EXPLORING EMPIRICALLY THE RELATIONSHIP BETWEEN LACK OF COHESION IN OBJECT-ORIENTED SYSTEMS AND COUPLING AND SIZE

Linda Badri, Mourad Badri and Fadel Toure
Software Engineering Research Laboratory, Department of Mathematics and Computer Science
University of Quebec at Trois-Rivières, Trois-Rivières, Quebec, Canada

Keywords: Software Quality, Quality Attributes, Software Attributes, Metrics, Lack of Cohesion, Coupling, Size.

Abstract: The study presented in this paper aims at exploring empirically the relationship between lack of cohesion of classes in object-oriented systems and their coupling and size. We designed and conducted an empirical study on various open source Java software systems. The experiment has been conducted using several well known code-based metrics related to cohesion, coupling and size. The results of this study provide evidence that a lack of cohesion may actually be associated with (high) coupling and (large) size.

1 INTRODUCTION

A large number of object-oriented (OO) metrics have been proposed in the literature. They are used to assess different software attributes. Software metrics can be calculated automatically from source code. The assessment of even large software systems can then be performed quickly at a low cost. Software metrics can be useful in predicting software quality and supporting various software engineering activities (Basili, 1996; Bansiya, 2002; Briand, 2000; Chidamber, 1998; Darcy, 2005; El Emam, 1999; Fenton, 1996).

Cohesion is considered as one of most important OO software attributes. Many metrics have been proposed in the last several years to measure class cohesion in OO systems. Class cohesion (more specifically, functional cohesion) is defined as the degree of relatedness between members of a class. In OO systems, a class should represent a single logical concept, and not to be a collection of miscellaneous features. OO analysis and design methods promote a modular design by creating classes with high cohesion and low coupling (Larman, 2003; Pressman, 2005; Sommerville, 2004). Improper assignment of responsibilities in the design phase can produce low cohesive classes with unrelated members. The reasoning is that classes that lack cohesion are poorly designed and will be difficult, among others, to understand, to test and to maintain. However, there is no empirical evidence on these claims. In fact, studies have failed to show a significant relationship between cohesion metrics and software quality attributes such as fault-proneness or changeability (Briand, 1998; Briand, 2000; Kabaili, 2001). Moreover, studies have noted that cohesion metrics fail to properly reflect cohesion of classes (Aman, 2002; Chae, 2000; Chae, 2004; Kabaili, 2000; Kabaili, 2001). As against, several studies have showed that there exist a significant relationship between software attributes such as complexity, coupling and size and software quality attributes such as fault-proneness, testability and maintainability.

One possible explanation of the lack of relationship between cohesion (according to experimented cohesion metrics) and some software quality attributes is due to the difficulty of measuring cohesion from syntactic elements of code (Briand, 1998; Briand, 2000; Henderson-sellers, 1996; Stein, 2005). Moreover, cohesion metrics are in our opinion based on restrictive criteria, in the sense that they do not consider some characteristics of classes, which lead in many situations to some inconsistency between the computed cohesion values and the intuitively expected ones (Badri, 2004; Chae, 2000; Chae, 2004; Kabaili, 2001). However, an empirical study performed by Stein et *al.* (Stein, 2005) pointed to a more basic relationship between cohesion and complexity: that a lack of

Badri L., Badri M. and Toure F. (2010).
EXPLORING EMPIRICALLY THE RELATIONSHIP BETWEEN LACK OF COHESION IN OBJECT-ORIENTED SYSTEMS AND COUPLING AND SIZE.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 317-324
DOI: 10.5220/0003012203170324

cohesion may be associated with high complexity.

In this paper, we decided to explore empirically the relationship between lack of cohesion (disparity of the code) of classes in OO systems and their coupling and size. Our hypothesis is that classes with high (strong) coupling and/or large size will lack cohesion. To test our hypothesis, we chose in our experiment two well-known lack of cohesion metrics: LCOM (Lack of COhesion in Methods) (Chidamber, 1994) and LCOM* (Henderson-sellers, 1996). In order to facilitate comparison with our class cohesion measurement approach (Badri, 2004; Badri, 2008), and knowing that the selected cohesion metrics are basically lack of cohesion metrics (inverse cohesion measures), we derive a lack of cohesion measure from the cohesion metric we proposed.

