Arab J Sci Eng (2017) 42:3551–3561
DOI 10.1007/s13369-017-2588-x

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Measuring Object-Oriented Class Cohesion Based on Complex Networks

Aihua Gu[1,2] · **Xiaofeng Zhou**[1] · **Zonghua Li**[1] · **Qinfeng Li**[1] · **Lu Li**[2]

Received: 1 November 2016 / Accepted: 8 May 2017 / Published online: 20 May 2017

Abstract Class cohesion has an immediate impact on maintainability, modifiability and understandability of the software. Here, a new metric of cohesion based on complex networks (CBCN) for measuring connectivity of class members was developed mainly relying on calculating class average clustering coefficient from graphs representing connectivity patterns of the various class members. In addition, the CBCN metric was assessed with theoretical validation according to four properties (nonnegativity and normalization, null and maximum values, monotonicity, cohesive modules) of the class cohesion theory. Based on data comparison with existing seventeen typical class cohesion metrics of class cohesion for a system, the CBCN metric was superior to others. Applying the CBCN metric to three open source software systems to calculate class average clustering coefficients, we found that understanding, modification and maintenance of classes in an open software system could be likely less difficult compared with those of others. Three open software systems have power-law distributions for the class average clustering coefficient, which makes possible the further understanding of the cohesion metric based on complex networks.

Keywords Object-oriented class · Software quality · Class cohesion metric · Complex networks · Software metrics

✉ Aihua Gu
guaihua1978@163.com

[1] School of Computer and Information Science, Hohai University, Nanjing 210098, China

[2] School of Information Engineering, Yancheng Teachers University, Yancheng 224002, China

1 Introduction

General tasks of software engineers include the development of methods which produce applications with high quality, stability and easy maintenance [1]. For analyzing as well as describing complex object-oriented software architectures, and reflecting their design, the software architecture should be modeled and analyzed at the class level [2].

The cohesion of class is an important indicator, and the notion of cohesion has been studied for a long time, starting with the classical work of Yourdon and Constantine [3,4]. Cohesion constitutes a general topic to achieve component cohesion based on component reuse in the research field, and also related to attractive software properties as well as effective modularity. For instance, understanding, modification and maintenance are much easier for a module with high cohesion, compared with one possessing lower cohesion. Therefore, cohesion can predict quality features of object-oriented software programs, including understandability, modifiability and maintainability. Consequently, it is an important topic to be studied. A less cohesive module is likely to be more difficult to understand, modify and maintain in comparison with a highly cohesive module [5,6].

Some evaluation methods have been developed to describe cohesion characteristics by including textual similarity measures or measures of co-evolution [7–9]. The metrics used in this study have been thoroughly defined in previous reports [1,10–24] in Table 1. Al Dallal further detailed the structural cohesion metrics [26–31]. The structural cohesion metrics of a class rely on referencing of characteristics of the methods within it, they could be well understood, and some have the ability to measure the cohesion for the actual object-oriented software development. However, other structural feature can have some defects. The metrics CC, SCOM, CBMC, and ICBMC do not comply with the critical cohesion feature

Table 1 Metrics of class cohesion in this paper

Class cohesion metric	Definition and/or Formula
The lack of cohesion in methods [10]	(LCOM1) LCOM1 = Number of method pairs with no common properties
LCOM2 [11]	P = Number of method pairs with no common properties. Q = Number of method pairs with common properties. LCOM2 = $\begin{cases} P-Q & \text{if } P-Q \geq 0 \\ 0 & \text{otherwise} \end{cases}$
LCOM3 [12]	LCOM3 = Number of linked graph components representing a given technique, forming a node and sharing ≥ 1 property as an edge
LCOM4 [13]	As with LCOM3, with more edges used to reflect the method invocations
LCOM5 [14]	LCOM5 = $(a-kl)/(l-kl)$, where k is the method number, l represents the attribute number, and a is the summation of the number of distinct attributes determined via a given technique in a class
Class cohesion (CC) [15]	CC = Ratio of summation of similarities between the considered method pairs to total number of method pairs. Similarity between techniques i and j was derived as follows: $Similarity\ (i,j) = \left\|I_i \cap I_j\right\| / \left\|I_i \cup I_j\right\|$, where I_i and I_j represent the properties of methods i and j, respectively
Class cohesion metric (SCOM) [16]	SCOM = Ratio of summation of similarities between the method pairs to total number of method pairs. Similarity between techniques i and j can be obtained as: $Similarity\ (i,j) = (\left\|I_i \cap I_j\right\| \cdot \left\|I_i \cap I_j\right\|)/(\min\ (\|I_i\| \cap \left\|I_j\right\|) \cdot l)$, where l is the number of attributes
Coh [17]	Coh = a/kl where a, k and l were defined as above
Tight class cohesion (TCC) [18]	TCC = relative number of directly linked method pairs, i.e., directly connected to an attribute
Loose class cohesion (LCC) [18]	LCC = Relative number of directly/transitively connected method pairs (methods are transitively connected when transitively linked to an attribute
Degree of cohesion-direct (DC_D) [19]	(DC_D) = Relative number of directly connected method pairs (as described for TCC or methods directly/transitively invoking the same technique
Degree of cohesion-indirect (DC_I) [19]	(DC_I) = Relative number of directly/transitively linked method pairs (as described for LCC)
Cohesion based on member connectivity (CBMC) [20]	CBMC $(G) = F_C\ (G) \times F_S\ (G)$, where $F_C\ (G) = \|M\ (G)\| / \|N\ (G)\|$, $M\ (G)$ and $N\ (G)$ are, respectively, numbers of gule and non-special methods in graph G, $F_S\ (G) = [\sum_{i=1}^{n} \text{CBMC}\ (G^i)]/n$; n represents the child node number. Gule methods constitutes the minimum group of methods, whose removal results in disconnected graph
Improved cohesion based on member connectivity (ICBMC) [21,22]	ICBMC $(G) = F_C\ (G) \times F_S\ (G)$, where $F_C\ (G) = \|M\ (G)\| / \|N\ (G)\|$; $M\ (G)$ represents the number of edges in the cut set of G, $N\ (G)$ is the number of non-special methods in graph G by that of attributes, $F_S\ (G) = [\sum_{i=1}^{2} \text{CBMC}\ (G^i)]/2$; the cut set represents the minimum set of edges, whose removal results in disjointed graph
OL_n [23]	OL_n = Average strength of attributes (strength of an attribute reflects the average strength of methods referencing the latter). The strength of a method is first set to 1 and computed in each iteration; n is the number of iterations utilized in OL computation.
Low-level design similarity-based class cohesion (LSCC) [24]	LSCC = $\begin{cases} 0 & \text{if } k=0 \text{ or } l=0 \\ 1 & \text{if } k=1 \\ \frac{\sum_{i=1}^{l} x_i\ (x_i-1)}{lk\ (k-1)} & \text{otherwise} \end{cases}$ where k and l represent the number of methods and attributes, respectively; x_i represents the number of methods referencing the attribute i
Path connectivity class cohesion (PCCC) [1]	PCCC = Ratio of the number of simple paths in reference graph G to that in the corresponding graph FG

