Research on Structural Holes and Closeness of Multi-Granularity Software Networks

Hui Li, Hui Zhang
School of Informational Science and Engineering, Shenyang University of Technology, Shenyang, China
Email: dialee6@yahoo.com.cn

Hai Zhao, Wei Cai
School of Information Science and Engineering, Northeastern University, Shenyang, China
Email: zhangluyi2007@126.com

***Abstract*—Object-Oriented software structures represent multi-level characteristics. In this paper, two parameters -- structural holes and closeness used in complex networks are introduced to study topological characteristics of software networks from multi-granularity perspective. By this method, software networks are observed and analyzed in three levels: package granularity, class granularity and method granularity. Then correlations between these two parameters and node degrees are investigated and analyzed by case studies, respectively. The results show that this method is useful for measuring the extent of dependence and centralization in software networks, and help us deeply understand different scales of software structural characteristics.**

***Index Terms*—multi granularity, software networks, structural holes, closeness**

I. Introduction

Complex networks have been widely studied across many fields of science. Examples include the movie actor collaboration [1], Internet [2], the World Wide Web [3], etc. In recent years, some researchers have studied on a great deal of orient-object software and found the structure of these software systems are not random and out of order, instead, most of them present the features of complex networks such as “small world” property and “scale-free” property. These findings led researchers to think about software structure from the perspective of complex network. The software is abstracted as a complex network to study, and then gradually formed a network view [4]. The combination between complex network theory and software engineering provides an unprecedented research means for software. By this way, the characteristics hidden in software structure can be obtained and we can better understand the development of software design rules.

As the software scale increase and the software complexity rise ceaselessly, software structure has appeared in multiple levels, multiple granularities, and multiple integration modes of organization method. Software attributes largely depend on its component elements and the complex interactions between two elements; these elements are small to methods and variable in the class, big to pack and subsystems. Nevertheless, it can be found that the work carried out in [5-7] is confined to a single level of granularity level. So whether the properties they got can fit in with other levels of granularity is still a problem faced by software structure. At the same time, it is difficult to observe the new properties showed in different levels. To solve this problem and give a more complete view of software structures, it is necessary to divide the object-oriented software into three levels of software networks (package level, class level and method level) and use them to explore software properties.

Based on the current research results of complex networks, we further study multi-granularity software networks nodes in the dependency and center degree by means of the two new parameters--structural holes [8] and closeness in this paper. This work provides a more comprehensive understanding of structural properties in different levels and is useful for the software development and software maintenance.

II. Related Work

Several researches regarding software networks at different levels of granularity have been performed. LaBelle et al [12] studied the inter-package dependency relationship of software networks at the package level, and found that these networks are of scale free and small world type. In [13-15], Valverde et al abstracted classes and methods of a class for nodes and discovered the same conclusion in [12].

In [16], collaboration graphs were produced at the package, class and method levels and formed networks which exhibited approximately scale-free properties at all three levels. The researchers analyzed the significant differences among the magnitudes of power law exponents at three levels of granularity.

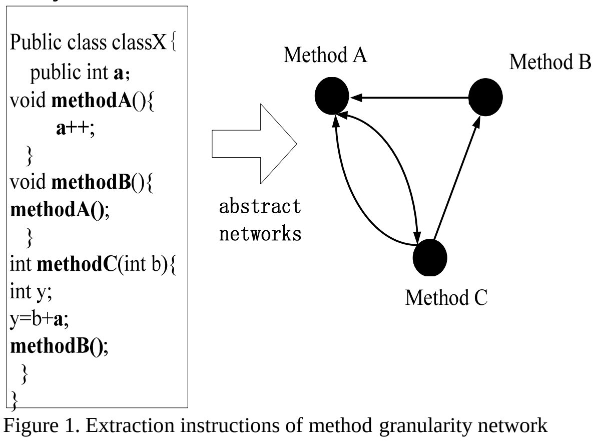
In [10], the definition of multi-granularity software networks was given exactly. Nodes represent the network component units (methods or classes or packages), links represent the relationships between unit and unit. The definition of interactions between two classes depended on the interactions among methods of classes; while the definition of interactions between two packages depended on the interactions among classes of packages. By this way, some basic parameters of multi-granularity software networks were researched to analysis software structure.

