A Bridge Role Metric Model for Nodes in Software Networks

Bo Li[1,2]*, Yanli Feng[1,2], Shiyu Ge[2], Dashe Li[1]

1 Key Laboratory of Intelligent Information Processing, Shan Dong Institute of Business and Technology, YanTai, Shandong, China, **2** Department of Computer Foundation Studies, Shan Dong Institute of Business and Technology, YanTai, Shandong, China

Abstract

A bridge role metric model is put forward in this paper. Compared with previous metric models, our solution of a large-scale object-oriented software system as a complex network is inherently more realistic. To acquire nodes and links in an undirected network, a new model that presents the crucial connectivity of a module or the hub instead of only centrality as in previous metric models is presented. Two previous metric models are described for comparison. In addition, it is obvious that the fitting curve between the *Bre* results and degrees can well be fitted by a power law. The model represents many realistic characteristics of actual software structures, and a hydropower simulation system is taken as an example. This paper makes additional contributions to an accurate understanding of module design of software systems and is expected to be beneficial to software engineering practices.

Citation: Li B, Feng Y, Ge S, Li D (2014) A Bridge Role Metric Model for Nodes in Software Networks. PLoS ONE 9(11): e111613. doi:10.1371/journal.pone.0111613

Editor: Matthias Dehmer, UMIT, Austria

Received May 26, 2014; **Accepted** September 30, 2014; **Published** November 3, 2014

Data Availability: The authors confirm that, for approved reasons, some access restrictions apply to the data underlying the findings. Data are owned by Jialiang Wang. Requests for data may be made to Dr. Wang (89166923@qq.com)

Funding: This work was funded by the National Natural Science Foundation (No. 61070175, http://isisn.nsfc.gov.cn/egrantindex/funcindex/prjsearch-list#); Shandong Province Natural Science Foundation(ZR2013FL017, ZR2013FL018, http://www.sdnsf.gov.cn/portal/); Shandong Province University Science and Technology of China (J12LJ03, http://jihlx.sdstc.gov.cn/stdpms/); project development plan of science and technology of yanntai(2013ZH347, 2013ZH091, http://www.ytstc.gov.cn/ArticleDetail.aspx?aid=11762), Dr. Scientific Research Foundation in shandong institute of business and technology (BS201407, http://www.ccec.edu.cn/). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

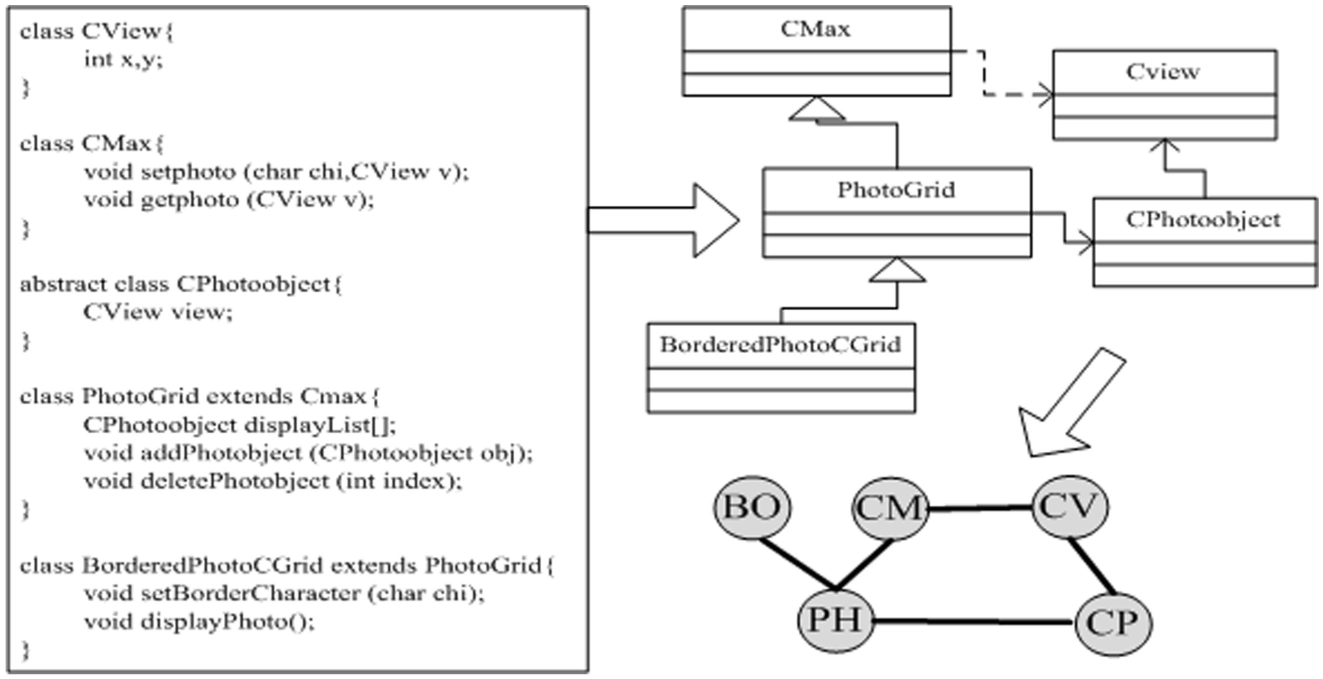
Competing Interests: The authors have declared that no competing interests exist.

* Email: 89646597@qq.com

Introduction

Large-scale software systems have developed quickly with the rapid development of software engineering. Hence, understanding, measuring, and controlling design are significant challenges for designers, which have attracted a significant amount of attention. There are many studies on software metric methods such as property-based [1],junction point [2],productivity [3] and combination [4], but "a common approach of using simple regression models to predict software defects [5–8] can lead to risk management decisions". In 2002, complex networks were applied to metric software structures by Valverde et al, where the software structure is represented by a complex network. Characteristics of scale-free and small-world networks have been determined [9], and subsequently, studies have also determined that software networks that are extracted from various software also follow power-law degree distributions [10–19], exhibit strong community phenomenon [10,20], and show some complex network behavior characteristics[13,21–26]. Furthermore, other studies have been analyzed in software systems, and subsequently, the methodology of dependability in software networks based on three dimensions of structure has been discussed, and the structural stability in software is analyzed on dimension of composition. Because of the reusability of design patterns in object-oriented software systems, the design patterns are regarded as a typical structure that has more effect on the whole [27,28]. [29] has studied nine large object-oriented software networks, recovering that graphs associated with these software networks are self-similar. They have also studied the time evolution of fractal dimensions during software system growth, and a significant correlation is found between the complexity metrics and the fractal dimension. [30] has presented a systematic empirical analysis of the statistical properties of communities.

