Study and Metric on Macro-Topology Characteristic Values of Software Networks

Zheng Liu, Hai Zhao
College of Information Science and Engineering, Northeastern University, Shenyang, China
Email: liuzheng@mail.neu.edu.cn

Qian Zhang
Department of TSD, NeuSoft Company, Shenyang, China
Email: zhangqian@neusoft.com

Abstract—**According to the research progress of software networks in these years, the software structural networks show the same characteristics with complex networks and the software topological structure has an effect on the quality characteristics of software systems. The research work begins with the statistics analysis of three characteristic values, which can reflect the efficiency, complexity and orderliness of software systems separately, of 500 open source software systems. Then D(AP), D(AD) and D(E) are presented for measuring the quality characteristics of software systems refer to the definition of deviation of characteristic value and quality characteristics relative to software scale. By testing on seven Linux kernels of different versions, the metrics on software networks based on deviation of characteristic values is validated effective.**

Index Terms—**Deviation of characteristic value, Metrics of software, Macro-topology, Software network**

I. INTRODUCTION

Nowadays, the software scale is increasing at a rapid rate in order to fit the more complex application, which results in the quality of software system beyond control. Those software metric sets applied for several years cannot deal with the enormous complexity of a software system. So how to find an adequate metric set for large-scale software systems is a challenge for software engineers. Some researchers have studied on a great deal of orient-object software networks and found the structure of these software systems are not random and out of order, instead, most of them present the global statistical features such as “small world” property and “scale free” property. According to massive experiment data, Myers pointed out that large-scale software systems represent a class of artificial complex networks, and this point are approved by more and more researchers. Since then, more and more study on the characters of software networks has shown that the statistic characteristics can reflect the quality of software system factually. So measuring the complexity of a large-scale software structure based on software networks provides the foundation for our research works.

According to the statistic characteristics of software networks macro-topology characteristic values, the definitions of deviation of characteristic values of software networks and relative quality characteristics of software on scale are presented in this paper. Then from this perspective, we evaluate the average shortest path length deviation, the average degree deviation as well as the standard structure entropy deviation to measure the efficiency, structural complexity and orderliness of software system. These new measure methods lay a good foundation for building an effective metrics set for large-scale software system in the future.

II. RELATED WORKS

In the later 1990s, the configuration, design and development method of a software system had changed from oriented-object paradigm to oriented-network paradigm along with the rise and development of technique of web service, SOA and semantic web [1]. The particle size of software object is becoming larger, the topology is getting more and more complex, and the coupling relation between objects is looser than before, which lead to the rapid increase of the scale and complexity of software systems. During this period, some mainstream metrics for OO software are recognized and applied extensively [2, 3]. Meanwhile, some new measure methods were proposed in succession, such as the property-based software engineering measurement proposed by Briand [4], the method using function points by Furey [5], the measurement on software size and productivity rating by Arnold and Pedross [6], and the method of using combinations of metrics by Bauer [7]. Later at the end of 20th century, Fenton et al summarized and evaluated all the methods presented above and considered that these methods modified Chidamber & Kemerer metrics (CK) and Metrics for OO design (MOOD) and enrich the metrics for OO software, but there were still limitation in predicting the quality of software system that due to the inadequate understanding of the complexity of software system.

Recently, researchers began to study the complexity of software system appearing in global aspect using theory of complex networks, which can describe the global

doi:10.4304/jsw.6.3.460-467

structure and actions of software systems more exactly. Started from the change of software structure, Vasa et al [8, 9] predicted the scale and constructed cost of a software system by the relation between the numbers of arcs and vertexes in software network and proposed a metrics set for the stability of software structure. This method plays a guiding role in software development. Girolamo et al [10] detected the structural faults using some properties of software network such as betweenness from class-level, network-level and design-level separately for evaluating the quality of software system. Ma et al [11] proposed a multi-level metrics set which gathered all the available methods and applied them measure a whole software system on code-level, class-level and system-level according to their characteristics. Li et al [12] introduced network evolution model into complexity metrics of software system, and proposed an evolution model CN-EM to describe the process of evolution that the properties of complex networks appearing in practical software system. Melton et al [13] studied the dependent relation between classes in 81 open source software systems, and revealed one of the causes of creating dependent cycle so that guide programmers to improve their work. Until now, studying the characteristics of macro-topology characteristic values in software network is starting for a while and there isn't integrated system to support such theory. So it is important to continue this research work further.