In order to explore the relationship between lack of cohesion and coupling and size, we investigate in this study a small selection of coupling and size metrics. We focus on measures defined at the class level. We chose the well-known coupling metrics: CBO (Coupling Between Objects) (Chidamber, 1994) and FO (Fan-Out) (Kitchenham, 1990), and size measures: LOC (Lines Of Code), NOA (Number of Attributes) and NOO (Number of Operations) (Henderson-sellers, 1996). Our aim in this project, as a next step, is to investigate lack of cohesion as a predictor of some relevant external software quality attributes such as testability and maintainability. These issues will be addressed in a future work. We designed and conducted an empirical study on several open source Java software systems. The achieved results provide evidence that a lack of cohesion may actually be associated with high coupling and large size, validating some fundamental design principles of software engineering.

The rest of the paper is organized as follows: We give in Section 2 an overview of major class cohesion metrics. Section 3 presents some related work. Section 4 presents briefly our approach for class cohesion measurement. Section 5 gives some characteristics of the systems we used in our experiment. We present and discuss in Section 6 and Section 7 the empirical investigation that we conducted to explore respectively the relationship between lack of cohesion and coupling and the relationship between lack of cohesion and size. Finally, Section 8 summarizes the contributions of this work, discusses some of its limitations and outlines directions for further research.

2 COHESION METRICS

Yourdon *et al.* (Yourdon, 1979) defined cohesion, in the procedural paradigm, as a measure of the extent of the functional relationships of the elements in a module. In the OO paradigm, Booch (Booch, 1994) described high functional cohesion as existing when the elements of a class all work together to provide some well-bounded behavior. There are several types of cohesion: functional cohesion, sequential cohesion, coincidental cohesion, etc. (Henderson-sellers, 1996; Yourdon, 1979). In this work, we focus on functional cohesion.

Many metrics have been proposed in the last several years in order to measure class cohesion in OO systems. The argument over the most meaningful of those metrics continues to be debated (Counsell, 2006). Major of proposed cohesion metrics are based on the notion of degree of similarity of methods, and usually capture cohesion in terms of connections between members of a class. They present, however, some differences in the definition of the relationships between members in a class (mechanism that defines cohesion and its measure). A class is more cohesive, as stated in (Chae, 2000), when a larger number of its instance variables are referenced by a method (LCOM* (Henderson-sellers, 1996), Coh (Briand, 1998)), or a larger number of methods pairs share instance variables (LCOM1 (Chidamber, 1991), LCOM2 (Chidamber, 1994), LCOM3 (Li, 1993), LCOM4 (Hitz, 1995), Co (Hitz, 1995), TCC and LCC (Bieman, 1995), DC_D and DC_I (Badri, 2004; Badri, 2008)). We chose in our study the cohesion metrics: LCOM (Chidamber, 1994) and LCOM* (Henderson-sellers, 1996). LCOM (referenced in the literature as LCOM2, as a refinement of LCOM1) is defined as the number of pairs of methods in a class having no common attributes minus the number of pairs of methods having at least one common attribute. LCOM is set to zero when the value is negative. LCOM* is somewhat different from the LCOM metric. LCOM* is different also from the other versions of the LCOM metric proposed by Li *et al.* (Li, 1993) and Hitz *et al.* (Hitz, 1995). It considers that cohesion is directly proportional to the number of instance variables that are referenced by the methods of a class.

These metrics are known as structural metrics, which is the most investigated category of cohesion metrics. They measure cohesion on structural information extracted from the source code. Several studies, using the Principal Component Analysis technique, have been conducted in order to

understand the underlying orthogonal dimensions captured by some of these metrics (Aggarwal, 2006; Briand, 1998; Chae, 2000; Etzkorn, 2004; Marcus, 2005). Briand *et al.* (Briand, 1998) developed a unified framework for cohesion measurement in OO systems that classifies and discusses several cohesion metrics. Development of metrics for class cohesion assessment still continues (Badri, 2008; Chae, 2004; Chen, 2002; Counsell, 2006; Marcus, 2005; Marcus, 2008; Meyers, 2004; Woo, 2009; Zhou, 2002; Zhou, 2003). Recent approaches for assessing class cohesion focus on semantic cohesion (De Lucia, 2008; Marcus, 2008). We focus in this work on structural cohesion metrics.