of monotonicity [25]. LCOM5 and Coh reflect the counting of various characteristics evaluated in all methods of the given class. Other metrics, such as LCOM3, LCOM4, CBMC, ICBMC and OL_n, reflect connectivity among class members. The metric PCCC is derived from the count of likely paths representing class member connectivity. Other metrics, including TCC, LCC, LCOM1, LCOM2, DC_D and DC_I, are derived from the count of method pairs sharing given features. Meanwhile, SCOM, CC and LSCC derived from a count of properties common in each method pair. In summary, all the above methods cannot well support class cohesion measuring, and this motivates us to develop a more practical metric to measure the cohesion, which can give effective suggestions for software developers.

Complex networks have been described in multiple research fields, for the discovery of statistical features regarding the small-world feature [32] of short links between two given nodes and very clustered connections, and the scale-free feature [33] of complex networks. As a consequence, analysis and control of the dynamic behavior in complex networks have become a hot topic in various disciplines. At the beginning of the current century, the software system has been decomposed into a series of collections of entities (e.g., class, subroutines, components) by statistical physicists and other researchers in the field of complex systems. It was found that the structure has the statistical characteristics of "scale-free" and "small world" of complex networks, attracting more attention in the field of software engineering [34–41]. Class dependence network is a graph depicting complex topology of network interconnections. It was found that cohesion properties contain some statistical properties of complex networks. Because the property of the clustering coefficient in complex networks tends to define the neighboring nodes in a network [32], the metric basis for calculating the class average clustering coefficient should be indicated for measuring connectivity level among various class members. However, based on actual class dependence graphs, the property of the clustering coefficient cannot fully satisfy class cohesion metrics. For example, when the average clustering coefficient in the actual class dependence graph (which has some edges) is zero, the index of clustering coefficient could compare cohesion values for such graphs, which have different edges. It is not reasonable to measure the cohesion value only using the clustering coefficient, and network connectivity is also proposed for such measurements, with the clustering coefficient in this paper.

The main contributions described by this article are:

1. Introduction of a connectivity-based metric, CBCN based on complex networks, which satisfies key mathematical cohesion properties.
2. Comparing the values of cohesion of eighteen metrics, the CBCN metric is superior to the other metrics for a simple system.
3. Applying the CBCN metric to three open source software systems and analyzing the effect of descriptive statistics as well as distributions of the class average clustering coefficient, which provides a view for further understanding of the cohesion metric according to the theory of complex networks.

The present article has the following organization: section 2 summarizes preliminaries. Section 3 defines the newly developed CBCN metric and discusses its theoretical validation. Section 3.2 compares the existing metrics with CBCN. Section 4 describes its application in three open source systems. Section 5 provides a conclusion and suggests perspectives.

2 Preliminaries

2.1 Class Cohesion Metric Properties

According to Briand [25], class cohesion metrics carry these features: (1) nonnegativity and normalization [0, Max], allowing easy comparison between various classes; (2) null and maximum values (no cohesive interactions, 0, all possible interactions within the class present, maximum); (3) monotonicity (even if the module is added cohesive interactions, the cohesion of the module cannot be decreased); and (4) cohesive modules (merging two unrelated modules into one cannot enhance the cohesion of the module). Thus, for 2 classes, e.g., $c1$ and $c2$, the cohesion of the merged class c' must adhere to the following conditions:

$$cohesion\ (c') \leq \max\{cohesion\ (c1), cohesion\ (c2)\}.$$

2.2 Clustering Coefficient and Network Connectivity

Clustering coefficient points to the tendency for the neighbors of a given node to be their neighbors in a network [32]. Therefore, it is employed to evaluate node tendency to cluster. The clustering coefficient is defined as follows: in a network with n nodes and e edges, if for each node i there are d_i other nodes connected to it and e_i links between those d_i nodes, the coefficient for node i would be:

$$C_i = 2e_i/d_i\ (d_i - 1) \tag{1}$$

Average value for a network is:

$$C = 2\sum_{i=1}^{n} e_i/d_i(d_i - 1)n \tag{2}$$

The network connectivity for a network is derived as:

$$\mathrm{NC} = 2e/n\ (n-1) \tag{3}$$

The value of average clustering coefficient of a complete graph is 1, and that of network connectivity is also 1.

3 Measurement of Class Cohesion Based on Complex Networks

3.1 Dependence Relations in Class and Dependence Complex Network

In the object-oriented software system, a software system is composed of a series of related classes. One class (c) describes a class in the software system, $c = (A, M)$, where $A = \{A_1, A_2, \cdots, A_a\}$ is a set of attributes, and describes attributes contained in the class c. $M = \{M_1, M_2, \cdots, M_m\}$ is a set of methods, and describes m methods comprised by the class c. Thus, dependence relations for a given class comprise: dependence relations between attributes and methods, among attributes and among methods. Based on such dependence relations, a class dependence network $\langle N, E\rangle$ can be constructed, where N and E are sets of nodes and edges, respectively. $N = N_A \cup N_M$, where N_A and N_M are sets of nodes representing the set of attributes A and methods M, respectively.

3.1.1 A Dependence Network Between Attributes

$\langle N_A, E_A\rangle$, where N_A is a set of attributes, E_A is the dependence relations between the attributes. If $A_i, A_j \in N_A$, A_i and A_j are referenced by the same and common method directly, then $\langle A_i, A_j\rangle \in E_A$.