III. MACRO-TOPOLOGY CHARACTERISTIC VALUES OF SOFTWARE NETWORK

A. Static structure of software network

Static structure of software network is a network model in which the function, class or model in source code is abstracted as vertex and the relation (including calling, inheritance and reference etc) between code blocks mentioned above is abstracted as arc. It is always called software network. Analyzing on such structure got by the method of re-engineering can help us obtain the global characteristics of software system.

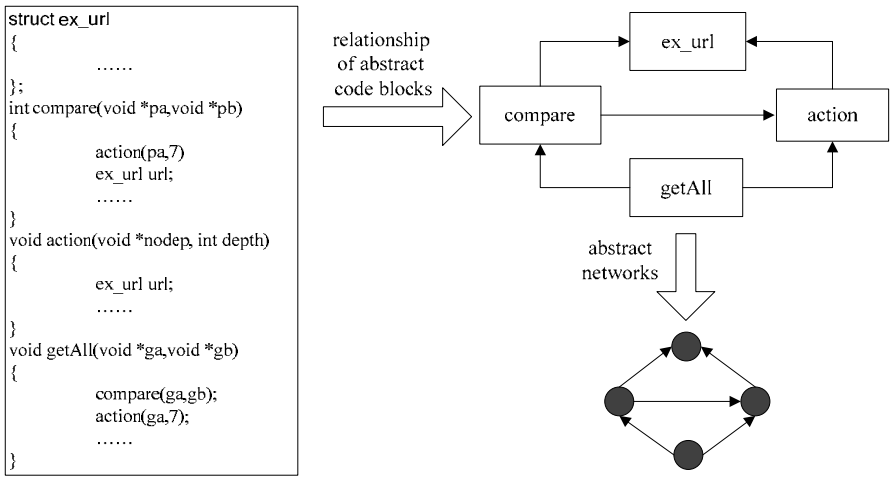

Figure 1. The analysis method of open-source software system architecture

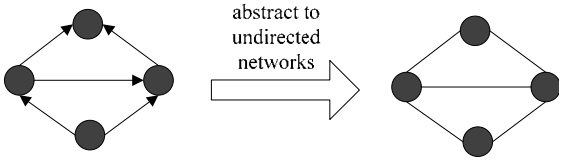

Figure 2. The analysis method of open-source software system architecture for undirected graph

The static macro-topology characteristic values of software network studied in this paper are based on undirected networks. So we should do more abstract work on the abstracted result shown as Figure 1 that is transforming the directed relation such as calling, inheritance and reference etc to simple undirected relation and then gaining undirected networks. This course is shown as Figure 2.

B. Macro-topology characteristic values

In order to evaluate large-scale software systems, they are always treated as networks. By analyzing macro-topology characteristic values of complex networks in software networks, researchers can get the global characteristics of them and then give exact evaluation of quality of software systems. In this paper, three characteristic values of them are used to measure a large-scale OO software system, which are average shortest path length, average degree of each vertex and the standard structural entropy.

Average shortest path length is also called characteristics path length, which presents the average value of the shortest distance between any pair of vertexes through a graph. This characteristic value describes the degree of separation of vertexes, which is called network diameter. Most of the large-scale networks are found that their diameters are much shorter than expectation, and this phenomenon is called "small world".

Degree of a vertex is defined as the number of neighbors connecting with it, and the average value of all the vertexes' degree is called average degree. Degree distribution, describing statistical property of system

structure, indicates the connectivity of each vertex in graph. On the other hand, degree of a vertex reflects the importance of this vertex in graph. The larger the in-degree is, the higher the possibility of being reused may be; and the larger the out-degree is, the more complex the vertex will be.

Entropy is a state function whose value keeps invariant when the state of a system is determined. For information entropy is a metrics for orderliness of systems that is larger when the system is in disordered state while is smaller when the system orderliness is improved. The evolution of systems can be described by the relation between entropy and orderliness of systems.