On the other research front, some studies are trying to develop a metric for the role of a module in software networks, but few models can describe the "bridge" role of a module more accurately. The Weighted OO Software Coupling Network as the node weight is proposed in [31], where the weight and out degree follow a power law distribution. [32,33] have introduced main metric parameters of software networks in detail and have integrated these metrics parameters into a hierarchical metric set. The analytic results in [34] have revealed that most of the parameters in complex systems can also be used to represent properties of software structures, some efficient metrics and methods are introduced which are based on basic parameters in other complex systems, and a practical example is used to demonstrate the validity and effectiveness of the proposed metrics. [35] has described some recent algorithms that appear to work as well as some algorithms based on betweenness, which is one of the most important metrics of the centrality of a module in a software network. [36] has introduced another important metric model: closeness. It makes regular and macroscopic analysis and subsequently, utilizes the method to measure important features and characteristics. The relativity among the integral measure and identities facilitates important proofs for the qualification of software qualities.



```
class CView{
        int x,y;
}

class CMax{
        void setphoto (char chi,CView v);
        void getphoto (CView v);
}

abstract class CPhotoobject{
        CView view;
}

class PhotoGrid extends Cmax{
        CPhotoobject displayList[];
        void addPhotobject (CPhotoobject obj);
        void deletePhotobject (int index);
}

class BorderedPhotoCGrid extends PhotoGrid{
        void setBorderCharacter (char chi);
        void displayPhoto();
}
```

Figure 1. The extraction from codes to software network. The process is as follows: The UML class diagram is first abstracted from the source code and subsequently converted to the undirected software network.
doi:10.1371/journal.pone.0111613.g001

This paper is motivated by the above considerations. [10,32–33] have proposed some metric parameters and models to represent properties of software structure that are independent of the connectivity role of a module, and modules in [34–36] represent the centrality of a module in software networks but are different from connectivity. Hence, a new model is proposed from a new perspective. Some modules behave stronger connectivity than other modules, and if a fault occurs, neighbors of these modules cannot connect to each other. A "bridge" is used to represent the connectivity of the module in the software network; therefore, a bridge role metric model that can more accurately serve as a metric for characteristics is proposed. The remainder of this paper is outlined as follows. After describing the bridge role metric model in section 2, we compare this model with two other previous models and analyze the correlation between the *Bre* results and other fundamental metrics. In section 4, an actual hydropower system is taken as an example to demonstrate the validity of the model and the implications of design principles for software structure are discussed. In section 5, the conclusion is presented and future studies are proposed.

Methods

Software networks

Software is a system which is composed of many interactional and collaborative units reflecting coding, design and execution. The extraction from codes to network is displayed in Fig. 1. Particularly, some modules are reused or rely on other modules, and the dependency relations between two modules A and B include two types: inheritance and association. If A makes reference to B (either through association or inheritance) in its definition, there is an edge directed from A to B and vice versa. Hence, a software network is defined as $G = <V,E,f>$, where $V = \{v_i | i = 1,......,N\}$, which represents modules, is a set of nodes

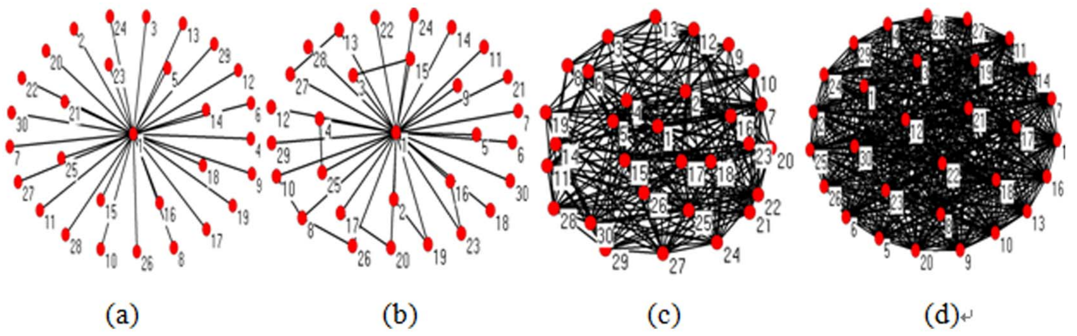


Figure 2. Several cases with number of edges gradually increasing and the fixed nodes. In Figure 2 (a), $N = 30$, $M = 29$, and $Bre_1 = 0.0345$. In Figure 2 (b), $N = 30$, $M = 37$, and $Bre_1 = 0.0577$. In Figure 2 (c), $N = 30$, $M = 275$, and $Bre_1 = 0.1319$. In Figure 2 (d), $N = 30$, $M = 435$, and $Bre_1 = 0.1332$. The value of Bre_1 reflects the connectivity of the node 1.
doi:10.1371/journal.pone.0111613.g002

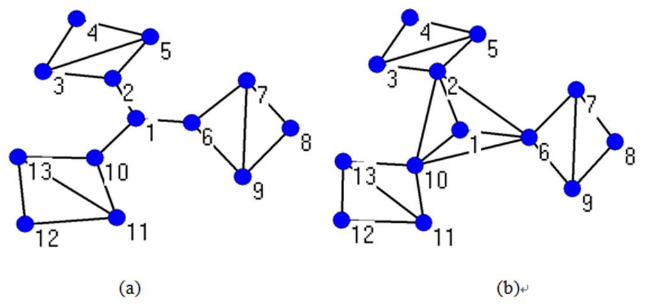


Figure 3. Comparisons of the value between closeness and bridge role in two identical networks. (a) $N = 13$ and $M = 18$. (b) $N = 13$ and $M = 21$, where there are three more edges in this network on the basis of the network in (a). Node 1 is in the center in these two networks.
doi:10.1371/journal.pone.0111613.g003