The above research results are useful for us to do further study, which will be discussed in Section III.

III. MULTI-GRANULARITY SOFTWARE NETWORK

In this paper, the object-oriented software networks are divided into package granularity, class granularity and method granularity at three different levels of networks [9-10]. Each granularity of software networks consists of nodes and links. Nodes represent the network component units; links represent the relationships between unit and unit. We give their formal descriptions as follows.

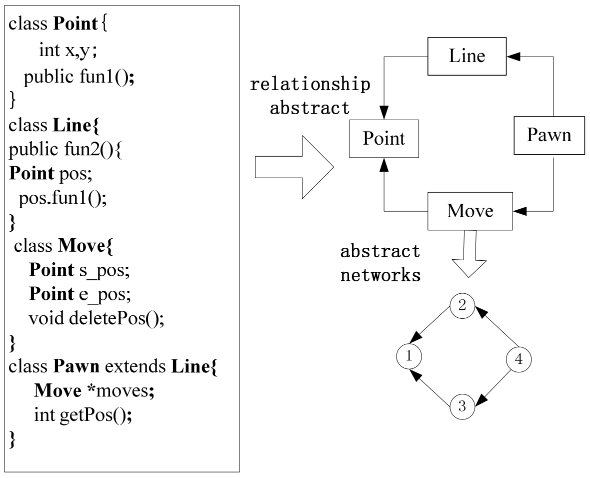
Definition 1. Method granularity of software networks (FN) is defined as a directed graph $FN = (V_F, E_F)$, where all methods of specific object-oriented software can be treated as nodes n_F (where $n_F \in V_F$).The call relationship between every pair of methods if exists forms a directed link l_F (where $l_F \in E_F$) in the graph. The direction of each link is from the caller (method that calls other methods) to the callee (method that other methods call). The interactions among the methods can be divided into two categories :(1) methods in the same class have indirect interactions by calling the same instance variables;(2) methods in the different classes happen interactions by calling each other. Fig. 1 shows the course that how to extract network from the method hierarchy.



```
Public class classX{
   public int a;
void methodA(){
       a++;
  }
void methodB(){
methodA();
  }
int methodC(int b){
int y;
y=b+a;
methodB();
  }
}
```

Figure 1. Extraction instructions of method granularity network

Definition 2. Class granularity of software networks (CN) is defined as a directed graph $CN = (V_C, E_C)$, where all classes and interfaces of specific object-oriented software can be treated as nodes n_c (where $n_c \in V_c$). The relationship between every pair of nodes if exists forms a directed link l_c (where $l_c \in E_c$) in the graph. In CN, the relationship occurs when one class uses the services provided by another class. Therefore, the links A → B can be defined if the following circumstances existing: (1) If class A inherits from another class B via keyword extends; (2) If class A realize interface B keyword implements; (3) If class B has an attribute with type of class A; (4) If one of class A's methods call a method on an object of class B, and so on. Fig. 2 shows the course that how to extract network from the class hierarchy.



```
class Point{
    int x,y;
  public fun1();
}
class Line{
public fun2(){
Point pos;
  pos.fun1();
}
 class Move{
   Point s_pos;
   Point e_pos;
   void deletePos();
}
class Pawn extends Line{
    Move *moves;
    int getPos();
}
```

Figure 2. Extraction instructions of class granularity network

Definition 3. Package granularity of software networks (PN) is defined as a directed graph $PN = (V_P, E_P)$, where all packages of specific object-oriented software can be treated as nodes n_p (where $n_p \in V_p$). The relationship between every pair of packages if exists forms a directed link l_p (where $l_p \in E_p$) in the graph. In PN, the relationships among packages indirectly derive from that among classes they contain, i. e., a relationship between two classes in two different packages indicates there exits a relationship between the two packages. Fig. 3 shows the course that how to extract network from the package hierarchy.

