Topology Structure and Centrality in A Java Source Code

Long YING
education department
national university of defense technology
Changsha, China
nudt_ylong@yahoo.com.cn

De-wu DING *
dept. of mathematics and computer science
chizhou college
Chizhou, China
dwding@yahoo.com.cn

***Abstract*—Call graph plays a very important role in program analysis, and it has been widely used in software engineering (e.g., program understanding, compiling optimization, regression test, etc). In the present paper, we studied the source code structure of a Java program from a call graph perspective. We check the basal network properties of the call graph. By comparing several familiar centrality measures (betweenness, closeness, eccentricity, HITS-Authority and PageRank) and applying them to the call graph, we also give the centrality analysis of the call graph.**

Keywords-call graph; centrality; complex network; Java program

I. INTRODUCTION

There has been a strong interest in the study of complex networks in many disciplines in the last decade, e.g., biological networks, social networks, and so on [1]. Call graph is represented as another important class of complex networks. Generally speaking, a call graph is a directed graph in which nodes are functions and arcs are call events between functions (figure 1). Call graph could used to represent the call relationship of the program, and thus plays a very important role in program analysis. It has been widely used in software engineering (e.g., program understanding, compiling optimization, regression test, etc) [2-4].

From a complex network perspective, the structure of a call graph would affect its function, and thus in helping for program understanding and analysis. We studied the source code structure of a Java program with its call graph in the present paper. We check the basal network properties of the call graph, and we also give the centrality analysis of the call graph.

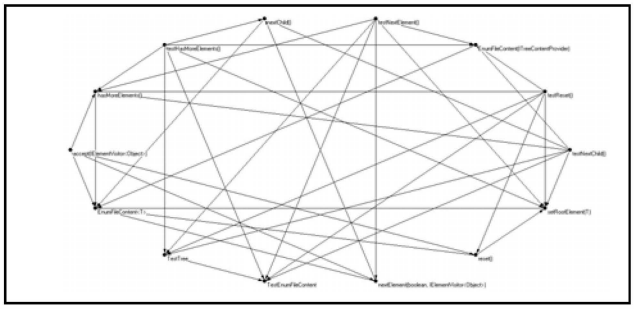

Figure 1. A simple call graph in the Java program as a demonstration

II. CALL GRAPH OF A JAVA PROGRAM

We firstly obtained the call graph of a Java program [5]. We then extracted a complex network model for the call graph, i.e., using nodes to represent functions in the source code of the Java program and using arcs to represent call events between these functions. The resulted network includes 724 nodes and 1025 arcs. We then give the topological characteristic of the call graph with the program pejak [6] (see figure 2).

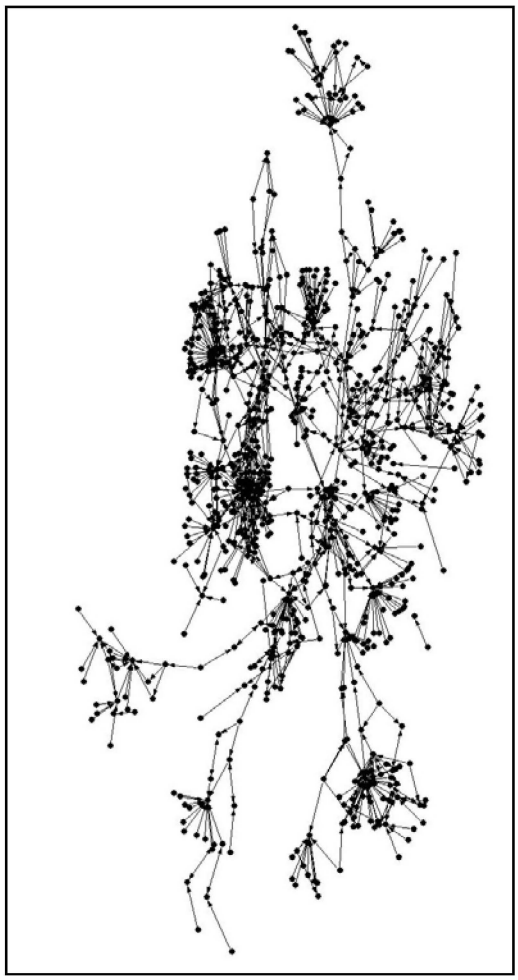

Figure 2. The topological characteristic of the call graph in the Java program, nodes are functions and arcs are call events between functions

* Corresponding author, E-mail: dwding@yahoo.com.cn

978-1-4673-2311-6/12/$31.00 ©2012 IEEE

III. COMPLEX NETWORK ANALYSIS

We firstly check the basal properties of the complex network model for the Java program source code. The average path length is 0.009 and the network diameter is 5 shown that the source code network of the Java program is a representative small-world network. The power law of occurrence degree distribution also shown it is a kind of scale-free network. Furthermore, the modularity of the network is 0.74 which suggested that the source code network is high modularization.