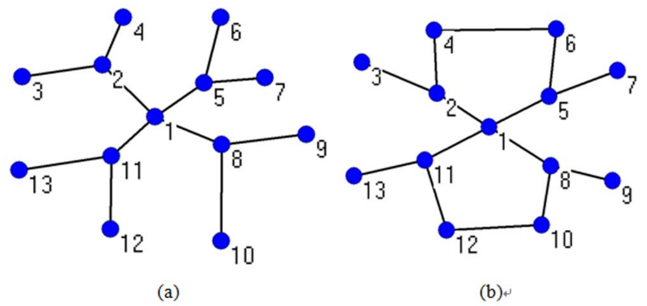


Figure 4. Comparisons of the value between the betweenness and bridge role in two identical networks. (a) $N = 13$ and $M = 12$. (b) $N = 13$ and $M = 16$. Node 1 is also in the center in these two networks as shown in Figure 3.
doi:10.1371/journal.pone.0111613.g004

and $E = \{e_{i,j} | i = 1,...m; j = 1...n\}(|E| = M)$, which denotes relations between modules. The repeated edges between modules are not considered. Software is regarded as an undirected network, and $e_{i,j} = e_{j,i}$. Node i is characterized by parameters such as the degreek_i, closeness $C_C(i)$, and the betweenness $C_B(i)$, which are presented in section 3. In this paper, approximately 100 randomly selected software (listed in table in Appendix S1) from the open source community (http://sourceforge.net, http://code.google.com/hosting/ and http://www.oschina.net) are chosen as empirical cases.

Bridge role metric model

As mentioned above, two metric models can better measure the centrality of a node in the software network, *i.e.* the closeness $C_C(i)$ [36] and the betweenness $C_B(i)$ [35], and their definitions are as follows: $C_C(i) = (N-1)\left[\sum_{j=1}^{N} d_{ij}\right]^{-1}$, where N is the number of nodes in the software network, $i <> j$, and $d_{ij} = d_{ji} = 1$ if there is an direct connection between node i and j; otherwise, $d_{ij} = d_{ji} = d_{iv_1} + d_{v_2 v_1} + \ldots\ldots + d_{v_2 v_m} + d_{v_m i}$, $m <= n$. $C_B(i) = \sum_{s \neq i \neq t \in V} \delta_{st}(i)/\delta_{st}$, where $\delta_{st}(i) = 1$ if node i is located on the shortest path between node s and t, and δ_{st} is the number of shortest paths between s and t.

Conversely, closeness and betweenness cannot show the connectivity role effectively; therefore, the bridge role metric model is proposed in this paper, and comparisons with the two metric models above are executed.

The definition is as follows:

$$Bre_i = \sum_{j \neq i} (p_{ij} + \sum_{k, k \neq i, k \neq j} p_{ik} p_{kj})^2 \tag{1}$$

$$p_{ij} = \frac{d_{ij}}{\sum_k (d_{ik})} \tag{2}$$

where node j is any neighbor of node i, and $d_{ij} = 1$ if there exists an edge between node i and node j; otherwise, $d_{ij} = 0$. We now discuss the value of equation (1). We suppose the number of all neighbors of node i is n ($1 \leq n \leq N\text{-}1$,); therefore, node i and all its neighbors can be considered as a community as follows: $p_{ij} = \frac{d_{ij}}{\sum_k (d_{ik})} = \frac{1}{n}$ and $Bre_i = \sum_{j \neq i} (\frac{1}{n} + \sum_{k, k \neq i, k \neq j} \frac{1}{n} p_{kj})^2$. We set the number of neighbors (including node i) of node k ($k \neq i, k \neq j$) as $n_k (1 \leq n_k \leq n)$ and the number of neighbors of node j as $n_j (1 \leq n_j \leq n)$. We obtain $p_{kj} = \frac{1}{n_k}$ and $Bre_i = \sum_{j \neq i} (\frac{1}{n} + \sum_{k, k \neq i, k \neq j} \frac{1}{n}\frac{1}{n_k})^2$. One of the two extreme cases is that when $n_j = 1 (j \neq i)$, $Bre_i = \sum_{j \neq i} (\frac{1}{n} + \sum_{k, k \neq i, k \neq j} \frac{1}{n} \times 0)^2 = \sum_{j \neq i} (\frac{1}{n})^2 = \frac{1}{n} (\lim_{n \to \infty} \frac{1}{n} = 0)$, and the other extreme case is that when $n_k \neq n (k \neq i, k \neq j)$, *i.e.*, the community is a n+1-clique, then $Bre_i = \sum_{j \neq i} (\frac{1}{n} + \sum_{k, k \neq i, k \neq j} \frac{1}{n^2})^2 = \sum_{j \neq i} (\frac{1}{n} + \frac{n-1}{n^2})^2 = \sum_{j \neq i} (\frac{2n-1}{n^2})^2 = (\frac{2n-1}{n^3})^2$. We set $Y = \frac{(2n-1)^2}{n^3}$, and subsequently, the solutions of equation $Y' = 0$ are $n = 0.5$ and $n = 1.5$, but n is an integer; hence, the extremal solution is $n = 2$ and $Bre_i = 1.125$. When $n \geq 3, (2n-1)^2 \leq n^3$; therefore, $Y < 1$. Finally, another case should be discussed in which the community is not a star network or a clique. In these cases, the maximum can be computed with the recurrence method. Suppose that there are $\frac{n(n-1)}{2} - 1$ edges between neighbors of node i, *i.e.* $M = \frac{n(n-1)}{2} - 1$(if $M = \frac{n(n-1)}{2}$, the community is an $n+1$-clique). We then set $Bre_i(\frac{n(n-1)}{2} - 1)$ to represent the value of the metric model. We obtain $Bre_i(\frac{n(n-1)}{2} - 1) = \sum_{j \neq i} (\frac{1}{n} + \sum_{k, k \neq i, k \neq j} \frac{1}{n}\frac{1}{n_k})^2 = (n-2)(\frac{1}{n} + \frac{n-1}{n^2})^2 + 2 (\frac{1}{n} + \frac{n-2}{n^2})^2 = \frac{(n-2)(2n-1)^2}{n^4} + \frac{2(2n-2)^2}{n^4} < Bre_i(\frac{n(n-1)}{2}) = \frac{(n-2)(2n-1)^2}{n^4} + \frac{2(2n-1)^2}{n^4}$; hence, we have $Bre_i(n-1) < Bre_i(n)$ and

$$Bre_i = \begin{cases} \in (0,1) \cup (1, 1.125] & n \in [2, +\infty] \\ 1 & n = 0,1 \end{cases},$$

where $n = 0$ means that the node is an isolated one.

