International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.1 (2016), pp.221-228
http://dx.doi.org/10.14257/ijmue.2016.11.1.22

Research of Structural Risks of Object-Oriented Software Based on Ripple Degree of Software Network

[1]Shujuan Cui, [1]Sheng Bin and [2]Gengxin Sun

[1] *Software Technical College of Qingdao University, Qingdao, China*
[2] *International College of Qingdao University, Qingdao, China*
qdu_csj@163.com

Abstract

Object-oriented software systems emerge from mere keystrokes to form intricate functional networks connecting many collaborating objects. In this paper, building on complex networks, software collaboration networks contained within several open-source software systems have been examined. Because of the association between objects, the ripple effect exists in those software networks. The distributions of forward and reversal ripple degree in software networks are analyzed, and focusing on high ripple degree nodes, a metric formula that evaluates the significance is presented. According to our metric, the fragile nodes, rigid nodes and a "hidden danger" node in software system can be selected, which can be used to provide guidance for design and remodeling of software systems.

Keywords*: Software network, ripple degree, complex network, software structure*

1. Introduction

Structural complexity, which arise functional complexity of software systems, makes the control methods of traditional software quality to get good effect difficultly. Object-oriented software systems are represented as complex networks, which to date have received more and more attention in software structure field. From 2002, a large number of class diagram of object oriented software systems are studied by researchers of complex system and statistical physics. Relationships between classes are represented as a directed graph, then the overall properties of software network were studied through network topologies, it is the method that structure of software systems are researched by complex network theories [1-2].

With the development of large-scale distributed software system based on Internet, there are more and more researchers who pay close attention to the change of software function. Bohner [3-4] proposed a process analysis framework of software change, which firstly used "ripple effect" to describe impact of software change. Ahmed Breech [5-7] studied effect of entity (function or variable) changes on other related entities in view of entity changes of software systems. Chen [8] proposed a model of the effect of software change based objects and its attributes.

During the process of software development and maintenance, the rationality of system internal structure would reduce software risk and reduce cost of system maintenance. In this paper, ripple effect which is caused by nodes change in object-oriented software network is analyzed, ripple scope and rule of nodes are determined. Our works can predict the risk of possible structural defects in the system, and provide guidance for the design and maintenance of software.

ISSN: 1975-0080 IJMUE

2. Ripple Effect in Software Network

Object-oriented software systems are the main research objects in this paper, the process of establishing of software systems network model as follows: Firstly, source code would be abstracted as class diagram, then class is regarded as node in network, relationship between classes is edge between nodes. Thus, the topology model of software network can be constructed.

The relationships between classed include Generalization, Realization, Association, Aggregation, Usage, Dependency, and so on. The close degrees of relationships between classes are different with different relationships, in order to more accurately describe the structure of software system, the different weights are given to edges of software network. Software weighting network model is defined as follows.

$$N = (V, A, T, W) \tag{1}$$

Where $V = \{v_i \mid i = 1, 2, \ldots, n\}$ represents nodes set of network, each node corresponds to a class in the software system, $T = \{t \mid t \in \{G, U, D\}\}$, where G represents Generalization relationship, U represents Usage relationship, D represents Dependency relationship. $A = \{(v_i, v_j, t) \mid v_i \in V, v_j \in V, t \in T, i \neq j\}$ represents edges set of network, each edge corresponds to a relationship between two classes in the software system. But the relationship between two classes may not be the only in actual software system, for example, there is not only dependency relationship but also usage relationship between class A and class B. So the type of each edge t is a stowed value, $W = \{w \mid w = \max(w(t)), t \in T\}$ represents weight of each edge. The size of the weights is decided by the close degree according to the mutual connection between two nodes. In the weighted software network, the weight of edge is larger, the interaction between two nodes is more closely.

In the UML class diagrams there are 9 relationships between classes, but there are no significant differences among aggregation relationship, usage relationship, association relationship, they can only be distinguished by the semantic, the above several relationships are unified into the usage relationship in this paper, at the same time, realization relationship and generalization relationship are unified into the generalization relationship. According to the close degree of relationship between classed in UML class diagrams, the weight of dependency relationship is assigned to 0.1, the weight of usage relationship is assigned to 0.4, and the weight of generalization relationship is assigned to 0.7. If there are two or more than two kinds of relationship between two nodes, the weight is the maximum one.