3 COUPLING AND SIZE METRICS

Coupling between two classes exists if one class access or uses some elements of the other class. Chidamber *et al.* (Chidamber, 1994) proposed the CBO (Coupling Between Objects) metric that counts for a class the number of other classes to which it is coupled. This metric has been validated by Basili *et al.* (Basili, 1996) as a fault prone indicator. Kitchenham *et al.* (Kitchenham, 1990) defined FO (Fan-Out) as a count of the number of classes that are called by a given class. Various other coupling metrics have been proposed in the literature (Briand, 1997; Hitz, 1995; Li, 1993; Li, 1995). Studies showed, in fact, that coupling metrics are good predictive indicators of software quality (Aggarwal, 2006; Briand, 2000; Bruntink, 2006; Chaumum, 2000; Harrison, 1998; Xu, 2008; Zhou, 2006). Well known practices of software engineering promote modular design with low coupling between classes in order to facilitate, among others, comprehension, testing, maintenance and evolution (Larman, 2003; Pressman, 2005; Sommerville, 2004). Furthermore, size is also a good indicator of various attributes of software quality. The metric LOC (Lines Of Code), widely accepted in the software engineering community as a size/complexity metric (Dunsmore, 1984; Henderson-sellers, 1996; Levitin, 1986; Pant, 1995; Weyuker, 1988), has been used for a number of different software development activities. Many empirical results showed its usefulness (Basili, 1996; Bruntink, 2006; Henderson-sellers, 1996; Dagpinar, 2003; El Emam, 1999; Xu, 2008). NOA (Number of Attributes) and NOO (Number of Operations) are alternative size metrics more appropriate to an OO context.

4 CLASS COHESION MEASUREMENT

We give, in this section, a brief overview of our approach for class cohesion measurement. For more details see (Badri, 2004; Badri, 2008). The approach is based on different cohesion criteria. It takes into account different ways of capturing functional cohesion in a class.

Used Attributes: Two methods M_i and M_j are directly related if there is at least one attribute shared by the two methods.

Invoked Methods: Two methods M_i and M_j are directly related if there is at least one method invoked by the two methods. We also consider that M_i and M_j are directly related if M_i invokes M_j, or vice-versa.

Common Objects Parameters: Two methods M_i and M_j are directly related if there is at least one parameter of object type used by the two methods.

Two methods M_i and M_j may be directly connected in many ways: they share at least one instance variable in common, or interact at least with another method of the same class, or share at least one object passed as parameter. Let us consider a class *C* with *n* methods. The maximum number of methods pairs is $[n * (n - 1) / 2]$. Consider an undirected graph G_D, where the vertices are the methods of the class *C*, and there is an edge between two vertices if the corresponding methods are directly related. Let E_D be the number of edges in the graph G_D. The cohesion of the class *C*, based on the direct relation between its methods, is defined as: $DC_D = |E_D| / [n * (n - 1) / 2] \in [0,1]$. DC_D gives the percentage of methods pairs, which are directly related.

In order to facilitate comparison with the metrics LCOM and LCOM*, we derive a lack of cohesion measure (following the same approach of LCOM) from our approach. We associate to a class *C* (with *n* methods) a lack of cohesion measure (not normalized) based on the direct relation given by: $LC_D = [n * (n - 1) / 2] - 2 * |E_D|$. When the difference is negative, LC_D is set to zero.

5 SELECTED SYSTEMS

In order to achieve significant results, the data used in our empirical study were collected from several open source Java software systems. We used in our experiment eight systems from different domains and of varying sizes. The analyzed systems consist of a total of more than 2 000 classes (more than

Table 1: Some characteristics of the used systems and mean values of the selected metrics.

	#Classes	#Attributes	#Methods	#LOC	CBO	FO	LOC	NOA	NOO	LCOM	LCOM*	LC_D
GNUJSP	79	207	373	5225	5.28	4	66.14	2.62	9.47	94.85	60.57	52.67
JFLEX	47	403	401	9086	6.66	3.87	193.32	8.57	8.53	38.2	55.17	59.6
DBUNIT	213	464	874	11562	6.03	4.12	54.28	2.18	3.82	12.97	44.42	8.58
FREECS	115	712	822	15244	9.43	4.74	132.56	6.19	7.15	71.60	72.89	74.30
JHOTDRAW	301	688	3109	20767	8.26	5.34	68.99	2.29	8.51	125.82	46.16	80.46
JMOL	294	1942	1972	28967	7.88	4.76	98.53	6.61	6.14	249.92	68.68	161.78
ANT	657	3244	5822	63518	6.81	4.87	96.68	4.94	7.87	76.74	61.76	61.80
JFREECHART	411	2344	5589	67481	9.72	6.92	164.19	5.70	12.78	198.32	54.08	236.04

200 000 lines of code). We provide in what follows some background on the systems that are used in this study.