Computing the *Bre* value can be described with the following algorithm. The method of creating the hierarchical network is by placing the nodes to a corresponding hierarchy based on its centralization. For example, the nodes that are in the center will be placed in the most inner, and the whole network will be similar to a multi-ring network.

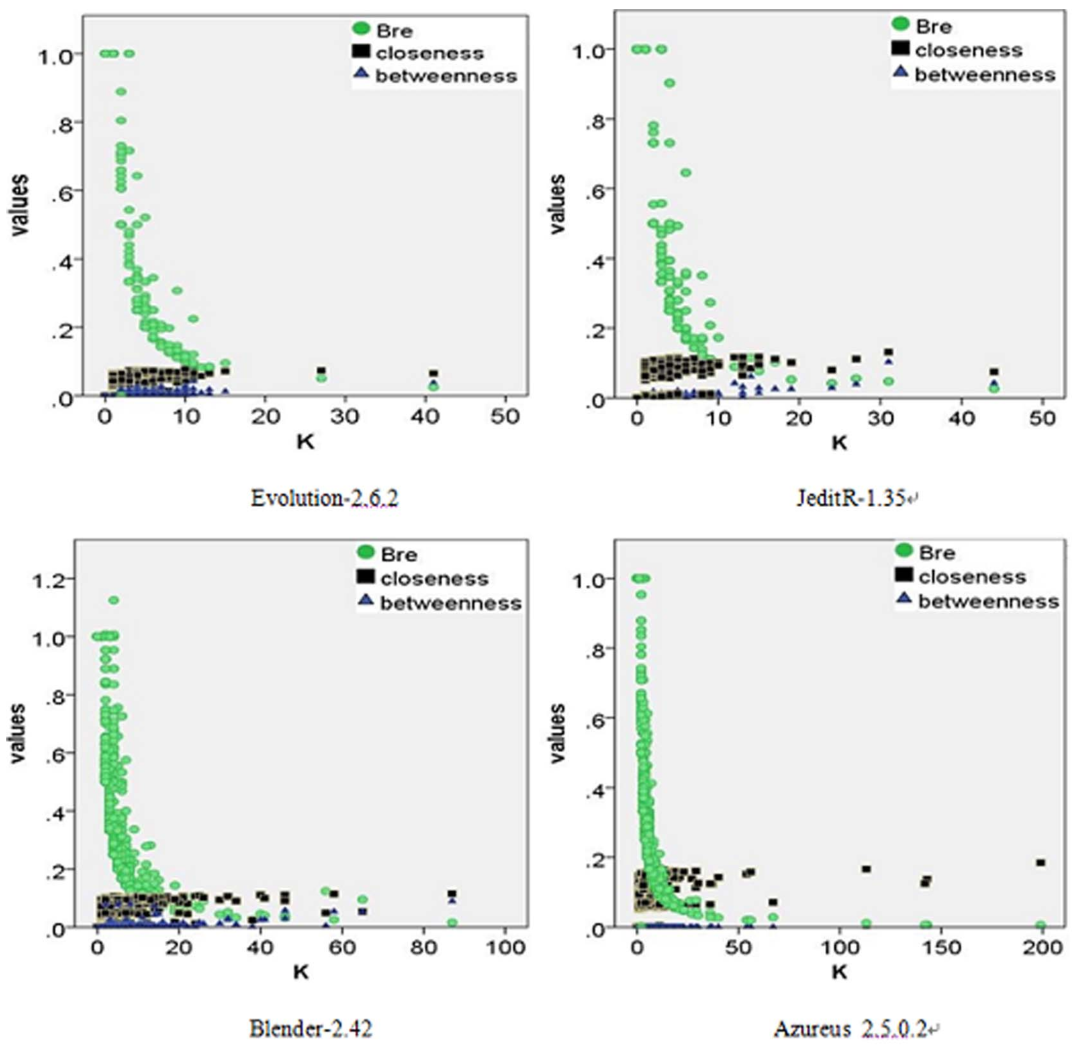


Figure 5. The correlations between the metric model *Bre* values, closeness, betweenness and the degree *K*. The corresponding data obtained from Evolution-2.6.2 ($N=1445$, $M=1129$), JeditR-1.35 ($N=822$, $M=718$), Blender-2.42 ($N=2426$, $M=2848$) and Azureus_2.5.0.2 ($N=2375$, $M=3278$), which are well-known software packages. The data points (●,■,▲) represent measurements of the three metric models.
doi:10.1371/journal.pone.0111613.g005

```
**Algorithm 1** $Bre(N,M)$
Create Hierarchical Network $H(N,M)$
If the number of nodes in the nodes set $|N| \neq \emptyset$
Computing the isolated and nodes (suppose the number of these nodes is $n1, Bre = 1$)
hierarchy = 1
    while $|N| \neq \emptyset$ do
        $|N| = |N|$-$n1$;
    Computing the nodes of the current hierarchy
    hierarchy = hierarchy+1;
        Removing the nodes of the current hierarchy from $|N|$;
        end while
        returning results
```

Results and Discussion

Comparisons

How does the bridge role metric model represent the connectivity of a node? Fig. 2 shows several cases with the number of edges gradually increasing, and node *1* is taken as an example to explain the function of the model. In Fig. 2(a), there are no edges between neighbors of node 1; hence, the neighbors cannot make contact with each other without node 1. In Fig. 2(b), four pair of nodes can connect to one another without node 1, and in Fig. 2(c), much more of these types of nodes exist. In Fig. 2(d), any node can connect to any other one. It can be concluded that the connectivity of a node in a given community becomes stronger with Bre_i value decreasing, and it has been theoretically proved in section 2.2 with equation (3), where connectivity means the ability to make other nodes communicate with each other.

