouver, BC, 2009.
[5] Blei, D. M., Ng, A. Y., and Jordan, M. I., "Latent Dirichlet Allocation", Journal of Machine Learning Research, vol. 3, 2003.
[6] Briand, L. C., Daly, J. W., and Wüst, J., "A Unified Framework for Cohesion measurement in OO Systems", ESE', 3/1, pp. 65-117, 1998.
[7] Briand, L. C., Wüst, J., Daly, J. W., and Porter, V. D., "Exploring the relationship between design cohesion metrics and software quality in object-oriented systems", JSS, 51/3, pp. 245-273 , May'2000.
[8] 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, pp. 826-832, November 2004.
[9] Chidamber, S. R. and Kemerer, C. F., "A Metrics Suite for OO Design", IEEE TSE, vol. 20, no. 6, pp. 476-493, 1994.
[10] De Lucia, A., Oliveto, R., and Vorraro, L., "Using structural and semantic metrics to improve class cohesion", IEEE, in ICSM'08, pp 27-36, 2008.
[11] Dempster, A., Laird, N., and Rubin, D., "Likelihood from incomplete data via the EM algorithm", JRSS, vol. 39, no. 1, pp. 1-38, 1977.
[12] Griffiths, T. and Steyvers, M., "Finding scientific topics", Proc. of the National Academy of Sciences 2004.
[13] 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, pp. 897-910, October 2005.
[14] Henderson-Sellers, B., "Software Metrics", Prentice Hall, 1996.
[15] 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.
[16] Lukins, S., Kraft, N., and Etzkorn, L., "Source Code Retrieval for Bug Location Using Latent Dirichlet Allocation", in Proc. Of WCRE'08, Antwerp, Belgium, , pp. 155-164, 2008.
[17] Maletic, J. I. and Marcus, A., "Supporting Program Comprehension Using Semantic and Structural Information", IEEE, in Proc. of ICSE'01, Toronto, Ontario, Canada, pp. 103-112, May 12-19 2001.
[18] 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, pp. 287-300,2008,.
[19] Maskeri, G., Sarkar, S., and Heafield, K., "Mining Business Topics in Source Code using LDA", ACM, in ISEC'08, India, pp. 113-120, 2008.
[20] Shannon, C. E., "A mathematical theory of communication", Bell System Technical Journal, vol. 27, pp. 379–423, July 1948.
[21] A. K. McCallum, "Mallet: A machine learning for language toolkit" , [Online]. Available: http://mallet.cs.umass.edu., 2002.
[22] M. F. Porter "An algorithm for suffix stripping" ,Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
[23] Zhou, Y., Xu, B., Zhao, J., and Yang, H., "ICBMC: an improved cohesion metric for classes", IEEE, ICSM'02, pp. 44-53, in 2002.
[24] Rhino: http://www.mozilla.org/rhino/.
[25] "Simple Code Metrics", http://plugins.netbeans.org/plugin/9494/simple-code-metrics, 1994.
[26] Jobson, J. D., "Applied Multivariable Data Analysis", Springer-Varlag, 1992.
[27] Jolliffe, I. T., "Principal Component Analysis", Springer Verlag, 1986.
[28] "Tanagra - A Data Mining Software", http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html.
[29] "IBM SPSS- A statistics tool", http://www-01.ibm.com/software/analytics/spss/downloads.