The network structure entropy is defined as follows [14]:

$$H = -\sum_{k=1}^{n} p(k)\log_2 p(k) \quad (1)$$

p(k) is degree distribution, n is the whole number of vertexes in graph. According to the extreme value of information entropy, the network structure entropy reaches its maximum when the structure of network is uniform, and $H_{\max} = \log_2 n$; when the structure of network is star-shaped, the entropy is minimum and $H_{\min} = \log_2 n - [n(n-1)]\log_2 (n-1)$. It is known from the analysis above that the network structure entropy and both of its maximum and minimum values all associate with the number of vertexes in software network. In order to eliminate the influence of different software samples, standard structure entropy of software network is defined as follows [14]:

$$H_s = \frac{H - H_{\min}}{H_{\max} - H_{\min}} \quad (2)$$

H_s is independent of the scale of software system and it has the same changing trend with network structure entropy. So we can study the evolution of network macro-topology structure by the variation of H_s.

IV. METRICS METHOD BASED ON DEVIATION OF CHARACTERISTIC VALUES

A. related definitions

Measuring the quality of software systems by quantifying the characteristics of software static macro-topology characteristic values is a metrics method discussed in this paper, and the main metrics parameters called deviation of characteristic values of software network is defined as follows:

Definition 1. Deviation of characteristic values of software network. It describes the deviation between the characteristic values of software measured and the average characteristic values of all the software widely used at present which have the same size, and it is marked as *D*.

According to the macro-topology characteristic values of software network, there are deviation of average shortest path length, deviation of average degree and deviation of standard structure entropy used for measuring the quality of software systems in this paper.

It is known that the scale of a software system determines if its quality level can be accepted. For example, the same complexity accepted by a large-scale software system with more than 5000 classes may not be accepted by a small-scale one with only 100 classes. So the quality characteristics of a software system discussed here is a value relative to the software scale, and its definition is as follows:

Definition 2. Quality characteristics relative to software scale. It indicates the quality characteristics that can be accepted for a software system with given scale.

According to the deviation of characteristic values of software network discussed, three quality characteristics relative to software scale are used for measuring a software system, which are efficiency, complexity and orderliness. We select 500 software systems from different open-source web as sample data for all the following statistical experiments.

B. Metrics on deviation of average shortest path length

(1) Average shortest path length and the efficiency of software relative to its scale

The average shortest path length means the average shortest path length between any pair of vertexes in software networks, which reflects the average minimum depth of function call in code structure. Because of the negative correlation between functions nesting depth and efficiency of software, the average shortest path length is the metrics for software efficiency: The longer the average shortest path length is the deeper the functions nesting are, and the efficiency of software will be affected negatively in the meantime.

(2) Deviation of average shortest path length

First, we do some statistical jobs on the sample software systems to obtain the distribution of average shortest path length, and the result is in table 1 shown as follows.

Table 1. Statistics of the average shortest path length

Stat.	Value
Average	2.4085457875
Median	2.3371044040
Minimum	1.0333333333
Maximum	5.6090730700

It is shown from table 1 that the distribution interval of the average shortest path length is from 1.0333333333 to 5.6090730700, and most of the path length values are around 2.35. Then, the changing trend of the average shortest path length along with increasing of the number of nodes is analyzed.

Figure 3 describes that the average shortest path length increases with the number of nodes increasing, and after the number of nodes reaches 4000, the increasing trend of the average shortest path length slow down. The fitting equation of the curve in Fig.3 is

$$Y_L = 0.6858670534562866 \times 1.69117523938839^{X} \quad (3)$$

The fitting coefficient R=0.895.