As mentioned in section 2.2, two previous metric model parameters are the closeness $C_C(i)$ [36] and the betweenness $C_B(i)$ [35]. How does the metric model in this paper work more effectively than these two models? Node 1 is taken as an example, and comparisons are described as follows.

In Fig.3, there are two networks that almost have the same structure except for a few edges. Node 1 is in the center of (a) and

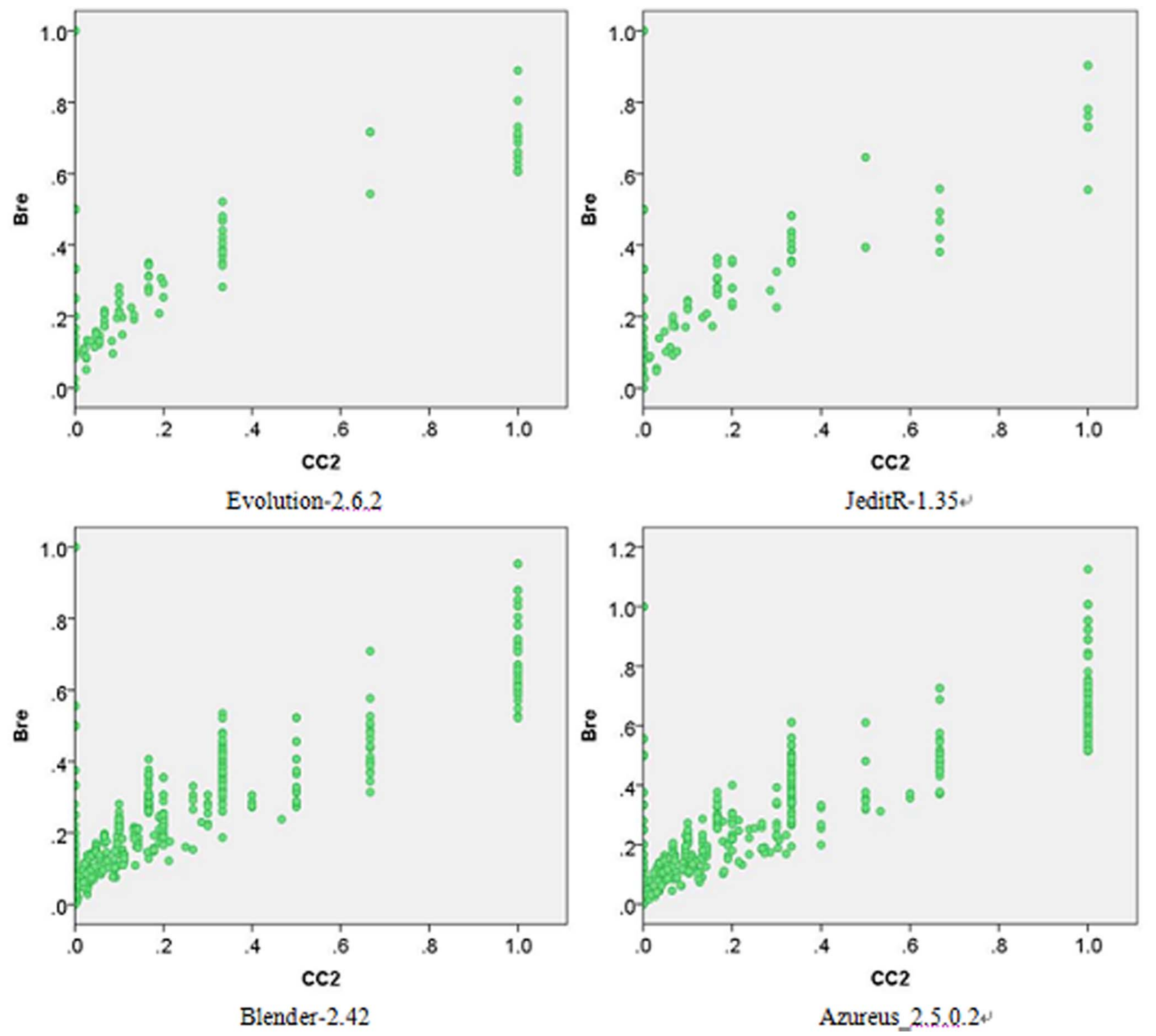


Figure 6. The correlations between the *CC2* and *Bre* values. The corresponding data are also obtained from Evolution-2.6.2, JeditR-1.35, Blender-2.42 and Azureus_2.5.0.2. The data points ● represent measurements of the *Bre* models.
doi:10.1371/journal.pone.0111613.g006

(b). In (a), nodes 2, 6, and 10 can communicate with each other only through node 1; hence, node 1 acts as a "bridge". In (b), all other nodes can connect to each other without node 1; therefore, in the latter network, the role of node 1 is not very important. The closeness of node 1 $C_C(1)$ is 0.5 in the two networks, which cannot reflect the evident different role of node 1 as an intermediate node. Nevertheless, in the former network, $Bre_1 = 0.3333$, and $Bre_1 = 0.6533$ in the latter network. It shows that the bridge role metric model can reflect the connectivity of a node more effectively than can the closeness.

We now concentrate on the other previous metric model parameter: betweenness. In addition, there are two networks in Fig. 4, where the latter one has two more edges than the former. In (a), there are a total of 66 shortest paths between the nodes excluding node 1, in which node 1 is located in 54 paths; therefore, $C_B(1) = 0.8182$. Meanwhile, node 1 acts as a bridge to allow nodes 2, 6, and 10 to communicate with each other, where $Bre_1 = 0.25$. In (b), the connectivity of node 1 for nodes 2, 6, and 10 does not change, and $Bre_1 = 0.25$; however, the shortest paths that node 1 is located in decrease, where $C_B(1) = 0.6667$. It should be noted that Bre_2,Bre_5,Bre_8 and Bre_{11} are altered because of the two extra edges.