In software network, ripple effect indicates that when a node arise failure or change, which would grow like a weed with association with other nodes, then it would influence other parts of the system. From the view of definition of ripple effect, node in software network can become the makers of ripple effect, also can be affected object by ripple effect. So ripple effect on a node is divided into forward ripple degree and reverse ripple degree.

Definition 1: forward ripple degree: from node v_i of software network, the product of weight on the shortest path of all nodes along the forward edge can be reached. It can be defined as follows.

$$RF_{WG}(v_i) = \sum_{j \neq i}^{v_j \in TF(v_i)} W_{ij}, W_{ij} = \prod_{i \neq j}^{w_{mn} \in SP_{F(ij)}} w_{mn} \tag{2}$$

Where $TF(v_i)$ is a set which includes nodes can be reached from node v_i along the forward edge. W_{ij} is the total weight of node v_i with one node v_j in $TF(v_i)$, it can be calculated by the product of weight on the shortest path between node v_i and node v_j along the forward edge. When the out-degree of node v_i is zero, $RF_{WG}(v_i) = 0$.

Forward ripple degree reflects affected level of a node by other nodes in the network, the nodes whose forward ripple degree are large easily lead to Fragility of software system, they should be highly concerned about in the design of software structure.

Definition 2: reverse ripple degree: from node v_i of software network, the product of weight on the shortest path of all nodes along the reverse edge can be reached. It can be defined as follows.

$$RR_{WG}(v_i) = \sum_{j \neq i}^{v_j \in TR(v_i)} W_{ij}, W_{ij} = \prod_{i \neq j}^{w_{mn} \in SP_{R(ij)}} w_{mn} \quad (3)$$

Where $TR(v_i)$ is a set which includes nodes can be reached from node v_i along the reverse edge. W_{ij} is the total weight of node v_i with one node v_j in $TR(v_i)$, it can be calculated by the product of weight on the shortest path between node v_i and node v_j along the reverse edge. When the in-degree of node v_i is zero, $RR_{WG}(v_i) = 0$.

Reverse ripple degree reflects affected level of a node for other nodes in the network, it most directly reflects the ripple effect. Ordinarily those nodes whose reverse ripple degrees are large may lead to rigidity of software system, they should be highly concerned about in the design and maintenance of software structure

3. Software Ripple Degree Analysis

According to definition of ripple degree, 125 open source software samples are selected to ripple degree analysis. These software samples are all object-oriented software, they cover the development tools, application software, system software, entertainment software and compile software *etc*. Programming languages include C++, Java and C#.

Firstly, according to definition of forward ripple degree and reverse ripple degree, ripple degree of each node in software network is calculated, the interval distribution frequency of node ripple degree were analyzed statistically by 0.5 as interval. We found that with increasing of interval value, occurences frequency of forward ripple degree and reverse ripple degree all present changing trend of quick decline firstly and approaching zero at last.

According to definition of forward ripple degree, we can know that those nodes whose forward ripple degree are large mostly located in the higher level of software system, they depend on a large number of other nodes from the view of software structure and become weak links of software system. Therefore, the number of such nodes can not be too much in the whole system for avoiding fragility appearance. On the contrary, those nodes whose reverse ripple degree are large mostly are base class or interface which located in the lower level of software system, if those nodes change, it would lead to nodes in a large range to make corresponding adjustments. So the number of such nodes also can not be too much in the whole system for avoiding rigidity appearance.

From the above analysis, we can see that distribution law of ripple degree reflect commendably design principles of software. For a node, if its forward ripple degree is larger, it means that it reuse a large number of low lever nodes and would be easy to suffer ripple effect. In order to avoid a greater range of ripple effect, such nodes should be tried to avoid reusing or associating, so their reverse ripple degree should not be too large. Likewise, for nodes whose reverse ripple degree are larger, it is very important to maintain their stable structure because of their important position in software system, so these nodes should be located outside the ripple range of other nodes as far as possible, it means that their forward ripple degree should not be too large. It can be seen that there is disassortative between forward ripple degree and reverse ripple degree of the same node.

To prove the point, joint distribution of forward ripple degree and reverse ripple degree are counted statistically for various software system, statistics results reveal that reverse ripple degree of those nodes whose forward ripple degree are larger are not too large and vice versa. Taking Firefox for example, its joint distribution of forward ripple degree and reverse ripple degree are shown as Figure 1.