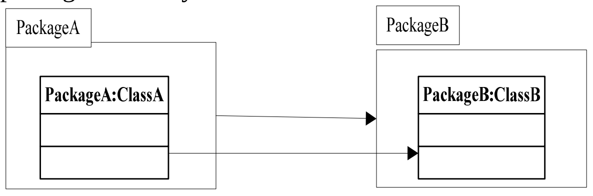


Figure 3. Extraction instructions of package granularity network

In the traditional object-oriented software design, most of the design principles and methods are all to the class level design for guidance. The previous jobs mainly analyzed and measured software networks characteristics at class level of granularity, so it will appear what kinds of feature at method level and package level, which are the main tasks of the section IV.

IV. MULTI-GRANULARITY ANALYSIS

In this section, we modify the operating parameters of Doxygen [11] software to analysis software source codes and generate the XML file including each unit of information as well as the interactive information between two units. Then we extract the XML file and finally get the nodes and links of software networks at three levels of granularity.

The analysis of object-oriented software from a multi-granularity perspective takes software system source codes as the research object, so it needs to obtain open source software. The granularity levels of software written by JAVA language are relatively clear, which contributes to abstracting software networks at different levels of granularity. Hundreds of open source software written by JAVA language are analyzed and calculated. The sets of software cover the extensive application fields and we take JBoss software as an example to analyze the multi-granularity software networks from the following aspects (other software researched in this paper also have similar properties). As we all know, JBoss is the joint efforts of worldwide developers and is an application server based on J2EE.

A. Structural Holes

Structural holes theory was first proposed by Ronald S. Burt [8] and was mainly used in the research of social network. The so-called structural hole is that, in the social network, one or some individuals have direct contact with some individuals, but have no direct contact with other individuals. Owing to the no direct contact or discontinuous contact phenomenon, the network structures appear the cave from the overall network perspective. In short, structural hole is a relationship of nonredundancy between two contacts.

The software networks formed by four nodes A, B, C, D are used to explain the structural holes. In Fig. 4 (a), node A has 3 structural holes: BC, BD and CD. There are no direct contacts among nodes B, C and D, and only node A directly connects with each of them. So, node A has distinctive advantages in getting the resources of network because other nodes connect with each other via node A. Fig. 4 (b) is a closed network and there is no structural holes in it.

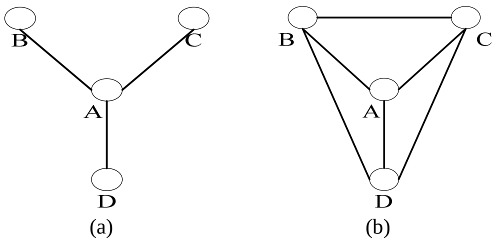


Figure 4. Examples of structural holes

Network Constraint Coefficient can be used to measure structural holes. This coefficient describes the close degree of direct connections or indirect connections between a network node and other nodes. The higher the coefficients are, the fewer structural holes are. The specific steps are as follows:

$$p_{ij} = \frac{d_{ij} + d_{ji}}{\sum_k (d_{ik} + d_{ki})} \quad (1)$$

Where p_{ij} is the ratio that the shortest path length between nodes i and j divided by the sum of the shortest path length between node i and its all adjacent nodes, while d_{ij} is the shortest path length between nodes i and j.

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

Where c_{ij} is the constraint degree when nodes i and j connect. When the only adjacent node for i is j, c_{ij} will take the maximum value 1; while j can not get indirect contact with i through other nodes, c_{ij} will take the minimum value p_{ij}^2. k is the adjacent node for nodes i and j.

