

## **Principal Component Analysis of Lack of Cohesion in Methods (LCOM) metrics**

Anuradha Lakshminarayana                      Timothy S. Newman  
Department of Computer Science  
University of Alabama in Huntsville

### **Abstract**

In this report, we study the Lack of Cohesion in Methods (LCOM) metric for an object-oriented system and examine the suitability of eight variations of this metric through a principal component analysis.

### **1. Introduction**

One concern in software engineering is how high-quality software can be produced with predictable costs and time. Software metrics provide a quantitative means to predict the software development process and evaluate the quality of the software products.

Several software metrics have been proposed to measure the complexity in the procedural paradigm. Some of the metrics which are frequently used in the procedural paradigm are McCabe's cyclomatic complexity metric [1] and Halstead's software science metric [2].

The object-oriented programming paradigm is often claimed to allow a faster development pace and higher quality of software. However, software metrics are less well studied in the object-oriented paradigm. A small number of metrics have been proposed to measure object-oriented systems. One of the first attempts at software metrics for object-oriented systems was made by Chidamber and Kemerer [3]. They proposed a set of six object-

oriented design metrics based on measurement theory. These include Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling between objects (CBO), Response for a Class (RFC), Weighted Methods per Class (WMC), and the Lack of Cohesion in Methods (LCOM) metrics.

In this paper, we study the LCOM metric. We examine the suitability of eight variations of the LCOM measure through a principal component analysis.

## **2. The Lack of Cohesion in Methods (LCOM) metric**

LCOM is one of the significant metrics that can be used to evaluate an object-oriented software system. LCOM is useful for estimating the amount of cohesion in the system. For example, in an object-oriented system, the LCOM can be used to measure the cohesion of each class of the system. A high LCOM value could indicate that the design of the class is poor (i.e., it might be worthwhile to split the class into two or more classes).

Several methods have been proposed to measure the LCOM metric. Some of the significant definitions are :

- (1) The original definition by Chidamber and Kemerer [3];
- (2) Li and Henry's definition [4,5];
- (3) Hitz and Montazerri's [6] re-definition of Li and Henry's LCOM; and
- (4) Chidamber and Kemerer's [7] re-definition of their original LCOM definition.

### **2.1 Chidamber and Kemerer's original LCOM**

Chidamber and Kemerer originally defined LCOM as follows :

“Let  $C_1$  be a class and let  $M_1, M_2, \dots, M_n$  be the member functions (methods) of  $C_1$ . Let  $\{I_i\}$  be the set of attributes (instance variables) of  $C_1$  used by method  $M_i$ .

(There will be 'n' such sets of attributes  $\{I_1\}, \{I_2\}, \dots, \{I_n\}$  corresponding to the 'n' methods of  $C_1$ );

LCOM = Number of disjoint sets formed by the intersection of the 'n' sets." [3]

This metric's definition has several implications :

- (a) First, cohesiveness of the methods in a class supports encapsulation of the class. A lack of cohesion would imply that it might be better to split up the class into subclasses.
- (b) LCOM is a measure of disparateness among methods in a class, so it could be used to identify flaws in the class design.

## 2.2 Li and Henry's LCOM

Li and Henry have defined LCOM as follows :

"LCOM = Number of disjoint sets of local methods.

Each set has one or more local methods of the class, and any two methods in the set access at least one attribute of the class in common; the number of common attributes ranging from 0 to N (where N is an integer greater than 0)" [4,5].

## 2.3 Hitz and Montazeri's LCOM

Hitz and Montazerri [6] re-defined Li and Henry's LCOM using a graph-theoretic representation. Their definition is :

"Consider a class X with  $I_x$  being the set of attributes and  $M_x$  being the set of methods. Construct a simple unidirected graph  $G_x(V,E)$  where :

vertices  $V = M_x$  and

edges  $E = \{ \langle m,n \rangle \text{ is an element of } V \times V : \text{ exactly those vertices (methods) are connected which share at least one attribute of X} \}$

Hence,  $LCOM(X) = \text{number of connected components of } G.$ ” [6].

## 2.4 Redefinition of LCOM by Chidamber and Kemerer

Chidamber and Kemerer have presented a re-definition of LCOM as :

“Let  $C_1$  be a class with methods  $M_1, M_2, \dots, M_n$ . Let.  $\{I_i\}$  be the set of attributes used by  $M_i$ .