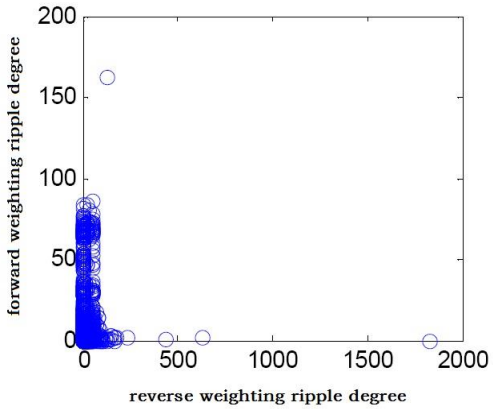

Figure 1. Joint Distribution of Forward and Reversal Ripple Degree of Firefox

From Figure 1 we can see that forward and reversal ripple degree of most nodes are small, those nodes located in the lower left corner of the figure.

In some other software, there would be another kind of nodes, whose forward and reversal ripple degree are all larger. For example, joint distribution of forward ripple degree and reverse ripple degree of eMule are shown as Figure 2. Several nodes of eMule are such nodes whose forward and reversal ripple degree are all larger. This kind of nodes should be avoided in software design.

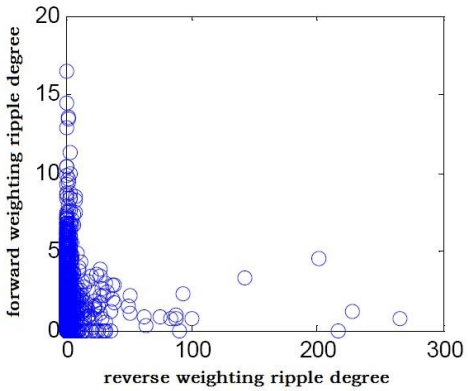

Figure 2. Joint Distribution of Forward and Reversal Ripple Degree of Emule

Through analysis of node ripple degree distribution, we can found that there are most nodes whose forward and reverse ripple degree are all smaller in software system, there are only a small part of nodes whose forward or reverse ripple degree is larger, there are scarcely any nodes whose forward and reverse ripple degree are all lager. Those nodes whose forward and reverse ripple degree are all lager are "hidden danger" nodes, which should be avoided in software design.

4. Significant Measures of Ripple Degree

In the process of nodes ripple degree analysis of sample software, we found that a wide variation in the max value of nodes forward and reverse ripple degree of each software. Then we fit the software scale with average value and max value of forward and reverse ripple degree, the results show that there are no significant correlation between the software scale and average ripple degree. In order to find fragile nodes, rigid nodes and "hidden danger" nodes in software structure, significant measures are used to judge above three kinds of nodes in this paper.

4.1. Significant Measures of Forward Ripple Degree

Because numerical range of forward ripple degree is very different for different software, in order to find those nodes whose forward ripple degree are larger, significant measure formula of forward ripple degree is proposed, it can be used to measure node forward ripple degree.

$$Z_{RF^{w}(v)} = \frac{RF_{WG}(v) - RF_{WG}avg}{RF_{SD}^{W}} \tag{4}$$

Where $RF_{WG}(v)$ represents forward ripple degree of node, $RF_{WG}avg$ represents average forward ripple degree of the whole software system, RF_{SD} represents standard deviations of forward local ripple degree.

Six sample software are randomly selected to calculate significant measure of forward ripple degree according to eq. (4), the statistical results are shown as Figure 3.