Then using the Gephi toolbox [7], the centers of the source code network are computed according to several familiar centrality measures (betweenness, closeness, eccentricity, HITS-Authority and PageRank).

Figure 3 gives the betweenness centrality on the source code network. And following the rank of betweenness centrality, the top 10 nodes of the source code network are: AbstractExporter, GEXFExporter, CodeData, ICodeData, IExporter, addDependency(String, IJavaElement), StdAnalyse, AbstractParser, ParserJava and Pair(T, T).

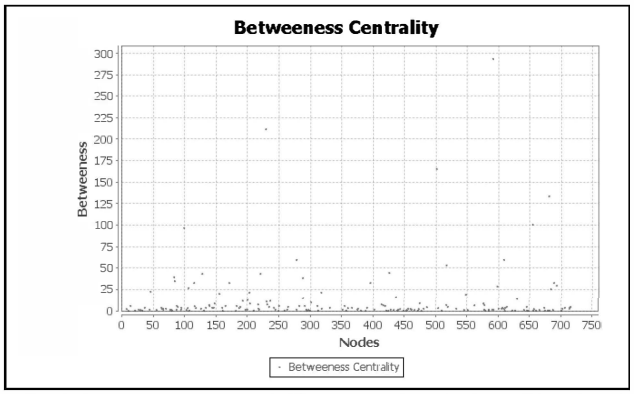

Figure 3. Betweenness centrality on the call graph in the Java program

Figure 4 gives the closeness centrality on the source code network. And following the rank of closeness centrality, the top 10 nodes of the source code network are: run(IProgressMonitor)(177), run(IProgressMonitor)(213), createParser(), run(IProgressMonitor)(263), init(), onChange(CodeDataChangesEvent), ParserJava(), launchAnalyze(), build(ICodeData, long, Collection<String>) and testIntrospectionFunction().

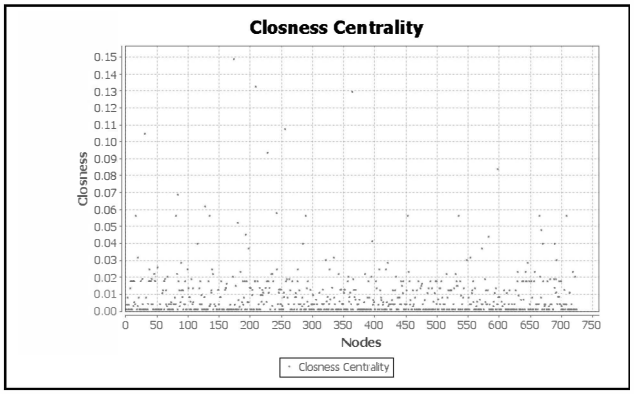

Figure 4. Closeness centrality on the call graph in the Java program

Figure 5 gives the eccentricity centrality on the source code network. And following the rank of eccentricity centrality, the top 10 nodes of the source code network are: visit(SuperConstructorInvocation), init(), visit(FieldAccess), run(IProgressMonitor)(177), run(IProgressMonitor)(213), run(), createParser(), visit(SuperFieldAccess), visit(SuperMethodInvocation) and nextElement().

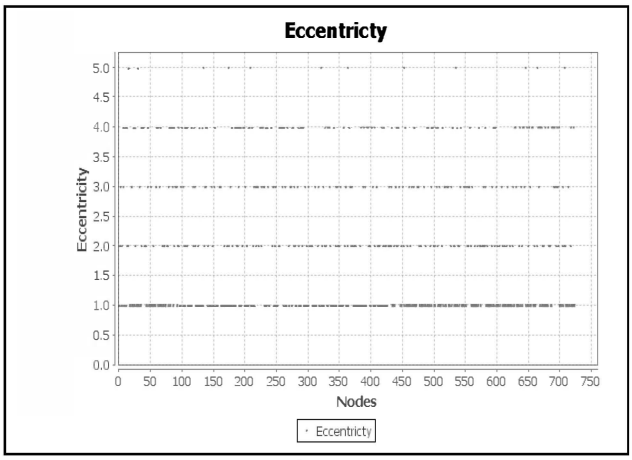

Figure 5. Eccentricity centrality on the call graph in the Java program

The HITS-Authority centrality on the source code network is then given in figure 6. And following the rank of HITS-Authority centrality, the top 10 nodes of the source code network are: GEXFExporter, CodeData, StdPage, ExtendedMethodDependanceDetector, StdAnalyse, DebugLogger, ExtensionUtil, AbstractExporter, EnumFileContent<T> and ExtensionListEditorControl.