Let  $P = \{ (I_i, I_j) / I_i \cap I_j = \emptyset \}$  and

$Q = \{ (I_i, I_j) / I_i \cap I_j \neq \emptyset \}$

If  $\{I_1\} = \{I_2\} = \dots = \{I_n\} = \emptyset$ , then  $P = \emptyset$ .

Hence,  $LCOM = \begin{cases} |P| - |Q|, & \text{if } |P| > |Q| \\ 0 & , \text{ otherwise} \end{cases}$  [7],

where  $P$  is the number of method pairs which access no common attribute of  $C_1$  and  $Q$  is the number of method pairs which access at least one common attribute of  $C_1$ . Hence, if the cardinality of  $P$  is greater than that of  $Q$ , LCOM is given by the difference of the number of method pairs without shared attributes and the number of method pairs with shared attributes. One flaw in this definition is that classes with widely different cohesion could have the same LCOM value.

The two LCOM definitions that are most widely used are Chidamber and Kemerer’s redefinition [7] (which we will call LCOM1) and Li and Henry’s definition [4,5] (which we will call LCOM2).

Some of the variations in measuring LCOM involve the inclusion or exclusion of inheritance and the inclusion or exclusion of the constructor member function. Using these variations, eight LCOMs can be identified [8]. We consider these eight variations in this paper. The variations are :

- (1) LCOM1-1 : the revised Chidamber and Kemerer definition, with consideration of inheritance and constructor.
- (2) LCOM1-2 : the revised Chidamber and Kemerer definition, with consideration of inheritance but not constructor.
- (3) LCOM1-3 : the revised Chidamber and Kemerer definition, without consideration of inheritance but with constructor.
- (4) LCOM1-4 : the revised Chidamber and Kemerer definition, without consideration of inheritance and without consideration of constructor.
- (5) LCOM2-1 : the Li and Henry definition with consideration of inheritance and constructor.
- (6) LCOM2-2 : the Li and Henry definition with consideration of inheritance but without consideration of constructor.
- (7) LCOM2-3 : the Li and Henry definition without consideration of inheritance but with constructor.
- (8) LCOM2-4 : the Li and Henry definition without consideration of inheritance and without consideration of constructor.

Using these different definitions of LCOM results in different measured LCOM values. Hence the question arises if there is a definition/implementation that is the most useful measure of cohesion. We address that question here.

### **3. Experiment Design**

To make study of cohesion as straightforward as possible, we want to determine a minimal number (ideally one) of LCOM metrics that best represent the cohesiveness of the object classes  $O_i$ . Thus, we studied the relative discriminatory power of each of the eight variations on LCOM.

Etzkorn et al. [8] have studied a set of eighteen object classes chosen from three independent C++ GUI packages. The eight variations of the LCOM metric (LCOM1-1 to LCOM2-4) for these object classes were calculated using the PATRicia system [9, 10, 11]. Also, the object classes were rated for cohesiveness by seven highly experienced domain experts. The experts categorized each object class using a value from non-cohesive (0%) to acceptably cohesive (100%).

Each set of the eight LCOM measures for an object class can be considered to form an eight-dimensional feature vector. Using the approach of principal component analysis, we can represent the feature vectors in a new  $p$ -dimensional space, where  $p \leq 8$ .

### 3.1. Principal Component Analysis

Principal component analysis [12] is typically used to reduce the dimensionality and/or to extract new uncorrelated features from the original data. Principal component analysis involves an eigen analysis on a covariance matrix. If the input data is represented as a matrix  $\mathbf{X}$  of 'n' rows and 'm' columns :

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix},$$

then, the sample mean  $\mu_j$  is computed for each column, where

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}, \quad \text{for } j = 1, 2, \dots, m.$$

Therefore,  $\mathbf{X}$  can be centered to form  $\mathbf{X}^*$  :

$$\mathbf{X}^* = \begin{bmatrix} x_{11} - \mu_1 & x_{12} - \mu_2 & \dots & x_{1m} - \mu_m \\ x_{21} - \mu_1 & x_{22} - \mu_2 & \dots & x_{2m} - \mu_m \\ \vdots & & & \\ x_{n1} - \mu_1 & x_{n2} - \mu_2 & \dots & x_{nm} - \mu_m \end{bmatrix}$$