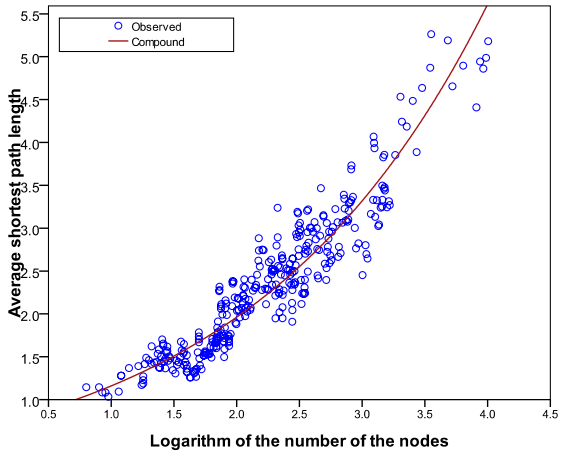

Figure 3. Average shortest path length and the number of nodes

According to the definition of deviation of characteristic value and the characteristics of average shortest path length, the deviation of average shortest path length in software networks is defined as follows:

Definition 3: Deviation of average shortest path length in software networks is the difference between the average shortest path length of given software system and its average level of those open-source software systems with the same scale which are used widely at present. It is marked as *D (AP)*, and there is $D(AP) = L - Y_L$. Among them, L is the average shortest path length of software system being measured and Y_L is the average level of average shortest path length calculated by formula (3).

Because the average shortest path length has negative influence on efficiency of software system, we draw the conclusion that if D(AP)>0, the average shortest path length of given software system is longer than the average level of other software systems with the same scale, that means there is still improvement room on efficiency by reducing the function nesting depth; while, if D(AP)$\leqslant$0, the average shortest path length of this software system is shorter or equal to the average level, that means this eigenvalue has little negative influence on its efficiency.

C. Metrics on deviation of average degree

(1) Average degree and complexity of software relative to its scale

Software networks are abstracted from software structure directly, so according to the definition and characteristics of average degree of software networks, average degree of a software system can reflect the structural complexity of this system: the larger the average degree is, the more complex the system is, and on the contrary, a system with lesser average degree always has lower complexity.

(2) Deviation of average degree

By analyzing on the average degree of all the software samples, we get the statistics data of average degree as table 2 listing.

Table 2. Statistics of the average degree

Stat.	Value
Average	2.4085457875
Median	2.3371044040
Minimum	1.0333333333
Maximum	5.6090730700

As it is shown from table 2 that the average degree distributes from 0.942 to 3.986, the span is a little wide. But the average value and median value of it are 1.97055 and 1.99620, which means the average degree of most software networks is around 1.98. Then in order to analyze on the relationship between the number of nodes and the average degree, plotting and fitting the change trend of average degree with the increasing of the number of nodes in Figure 4.

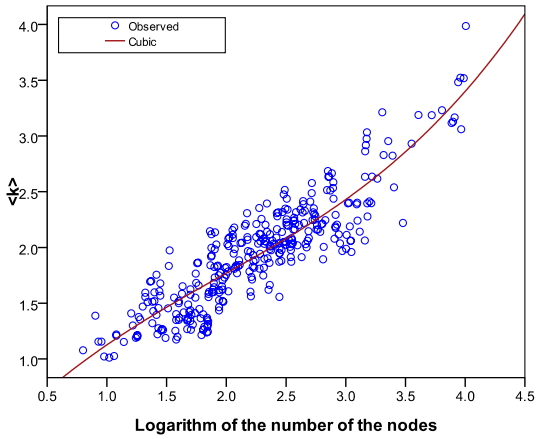

Figure 4. Average degree and the number of nodes

From the picture above, it is found the average degree of software networks concentrating on the interval from 1 to 3, and its value becomes larger along with the increase of the number of nodes. Especially when the logarithm of number of nodes is beyond 3.5, the value increases more rapidly. In order to analyze the relationship between average degree and the number of nodes, we do some fitting job on the distribution of average degree. The fitting equation in Figure 4 is

$$Y_{<k>} = 0.2063910019675122 + 1.15578778028317 \times X - 0.2847639420603531 \times X^2 + 0.04886675471683427 \times X^3 \quad (4)$$

The fitting coefficient R=0.801

According to the definition of deviation of characteristic value and the characteristics of average degree, the deviation of average degree of software networks is defined as follows:

Definition 4: Deviation of average degree of software networks is the difference between the average degree of given software system and its average level of those open-source software systems with the same scale which are used widely at present. It is marked as *D(AD)*, and there is $D(AD) = K_{<k>} - Y_{<k>}$. Among this, $K_{<k>}$ is the average degree of the software system being measured, and $Y_{<k>}$ is the average level of that calculated by formula (4).