The conclusion can be drawn as discussed above that the metric model proposed in this paper can reflect the connectivity of nodes more effectively than the closeness or betweenness.

Simulations

Some studies have revealed that software networks follow power law distributions over an extent of degree$K$, which is the number of edges attached to the node $P(K) \sim K^{\lambda}$ [36]. It is natural to consider the correlations between the bridge role metric model and other metrics. Fig.5 shows the correlations of *Bre*, between-

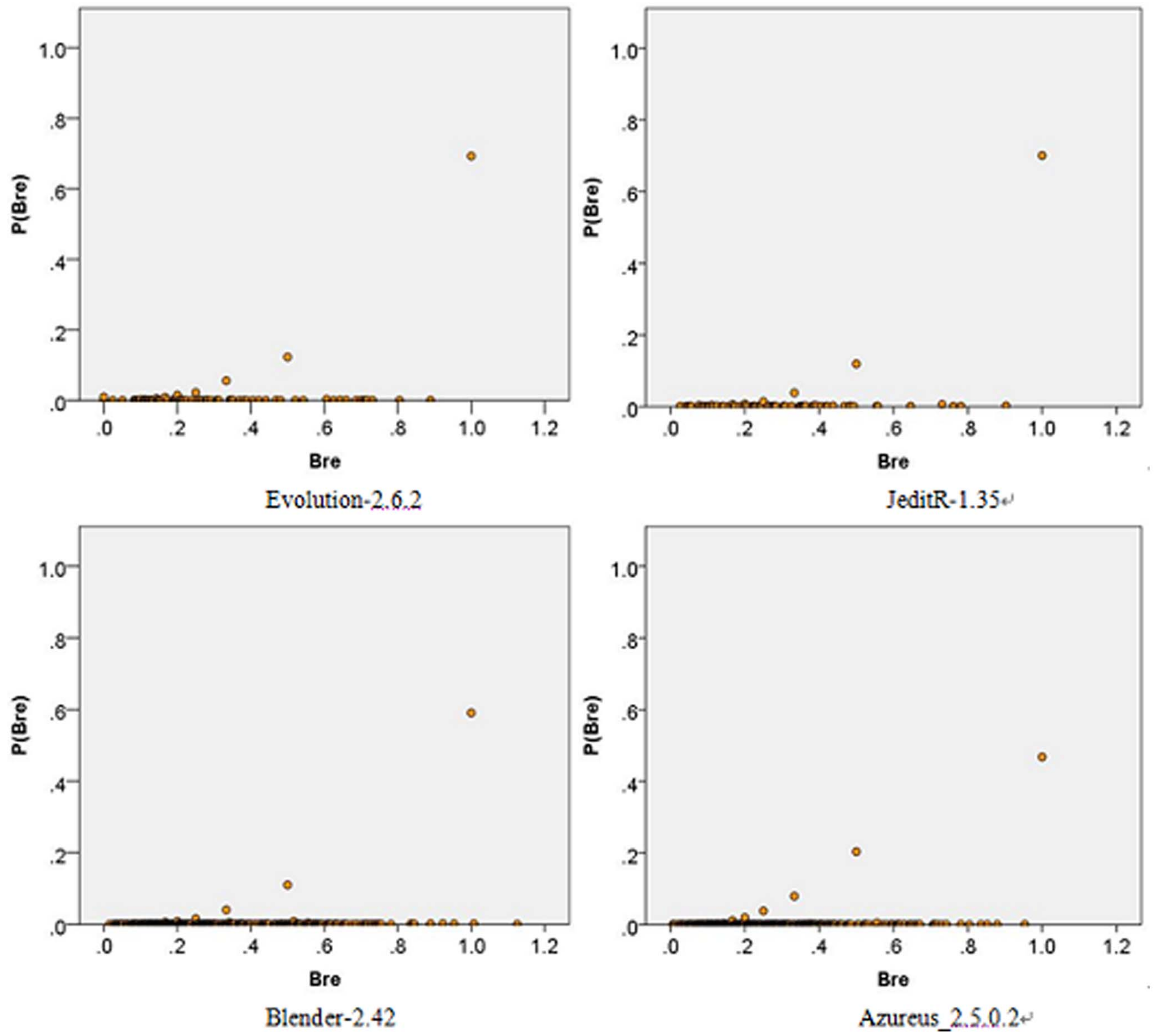


Figure 7. The distributions of the *Bre* values. The data points ● represent $P(Bre)$. The distribution can be plotted with two branches, one of which seems to be most likely proportional to Bre but with few data points; therefore, there is no correlation between Bre and $P(Bre)$.
doi:10.1371/journal.pone.0111613.g007

ness, closeness, and the degree K in four familiar software networks.

Typically, centrality (closeness or betweenness) has a significant correlation with the degree; nevertheless, it can be seen in Fig. 5 that the closeness or betweenness increases but is less pronounced as the degree K increases, and the centrality of a node does not significantly depend on its degree. Specially, it is determined that Bre values are logarithmic with the degrees, and it indicates that the node plays a less important connectivity role with increasing degree. Meanwhile, there are more edges between the neighbors of the corresponding node. The correlation contributes more to an accurate understanding of the module for software engineering practices. If there are some reusable modules in a software system, they will obey the engineering principle where if the reusable rate is high. The corresponding module is often redesigned as several additional modules, the neighbors often use or rely on more than one modules, and hence making the neighbors more “close”.