The sample covariance matrix  $\mathbf{R} = (1/n)[\mathbf{X}^*]^T\mathbf{X}^*$  is then computed. An eigen analysis on the covariance matrix  $\mathbf{R}$  yields a set of positive eigen values  $\{\lambda_1, \lambda_2, \dots, \lambda_m\}$  [12]. If the eigen values are sorted in descending order (i.e.,  $\lambda_1 > \lambda_2 > \dots > \lambda_m$ ), their corresponding eigen vectors,  $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$ , are the principal components. The first principal component retains the most variance; if the feature vectors are projected onto the first principal component, more variance will be retained than if the vectors are projected onto any other principal component. The second component retains the next highest residual variance, and so on. A smaller eigen value contributes much less weight to the total variance, hence if the feature vectors are projected onto a subset of principal components, omission of later components tends to introduce less classification error than if earlier components are omitted. In many cases, the first few components can retain nearly all of the variance, enabling satisfactory classification. If the ‘ $d$ ’ most significant principal components are selected for projection of the data, then the variance retained by this approximation is [12] :

$$\text{Variance} = \frac{\sum_{i=1}^d \lambda_i}{\sum_{i=1}^n \lambda_i} \quad (1)$$

## 4. Results

For the set of eighteen C++ classes (from the three GUI packages), the eight LCOM metrics were measured using the PATRicia system. This data (which can be considered as a set of eighteen data points in eight-dimensional space) was normalized and the data was centered before computation of covariance. The principal component analysis produces the following components in descending eigen value order (Table 1) :

Principal comp. #	Component vector	Eigen value
1	(.3363, .3341, .3426, .3417, .3757, .3589, .3765, .3594)	0.45105
2	(-.4006, -.4309, -.2171, -.2468, .2036, .01205, .6243, .3381)	0.02109
3	(.03590, .0407, -.3444, -.3344, .6334, .4676, -.2688, -.2727)	0.01107
4	(-.20809, -.4194, .4205, .3943, .5023, -.2844, -.1044, -.3228)	0.000314
5	(.32208, .2060, -.0066, -.2343, .04861, -.3066, .5745, -.6102)	0.0001706
6	(-.3478, -.06706, .1263, .2337, -.3649, .6497, .2165, -.4490)	0.0000444
7	(-.4676, .5392, -.4617, .4424, .1524, -.2289, .06406, -.04219)	0.0000174
8	(.4882, -.4326, -.5574, .4996, -.07030, .05291, .06482, -.04539)	0.000000926

Table 1 : The principal components and their eigen values

Using the relation (1) above, we have :

<u># of components</u>	<u>% variance retained</u>
1	93.23%
2	97.59%
3	99.88%
4	99.95%

Thus, projection of the object class feature vectors onto the first principal component retains 93.23% of the total variance and projection onto the first two principal components retains 97.59% to the total variance.

Examination of the first eigen vector reveals that LCOM2-1 and LCOM2-3 are more significantly weighted in the projection, and thus they contribute more than the other factors. We can also examine individual features (i.e., LCOM metrics) to determine if projection onto a subset of LCOM measures can retain a sufficient amount of variance. The dataset's total variance is

$$\text{Variance} = \frac{1}{(n-1)} \sum_{i=1}^N (X_i - \bar{X})^2,$$

where  $X_i$  is a row vector corresponding to each row of the matrix  $\mathbf{X}$  (i.e.,  $X_i$  is the LCOM feature vector for object class  $O_i$ ),  $\bar{X} = [\mu_1 \ \mu_2 \ \dots \ \mu_m]$  and  $(X_i - \bar{X})$  is the Euclidean distance between  $X_i$  and  $\bar{X}$ .

For the normalized dataset, the total variance was determined to be 0.512232. Computation of the variance of each of the eight LCOM metrics reveals that LCOM2-3 and LCOM2-1 retain more of the total variance than the rest. LCOM2-3 accounts for 15.1%.of the