Because of the relation between average degree in software networks and the complexity of this system, we can draw some conclusions that if *D (AD)*>0, the average degree of given software network is lager than the average level of other software systems with the same scale, which means the complexity of this software structure is too high and there is still improvement room to reduce it by decreasing average degree; on the contrary, the complexity of software structure is acceptable if *D (AD)*⩽0.

D. Metrics on deviation of standard structure entropy

(1) Standard structure entropy and the orderliness of software structure

According to the definition of standard structure entropy of software networks, the standard structure entropy is metric for the orderliness of software networks, and further, the orderliness of software networks can reflect the orderliness of software code structure. So we can draw a conclusion that the lesser the standard structure entropy is, the more orderly the software code structure and on the contrary, the larger the standard structure entropy is, the more complex and disorderly the software structure is. On the other hand, the maintainability of software system is affected by the complexity of software structure, so the standard structure entropy also bring effect on the maintainability.

(2) Deviation of standard structure entropy

Analyzing on standard structure entropy of all the sample software networks, we obtain the statistical data about the standard structure entropy shown in table 3 as follow:

Table 3. Statistics of the standard structure entropy

Stat.	Value
Average	0.3263393388
Median	0.326334383
Minimum	0.208619728
Maximum	0.52204386

It is shown in table 3 that the standard structure entropy distribute in a large span interval of [0.208619728 , 0.52204386], which indicates the orderliness of software systems are difference greatly. But according to the average and median values, the entropy values are determined concentrating around 0.32633. In order to observe the change trend of standard structure entropy along with the increasing of the number of nodes, we plot the relationship between these two parameters in Figure 5.

It is shown in Figure 5 that along with the increase of the number of nodes, standard structure entropy turns to lesser and lesser till the number reaches 2000. The curve fitting equation in Fig.5 is

$$Y_{H_S} = 0.5544837380342325 \times e^{(-0.2371493149006926 \times X)} \quad (5)$$

The fitting coefficient R=0.845.

According to the definition of deviation of characteristic value and the characteristics of standard structure entropy, the deviation of standard structure entropy of software networks is defined as follows:

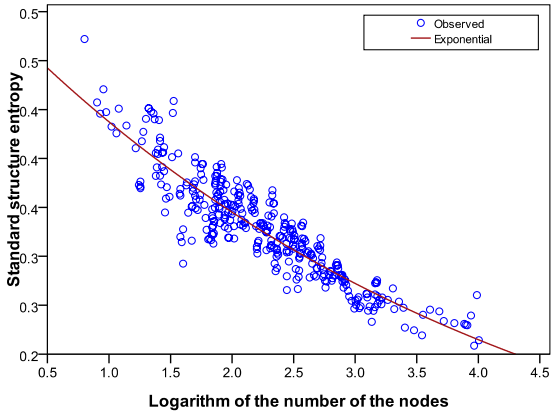

Figure 5 Standard structure entropy and the number of nodes

Definition 5: Deviation of standard structure entropy of software networks is the difference between the standard structure entropy of given software system and its average level of those open-source software systems with the same scale which are used widely at present. It is marked as *D(E)*, and $D(E) = H_s - Y_H$. Among this, H_S is the standard structure entropy of the software system being measured, and Y_H is the average level of that calculated by formula (5).

Some conclusions can be drawn based on the characteristics of standard structure entropy and the relationship between the entropy and the number of nodes as follows: If D(E)>0, the entropy value is larger than the average level, which means the code structure is complex and disorderly, and there is still room for improving it. On the contrary, if D(E)⩽0, it is means the orderliness of code structure of the given software system is acceptable.

V. EXPERIMENT AND ANALYSIS

In order to validate the metrics for software static structure based on deviation of characteristic value can reflect the effect on quality characteristics of software brought by macro-topology characteristic values authentically, we measure several software systems using metrics mentioned above and then do more analysis work on the result data.

We choose Linux kernel, one of the famous operation systems, as our test sample to carry out the experiment. Firstly, the characteristic values of different edition Linux kernels are calculated, and the statistical results are shown in table 4.