By formulas (1) and (2), the Network Constraint Coefficient of node i can be calculated.

$$C_i = \sum_j c_{ij} \quad (3)$$

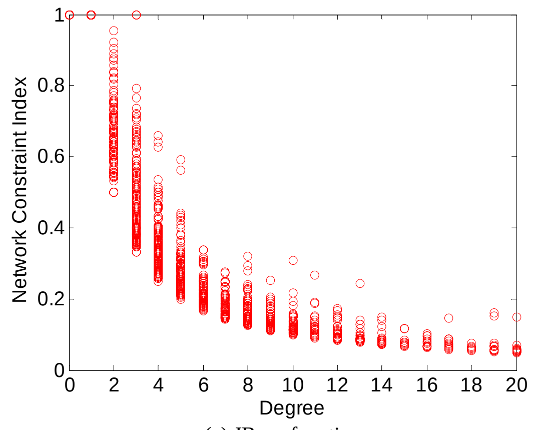
Structural holes are used to illustrate a node to other nodes in the dependency degree in software networks. Structural holes are fewer that show nodes to other nodes have stronger dependence. Structural holes are quantified by Network Constraint Coefficients. Through calculating the Network Constraint Coefficients of nodes, we can understand the extent of structural holes within software networks and get the distribution of node structural holes.

B. Structural Holes and Degree

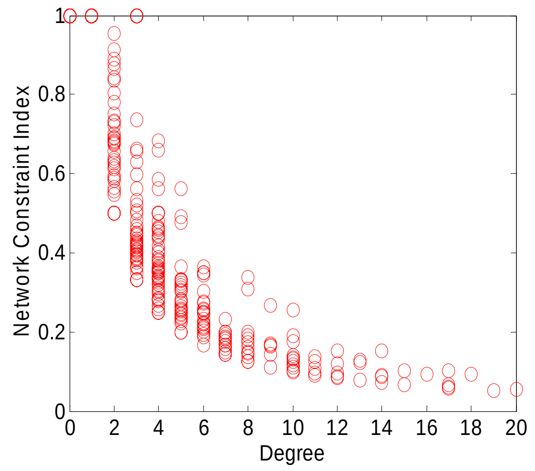
Degree of a node is defined as the number of neighbors connecting with it. We research the correlation between structural holes and degree at different levels of granularity in this paper. Structural holes are quantified by Network Constraint Coefficients, so correlation analysis between degree and structural hole that is correlation analysis between degree and Network Constraint Coefficient. Network Constraint Coefficient describes the close degree about a node with other nodes directly or indirectly link to. The higher the coefficients are, the higher network closure is, so the structural holes are fewer; while the smaller the coefficients are, the less connection among the network nodes is, so the structural holes are more.

In the case of JBoss, our observations of Fig. 5 (partial enlargement) shows that the software networks at all levels of granularity appear similar distributions. Along with the network node degree increasing, the Network Constraint Coefficients reduce gradually. That is to say, structural holes get more and more, and the corresponding nodes rely on their surrounding nodes relatively weak.

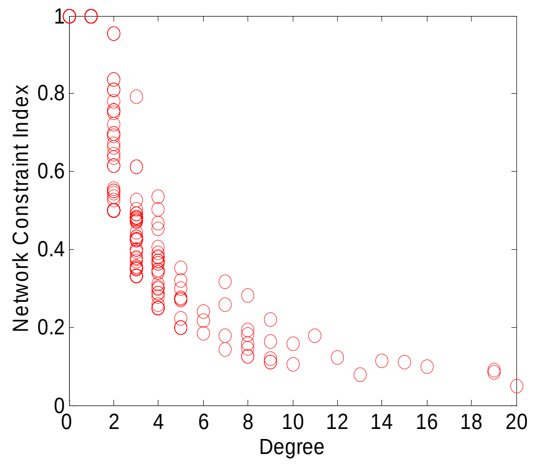
In addition, when some node degrees are small and Network Constraint Coefficients are 1, the software networks will appear “no hole"(see Fig. 4 (b)) structure. In this case, there are direct connections among the nodes and the corresponding software reuse is very obvious. The Network Constraint Coefficient of isolated nodes (degree is 0) is 1, there also will be “no hole"(see Fig. 4 (b)) structure. Figure 5 shows that the Network Constraint Coefficient of PN is 1 when the node degrees are 0 and 1; the Network Constraint Coefficient of CN, FN are 1 when the node degrees are 0, 1and 3.