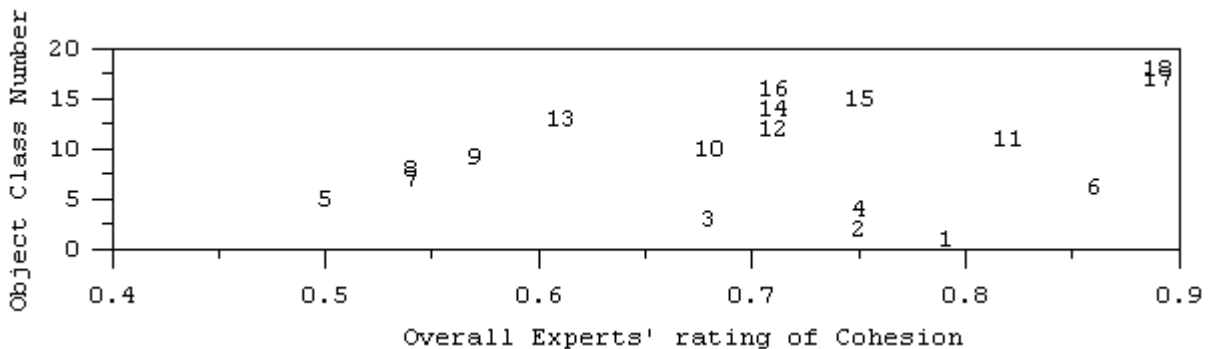


Fig.1. Average Experts' ranking for Cohesion for the 18 object classes

variance of the entire data set and LCOM2-1 accounts for 14.2%. Thus by considering only individual variances, LCOM2-1 and LCOM2-3 are the most significant features. When the dataset is projected onto LCOM2-1 and LCOM2-3, the variance retained is 0.150494 (29.3%).

We also compared the LCOM measures to the evaluation of experts. Comparison of each of the eight LCOM values for the eighteen classes with the experts' cohesion ratings for the classes suggests there is a closer correlation for LCOM2-1 and LCOM2-3 with the experts' ratings. Figure 1 shows the overall mean experts' ranking of cohesion for the 18 object classes. Figures 2 and 3 show the object classes ranked according to their LCOM2-1 and LCOM2-3 values, respectively. Figures 2 and 3 show object class 7 to be an outlier (the LCOM value is high). The experts have also rated object class 7 as an outlier (they assigned a low cohesion value). The experts have rated object classes 6, 17, and 18 as being the most cohesive, and from Figures 2 and 3 it can be observed that these classes have a low LCOM value. The other object classes (as rated by the experts) are quite cohesive, and this is indicated by the relatively low LCOM2-1 and LCOM2-3 values.

However, there is some disagreement between these LCOM measures and the experts' cohesion assessment of object class 5. The experts rate object class 5 as the least cohesive, while LCOM2-1 and LCOM2-3 rank it as less cohesive than the vast majority of object classes, but not as non-cohesive as object class 7. The reason for this is that a few of the

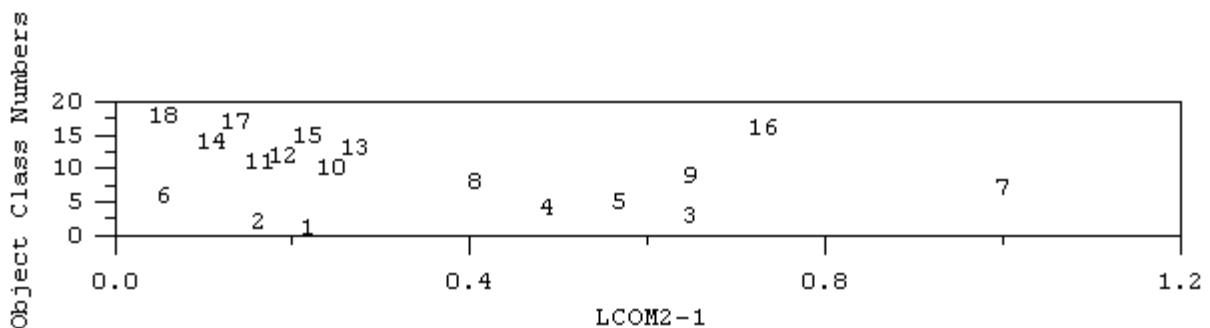


Fig.2. Ranking of the C++ object classes based on LCOM2-1 values. Higher values indicate less cohesion.

experts have rated object class 5 as being very poorly cohesive, while other experts ranked it only poorly cohesive. Thus, the former experts are consistent with LCOM. Perhaps not all of the experts were evaluating cohesion according to the same criteria.

Another object class that shows some discrepancy is object class 8. The experts rate object class 8 as fairly non-cohesive. The LCOM2-3 measure also rates object class 8 as very non-cohesive, whereas the LCOM2-1 measure for object class 8 indicates that it is only moderately non-cohesive. LCOM2-1 indicates a higher cohesion for class 8 because it takes into account inherited variables, whereas LCOM2-3 does not.