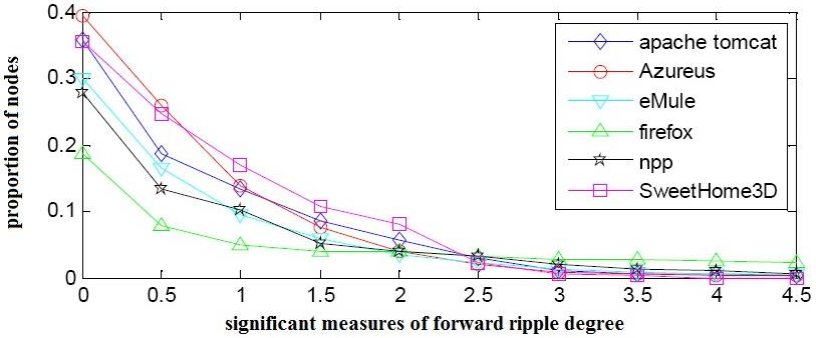

Figure 3. Statistics of Forward Ripple Degree's Significant Measure

In Figure 3, abscissa represents significant measure of forward ripple degree, ordinate represents the proportion of those nodes whose forward ripple degree are greater than $Z_{RF}(v)$ in the total number of nodes. From Figure 3 we can see that distribution curves of software forward ripple degree' significant measure show the similar trend, that is, with increasing of measure value, node ratio decreased rapidly, when measure value is greater than 2.5, change trend of node ration would be stable. The same measure have been done for the remaining samples in software sample, the same statistical results have been got.

Through access to relevant software source code and its open development document, we found that those nodes whose measure value is greater than 2.5 depend on or correlate with a large number of other classes, and they belong to fragile nodes and should be paid special attention by relevant personnel in later software maintenance.

4.2. Significant Measures of Reverse Ripple Degree

Similarly to forward ripple degree, numerical range of node reverse ripple degree is very different for different software, so significant measure formula of node reverse ripple degree is proposed.

$$Z_{RR^{w}(v)} = \frac{RR_{WG}(v) - RR_{WG}avg}{RR_{SD}^{W}} \tag{5}$$

Where $RR_{WG}(v)$ represents reverse ripple degree of node, $RR_{WG}avg$ represents average reverse ripple degree of the whole software system, RR_{SD} represents standard deviations of reverse local ripple degree.

We also select six sample software randomly to calculate significant measure of reserve ripple degree according to eq. (5), the statistical results are shown as Figure 4.

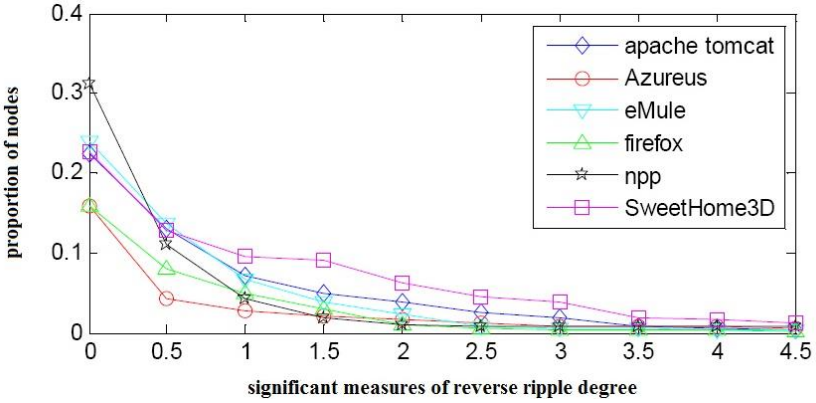

Figure 4. Statistics of Reverse Ripple Degree's Significant Measure

From Figure 3 we can see that with increasing of measure value, firstly node ratio decreased rapidly, then it would flatten out. When measure value of node is greater than 2.5, the proportion of nodes reduces to below 3%.

Through access to relevant software source code, we found that those nodes whose measure value is greater than 2.5 are all base classes or interfaces in software system, they would be inherited, depended or used by a great deal of nodes. They belong to rigid nodes and should be ensured their stability, and try to minimize the change in the latter maintenance process.

4.3. Significant Measures of Joint Ripple Degree

Those nodes whose forward and reverse ripple degree are all larger are the key nodes in the structure of software system. For such nodes, the formula of node joint ripple degree is proposed.

$$R_{WG}(v) = RR_{WG}(v) * (1 + RF_{WG}(v)) \tag{6}$$

Where $RF_{WG}(v)$ represents forward ripple degree of node, $RR_{WG}(v)$ represents reverse ripple degree of node, $1+RF_{WG}(v)$ represents ripple effect caused by node v (including itself) . The significant measure formula of node joint ripple degree as follows.

$$Z_{R_{WG}(v)} = \frac{R_{WG}(v) - R_{WG}avg}{R_{SD}^{W}} \tag{7}$$

Where $R_{WG}(v)$ represents joint ripple degree of node, $R_{WG}avg$ represents average joint ripple degree of the whole software system, R_{SD} represents standard deviations of joint local ripple degree.

We measure statistically significant measure of joint ripple degree for all sample software, statistical results show that distribution law of significant measure of joint ripple degree for all sample software are nearly consistent. Six random sample software are taken as example, their distribution of measure value shown as follows.