Table 4. Data of Linux kernel static structure macro-topology characteristic values in metric

Version	Nodes number	Arcs number	Average degree	Average shortest path length	Standard structure entropy
1.2.13	552	347	1.26	2.1674	0.1773
1.3.64	961	617	1.28	2.4188	0.1674
2.0.37	1776	1182	1.33	2.0411	0.1577
2.1.36	1989	1374	1.38	2.6518	0.1585
2.2.11	3593	2662	1.48	2.7243	0.1521
2.3.50	5056	4055	1.60	2.85	0.1511
2.4.36	9453	8452	1.79	2.74	0.1456

Table 5. Metric results of Linux kernel static structure

Version	Node number	D(AP)	D(AD)	D(E)
1.2.13	552	0.0180650014	0.2705953274	-0.0620938333
1.3.64	961	-0.1370829018	0.1313174998	-0.0559318627
2.0.37	1776	-0.3019109596	-0.4093739719	-0.0488784093
2.1.36	1989	-0.2943446907	-0.2292642494	-0.0451026064
2.2.11	3593	-0.4323069234	-0.4717456914	-0.0365178303
2.3.50	5056	-0.4636572138	-0.5182726067	-0.0292689539
2.4.36	9453	-0.5810855087	-0.6288507335	-0.0203903776

Furthermore, according to the metrics based on deviation of characteristic values, these Linux kernels are measured in this way. The result is shown as table 5.

Then we plot the metric results of D(AP), D(AD) and D(E) as Fig.6, Fig.7 and Fig.8 separately, and discuss the efficiency of the static structure metrics based on deviation of characteristic values.

(1) analysis on deviation of average shortest path length

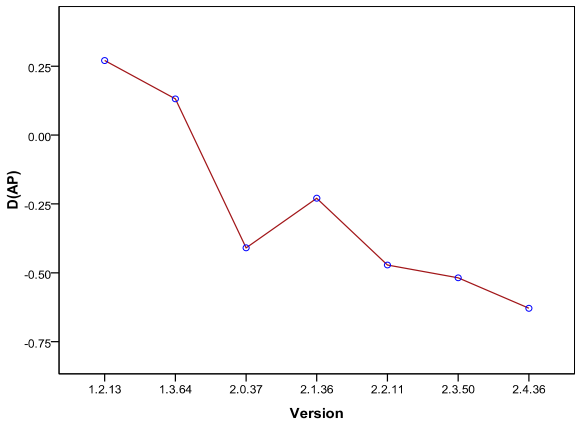

Figure 6. D(AP) of Linux kernels of different version

It can be found from the curve change that the deviation of average shortest path length presents descending tendency as a whole along with the increase of version. Because D(AP) has negative correlation with the efficiency of software system, the efficiency of Linux kernel is improved, which is consistent with the testing result announced by Linux official website. The Stat. data indicates that the execution efficiency would improve with the degree from 10% to 23% when kernel of latest version appeared [15], just as what Fig.6 shows.

Although D(AP) presents descending tendency as a whole, there are still some abnormal change we can find during the development of Linux kernel. Firstly, the value of D(AP) decreases greatly from the version 1.3.64 to 2.0.37. By consulting related references about the development of Linux kernel and analyzing related data, the reasons can be obtained: one of them is there are several different version kernels existing in the interval from 1.3.64 to 2.0.37, so the accumulative value is a little larger; another is large quantities of professional programmers attended the development of Linux kernel after Linux was well known as operation system, which made the structure designed more properly.

Secondly, the value of D(AP) increases from the version 2.0.37 to 2.1.36 instead of decreasing that is because from the version 2.1.XX, Linux system began to march on high-end market and expand server market, which demanded it can support various hardware including CPU and mainboard, etc. Massive function modules were added into existing system in this condition, and it lead to the average function calling depth of the kernel increasing greatly. Therefore, the average shortest path length increases greatly and D(AP) is larger than that of previous version.