3.1.2 A Dependence Network Between Methods

$\langle N_M, E_M\rangle$, where N_M is a set of methods, E_M is the dependence relations among the methods. If $M_i, M_j \in N_M$, and there is a dependence relation between M_i and M_j (M_i references M_j directly, or M_j references M_i directly), or M_i and M_j reference a common attribute directly, then $\langle M_i, M_j\rangle \in E_M$.

3.1.3 A Dependence Network Between Methods and Attributes

$\langle N, E_{MA}\rangle$, where $N = N_A \cup N_M$, E_{MA} is the dependence relations between the methods and attributes directly. If $A_i \in N_A$, $M_j \in N_M$, and there is a dependence relation between A_i and M_j (M_j references a common attribute A_i directly), then $\langle A_i, M_j\rangle \in E_{MA}$.

A class dependence network is composed of three dependence networks, which are, respectively, between attributes, between methods, and between attributes and methods.

3.2 Method of Measuring the Class Cohesion

A class is composed of attributes and methods. There are three kinds of dependencies among attributes and methods. Therefore, the cohesive degree of the class can also be measured from these three aspects. A class dependence network is a description about the various components and dependence relations in the class. The edge of a dependence network means dependence relations. Consequently, the metrics for measuring cohesion in the class could be done in accordance with a class dependence network.

It is known that a small-world network exhibits a high clustering coefficient [32], which indicates the aggregation of given network nodes. The larger the mean clustering coefficient of the class dependence network is, the tighter the connection is and the higher the cohesion is in the class. Cohesion is a measurement of the togetherness of the elements in a given module. So our metric should be not related to class size, and a kind of connectivity cohesion metric can be put forward based on the complex network, which is called CBCN. Thus, formally, for any class c, n is the number of the methods and attributes within the class c ($n = |N|$), e is the number of the edges within the class c ($e = |E|$), C is the average clustering coefficient of the class c ($C = \sum_{i=1}^{n} C_i/n = 2\sum_{i=1}^{n} e/d_i\ (d_i - 1)/n$). If there is $\sum_{i=1}^{N} C_i = 0$ in the network, the class average clustering coefficient in the formula (3) could be substitute by the network connectivity NC, $\mathrm{NC} = 2e/(n \times (n-1))$. So the model of the CBCN metric is defined as follows:

$$\mathrm{CBCN}\ (c) = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ 1 & \text{if } (n = 2) \text{ and } (\langle N_i, N_j\rangle \in E), \\ C & \text{if } (n > 2) \text{ and } (C \neq 0), \\ \mathrm{NC} & \text{if } (n > 2) \text{ and } (C = 0). \end{cases} \tag{4}$$

3.3 Theory Validation

The CBCN was validated based on the features of a class cohesion metric discussed in Sect. 2.1.

Properties CBCN 1 CBCN has the nonnegativity and normalization properties.

Proof CBCN minimum is zero with class reference graph having no edges. Therefore, no method references an feature, no method references a method, and no attribute references an attribute. Maximum CBCN is achieved with an edge

between each node unless there is only one node. Consequently, CBCN varies from 0 to 1, satisfying nonnegativity and normalization conditions.

If $\sum C_i \neq 0$, then

$$0 \leq C_i = 2e_i/(d_i \times (d_i - 1)) \leq 1 \tag{5}$$

$$0 \leq C = \left(\sum_{i=1}^{N} C_i\right) \Big/ n \leq 1 \tag{6}$$

If $\sum C_i = 0$, then

$$0 \leq 2e/(n \times (n-1)) \leq 1 \tag{7}$$

CBCN metric belongs to the interval [0, 1], satisfying the following formula:

$$0 \leq \text{CBCN} = \sum_{i=1}^{n} C_i/n \quad \leq 1 \tag{8}$$

$$0 \leq \text{CBCN} = \text{NC} \leq 1 \tag{9}$$

As a result, the CBCN metric ranges over the interval [0, 1], and it therefore satisfies the nonnegativity and normalization property.

CBCN 2 CBCN complies with the null and maximum values property.

Proof In a class comprising a set of methods and attributes, if the class lacks interactions, CBCN would be equal to zero. Conversely, a value of 1 would be obtained, when the class displays all possible interactions. This indicated that CBCN complies with the null and maximum value property. With a value of zero for CBCN, there are no nodes or edges in the class reference graph. On the contrary, if the value of CBCN is 1, there is only one node in the class dependence network or the network is a complete graph. Hence, the CBCN metric satisfies the null and maximum values property.

Properties CBCN 2 CBCN has monotonicity.

Proof Addition of an edge to the class dependence network results in no reduction in CBCN, indicating monotonicity.

If $\sum C_i \neq 0$, two unconnected nodes i and j in the class dependence network, and $\exists\, d_i \geq 2$, $\exists\, d_j \geq 2$. Suppose that one node k ($\exists d_k \geq 2$) has connection with i and j.

$$C_i = 2e_i/(d_i \times (d_i - 1)) \tag{10}$$

$$C_j = 2e_j/(d_j \times (d_j - 1)) \tag{11}$$

$$C_k = 2e_k/(d_k \times (d_k - 1)) \tag{12}$$

If increasing the edge between i and j, then satisfy the following formulas:

$$\begin{aligned} &2\,(e_i+1)/(d_i \times (d_i+1)) - 2e_i/(d_i \times (d_i-1)) \\ &= (2d_i - 2)/(d_i \times (d_i+1) \times (d_i-1)) \\ &> 0 \end{aligned} \tag{13}$$

$$\begin{aligned} &2\,(e_j+1)/(d_j \times (d_j+1)) - 2e_j/(d_j \times (d_j-1)) \\ &= (2d_j - 2)/(d_j \times (d_j+1) \times (d_j-1)) \\ &> 0 \end{aligned} \tag{14}$$

$$\begin{aligned} &2\,(e_i+1)/(d_i \times (d_i+1)) + 2\,(e_j+1)/(d_j \times (d_j+1)) \\ &\quad + 2\,(e_k+1)/(d_k \times (d_k-1)) \\ &\quad > 2e_i/(d_i \times (d_i-1)) + 2e_j/(d_j \times (d_j-1)) \\ &\quad + 2e_k/(d_k \times (d_k-1)) \end{aligned} \tag{15}$$

If $d_i = 1$ and $d_j = 1$, $C_i = 0$ and $C_j = 0$

$$\begin{aligned} &2\,(e_i+1)/(d_i \times (d_i+1)) + 2\,(e_j+1)/(d_j \times (d_j+1)) \\ &\quad + 2\,(e_k+1)/(d_k \times (d_k-1)) \\ &\quad > 0 + 0 + 2e_k/(d_k \times (d_k-1)) \end{aligned} \tag{16}$$

The average clustering coefficient is increased when $\sum C_i \neq 0$. If $\sum C_i = 0$, the network connectivity would be $2e/(n \times (n-1))$. After increasing an edge in the class dependence network, the network connectivity becomes $2(e + 1)/(n \times (n-1))$ which is obviously increased. Therefore, adding a cohesive interaction always increases the CBCN metric value, and the CBCN metric satisfies the monotonicity property.