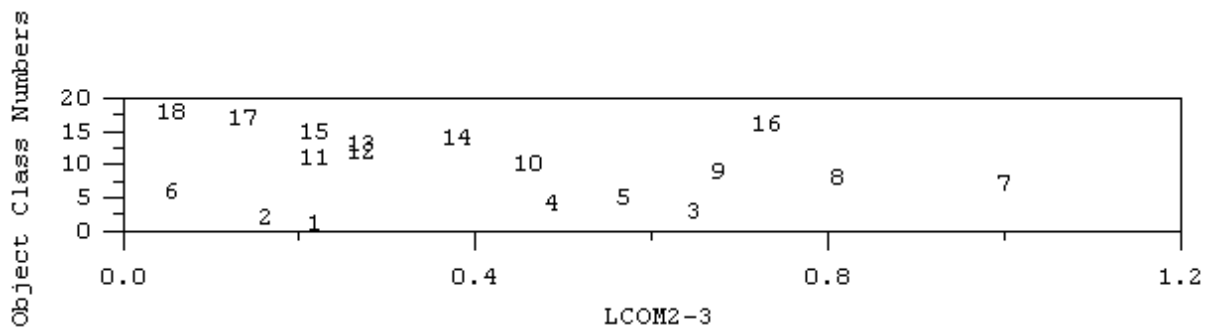


Fig.3. Ranking of the object classes based on LCOM2-3 values. Higher values indicate less cohesion.

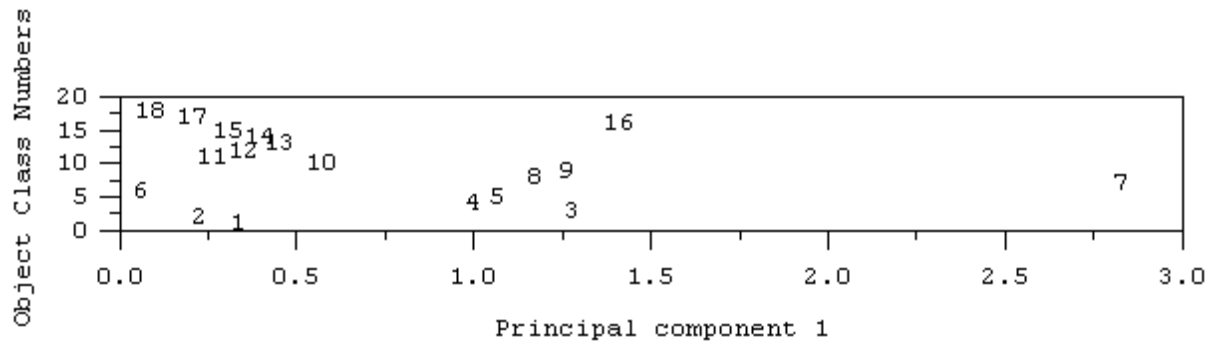


Fig. 4. Projection of the 18 object classes onto the first principal component. Higher values indicate less cohesion.

Figure 4 shows the projection of the eighteen object classes onto the first principal component. Figure 5 shows a projection of the classes onto LCOM2-1 and LCOM2-3. These two plots tend to organize the object classes similarly. Object class 7 is an outlier in both the plots, i.e., its lack of cohesion is very high. Object classes 18 and 6 are shown to have high cohesion in both the plots. The relative positions of the other object classes are almost identical in both the plots. There is a slight discrepancy in the ratings of object classes 8 and 14. Principal component analysis rates these object classes as less cohesive whereas the projection onto the dominant LCOMs rates these object classes as more cohesive than the object classes around them. This suggests that the two predominant LCOM measures (LCOM2-1 and LCOM2-3) as obtained from the principal component analysis on the entire dataset are sufficient to represent the cohesion for the object classes.