(2) analysis on deviation of average degree

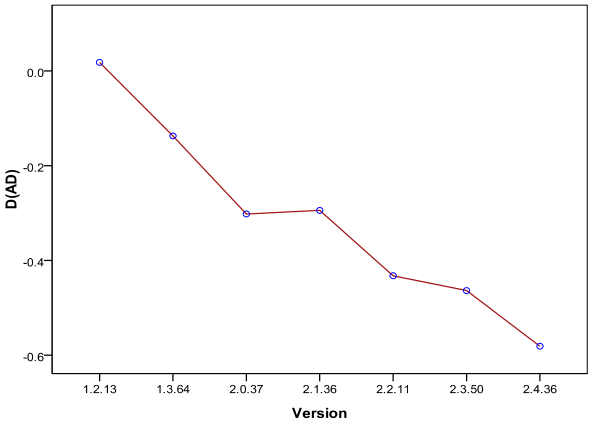

Figure 7. D(AD) of Linux kernels of different version

The evolution curve in Figure 7 indicates that the deviation of average degree D(AD) decreases obviously along with the increase of version, that is the complexity of software static structure is declined. According to the information from Linux official website, it is known the core team that is responsible for modifying and optimizing the kernel code before version 1.0 is only consist of five members. But after version 1.0, Linux kernel issued accordance with GPL protocol. From then on, many professional senior programmers attend the core team, especially after version 2.1, quantities of laboratories and companies began to participate in the development of Linux kernel (such as the foundation of Redhat Lab, the investment by Intel and Netscape, etc). All of above impelled the optimization of code and modules, which leads to the decrease of complexity of code structure eventually.

The light increase of D(AD) from version 2.0.37 to 2.1.36 shown in Fig.7 is because of the addition of massive modules supporting various hardware in order

to improve the portability of Linux kernel between different platforms.

(3) analysis on standard structure entropy

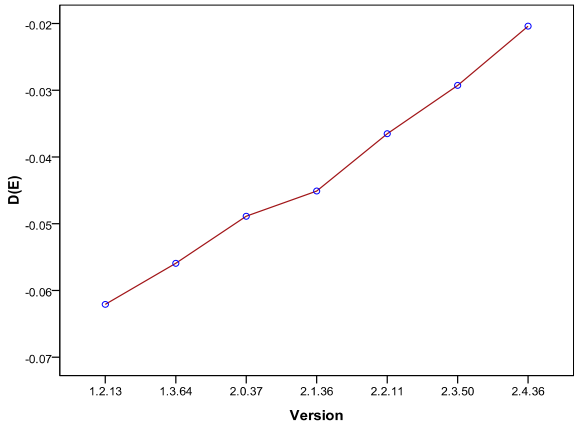

Figure 8. D(E) of Linux kernels of different version

It can be seen from Figure 8 that change trend of D(E) is opposite to that of D(AP) or D(AD), that is the value of D(E) increases gradually along with the increase of kernel version, which means the orderliness of Linux kernel is decreasing. But up till now this result is still acceptable because the value of D(E) is less than average level. According to reference [15], it is found code amount, such as the number of code line in Linux kernel, has changed greatly with the development of Linux kernel. The increasing speed of code amount is so high that the amount of kernel code will be two times than that of previous version at least when a new version Linux kernel is issued. For example, there are about 400 thousand code lines in kernel of version 2.0.37, while the number of code lines in kernel of version 2.1.36 is nearly one million. The explosion of code amount leads to the decrease of orderliness of Linux kernel, which can also be validated by study the source code of different version kernels.

Based on these test results and analyses, it can be concluded the static structure metrics of software based on deviation of characteristic values can really reflect the effects of static topology characteristic value on the quality characteristics of software systems and evolution a software system by their efficiency, complexity and structural orderliness effectively.

VI. CONCLUSIONS

Along with the rapid increase of software scale, complexity becomes a natural property of each software system, and how to measure the quality of a software system has been a challenge for software engineers. Many research results indicate that software systems presents significant characteristics of complex networks, so our study started with the characteristic values of complex networks and propose static structure metrics on software based on deviation of characteristic values. First, static topology characteristic values of software networks are analyzed and studied refer to the research idea and methods in complex networks by means of statistics. The results show the average shortest path length, the average degree and the standard structure entropy present high correlation with the number of nodes, which provides important theoretical basis for the following research work. Then, deviation of characteristic values including D(AP), D(AD) and D(E) are discussed based on the conceptions of deviation of characteristic value and quality characteristics relative to software scale. The metrics on the deviation of characteristic values between the given software and average level of the open-source software widely used with the same scale is used to reflect the effective of characteristic values of software networks on the efficiency, complexity and orderliness of software systems. In the latter part, seven Linux kernels of different versions are measured by the metrics proposed in this thesis which is validated through analyzing the test results. The metrics on software networks based on static structure characteristic values lay a foundation for the establishment of large-scale software metrics set by quantifying the characteristic values of software networks which can reflect the quality characteristics of software systems, and provide references for the measure methods of software based on quantification being proposed in the future.