Properties CBCN 3 The CBCN metric satisfies the cohesive module property.

Proof Merging two independent classes into one implies that the average clustering coefficient and entropy of class c_3 (merging product) do not increase relativity to class c_1 and c_2 (independent classes). Therefore, Max {CBCN (c_1), CBCN (c_2)} $\geq$ CBCN (c_3)}, the class dependence network of c_1 has l_1 nodes, and that of c_2 has l_2 nodes. If the average clustering coefficient of the two class dependence networks are not zero, the average clustering coefficient satisfies the following formulas:

$$\begin{aligned} &\left(\sum_{i=1}^{l_1} C_i + \sum_{j=1}^{l_2} C_j\right) / (l_1 + l_2) \\ &\quad \leq \max\left\{\sum_{i=1}^{l_1} C_i/l_1, \sum_{j=1}^{l_2} C_j/l_2\right\} \end{aligned} \tag{17}$$

$$\begin{aligned} &\left(\sum_{i=1}^{l_1} C_i + \sum_{j=1}^{l_2} C_j\right) \Big/ (l_1 + l_2) \leq \sum_{i=1}^{l_1} C_i/l_1 \\ &\quad \text{or } \left(\sum_{i=1}^{l_1} C_i + \sum_{j=1}^{l_2} C_j\right) \Big/ (l_1 + l_2) \leq \sum_{j=1}^{l_2} C_j/l_2 \end{aligned} \tag{18}$$

$$\sum_{i=1}^{l_1} C_i + \sum_{j=1}^{l_2} C_j \le \sum_{i=1}^{l_1} C_i + l_2 \sum_{i=1}^{l_1} C_i/l_1 \quad \text{or} \quad \sum_{i=1}^{l_1} C_i + \sum_{j=1}^{l_2} C_j \le \sum_{j=1}^{l_2} C_j + l_1 \sum_{j=1}^{l_2} C_j/l_2 \tag{19}$$

If

$$\sum_{i=1}^{l_1} C_i + l_2 \sum_{i=1}^{l_1} C_i/l_1 \ge \sum_{j=1}^{l_2} C_j + l_1 \sum_{j=1}^{l_2} C_j/l_2 \tag{20}$$

$$(1 + l_2/l_1) \times \sum_{i=1}^{l_1} C_i \ge (1 + l_1/l_2) \times \sum_{j=1}^{l_2} C_j \tag{21}$$

$$\sum_{i=1}^{l_1} C_i \ge [(1 + l_1/l_2)/(1 + l_2/l_1)] \times \sum_{j=1}^{l_2} C_j \tag{22}$$

Therefore,

$$\begin{aligned} &\sum_{i=1}^{l_1} C_i + l_2 \sum_{i=1}^{l_1} C_i/l_1 \\ &\ge \sum_{i=1}^{l_1} C_i + (l_2/l_1) \times [(1 + l_1/l_2)/(1 + l_2/l_1)] \times \sum_{j=1}^{l_2} C_j \\ &= \sum_{i=1}^{l_1} C_i + \sum_{j=1}^{l_2} C_j \end{aligned} \tag{23}$$

The formula (23) is same as the formula (18), so if $\sum_{i=1}^{l_1} C_i + l_2 \sum_{i=1}^{l_1} C_i/l_1 \ge \sum_{j=1}^{l_2} C_j + l_1 \sum_{j=1}^{l_2} C_j/l_2$, the formula (17) is proved correct. Similarly, if $\sum_{i=1}^{l_1} C_i + l_2 \sum_{i=1}^{l_1} C_i/l_1 \le \sum_{j=1}^{l_2} C_j + l_1 \sum_{j=1}^{l_2} C_j/l_2$, the formula (17) is also correct.

If the average clustering coefficient of the two class dependence networks are zero, the network connectivity satisfies the following formula:

$$2\,(e_1 + e_2)/(\,(l_1 + l_2) \times (l_1 + l_2 - 1)) < \max\{2e_1/(l_1 \times (l_1 - 1)), 2e_2/(l_2 \times (l_2 - 1))\} \tag{24}$$

$$(e_1 + e_2)/((l_1 + l_2) \times (l_1 + l_2 - 1)) < e_1/(l_1 \times (l_1 - 1)) \quad \text{or} \quad (e_1 + e_2)/((l_1 + l_2) \times (l_1 + l_2 - 1)) < e_2/(l_2 \times (l_2 - 1)) \tag{25}$$

If

$$e_1/(l_1 \times (l_1 - 1)) \ge e_2/(l_2 \times (l_2 - 1)) \tag{26}$$

$$e_1 l_2^2 - e_1 l_2 - e_2 l_1^2 + e_2 l_1 \ge 0 \tag{27}$$

$$\begin{aligned} &e_1/(l_1 \times (l_1 - 1)) - (e_1 + e_2) \\ &/(l_1 + l_2) \times (l_1 + l_2 - 1) \\ &= (2el_1 l_2 + (e_1 l_2^2 - e_1 l_2 - e_2 l_1^2 + e_2 l_1)) \\ &/(l_1 \times (l_1 - 1) \times (l_1 + l_2) \times (l_1 + l_2 - 1)) \\ &> 0 \end{aligned} \tag{28}$$

The formula (28) is same as the formula (25), so if $e_1/(l_1 \times (l_1 - 1)) \ge e_2/(l_2 \times (l_2 - 1))$, the formula (24) is proved correct. Similarly, if $e_1/(l_1 \times (l_1 - 1)) \ge e_2/(l_2 \times (l_2 - 1))$, the formula (24) is also correct.

Therefore, Max {CBCN (c_1), CBCN (c_2)} $\ge$ CBCN (c_3), which means that the CBCN metric satisfies the cohesive module property.

Proving that the CBCN metric satisfies expected mathematical properties increases the chances that it would be a meaningful class cohesion indicator.