(a) JBoss-function



(b) JBoss-class



(c) JBoss-package

Figure 5. The distribution of network Constraint Coefficient and degree of JBoss in multi-granularity software networks

Isolated node does not affect the analysis of the whole network characteristics, so we remove the isolated nodes and research the relation curve fitting between degree and Network Constraint Coefficient. The results are shown in Fig. 6.We choose the power function approximation as curve fitting, the equation as follows:

$$Y = a * X^{-b} \tag{4}$$

Where a is a constant, while b is the power index. Goodness-of-fit can well reflect the quality of curve fitting, and the Coefficient of Determination is usually used to judge the quality of regression model fitting degree. The correlation coefficient r is defined as follows:

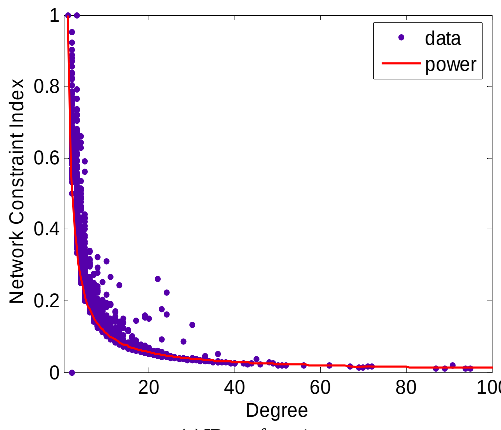
$$r = \frac{N\sum XY - \sum X \sum Y}{\sqrt{N\sum X^2 - (\sum X)^2}\sqrt{N\sum Y^2 - (\sum Y)^2}} \tag{5}$$

Where X , Y are Degree and Network Constraint Coefficient, N is the number of network nodes.

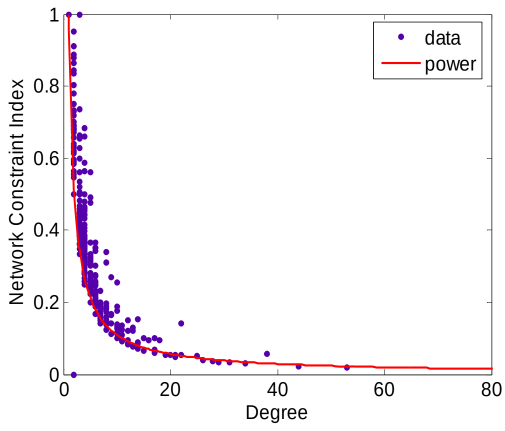
The Coefficient of Determination R is defined as follows:

$$R = r^2 \tag{6}$$

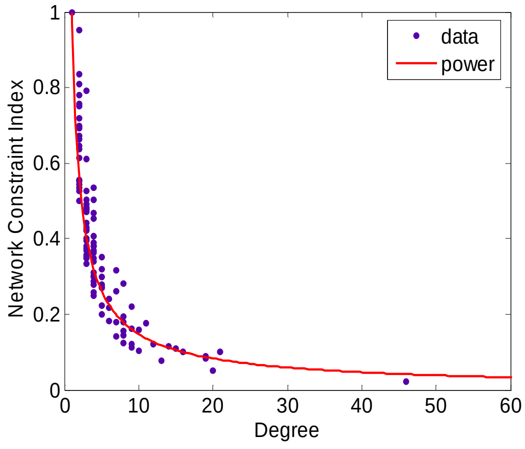
Coefficient of Determination and model fitting degree are positively correlated. We calculate the Coefficient of Determination and count the results (See table 1).