- GNUJSP (http://www.klomp.org/gnujsp/). A free implementation of JSP (Java Server Pages). The analyzed version (1.0.1) contains 79 classes.
- JMOL (http://www.openscience.org). A software for visualizing molecules for students, educators and researchers. The analyzed version (7) contains 294 classes.
- FREECS (http://freecs.sourceforge.net/). An online chat server. The analyzed version (1.2.2) contains 115 classes.
- JFLEX (http://jflex.de/). A lexical analyzer generator. The analyzed version (1.4) contains 47 classes.
- ANT (www.apache.org). A Java-based build tool, with functionalities similar to the Unix "make" utility. The analyzed version (1.5.3) contains 657 classes.
- JHOTDRAW (http://www.jhotdraw.org). A Java GUI framework for technical and structured graphics. The analyzed version (5.4) contains 301 classes.
- JFREECHART (http://www.jfree.org/jfreechart). A chart library for the Java (tm) platform. The analyzed version (1.1.0) contains 411 classes.
- DBUNIT (http://www.dbunit.org). A Database Testing Framework. The analyzed version (2.1) contains 213 classes.

Table 1 summarizes some characteristics of the used systems and gives the mean values of the selected metrics.

6 RELATIONSHIP BETWEEN LACK OF COHESION AND COUPLING

We present, in this section, the empirical study we conducted in order to assess the relationship between lack of cohesion and coupling. We performed statistical tests using correlation. The null and alternative hypotheses were:

- H0: There is no significant correlation between lack of cohesion and coupling.
- H1: There is a significant correlation between lack of cohesion and coupling.

In this experiment, rejecting the null hypothesis indicates that there is a statistically significant relationship between lack of cohesion and coupling (chosen significance level α=0.05). We used the coupling metrics CBO and FO, and the lack of cohesion metrics LCOM, LCOM* and LC_D. The metrics LCOM, LCOM*, CBO and FO have been computed using the Borland Together tool. The LC_D metric has been computed using the tool we developed. Several classes in the selected systems have, in fact, only one method. These classes were considered as special classes and have been excluded from our measurements. We also excluded all abstract classes. Special methods like constructors were also removed. These methods may artificially increase or decrease the cohesion of a class.

We collected the metrics data from the eight selected systems. Given the distribution of the measures we observed, we preferred a non-parametric measure of correlation in order to test the correlation between lack of cohesion and coupling. We used the Spearman's correlation coefficient. This technique, based on ranks of the observations, is widely used for measuring the degree of linear relationship between the ranks of two variables (two sets of ranked data). It measures how tightly the ranked data clusters around a straight line. Spearman's correlation coefficient will take a value between -1 and +1. A positive correlation is one in which the ranks of both variables increase together. A negative correlation is one in which the ranks of one variable increase as the ranks of the other variable decrease. A correlation close to zero means that there is no linear relationship between the ranks.

We also used the Kendall method to investigate if the two rankings (lack of cohesion and coupling) are consistent. Kendall's rank correlation reflects the strength of the dependence between the variables

Table 2: Correlations between Lack of cohesion and Coupling metrics.

		Spearman		Kendall				Spearman		Kendall				Spearman		Kendall				Spearman		Kendall	
		CBO	FO	CBO	FO			CBO	FO	CBO	FO			CBO	FO	CBO	FO			CBO	FO	CBO	FO
JHOTDRAW	LCOM	0.568	0.608	0.431	0.474	GNUJSP	LCOM	0.517	0.467	0.373	0.353	ANT	LCOM	0.508	0.519	0.373	0.388	JMOL	LCOM	0.338	0.393	0.242	0.292
	210	0	0	0	0		26	0.007	0.016	0.011	0.017		498	0	0	0	0		161	0	0	0	0
	LCOM*	0.368	0.366	0.275	0.28		LCOM*	0.58	0.569	0.468	0.459		LCOM*	0.466	0.45	0.346	0.333		LCOM*	0.167	0.154	0.128	0.117
	176	0	0	0	0		21	0.006	0.007	0.005	0.006		456	0	0	0	0		123	0.064	0.089	0.044	0.068
	LC_D	0.598	0.61	0.459	0.477		LC_D	0.497	0.478	0.366	0.348		LC_D	0.598	0.622	0.46	0.486		LC_D	0.33	0.438	0.258	0.357
	301	0	0	0	0		36	0.002	0.003	0.004	0.006		657	0	0	0	0		294	0	0	0	0
		CBO	FO	CBO	FO			CBO	FO	CBO	FO			CBO	FO	CBO	FO			CBO	FO	CBO	FO
FREECS	LCOM	0.545	0.553	0.414	0.432	JFLEX	LCOM	0.508	0.478	0.382	0.359	JFREECHART	LCOM	0.568	0.562	0.42	0.419	DBUNIT	LCOM	0.368	0.294	0.267	0.214
	105	0	0	0	0		41	0.001	0.002	0.001	0.003		362	0	0	0	0		130	0	0.001	0	0.001
	LCOM*	0.457	0.445	0.32	0.314		LCOM*	0.563	0.499	0.437	0.408		LCOM*	0.276	0.304	0.202	0.223		LCOM*	0.246	0.294	0.19	0.23
	111	0.001	0.002	0.002	0.003		35	0	0.002	0	0.001		392	0	0	0	0		99	0.014	0.003	0.01	0.002
	LC_D	0.509	0.542	0.386	0.44		LC_D	0.606	0.588	0.456	0.445		LC_D	0.566	0.566	0.421	0.423		LC_D	0.424	0.46	0.323	0.353
	115	0	0	0	0		47	0	0	0	0		411	0	0	0	0		213	0	0	0	0