4 Contrastive Analysis

The values for the cohesion of eighteen considered metrics of CBCN, PCCC, CBMC, OL2, OL3, LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, Coh, TCC, LCC, DCD, DCI, LSCC, CC and SCOM were assessed, comparing them with the current CBCN. In order to identify faulty classes, empirical evidence that individual or combined metrics are markedly associated with detectable quality features is required. Upon confirmation of such metric, it would represent a potential well-defined cohesion measure. Figure 1 shows Java code examples of classes for comparative analysis from a previous paper [1].

The value of CBCN cohesion is the class average clustering coefficient C, which is calculated by $C = 2\sum_{i=1}^{n} e_i/d_i\ (d_i - 1)n$, if $C > 0$. Otherwise, the value of CBCN cohesion is the network connectivity NC, which is calculated by NC $= 2e/(n \times (n - 1))$. To show the cohesions of the classes in Fig. 1, we calculated the class average clustering coefficient C and the network connectivity NC for each class.

$$\begin{aligned} C\ (\text{Person1}) &= 2\sum_{i=1}^{n} e_i/d_i\ (d_i - 1)n \\ &= (1 + 1 + 1 + 0 + 0)/5 = 0.6 > 0 \end{aligned} \tag{29}$$

$$\text{CBCN (Person1)} = C\ (\text{Person1}) = 0.6 \tag{30}$$

$$\begin{aligned} C\ (\text{Collection}) &= 2\sum_{i=1}^{n} e_i/d_i\ (d_i - 1)n \\ &= (0 + 0 + 0 + 0 + 0)/5 = 0 \end{aligned} \tag{31}$$

$$\begin{aligned} \text{NC (Collection)} &= 2e/n\ (n - 1) \\ &= 2 \times 1/(5 \times (5 - 1)) = 0.1 \end{aligned} \tag{32}$$

$$\text{CBCN (Collection)} = \text{NC (Collection)} = 0.1 \tag{33}$$

$$\begin{aligned} C\ (\text{Person2}) &= 2\sum_{i=1}^{n} e_i/d_i\ (d_i - 1)n \\ &= (1 + 1 + 1 + 1 + 0 + 0)/6 = 0.67 > 0 \end{aligned} \tag{34}$$

$$\text{CBCN (Person2)} = C\ (\text{Person2}) = 0.67 \tag{35}$$

Fig. 1 Java class

```
public class Person1{
        String firstName, lastName;
        Date date;
        void printName() {
            /*code that accesses firtName
              and lastName attributes*/
        }
        void printDate() {
            //code that accesses date attribute
        }
}
```

(a) Person1

```
public class Collection{
        String color, name;
        int value;
        void printColor() {
            //code that accesses color attribute
        }
        void printCurrentDate() {
          //code that does not access any attribute
        }
}
```

(b) Collection

```
public class Person2{
        String firstName, lastName;
        Date date;
        void printName1() {
          /*code that prints first the firstName
            and then the lastName*/
        }
        void printName2() {
          /*code that prints first the lastName
            and then the firstName*/
        }
        void printDate() {
          //code that accesses date attribute
        }
}
```

(c) Person2

```
public class Person3{
        String firstName, lastName;
        Date birth_date;
        void printFirstName() {
            //code that prints the firstName
        }
        void printName() {
            /*code that accesses firtName
              and lastName attributes*/
        }
        void printPersonInfo() {
            /*code that prints lastName
              and birth_date
        }
}
```

(d) Person3

Table 2 Parameters calculated for cohesion based on the CBCN metric

Class	Person1	Collection	Person2	Person3
No. of methods	2	2	3	3
No. of attributes	3	3	3	3
No. of edges	4	1	7	9
C	0.6	0	0.67	0.72
NC	0.4	0.1	0.47	0.6
CBCN	0.6	0.1	0.67	0.72

Table 3 Cohesion values using the considered metrics

Metric	Person1	Collection	Person2	Person3
CBCN	0.6	0.1	0.67	0.72
PCCC	0.12	0.03	0.1	0.11
CBMC	0	0	0	0.33
OL2	0	0	0	0.2
OL3	0	0	0	0.07
LCOM3	2	2	2	1
LCOM4	2	2	2	1
LCOM1	1	1	2	1
LCOM2	1	1	1	0
LCOM5	1	1.67	0.67	0.67
Coh	0.5	0.17	0.55	0.55
TCC	0	0	0.33	0.67
LCC	0	0	0.33	1
DCD	0	0	0.33	0.67
DCI	0	0	0.33	1
LSCC	0	0	0.22	0.22
CC	0	0	0.33	0.28
SCOM	0	0	0.22	0.39

$$\begin{aligned} C\ (\text{Person3}) &= 2\sum_{i=1}^{n} e_i/d_i\ (d_i-1)n \\ &= (1\times 2+2\times 2/(3\times(3-1))\times 2 \\ &\quad +2\times 3/(4\times(4-1))\times 2)/6 \\ &= 0.72>0 \end{aligned} \tag{36}$$

$$\text{CBCN (Person3)} = C\ (\text{Person3}) = 0.72 \tag{37}$$

The results for a simple system are shown based on our CBCN in Table 2.

LCOM measure the lack of cohesion (the value of cohesion gets lower along with increasing value based on LCOM measure), while other metrics measure cohesion itself. Comparison results are shown in Table 3, which presents data derived from Al Dallal [1] except for CBCN. Compared with existing classical metrics, except for PCCC and CBCN, at least two cohesion values of classes are the same in the four values based on other metrics, so those metrics may not measure the cohesion of these classes more effectively. The distribution of the four cohesion values based on PCCC is relatively narrow, and the cohesion value for Person1 class should be lower compared with that of Person3 class, intuitively from Fig. 2 (four class dependence networks are shown in Fig. 2). Therefore, CBCN appears superior to existing metrics in different connectivity patterns of cohesive interactions; therefore, CBCN capacity is a nice feature for the four tiny benchmark examples proposed by Al Dallal.