(a)JBoss-function



(b) JBoss-class



(c) JBoss-package
Figure 6. The fitness graph of relationship between network Constraint Coefficient and degree in multi-granularity software networks

TABLE1.
FITTING PARAMETERS OF NETWORK CONSTRAINT COEFFICIENT AND DEGREE

software	R	a	b
JBoss-package	0.9598	0.999	0.8308
JBoss-class	0.9233	0.9113	0.9316
JBoss-function	0.9876	0.9998	0.9645
JDK-package	0.9794	0.9958	0.8146
JDK-class	0.9162	0.9965	0.8801
JDK-function	0.9817	0.9989	0.9356
Vuze-package	0.9659	0.995	0.8673
Vuze-class	0.915	0.9964	0.8765
Vuze-function	0.9843	0.9997	0.9507

From the Table 1, we can see that the Coefficients of Determination of PN, CN and FN are in 0.91~ 0.99 ranges. It indicates that the goodness-of-fit of three networks are all good, the node degree and Coefficient of Determination are very accord with a power function relation, which reflect the modular design concept of object-oriented software. We then explain this phenomenon in software networks. The complex modules corresponding with low degree nodes are often broken down into more simple modules, and these simple modules cooperate to perform complicated work between each other. So the Network Constraint Coefficients of corresponding nodes are relatively high, the structural holes will reduce. Some modules corresponding with high degree nodes, whose internal logic and polymerization degree often have been very complex. So it should try to reduce the interactions between these modules and other modules. Reflecting in the multi-granularity software networks, the Network Constraint Coefficients of corresponding nodes are relatively small, the structural holes will increase accordingly.

According to the analysis of the above, researching the correlation between degree and structural holes contributes to exploring the collaborative relationship among different types of software entities, even finding the software entities with problems. On the other hand, it is useful for researchers to analysis the system hierarchy and modularity.

C. Closeness

Closeness is used to characterize the difficulty level that the network nodes through the network to reach other nodes in complex network. Closeness is defined as the reciprocal of the total distance from a node to all other nodes. Because the software is abstracted into a network, so closeness can be applied to software networks to measure the central degree of nodes.

$$C_c(x) = \left[\sum_{y=1}^{n} d_{xy}\right]^{-1} \tag{7}$$

Where d_{xy} is the shortest path length between nodes x and y, while n is the number of network nodes. In the network with n nodes, the sum of the distance from a node to all other nodes is not less than $n-1$, so the normalized closeness index is defined as follows:

$$C_C(x) = (n-1)C_c(x) \tag{8}$$

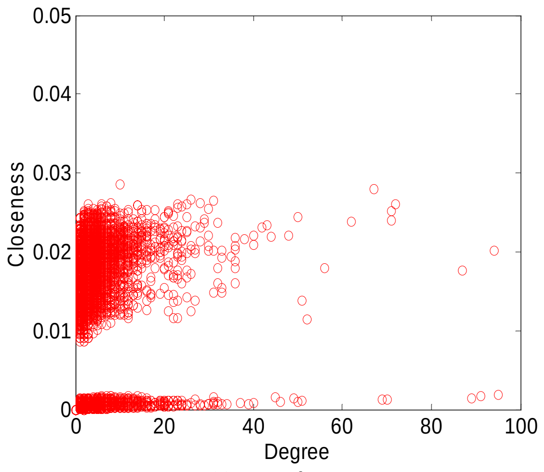
In software networks, node degree reflects a node to other nodes of the direct influence, while closeness reflects the ability of a node through the network exerting an influence on the other nodes. Degree mainly considers the reusability and complexity of nodes, while closeness not only takes into account the reusability and complexity, but also takes into account the nodes center level in the whole network, so closeness can reflect the overall structure of the network better.

D. Closeness and Degree

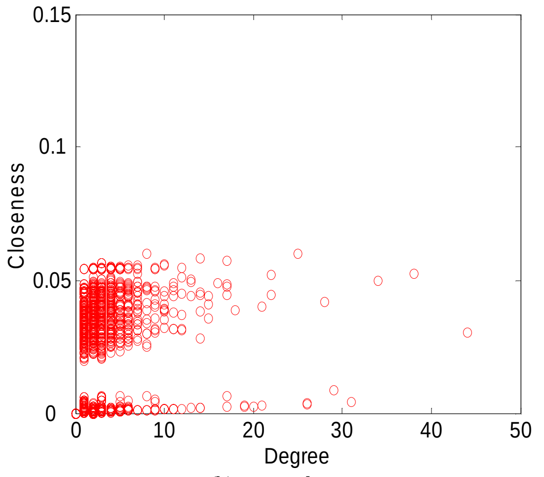
Compared to the degree, closeness considers more about the nodes center level in the whole network. According to the closeness size, nodes can be divided into the center nodes and edge nodes. Owing to the closeness is defined as the reciprocal of the total distance from a node to all other nodes. So we calculate the total distance, if a node has a small numerical value, its closeness will be larger and tends to be in the center of network. Instead, its closeness is very small, the node tends to be at the edge of network.