The Bre values have a close relation with the edges between their neighbors; therefore, there is most likely a correlation between them and another metric model called Clustering Coefficients ($CC2$) [34], which also depends on the edges. $CC2 = \frac{E(G_1(v))}{E(G_2(v))}$, where $E(G_k(v))$ is the number of edges among nodes in the k-neighborhood of node $v(k = 1,2)$. As seen in Fig.6, there is an approximately linear correlation between CC2 and Bre. The correlation indicates that increased use between parts of neighbors $(k = 1,2)$ will inevitably lead to a decrease of the connectivity of the corresponding module. Because of the scale-free characteristic $(P(K) \sim K^{\lambda})$ mentioned above, it is clear that $P(Bre)$is not proportional to Bre^{λ}, which represents the difference between Bre and K from another point of view. The

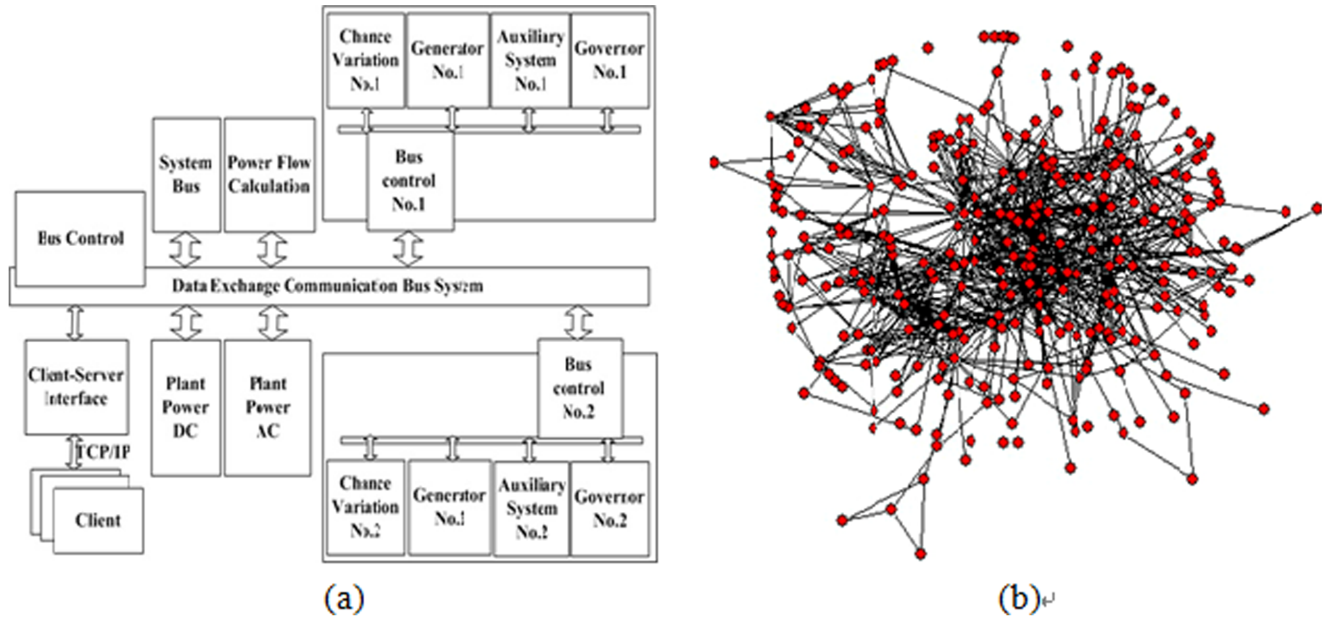


Figure 8. Architecture and network of the hydropower simulation system. (a) Architecture. (b) Network ($N = 310$, $M = 850$). The software system is developed by Embedded Technology Key Lab in Northeastern University (www.netology.cn) for the Fengman hydropower station in China.
doi:10.1371/journal.pone.0111613.g008

Bre distribution certainly does not reflect the scale-free [26] nature of the software system, which is shown in Fig. 7.

To verify the validity of the metric model proposed in this paper for software engineering practices, a hydropower simulation system [34] is taken as an example. The architecture and corresponding networks are shown in Fig. 8. It is developed by Embedded Technology Lab in Northeastern for the Fengman station, which was the earliest established large hydropower station. The software has access to two national software copyright studies (No. 0009448 and No. 050963) and has been working for more than ten years. The metric model in this paper is used for fault detection in developing version 2.0 software. First, modules are sorted based on the *Bre* values, subsequently, source codes of the modules that have lower values and are not isolated are analyzed. The studies determined that there are fault-pronesses [37] in four modules (the XJ, RD, LP and VoltCurr modules), which lead to overall instability. These modules are basic control units and plays significant bridge roles in the system because other modules inherit or use them. The studies facilitated redesigns to reduce the fault-proness and enhance stability.

Conclusions

The contribution of this paper is the proposed bridge role metric model. Because of the different connectivity role of a node in a software network, we use the *Bre* metric model instead of the previous two metric models: betweenness and closeness. After providing a definition, the range of the metric value was discussed. The metric model's function is illustrated with different cases as well as theoretically. Comparisons are also carried out, and the analysis indicates that the model can reflect the connectivity more effectively. Furthermore, it is determined that *Bre* values are logarithmic with the degrees and are proportional to another metric model-Clustering Coefficients-CC2, which indicates that the node plays a less important connectivity role as the degree increases. Nevertheless, $P(Bre)$ is not proportional to Bre^{λ}. To verify the validity of the model in software engineering practice, a hydropower simulation system is taken as an example to detect the fault-proness in modules.

However, we still require further work to improve the application of the model in software structure designs. Most likely, we can detect fault-proness through a combination of this model and others (K, closeness, etc.). Second, it also required some other proof, we will use some software engineering metrics such as coupling, Cohesion to support the solution proposed. Additionally, further investigations to extend the metric model to macro- and micro-structure should be carried out to emphasize estimating the role of a node in the entire software network more effectively.

The work in this paper could facilitate a better understanding of the role of modules in systems. Actually, because local instability most likely leads to global failures, the structure is very important for designers to predict the fault-proneness of a module. The metric can help us to redesign the structure of software, improve the quality of software, and subsequently shorten the development life cycle.

Supporting Information

Appendix S1 Appendix to the manuscript.
(DOC)

Acknowledgments

The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

Author Contributions

Conceived and designed the experiments: BL YF SG DL. Performed the experiments: BL YF SG DL. Analyzed the data: BL YF SG DL. Contributed reagents/materials/analysis tools: BL YF SG DL. Wrote the paper: BL YF SG DL.