5 Application in Three Java Open Source Software Systems (JOSSS)

5.1 Experimental Data

Three Java open source software systems (JOSSS) were selected in various domains: Art of Illusion [42], JabRef [43] and GanttProject [44]. Art of Illusion is a 3D rendering, modeling and animation studio system. JabRef is a graphical application for managing bibliographical databases. GanttProject is a project scheduling application featuring resource calendaring, management and importing or exporting (MS Project, PDF, HTML, spreadsheets). The restrictions placed on the choice of these systems were that they (1) are implemented using Java, (2) are relatively large in terms of number of classes, (3) are from different domains, and (4) have available source code and fault repositories. The three JOSSS were employed by Al Dallal and Briand [1,24,26–31] for validating the cohesion metric.

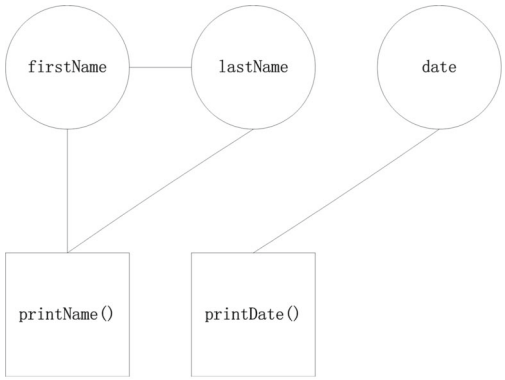
5.2 Experimental Environment

In the three JOSSS, each system is composed of a series of related classes. The dependence relations for a given class comprise: dependence relations between attributes and methods, among attributes, and among methods. A Java tool was developed by our team and applied to automate cohesion measurements based on the CBCN metric. Java source code in the three JOSSS was assessed, extracting required data for model building, and calculating the number of methods and attributes, total number of edges, average clustering coefficients, network connectivities and cohesion values using the CBCN metric for each class in the three JOSSS.

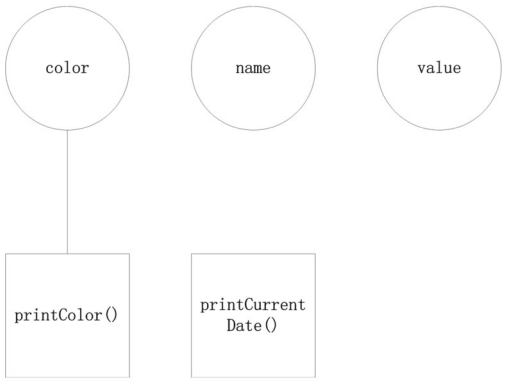
5.3 Experimental Data and Assessment

5.3.1 Descriptive Statistics for Three JOSSS

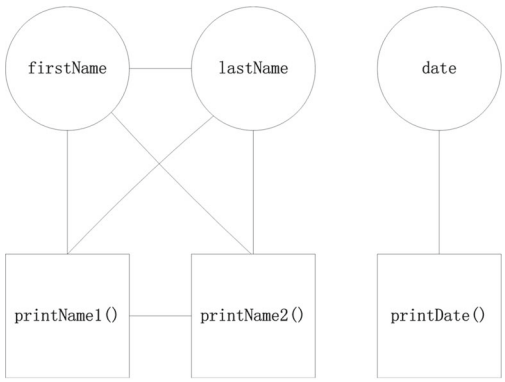
Our cohesion measuring tool was applied to the three JOSSS and calculated the class average clustering coefficients, the network connectivity, cohesion values for each class in the three JOSSS. The ratio of classes whose CBCN value is zero



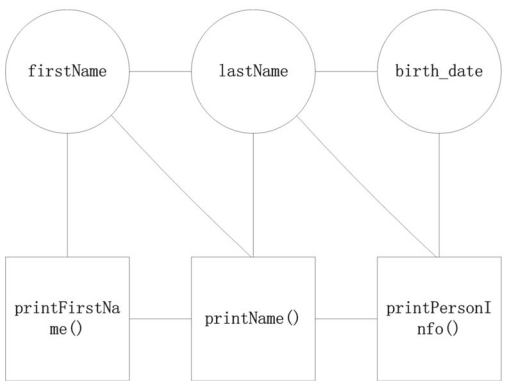
(a) Person1



(b) Collection



(c) Person2



(d) Person3

Fig. 2 Class dependence network of Java class

Table 4 Descriptive statistics of the CBCN metric for the three open source software systems

System	Illusion	JabRef	GanttProject
Min	0	0	0
Max	1	1	1
25%	0.41	0	0
Med	0.68	0.48	0.31
75%	0.81	0.75	0.62
Mean	0.58	0.43	0.34
Std.dev	0.31	0.36	0.34

relative to the number of all classes in the Illusion was 12.4%, that of the JabRef tool was 30.6%, while 37.6% was obtained for GanttProject. Table 4 reports the descriptive statistics for all cohesion measures, with minimum, 25% quartile, mean, median, 75% quartile, maximum value and standard deviation. Expectedly, results in Table 4 showed that some descriptive statistics of the Illusion are different from those of JabRef and GanttProject.

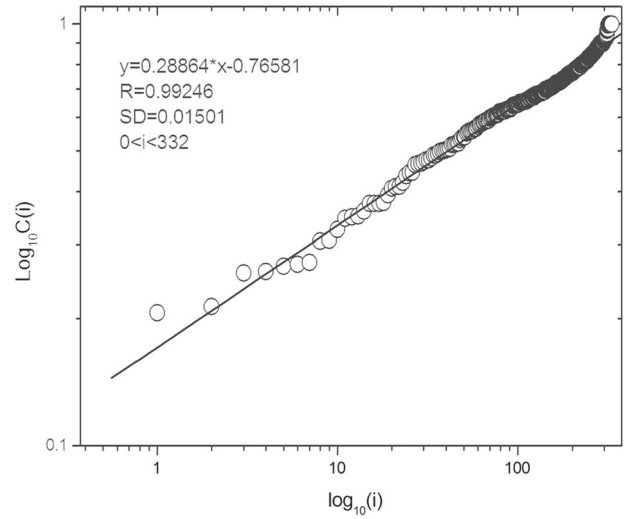
Because a less cohesive module is likely to be more difficult to understand, modify and maintain in comparison with a highly cohesive module [5,6], the classes in the Illusion could be likely easier to understand, modify and maintain in comparison with the classes in the JabRef and GanttProject from Table 4.