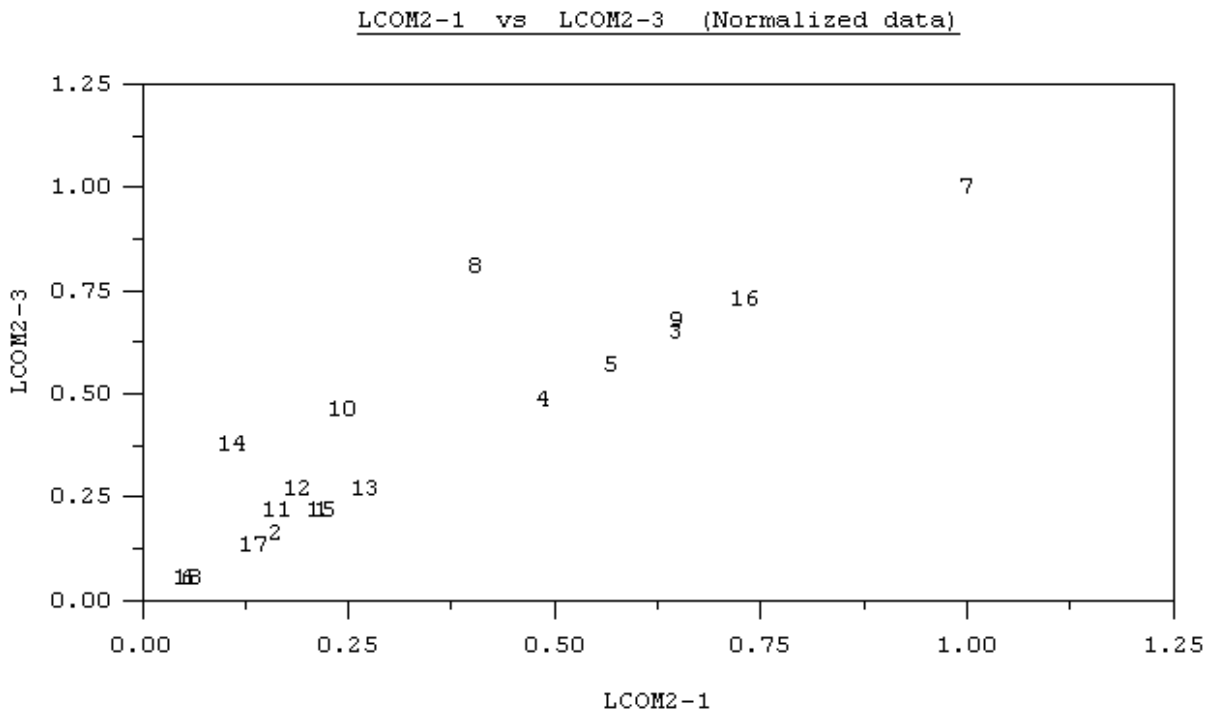


Fig. 5. Projection of the 18 object classes onto LCOM2-1 and LCOM2-3. Higher values indicate less cohesion.

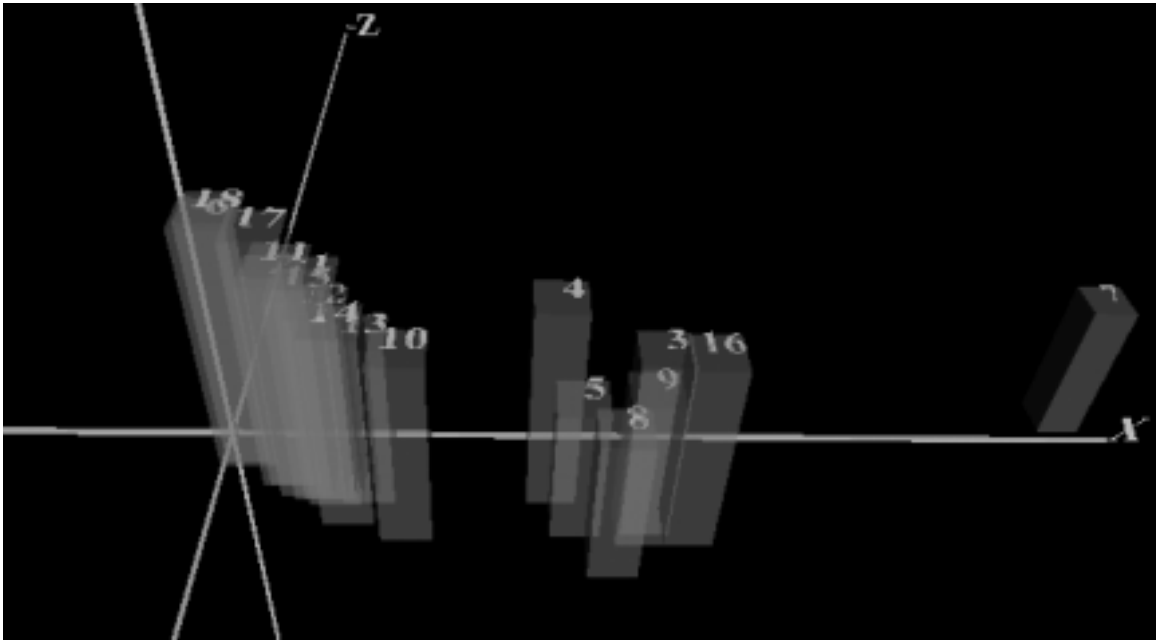


Fig. 6. A 3D plot showing the 18 object classes projected onto the first two principal components. The x-axis shows the first principal component direction while the z-axis shows the second principal component direction. Height demonstrates average expert ratings. Each column shows the object class label.