We examine the correlation between degree and closeness and notice that the three levels of networks have a similar distribution rule. From Fig. 7, we know that, as the degree increases, the closeness of PN, CN and FN grow slowly and tend to be stable, close to a numerical. In the case of JBoss, the closeness of PN, CN and FN are close to 0.1, 0.025 and 0.05, respectively.

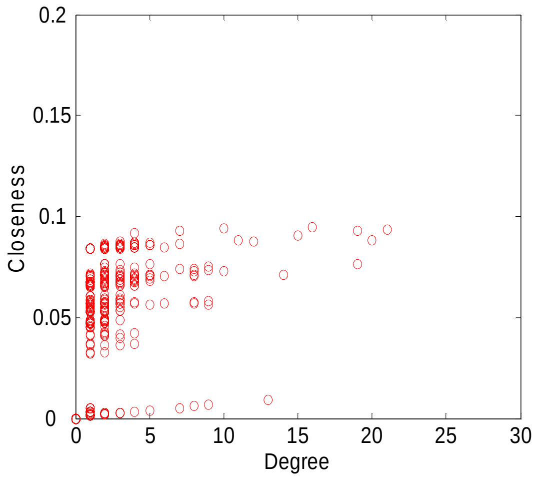
We also observe that the closeness of nodes is very small when its degree is relatively small, such kind of nodes are at the edge of network. This phenomenon is more obvious at the class level and method level .It indicates that the three levels of granularity networks all have edge nodes. The modules corresponding to these nodes do not play a significant role in intermediary, but only concern about their own internal logic within the software structure. Another case, some nodes with larger closeness are in the center of network, and eventually become stable, close to a certain numerical. The modules corresponding to such kind of nodes often have an obvious intermediary function within the software structure.



(a) JBoss-function



(b) JBoss-class



(c) JBoss-package

Figure 7. The relationship between degree and closeness of JBoss in multi-granularity software networks

For the software design, the edge node and central node all have their own important roles. The modules corresponding to the edge nodes should try to avoid being centralized. Relatively, we should try to avoid making the modules corresponding to the central nodes marginalize excessively. Because, edge nodes that being centralized too much will lead to module complexity substantially increased, and this complexity is often repeated and unnecessary. While central nodes that being marginalized redundantly will cause the cohesion degree of modules reduce greatly. Therefore, it is need to keep the mutual cooperation between central nodes and edge nodes in each granularity. Otherwise it will increase the complexity of software, which enlarges the probability of software error and may bring great loss to the company undertaking this project.

Further more, the closeness also needs to maintain stable. In the process of software evolution, if the closeness keeps stable, we can think that module cooperation exist stability. If closeness changes a lot, it will destroy the stable structure of software and reduce the stability of software.

V. Conclusions And Future Works

In this paper, we proposed an approach to study object-oriented software from different levels of granularity using the tools of complex network theory. We extracted the package granularity, class granularity and method granularity of three different levels of software networks. We introduced structural holes and closeness the two new parameters to software networks. Then we calculated and analyzed the parameters in multi-granularity software networks.The above results can provide valuable insights and a different dimension to our understanding of software topology properties and system structure. At the same time, it has great practical significance in controlling the complexity of software and researching the software design ideas how to influence of software architecture.

As a future work, we need more investigation on the following aspects: (1) researching multi-granularity software networks using other parameters so that we can gain more knowledge about software structure;(2) analyzing multi-granularity software networks constructed from other large Java software projects in order to obtain further evidence of software characteristics;(3) exploring the software evolution or weighted networks at different levels of granularity.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under grant No. 60973022, the Cultivation Fund of the Key Scientific and Technical Innovation Project , Ministry of Education of China under grant No.708026. Thanks for anonymous reviewers' valuable comments!