5.3.2 Distributions

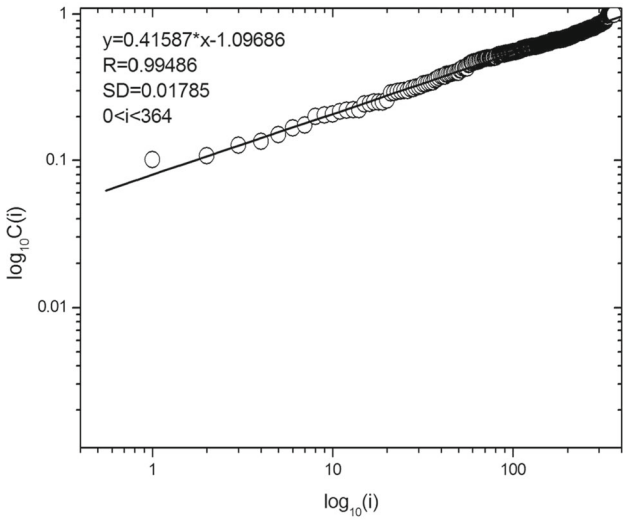
From our data, $C(i)$ indicates the average clustering coefficient of the class c_i. $C(i)$ is a non-normalized integral of the probability distribution $P(i)$, when $P(i) \sim i^{-r}$, $C(i) \sim i^{-r}$. With $C(i) \neq 0$, the gamma exponent was obtained by fitting to the linear range of log–log curves of distributions. Figure 3 shows distributions of the class average clustering coefficient for Illusion, JabRef and GanttProject, respectively.

The Pearson correlation coefficient (R) describes the quality of a given linear fit. As shown in Table 5, when 0.95 is considered the minimum reliable value [40], all distributions in Fig. 3 can be described as power-law. The errors of gamma exponent values were all very small, and gamma exponent values for the distributions of the class average clustering coefficient for JabRef were similar to those obtained for GanttProject.

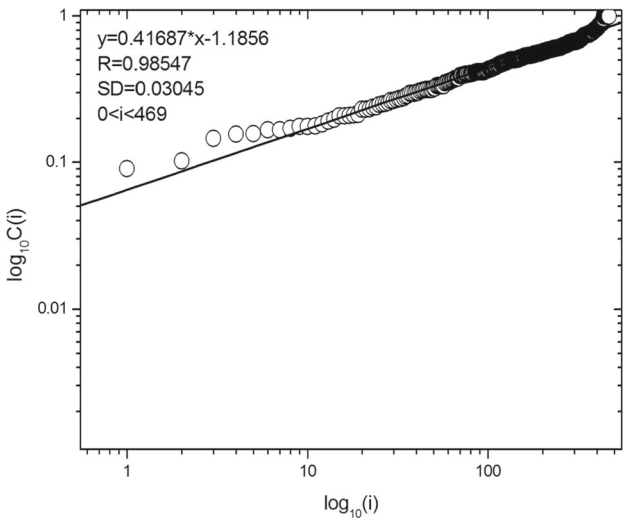
Log–log plot is a two-dimensional graph of numerical data that uses logarithmic scales on both the horizontal and vertical axes. Monomials (relationships of the form $y = ax^k$) appear as straight lines in a log–log graph, with the power and constant term corresponding to slope and intercept of the line, and thus, these graphs are very useful for recognizing these relationships and estimating parameters. Any base can



(a) Illusion



(b) JabRef



(c) GanttProject

Fig. 3 Log–log plot of class average clustering coefficients for three source software systems

Table 5 Gamma exponents with square of the Pearson correlation coefficient

	γ	error of γ	R
Illusion	0.28864	0.00197	0.99246
JabRef	0.41587	0.00223	0.99486
GanttProject	0.41687	0.00333	0.98547

be used for the logarithm, though most common are 10, e and 2. The fact that the distributions presented in Fig. 3 can be well fitted by power laws indicated that cohesion values of the three open source software systems are not all very high or all very low. Class average clustering coefficients could be subject to "scale-free" properties of complex networks when class average clustering coefficients are not zero.

6 Conclusions and Perspectives

A newly developed metric is presented in this work, termed connectivity cohesion metric based on complex networks (CBCN). CBCN which could allow assessment of previous systems for cohesion, adhering strongly to the theory. Through application of the CBCN metric in three Java open source software systems, which well assess the quality of different classes, our data showed that classes in Illusion were likely more understandable, modifiable and maintainable compared with the classes in JabRef and GanttProject. The three systems have power-law distributions for the class average clustering coefficient, which makes possible the further understanding of the cohesion metric based on the theory of complex networks.

Access methods and constructors may alter the values for cohesion measures. Each method is associated with the referenced attributes but might have transitive relationship with attributes referenced by the methods it invokes. This could lead to increased values for cohesion measures. We are currently exploring how to account for such questions regarding complex networks, and data will be published in the future.

Acknowledgements The current study was funded by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under Grants Nos. 2013BAB05B00 and 2013BAB06B04. This work was also supported by the National Natural Science Foundation of China under Grant No. 61602400 and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China No. 16KJB520043.

References

1. Al Dallal, J.: Fault prediction and the discriminative powers of connectivity-based object-oriented class cohesion metrics. Inf. Softw. Technol. **54**, 396–416 (2012)