being compared. High values of the Kendall's correlation coefficient means that the most pairs of values are concordant, indicating that the two rankings (lack of cohesion and coupling) are consistent. Analysis of the data sets is done by calculating the Spearman's and Kendall's correlation coefficients for each pair of metrics (lack of cohesion metric, coupling metric). We have a total of six pairs of metrics.

Table 2 summarises the results of the correlation analysis. It shows, for each used system and between each distinct pair of metrics, the obtained values for the Spearman's and Kendall's correlation coefficients with their corresponding *p*-values. Moreover, under each metric name in Table 2, we mention the number of classes that were actually used in the analysis. The cohesion values of several classes (like classes having no attributes) are not, in fact, computed by the Together tool for the metrics LCOM and LCOM*. This is due to the definition of measures themselves. The cohesion values of such classes are, however, computed (using our tool) for the metric LC_D. These classes have, in fact, several methods which are connected according to the cohesion criteria defined in section 4.

The obtained Spearman's correlation coefficients are significant (at α=0.05). Moreover, the measures have positive correlation. Since the used cohesion metrics are, in fact, lack of cohesion measures (inverse cohesion measures), the positive coefficients indicate that the ranks of both lack of cohesion and coupling increase together, which is consistent with the idea on cohesion and coupling in the software engineering community. Moreover, the obtained values of the Kendall's correlation coefficient are also significant (at α=0.05). They confirm that there is more concordance than discordance in the pairs of metrics, confirming that the two rankings are consistent. Overall, the results of the correlation analysis support the idea that the more strongly a class is coupled to other classes, the less cohesive the class is likely to be.

We can also see from Table 2 that the metric LC_D is more strongly correlated with the coupling measures than the metrics LCOM and LCOM*. The higher correlation values are observed for systems JHOTDRAW, ANT and JFLEX (metric LC_D). The fact that the metrics LCOM and LCOM* are based on the concept of sharing instance variables only, which is a restrictive way of capturing cohesion in our opinion, leads to lack of cohesion values that do not, in fact, reflect properly the cohesion of classes. The metric LC_D, compared to the metrics LCOM and LCOM*, is based on various and complementary cohesion criteria. It captures more pairs of connected methods than LCOM and LCOM* metrics. It captures additional dimensions of cohesion measurement. This explains, in our opinion, why LC_D obtains higher correlation values with coupling measures than the metrics LCOM and LCOM*.

7 RELATIONSHIP BETWEEN LACK OF COHESION AND SIZE

We present, in this section, the empirical study we conducted in order to assess the relationship between lack of cohesion and size. We performed statistical tests using correlation. The null and alternative hypotheses were:

- H0: There is no significant correlation between lack of cohesion and size.
- H1: There is a significant correlation between lack of cohesion and size.

Rejecting the null hypothesis, in this experiment also, indicates that there is a statistically significant relationship between lack of cohesion and size

Table 3: Correlations between Lack of cohesion and Size metrics.

		Spearman															
		LOC	NOO	NOA		LOC	NOO	NOA		LOC	NOO	NOA		LOC	NOO	NOA	
JHOTDRAW	LCOM 210	0.688 0	0.81 0	0.441 0	GNUJSP LCOM 26	0.749 0.007	0.873 0	0.269 0.183	JMOL LCOM 161	0.698 0	0.805 0	457 0	ANT LCOM 498	0.752 0	0.852 0	0.692 0
	LCOM* 176	0.384 0	0.81 0	0.731 0	LCOM* 21	0.68 0.001	0.253 0.267	0.719 0	LCOM* 123	0.425 0	0.34 0	0.467 0	LCOM* 456	0.652 0	0.688 0	0.761 0
	LCD 301	0.744 0	0.853 0	0.578 0	LC_D 36	0.67 0.002	0.67 0	0.419 0.011	LC_D 294	0.705 0	0.845 0	0.634 0	LC_D 657	0.797 0	0.886 0	0.793 0
		LOC	NOO	NOA		LOC	NOO	NOA		LOC	NOO	NOA		LOC	NOO	NOA	
FREECS	LCOM 105	0,590 0	0.749 0	0.463 0	JFLEX LCOM 41	0.512 0.001	0.483 0.002	0.419 0.007	DBUNIT LCOM 130	0.529 0	0.574 0	0.146 0.099	JFREECHART LCOM 362	0.689 0	0.721 0	0.561 0
	LCOM* 111	0.753 0	0.601 0	0.575 0	LCOM* 35	0.545 0.001	0.674 0	0.561 0.001	LCOM* 99	0.478 0	0.384 0	0.649 0	LCOM* 392	0.359 0	0.446 0	0.672 0
	LC_D 115	0.678 0	0.897 0	0.631 0	LC_D 47	0.802 0	0.933 0	0.469 0.001	LC_D 213	0.719 0	0.847 0	0.432 0	LC_D 411	0.826 0	0.992 0	0.736 0