References

[1] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks", Nature, vo. 393, no. 6684, pp. 440-442, 1998.

[2] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," ACM

SIGCOMM Computer Communication Review, vol. 29, no. 4, pp. 251-262, 1999.
[3] R. Albert, H. Jeong, A. L. Barabasi, "Diameter of the world-wide web," Nature, vol. 401, pp. 130-131, 1999.
[4] G. Concas, M. Marchesi, S. Pinna, and N. Serra, "Power-laws in a large object-oriented software system," IEEE Transactions on Software Engineering, vol. 33, no. 10, pp. 687-708, 2007.
[5] Zhang HH, Zhao H, Cai W,et al. A metrics suite for static structure of large-scale software based on complex networks[C] // Intelligent Information Hiding and Multimedia Signal Processing. Washington D C: IEEE Computer Society, 2008:512-515.
[6] Cai W, Zhao H, Zhang H H,et al. Static structural complexity metrics for large-scale software[J].Special Issue on Software Engineering and Complex Networks of Dynamics of Continuous, Discrete and Impulsive Systems Series B, 2007,14(S6):12-17.
[7] Zhang HH, Zhao H, Cai W, Zhao M, Luo GL.A qualitative method for analysis the structure of software systems based on k-core[J].Special Issue on Software Engineering and Complex Networks of Dynamics of Continuous, Discrete and Impulsive Systems Series B,2007,14(S6):18-24.
[8] Ronald S B. Structural Holes [J]. The social structure of competition, 1992, 18(55):178-182.
[9] Li B, Pan WF, Lu JH. Multi-Granularity Dynamic Analysis of Complex Software Networks[C], Circuits and Systems (ISCAS), 2011 IEEE International Symposium on Digital Object Identifier, 2011: 2119 - 2124
[10] He KQ, Ma YT, Liu J, et al. Software networks [M]. Beijing: Science Press, 2008:1-117.
[11] Doxygen [EB/OL], http://www.doxygen.org, 2006.
[12] LaBelle N, Wallingford E. Inter-Package Dependency Networks in Open-Source Software, Submitted to Journal of Theoretical Computer Science, 2004
[13] Valverde S, Ferrer Cancho R and Sole R V. Scale-free networks from optimal design [J]. Europhysics Letters, 2002, 60(4):512-517.
[14] Valverde S, Sole R V. Hierarchical Small Worlds in Software Architecture. Working paper of Santa Fe Institute.2003, SFI/03-07-44.
[15] Valverde S, Sole R V. Universal properties of bipartite software graphs[C].In Proceedings of 9th IEEL International Conference on Engineering of Complex Computer Systems (Workshop on Software and complex systems), Italy, http://complex. Upf. Es/~sergi/valverdefir. PDF, 2004
[16] Hyland-Wood D, Carrington D and Kaplan S. Scale-Free Nature of Java Software Package, Class and Method Collaboration Graphs[R].Technical Report of MIND Laboratory (No. TR-MS1286), University of Maryland College Park. 2006.

Hui Li is an Associate Professor at college of Information Science and Engineering, Shenyang University of Technology, Shenyang, P.R. China. She has received B.S. degree in Measurement technology and instruments and M.S. degree in Communication and information system at Northeastern University. Now, she is working for PH.D degree in Navigation. Her main research interests are sensor network, complex network and navigation system.

Hui Zhang is a student at college of Information Science and Engineering, Shenyang University of Technology, Shenyang, P.R. China. Now, she is working for M.S. degree in Signal and information processing. She mainly researches software networks.

Hai Zhao is a professor at college of Information Science and Engineering, Northeastern University, Shenyang, P.R. China. His current interests include complex network, real-time system, and sensor network.

Wei Cai is a student at college of Information Science and Engineering, Northeastern University, Shenyang, P.R. China. Now, he is working for PH.D degree in Computer System Architecture. His current interest is in Software engineering.