References

1. Briand LC, Morasca S, Basili VR (1996) Property Based Software Engineering Measurement. IEEE Transaction on Software Engineering 22(1): 68–86.
2. Furey S (1997) Why we should use function points. IEEE Software 14(2): 28–30.
3. Arnold M, Pedross P (1998) Software Size Measurement and Productivity Rating in a Large-Scale Software Development Department. In: Proceedings of 1998 International Conference on Software Engineering[C] Kyoto: 503–506.
4. Bauer M (1999) Analysing Software Systems by Using Combinations of Metrics. In: Proceedings of ECOOP'99 Workshops[C] Lisbon: 170–171.
5. Kpodjedo S, Ricca F, Antoniol G, Galinier P (2009) Evolution and Search Based Metrics to Improve Defects Prediction. In: Proceedings of the1st International Symposium on Search Based Software Engineering, Windsor, UK: 23–32.
6. Fenton NE, Neil M (2000) Software metrics: successes, failures and new directions. Journal of Systems and Software 47(23): 149–157.
7. Mens T, Demeyer S (2001) Future trends in software evolution metrics. In: Proceedings of 4th International Workshop on Principles of Software Evolution, Austria: 83–86.
8. Fenton NE, Krause P, Neil M (2002) Software Measurement: Uncertainty and Causal Modeling. IEEE Software 19(4): 116–122.
9. Valverdes, Cancho RF, Sole RV (2002) Seale free networks from optimal design. Europhysics Letter 60: 512–517
10. Myers CR (2003) Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. Physics Review E 68: 1–15.
11. Valverdes S, Sole R (2003) Hierarchical small worlds, in software architecture. Working Paper of Santa Fe- institute, SFI/030744.
12. De Moura APS, Lai YC, Motter AE (2003) Signatures of small-world and scale-free properties in large computer programs. Physical Review E 68: 017102.
13. Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286: 509–512.
14. Wen L, Dromey G, Kirk D (2009) Software engineering and scale-free networks. IEEE Transactions on Systems, Man, and Cybernatics PartB: Cybernatics 39(4): 845–854.
15. Concas G, Marchesi M, Pinna S, Serra N (2007) Power-laws in a large object-oriented software system. IEEE Transactions on Software Engineering 33(10): 687–708.
16. Clauset A, Shalizi CR, Newman ME (2009) Power-law distributions in empirical data. SIAM Review 51(4): 661–703.
17. Ma YT, He KQ, Li B, Liu J, Zhou XY (2010) A Hybrid Set of Complexity Metrics for Large-Scale Object-Oriented Software Systems. Journal of Computer Science and Technology 25(6): 1184–1201.
18. Potanin A, Noble J, Frean M, Biddle R (2005) Scale-free geometry in OO programs. Communications of the ACM-Adaptive complex enterprises 48(5): 99–103.
19. Liu J, He KQ, Ma YT, Peng R (2006) Scale free in software metrics. In: Proceedings of 30th Annual International Computer Software and Applications Conference pp. 229–235.
20. Subelj S, Bajec M (2011) Community structure of complex software systems: analysis and applications. Physica A 390(16): 2968–2975
21. Jenkins S, Kirk SR (2007) Software architecture graphs as complex networks: a novel partitioning scheme to measure stability and evolution. Information Sciences 177(12) 2587–2601.
22. Li B, Pan WF, Lu JH (2011) Multi-granularity dynamic analysis of complex software networks. in: Proceeding of IEEE International Symposium on Circuits and Systems pp. 2119–2124.
23. Pan WF, Li B, Ma YT, Qin YY, Zhou XY (2010) Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks. Journal of Computer Science and Technology 25(6): 1202–1213.
24. Roach C, Menezes R (2011) Using networks to understand the dynamics of software development. Communications in Computer and Information Science 116(2): 119–129.
25. Dabrowski R, Stencel K, Timoszuk G (2011) Software is a directed multi-graph. In: Proceedings of the 5th European conference on Software architecture pp. 360–369.
26. Zhang HH, Zhao H, Cai W, Zhao M (2008) Visualization and cognition of large-scale software structure using the k-core analysis. In: Proceedings of the 2008 International Conference on Intelligent Information Hidingand Multimedia Signal Processing, pp. 954–957.
27. Wang W, Zhao H, Li H, Zhang J, Li P, et al. (2010) Research on LFS Algorithm in Software Network. Journal of Software Engineering and Applications 3(2): 185–189.
28. Wang W, Zhao H, Li H, Li P, Yao D, et al. (2010) Application of Design Patterns in Process of Large-scale Software Evolving. Journal of Software Engineering and Applications 3(1): 58–64
29. Concas G, Locci MF, Marchesi M, Pinna S, Turnu L (2006) Fractal Dimension in Software Networks. Europhysics Letters 76(6): 1221–1227.
30. Lancichinetti A, Kivelä M, Saramäki J, Fortunato S (2010) Characterizing the Community Structure of Complex Networks. PloS ONE 5(8): e11976.
31. Liu J, He KQ, Ma YT, Peng R (2006) Scale Free in Software Metrics. In: Proceedings of the 30th Annual International Computer Software and Applications Conference (1) 229–235.
32. Liu J, He KQ, Peng R, Ma YT (2006) A Study on the Weight and Topology Correlation of Object Oriented Software Coupling Network, In: Proceedings of the 1st International Conference on Complex Systems and Applications pp. 955–959.
33. Ma YT, He KQ, Liu J, Yan YL (2005) A Complexity Metrics Set for Large-scale Object-Oriented Software Systems. In: Proceedings of the 6th International Conference on Computer and Information Technology pp. 189.
34. Cai W, Zhao H, Zhang HH (2007) Static Structural Complexity Metrics for Large-scale Software. Special Issue on Software Engineering and Complex Networks of Dynamics of Continuous, Discrete and Impulsive Systems Series B 14(S6): 12–17.
35. Newman ME (2004) Detecting community structure in networks. Euro. Phys. J. B, 38(2): 321–330.
36. Peng L (2011) Research on Structure Characteristics and Information Metabolism of Software Network. Shen Yang: Northeastern University. 344 p.
37. Huang P, Zhu J (2008) Predicting the fault-proneness of class hierarchy in object-oriented software using a layered kernel. Journal of Zhejiang University 9(10): 1390–1397.