(chosen significance level α=0.05). We used the well-known size metrics LOC (Lines Of Code), NOA (Number of Attributes) and NOO (Number of Operations). We used also, as for the previous empirical study, the lack of cohesion metrics LCOM, LCOM* and LC_D. In this experiment also, special classes and methods have been excluded from our measurements. We collected the metrics data from the eight selected systems. We used the Spearman technique to assess the correlation. Table 3 summarises the results of the correlation analysis. It shows, for each used system and between each distinct pair of metrics (lack of cohesion metric, size metric), the obtained values for the Spearman's correlation coefficient (with their corresponding *p*-values). We have a total of nine pairs of metrics. The obtained correlations in this experiment also are significant (at α=0.05). Furthermore, the correlation values are positive. This indicates that the ranks of both lack of cohesion and size increase together, which is consistent with the idea on cohesion and size in the software engineering community.

These findings indicate that there is a relative strong correlation between lack of cohesion and the quantities measured by size (particularly the number of methods). The obtained results support the idea that the larger the class is, the less cohesive the class is likely to be. It is, in fact, plausible that the larger a class is, particularly in terms of number of attributes and methods, the more tasks it includes, so there is an increased likelihood that some of those tasks are unrelated which reduces the cohesion of the class.

Overall, we can observe from Table 3 that the obtained correlation values are higher than those obtained with coupling (Table 2). Moreover, the metric LC_D is more strongly correlated, in this experiment also, with the size measures than the metrics LCOM and LCOM*. The higher correlation values between LC_D and LOC are observed for systems JFREECHART, JFLEX, ANT, JHOTDRAW and DBUNIT. The same trend is also observed with the metric NOO. The correlation between LC_D and particularly the metric NOO is very strong. As it was the case in the previous empirical investigation, this confirms that LC_D captures additional dimensions of cohesion measurement compared to the metrics LCOM and LCOM*. Moreover, the number of classes in a system seems not influencing the correlation values. In fact, the higher values of correlations are observed in relative small systems (such as JFLEX with 47 classes) as well as in relatively large systems (such as ANT with 657 classes).

8 CONCLUSIONS AND FUTURE WORK

The paper investigates the relationship between lack of cohesion in OO systems and coupling and size. We performed an empirical study on various open source Java software systems. We used several well-known code-based metrics related to cohesion, coupling and size. The achieved results support the idea that a lack of cohesion may actually be associated with (high) coupling and (large) size, validating some fundamental design principles of software engineering. The correlation analysis showed also that, essentially, the metric LC_D has higher correlation values with coupling and size measures than the metrics LCOM and LCOM*, which confirms that it captures additional dimensions in class cohesion measurement. We hope this study will contribute to a better understanding of the relationship between cohesion (lack of cohesion) and other OO software attributes.

The analysis performed here is correlational in nature. We have demonstrated that there is a statistically and practically significant relationship between lack of cohesion of classes in OO systems and coupling and size. Such statistical relationships do not imply causality. They only provide empirical evidence of it. Only controlled experiments, where the measures would be varied in a controlled manner, could really demonstrate causality. However, such experiments would be difficult to perform in practice as stated by Briand *et al.* in (Briand, 2000).

The study performed in this paper should be replicated across many environments and systems, particularly large-scale systems, in order to draw more general conclusions about the relationship between lack of cohesion and coupling and size. In fact, there are a number of limitations that may affect the results of the study or limit their interpretation and generalization.

Firstly, the obtained results are based on the data set we collected from the analyzed systems. Our study involved a relatively small number of systems considering the fact that there are plenty of OO systems available. This may pose a threat to the scalability of the results. Even if we believe that the total number of analyzed classes is enough large to allow obtaining significant results, the study should be, however, replicated on a large number of OO systems to increase the generality of the results. Secondly, the systems used in the experiment are rather small or medium. Our study should be replicated on large systems. Thirdly, it is also possible that facts such as the development process used to develop the analyzed systems (for example, using design heuristics to guide the design process) and the development style of a given development team might affect the results or produce different results for specific applications. Moreover, our study involved only software systems written in Java. While there is no reason to suspect that the results would be different with systems written in other OO languages (such as C++), it would be interesting to study systems written in other languages.