2. Bansiya, J.; Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Trans. Softw. Eng. **28**, 4–17 (2002)
3. Yourdon, E.; Constantine, L.L.: Structured Design. Yourdon Press, Shepherdstown (1978)
4. Yourdon, E.: Structured Design Fundamentals of a Discipline of Computer Program and Systems Design. Prentice-Hall Inc, Reno (1979)
5. Briand, L.C.; Bunse, C.; Daly, J.: A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. IEEE Trans. Softw. Eng. **27**, 513–530 (2001)
6. Ujhazi, B.; Ferenc, R.; Poshyvanyk, D.; Gyimothy, T.: New conceptual coupling and cohesion metrics for object-Oriented systems. In: 2010 10th IEEE Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 33–42. IEEE (2010)
7. Cox, G.W.; Etzkorn, L.H.; Hughes, W.E.: Cohesion metric for object-oriented systems based on semantic closeness from disambiguity. Appl. Artif. Intell. (AAI) **20**, 419–436 (2006)
8. Etzkorn, L.H.; Gholston, S.E.; Fortune, J.L.; Stein, C.E.; Utley, D.; Farrington, P.A.; Cox, G.W.: A comparison of cohesion metrics for object-oriented systems. Inf. Softw. Technol. (INFSOF) **46**, 677–687 (2004)
9. Chen, Z.; Zhou, Y.; Xu, B.: A novel approach to measuring class cohesion based on dependence analysis. In: 2002 Proceedings of the International Conference on Software Maintenance, pp. 377–384 (2002)
10. Chidamber, S.R.; Kemerer, C.F.: Towards a metrics suite for object-oriented design. In: Object-Oriented Programming Systems, Languages and Applications. Special Issue of SIGPLAN Notice, vol. 26, pp. 197–211 (1991)
11. Chidamber, S.R.; Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**, 476–493 (1994)
12. Li, W.; Henry, S.: Maintenance metrics for the object oriented paradigm. In: 1993 Proceedings of 1st International Software Metrics Symposium, Baltimore, MD, pp. 52–60 (1993)
13. Hitz, M.; Montazeri, B.: Measuring coupling and cohesion in object oriented systems. In: 1995 Proceedings of the International Symposium on Applied Corporate Computing, Mexico, pp. 25–27 (1995)
14. Sellers, B.H.: Object-Oriented Metrics Measures of Complexity. Prentice-Hall, Englewood Cliffs (1996)
15. Bonja, C.; Kidanmariam, E.: Metrics for class cohesion and similarity between methods. In: 2006 Proceedings of the 44th Annual ACM Southeast Regional Conference, Melbourne, Florida, pp. 91–95 (2006)
16. Fernández, L.; Peña, R.: A sensitive metric of class cohesion. Int. J. Inf. Theor. Appl. **13**, 82–91 (2006)
17. Briand, L.C.; Daly, J.W.; Wst, J.: A unified framework for cohesion measurement in object-oriented systems. Empir. Softw. Eng. **3**, 65–117 (1998)
18. Bieman, J.M.; Kang, B.K.: Cohesion and reuse in an object-oriented system. In: 1995 Proceedings of the 1995 Symposium on Software reusability. Seattle, Washington, USA, pp. 259–262 (1995)
19. Badri, L.; Badri, M.: A proposal of a new class cohesion criterion: an empirical study. J. Object Technol. **3**, 145–159 (2004)
20. Chae, H.S.; Kwon, Y.R.; Bae, D.H.: A cohesion measure for object-oriented classes. Softw. Pract. Exp. **30**, 1405–1431 (2000)
21. Xu, B.W.; Zhou, Y.M.: Comments on 'A cohesion measure for object-oriented classes' by Chae HS, Kwon YR, Bae DH (Softw. Pract. Exper. 2000; 30: 1405–1431). Softw. Pract. Exp. **31**, 1381–1388 (2001)
22. Xu, B.W.; Zhou, Y.M.: More comments on 'A cohesion measure for object-oriented classes' by Chae HS, Kwon YR, Bae DH (Softw. Pract Exper. 2000; 30: 1405–1431). Softw. Pract. Exp. **33**, 583–588 (2003)
23. Yang, X.: Research on class cohesion measures. M.S. Thesis, Department of Computer Science and Engineering, Southeast University (2002)
24. Al Dallal, J.; Briand, L.C.: A precise method-method interaction-based cohesion metric for object-oriented classes. ACM Trans. Softw. Eng. Methodol. (TOSEM) **21**, 8 (2012)
25. Briand, L.C.; Morasca, S.; Basili, V.R.: Property-based software engineering measurement. IEEE Trans. Softw. Eng. **22**, 68–86 (1996)
26. Al Dallal, J.; Briand, L.C.: An object-oriented high-level design-based class cohesion metric. Inf. Softw. Technol. **52**, 1346–1361 (2010)
27. Al Dallal, J.: Measuring the discriminative power of object-oriented class cohesion metrics. IEEE Trans. Softw. Eng. **37**, 778–804 (2011)
28. Al Dallal, J.: Transitive-based object-oriented lack-of-cohesion metric. Proc. Comput. Sci. **3**, 1581–1587 (2011)
29. Al Dallal J.: Theoretical analysis for the impact of including special methods in lack-of-cohesion computation. In: 2012 First World Conference on Innovation and Computer Sciences, pp. 167–171 (2012)
30. Al Dallal, J.: Incorporating transitive relations in low-level design-based class cohesion measurement. Softw. Pract. Exp. **43**, 685–704 (2013)
31. Al Dallal, J.: The impact of accounting for special methods in the measurement of object-oriented class cohesion on refactoring and fault prediction activities. J. Syst. Softw. **85**, 1042–1057 (2012)
32. Watts, D.J.; Strogatz, S.H.: Collective dynamics of small-world networks. Nature **393**, 440–442 (1998)
33. Barabási, A.L.; Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
34. Valverds, S.; Cancho, R.F.; Sole, R.V.: Scale-free networks from optimal design. Europhys. Lett. **60**, 512–517 (2002)
35. Myers, C.R.: Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. Phys. Rev. E **68**, 046116.1–046116.15 (2003)
36. LaBelle, N.; Wallingford, E.: Inter package dependency networks in open source software. arXiv: cs.SE/0411096 (2004)
37. Ma, Y.T.; He, K.Q.; Du, D.H.: A qualitative method for measuring the structural complexity of software systems Based on complex networks. In: 2005 Proceedings of 12th Asia-Pacific Software Engineering Conference (APSEC05), Taipei, Taiwan, December, pp. 257–263 (2005)
38. Hyland Wood D.; Carrington, D.; Kaplan, S.: Scale-free nature of Java software package, class and method collaboration graphs (Tech. Rep. No. TR-MS1286), University of Maryland College Park (2006)
39. Gao, Y.; Xu, G.A.; Yang, Y.X.; Niu, X.X.; Guo, S.Z.: Empirical analysis of software coupling networks in object-oriented software systems. In: 2010 IEEE International Conference on Software Engineering and Service Sciences (ICSESS), Beijing, pp. 178–181 (2010)
40. Savić, M.; Ivanović, M.; Radovanović, M.: Characteristics of class collaboration networks in Large Java Software Projects. Inf. Technol. Control J. **40**, 48–58 (2011)
41. Subelj, L.; Bajec, M.: Software systems through complex networks science: review, analysis and applications. In: 2012 Proceedings of the First International Workshop on Software Mining, Software Mining, pp. 9–16. ACM, New York (2012)
42. Illusion (June 2012). http://sourceforge.net/projects/aoi/
43. JabRef (June 2012). http://sourceforge.net/projects/JabRef/
44. GanttProject (June 2012). http://sourceforge.net/projects/GanttProject/