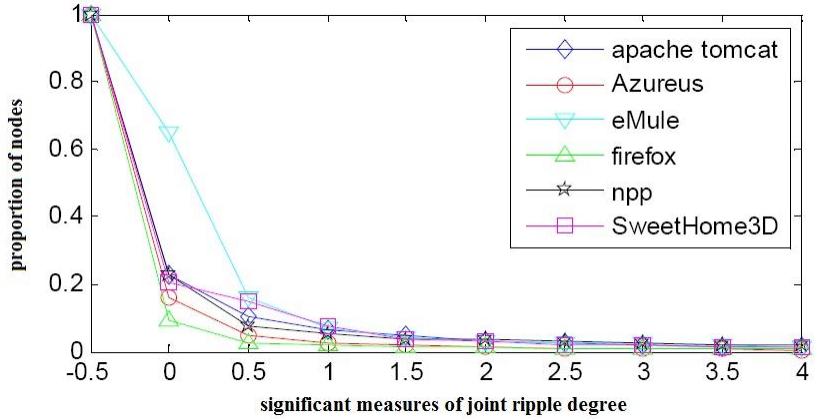

Figure 5. Statistics of Joint Ripple Degree's Significant Measure

From Figure 5 we can see that the significant measure of joint ripple degree of all nodes are higher than -0.5, and measure value of over 80% nodes are located in [-0.5,0.5].

According to the software test report and CVS records, when the significant measure value of joint ripple degree are higher than 6, the number of appearing problems or needing be modified are obviously higher than other nodes. So we select 6 as threshold value of significant measure, when the significant measure value of joint ripple degree are higher than 6, the node would be regarded as the key node in software system, once it suffers attack, the whole software network will be seriously threatened, even would cause the entire software system crashed.

5. Conclusion

Ripple effect of software network reflects interactive influence in software structure. In the process of software development, the software structure which has been established make analysis and measurement of ripple effect in time, it can find fragile nodes, rigid nodes and "hidden danger" nodes in software structure as early as possible so as to reduce software risk. In this paper through analysis of ripple degree, significant measures of ripple degree are proposed, according to the significant measures, above three kinds of nodes would be easy to find in software system.

References

[1] R. Vasa, J. G. Schneider and C. Woodward, "Detecting structural changes in object oriented software systems", Proceedings of the 10th 2005 International Symposium on Empirical Software Engineering, **(2005)**.
[2] R. Vasa, J. G. Schneider and O. Nierstrasz, "The inevitable stability of software change", Proceedings of the International Conference on Software Maintenance, **(2007)**.
[3] S. A. Bohner, "Impact analysis in the software change process: A year 2000 perspective", Proceedings of the 1996 International Conference on Software Maintenance, **(1996)**.
[4] S. A. Bohner, "Software change impacts: An evolving perspective", Proceedings of the International Conference of software Maintenance, **(2002)**.
[5] A. E. Hassan and C. H. Richard, "Predicting change propagation in software system", Proceedings of the 20th IEEE International Conference on Software Maintenance, **(2004)**.
[6] B. Breech, A. Danalis and S. Shindo, "Online impact analysis via dynamic compilation technology", Proceedings of the 20th IEEE International Conference on Software Maintenance, **(2004)**.
[7] H. Malik and A. E. Hassan, "Supporting software evolution using adaptive change propagation heuristics", Proceedings of the 20th IEEE International Conference on Software Maintenance, **(2004)**.
[8] C. Y. Chen, "An object-based, attribute-oriented approach for software change impact analysis", Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management, **(2007)**.

Authors

Shujuan Cui, She is currently a lecturer in Software Technology College of Qingdao University. Her main research interests include complex networks, web information retrieval and object- oriented programming.

Sheng Bin, She received her Ph.D. degree in Computer Science from Shandong University of Science and Technology, China in 2009. She is currently a lecturer in the School of Software Technology at Qingdao University, China. Her main research interests include embedded system, operating system, complex networks, cloud computing and data mining.

Gengxin Sun, He received his Ph.D. degree in Computer Science from Qingdao University, China in 2013. He is currently an Associate Professor in the School of Computer Science and Engineering at Qingdao University. His main research interests include embedded system, operating system, complex networks, web information retrieval and data mining.