As future work, we plan to investigate the relationship between lack of cohesion and some relevant external software quality attributes such as testability. In addition, we will extend this study by integrating other class cohesion metrics, particularly semantic cohesion metrics.

ACKNOWLEDGEMENTS

This project was financially supported by NSERC (National Sciences and Engineering Research Council of Canada).

REFERENCES

Aggarwal, K. K., Yogesh, S., Arvinder K., Ruchika, M., 2006. Empirical study of object-oriented metrics. *Journal of Object Technology*. vol. 5. no. 8.

Aggarwal, K. K., Yogesh, S., Arvinder, K., Ruchika, M., 2006. Investigating the effect of coupling metrics on fault proneness in object-oriented systems. *SQP*, vol. 8 no. 4.

Aman, H., Yamasaki, K., Yamada, H., Noda, M.T., 2002. A proposal of class cohesion metrics using sizes of cohesive parts. *Knowledge-Based Sof. Engineering*. T. Welzer et al. (Eds) IOS Press.

Badri, L., Badri, M., 2004. A proposal of a new class cohesion criterion: An empirical Study. *Journal of Object Technology*. vol. 3, no. 4, Special issue: TOOLS USA.

Badri, L., Badri, M., Gueye, A., 2008. Revisiting class cohesion, An empirical investigation on several systems. *Journal of Object Technology*. vol. 7, no. 6.

Basili, V. R., Briand, L.C., Melo, W., 1996. A validation of object-oriented design metrics as quality indicators. *IEEE TSE*, 22 (10).

Bansiya, J., Davis, C.G., 2002. A hierarchical model for object-oriented design quality assessment. *IEEE TSE*, vol. 28, no. 1.

Bieman, J. M., Kang, B. K., 1995. Cohesion and reuse in an object-oriented system. *Proc. of the Symposium on Software Reusability.*

Booch, G., 1994. *Object-Oriented Analysis and Design With Applications*, Benjamin/Cummings. 2nd edition.

Briand, L. C., Daly, J., Porter, V., Wuest, J., 1997. The dimensions of coupling in object-oriented design. *Proceedings of OOPSLA'97*.

Briand, L.C., Daly, J., Porter, V., Wuest, J., 1998. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering*, *3 (1)*.

Briand, L. C., Daly, J., Porter, V., Wuest, J., 2000. Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software*, No. 51.

Bruntink, M., Van Deursen, A., 2006. An empirical study into class testability. *Journal of Sysems and Software,* 79, 9.

Chae, H. S., Kwon, Y. R., Bae, D. H., 2000. A cohesion measure for object-oriented classes. *Software Practice and Experience*, No. 30.

Chae, H. S., Kwon, Y. R., Bae, D. H., 2004 Improving cohesion metrics for classes by considering dependent instance variables. *IEEE TSE*. vol. 30, no. 11.

Chaumun, M. A., Kabaili, H., Keller, R. K., Lustman, F., St-Denis, G., 2000. Design properties and object-oriented sotware changeability. *Proc. of the 4th Euromicro Working Conference on Software Maintenance and Reengineering*.

Chen, Z., Zhou, Y., Xu, B., Zhao, J., Yang, H., 2002. A novel approach to measuring class cohesion based on dependence analysis. *Proc. 18th International Conferrence on Software Maintenance*.

Chidamber, S. R., Kemerer, C.F., 1991. Towards a Metrics Suite for Object-Oriented Design. *OOPSLA*. Special Issue of SIGPLAN Notices, Vol. 26, No. 10.

Chidamber, S. R., Kemerer, C. F., 1994. A Metrics suite for object Oriented Design. *IEEE TSE*. Vol. 20, No. 6.

Chidamber, S. R., Darcy, D. P., Kemerer, C. F., 1998. Managerial use of metrics for object-oriented software : An exploratory analysis. *IEEE TSE*, Vol. 24, No. 8.

Counsell, S., Swift, S., 2006. The interpretation and utility of three cohesion metrics for object-oriented design. *ACM TSEM*. vol. 15, no. 2.

Dagpinar, M., Jahnke, J. H., 2003. Predicting maintenability with object-oriented metrics – An empirical comparaison. *Proc. of the 10th working conference on reverse engineering (WCRE'03)*. IEEE computer society.

Darcy, D., Kemerer, K., 2005. OO metrics in practice, *IEEE Software*. vol. 22, no. 6.