But because the research of software networks is still in its primary stage and more research methods are being explored, the metrics on static structure of software proposed in this thesis is not perfect. For example, not all the characteristic values of software networks are used in our research work and the metrics on different characteristic value are the same and simple. In next research, more characteristic values will be studied for measuring the quality of software systems and the metrics will more refined according to the different characteristics of each characteristic value.

ACKNOWLEDGEMENT

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] C. V. Ramamoorthy, W.-T. Tsai, T. Yamaura, et al.. Metrics Guided Methodology[C]. In Proceedings of 9th Annual International Computer Software and Applications Conference, 1985, 111-120.

[2] Y. T. Ma, J. X. Chen, and J. H. Wu. Research on the phenomenon of software drift in software processes [C]. In Proceedings of 8th International Workshop on Principles of Software Evolution, Lisbon, 2005, 195-198.

[3] M. Alshayeb and W. Li. An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes [J]. IEEE Transactions on Software Engineering, 2003, 29(11): 1043-1049.

[4] L. C. Briand, S. Morasca, V. R. Basili. Property-Based Software Engineering Measurement [J]. IEEE

Transaction on Software Engineering, 1996, 22(1): 68-86.
[5] S. Furey. Why we should use function points [J]. IEEE Software, 1997, 14(2): 28.
[6] M. Arnold and P. Pedross. Software Size Measurement and Productivity Rating in a Large-Scale Software Development Department[C], In Proceedings of 1998 International Conference on Software Engineering, Kyoto, 1998, 503-506.
[7] M. Bauer. Analysing Software Systems by Using Combinations of Metrics[C]. In Proceedings of ECOOP'99 Workshops, Springer-Verlag LNCS 1743, Lisbon, 1999, 170-171.
[8] Vasa R, Schneider J G, Woodward C, et al. Detecting structural changes in object oriented software systems[C].In proceeding of International Symposium on Empirical Software Engineering. Noosa Heads, Australia. 2005:479-486
[9] Vasa R, Schneider J G, Nierstrasz O. The inevitable stability of software change[C]. In proceeding of The 23nd IEEE International Conference on Software Maintenance. Paris, France, 2007:4-13
[10] Girolamo A, Newman L I. Rao R. The structure and behavior of class networks in object-oriented software design. www.eecs.umich.edu/~leenewm/documents/classnetworks.pdf, 2005
[11] Ma Y T, He K Q, Du D H, et al. A complexity metrics set for large-scale object-oriented software systems[C]. In proceeding of The 6th IEEE International Conference on Computer and Information Technology. Seoul, Korea, 2006:189-194
[12] Li B, Wang H, Li Z Y, et al. Software Complexity Metric s Based on Complex Networks[J]. Chinese Journal of Electronic, 2006, 34(12A):2371-2375
[13] Melton H, Tempero E. Static members and cycles in java software[C]. In proceeding of the first International Symposium on Empirical Software Engineering and Measurement. Madrid, Spain, 2007:136-145
[14] Zhang W B. Research on the Life Characteristic of Internet Macroscopic Topology [D]. Shenyang: Northeastern University,2006
[15] Fan L. Source Code of Linux Kernel [M]. Beijing: Posts& Telecom Press,2002

Zheng Liu is a lecturer at college of Information Science and Engineering, Northeastern University, Shenyang, P. R. China. She received B.S. degree in Computer Science and Technology and M.S. degree in Computer Application Technology at Northeastern University. Now, she is working for PH.D degree in Computer System Architecture. Her main research interests are complex network and software engineering.

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

Qian Zhang is a senior software engineer at Neusoft. He receive M.S. degree and PH.D degree in Computer Application Technology, Northeastern University , Shenyang, P. R. China. His current interest is software engineering.