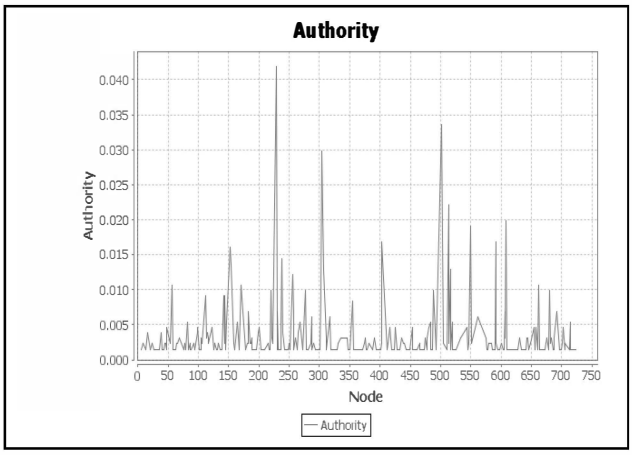

Figure 6. HITS-Authority centrality on the call graph in the Java program

At last, PageRank centrality on the source code network is given in figure 7. And following the rank of PageRank centrality, the top 10 nodes of the source code network are: AbstractExporter, ICodeMinerModule, IInfoElement<T>, ICodeData, GEXFExporter, CodeData, StdPage, IExporter, CodeDataChangesListener and IWriteElement<T>.

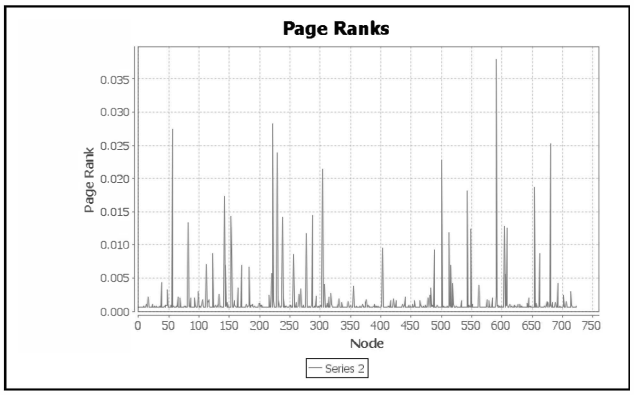

Figure 7. PageRank centrality on the call graph in the Java program

IV. CONCLUSION

Complex networks are widely used in many disciplines in nowadays, and would play an important role in software engineering [2-4]. From this point, we studied the source code structure of a Java program in the present paper. We obtained the call graph of the Java program, and extracted a complex network model for it. We also gave the basal network properties analysis and some centrality analysis on the model with complex networks methods.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments on this study. Support for this work is provided by Natural Science Fund of Chizhou College (2012ZRZ002).

REFERENCES

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, "Complex networks: structure and dynamics," Phys. Rep., vol. 424, pp. 175-308, February 2006.

[2] L. Wang, Z. Wang, C. Yang, L. Zhang, and Q. Ye, "Linux kernels as complex networks: a novel method to study evolution," Proc. ICSM, vol. 1, pp. 41-50, September 2009.

[3] K. K. Yan, G. Fang, N. Bhardwaj, R. P. Alexander, and M. Gerstein, "Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks," PNAS, vol. 107, pp. 9186-9191, May 2010.

[4] D. W. Ding, "Network motifs in Linux kernel v2.6.27," unpublished.

[5] S. Heymann and J. Palmier, "Source code structure of a java program," http://wiki.gephi.org/index.php/Datasets, February 2012.

[6] V. Batagelj and A. Mrvar, "Pajek - program for large network analysis," Connections, vol. 21, pp. 47-57, February 1998.

[7] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," Proceedings of the Third International AAAI Conference on Weblogs and Social Media, vol. 1, pp. 361-362, May 2009.