De Lucia, A., Oliveto, R., Vorraro, L., 2008. Using structural and semantic metrics to improve class cohesion. *Proc. of the ICSM*.

Dunsmore, H.E., 1984. Software metrics: An overview of an evolving methodology. *Information Processing and Management*, 20(1-2), 183-192.

El Emam, K., Melo, W., 1999. The prediction of faulty class using object-oriented design metrics. *National Research Council of Canada NRC/ERB 1064*.

Etzkorn, L. H., Gholston, S.E., Fortune, J. L., Stein, C. E., Utley, D., 2004. A comparison of cohesion metrics for object-oriented systems. *Information and Software Technology*, 46.

Fenton, N., Pfleeger, S. L., 1996. *Software Metrics: A Rigorous and Practical Approach*. Int. Thomson Computer Press. 2nd edition.

Harrison, R., Counsell, S. J., Nithi, R., 1998. An investigation into the applicability and validity of object-oriented design metrics. *Empirical Software Engineering*, vol. 3, no. 3.

Henderson-Sellers, B., 1996. *Object-Oriented Metrics Measures of Complexity*, Prentice-Hall.

Hitz, M., Montazeri, B., 1995. Measuring coupling and cohesion in object-oriented systems. *Proc. of the Int. Symp. on Applied Corporate Computing*.

Kabaili, H., Keller, R. K., Lustman, F., Saint-Denis, G., 2000. Class Cohesion Revisited: An Empirical Study on Industrial Systems. *Proc. of the Workshop on Quantitative Approaches Object-Oriented Software Engineering*. France.

Kabaili, H., Keller, R.K., Lustman, F., 2001. Cohesion as Changeability Indicator in Object-Oriented Systems. *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR 2001)*. Estoril Coast (Lisbon), Portugal.

Kitchenham, B. A. Linkman, S.J., 1990. Design metrics in practice. *Information Software Technology*.

Larman, G., 2003. *Applying UML and Design Patterns, An introduction to object-oriented analysis and design and the unified process*, Prentice Hall.

Levitin, A. V., 1986. How to measure size and how not to, *Proc. 10th COMPSAC*. Chicago, oct 8-10, IEEE Computer Society Press.

Li, W., Henry, S., 1993. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*. Vol. 23.

Li, W., Henry, S., Kafura, D., Schulman, R., 1995. Measuring Object-Oriented Design. *Journal of Object-Oriented Programming*. Vol. 8, No. 4.

Marcus, A., Poshyvanyk, D., 2005. The conceptual cohesion of classes. *Proc. 21th IEEE Int. Conf. on Software Maintenance*.

Marcus, A., Poshyvanyk, D., Ferenc, R., 2008. Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems. *IEEE TSE*. Vol. 34, NO. 2.

Meyers, T.M., Binkley, D., 2004. Slice-Based cohesion metrics and software intervention. *Proc. 11th IEEE WCRE*.

Pant, Y. R., Verner, J. M., Hendreson-Sellers, B., 1995. *S/C: a software size/complexity measure*, chapter 50 in *Software Quality and Productivity*: *Theory, Practice, Education and Training*, eds. M. Lee, B.-Z. Barta, and P. Juliff, Chapman & Hall, London.

Pressman, R.S., 2005. *Software Engineering, A practitioner's approach*, Mc Graw Hill.

Sommervile, I., 2004. *Software Engineering*.

Stein, C., Cox, G., Etzkorn, L., 2005. Exploring the relationship between cohesion and complexity. *Journal of Computer Science.* 1 (2),

Weyuker, E. J., 1988. Evaluating software complexity measures. *IEEE TSE*. 14(9), 1357-11365.

Woo, G., Chae, H. S., Cui, J. F., Ji, J.H., 2009. Revising cohesion measures by considering the impact of write interactions between class members. *Information and Software Technology*, 51.

Xu, J., Ho, D., Capretz, L.F., 2008. An Empirical Validation of Object-Oriented Design Metrics for Fault Prediction. *Journal of Computer Science*, 4 (7), 571-577.

Yourdon, E., Constantine, L., 1979. *Structured Design*, Prentice Hall, Englewood Cliffs. N.J.

Zhou, Y., Xu, B., Zhao, J., Yang, H., 2002. ICBMC: An improved cohesion measure for classes. *Proc. 18th ICSM*.

Zhou, Y., Wen, L., Wang, J., Chen., Y., Lu, H., Xu, B., 2003. DRC: dependence-relationships-based cohesion measure for classes. *Proc. 10th APSEC*.

Zhou, Y., Leung, H., 2006. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE TSE*. vol. 32, no. 10.