Fig. 6 shows a 3D plot of the eighteen object classes projected onto the first two principal components (taken along the  $x$  and  $z$  axes respectively). Each block represents an object class (as numbered) and the height of each block indicates the overall experts' rating of cohesion for the object class. Object classes 18, 6 and 7 have the maximum height, and hence are the most cohesive among the 18 object classes as rated by the experts. Object class 7 has a considerably smaller height, suggesting that the experts have rated object class 7 as being quite non-cohesive. The positions of the boxes represents the relative cohesion ratings for the object classes as obtained from the principal component analysis – object class 7 has the least cohesion among the eighteen object classes, and object classes 6 and 18 are the most cohesive.

## 5. Conclusion

Through principal component analysis on a set of 18 C++ object classes, we examined the capability of individual LCOM measures or combinations of LCOM measures to adequately express cohesion. Projecting the object class feature vectors onto the first two principal components retained upto 97.59% of the total variance, hence two components are sufficient to represent the entire dataset with less error. We also compared the LCOM measures to experts' ratings of cohesion. It appears that LCOM2-1 and LCOM2-3 are the most powerful LCOM measures based on their maintenance of variance in the set. Thus the principal component analysis can be used to reduce the dimensionality of a dataset, and used to represent the entire dataset with lesser loss of information from the original dataset.

## Acknowledgements

We wish to thank the National Science Foundation and Acxiom Corporation for providing partial support for our research.

## References

- [1] McCabe, T.J. A complexity measure, *IEEE Transactions on Software Engineering* 2(4):308-320, 1976.
- [2] Halstead, M.H. *Elements of Software Science*, Elsevier, North-Holland, New York, 1977.
- [3] Chidamber, S.R, and C.F. Kemerer. Towards a metric suite for object-oriented design, *Proceedings : OOPSLA '91*, Phoenix, AZ, July 1991, pp. 197-211.
- [4] Li, W., and S. Henry. Maintenance metrics for the object-oriented paradigm. *Proceedings of the First International Software Metrics Symposium*, Baltimore, MD, May 1993, pp. 52-60.
- [5] Li, W., S. Henry, D. Kafura, and R. Schulman. Measuring Object-oriented design. *Journal of Object-Oriented Programming*, Vol. 8, No. 4, July 1995, pp. 48-55.

- [6] Hitz, M., and B. Montazeri. Chidamber and Kemerer's metric suite : A Measurement Theory Perspective, *IEEE Transactions on Software Engineering*, Vol. 4, April 1996, pp. 267-271.
- [7] Chidamber, S.R, and C.F. Kemerer. A metrics suite for object-oriented design, *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, June 1994, pp. 476-493.
- [8] Etzkorn, L.H., C.G. Davis, and W. Li. A practical look at the Lack of Cohesion in Methods metric, *Journal of Object-Oriented Programming*, Vol. 11, No. 5, Sept. 1998, pp. 27-34.
- [9] Etzkorn, L.H., and C.G. Davis. Automated Object-oriented reusable component identification, *Knowledge-Based Systems*, Vol. 9, Issue 8, Dec. 1996, pp. 517-524.
- [10] Etzkorn, L.H., C.G. Davis, L.L. Bowen, D.B. Etzkorn, L.W. Lewis, B.L. Vinz, and J.C. Wolf. A knowledge-based approach to object-oriented legacy code reuse, *Proceedings of the Second IEEE International Conference on Engineering of Complex Computer Systems*, Montreal, Canada, October 1996, pp. 493-496.
- [11] Etzkorn, L.H., A metrics-based approach to the automated identification of object-oriented reusable components : a short overview. *Proc., OOPSLA 1995 Doctoral Symposium*, Austin, TX, October 1995 .
- [12] P.C. Wong, R.D. Bergeron. Multivariate visualization using metric scaling. *Proc., Visualization 97*, Phoenix, October 1997, pp. 111-118.