International Journal of Information Processing, 10(1), 92-102, 2016
ISSN : 0973-8215
IK International Publishing House Pvt. Ltd., New Delhi, India

Analysis of Structural Coupling Strength in Improving Software Module Structure

Amarjeeta[a], Jitender Kumar Chhabra[a]

[a]Department of Computer Engineering, NIT Kurukshetra, Haryana, India, Contact: amarjeetnitkkr@gmail.com

The organization of a software system widely affects the various quality parameters such as modifiability, understandability, and testability. Hence, the success of medium or large scale software systems heavily depends on how well they are organized. However, systems often require to be modified to satisfy the environment and requirements. Mostly, it is carried out without following the original design principles of the system. Over a period of time, such a continuous modification deteriorates the system organization, hence increases the system complexity. To improve the system structure, the software clustering seems more realistic technique. Recently, the search-based software engineering approach gain more attention to solve the software clustering problem. In this paper, we propose a search-based multi-objective optimization to improve the structure of the object-oriented software systems. To determine the coupling strength between the software artifacts, we uses different coupling strength scheme such as binary coupling, additive coupling and relative coupling scheme. The approach is evaluated over four real-world and three random software applications. The experimentation results show that how the use of additive and relative coupling strength scheme leads to generate more effective solutions compared binary coupling strength.

Keywords: Modularization Quality, Multi-objective Optimization, Search based Software Engineering, Software Restructuring.

1. INTRODUCTION

A medium or large scale software system often needs constant modification that is triggered due to evolving technologies, changing requirements and stakeholder [1]. It has been revealed that the maintenance of software system, cost up to 75% of total software development cost [2]. In order to fix the bugs or satisfy the new requirements, maintainers modify the software system. Generally, the software developers do not follow the guidelines given in the software documentation during the modification, which leads system structure erosion [3]. There can be many reasons not to follow the documentation guidelines, for instance restrict delivery, lack of proper developer training, and absence of long-term commitment developers to the project [4]. Such maintenance practices deteriorate the system structure and leads system more complex for future maintenance [5]. Hence, to make system more modifiable it requires improving the system structure. To improve the system structure, software refactoring plays an important role, where the system structure is modified without changing the external behavior [6]. Applying software refactoring to re-structure the whole system is often challenging and costly [7]. Software clustering is a most feasible technique that can be used to obtain refactoring of whole system more efficiently. Hence, software clustering is widely used as an activity of software refactoring for whole system. Software clustering techniques has been an active research for more than twenty years. In literature, the software clustering has been performed at various levels of software granularity, with different clustering approaches [8]. A basic concept often used by the existing approaches is that soft-

TABLE I. INFORMATION ABOUT OF THE ORIGINAL SOFTWARE APPLICATIONS

Systems	Version	#Classes	#Connection	#Modules
JavaCC	1.5	154	722	6
JUnit	3.81	100	276	6
Java Servlet API	2.3	63	131	4
XML API DOM	1.0.b2	119	209	9
DOM 4J	1.5.2	195	930	16
Random50	NA	50	134	7
Random100	NA	100	285	12
Random150	NA	150	512	18

TABLE II. MEAN AND STANDARD DEVIATION OF MQ

	Binary Weighted			Additive Weighted			Relative Weighted		
Systems	Mean	SD	%-IM	Mean	SD	%-IM	Mean	SD	%-IM
Real Problems									
JavaCC	3.17	0.15	39.55	3.91	0.15	25.32	3.86	0.11	37.56
JUnit	3.84	0.13	32.34	3.89	0.12	36.86	3.94	0.17	54.98
Java Servlet	3.21	0.14	25.23	3.41	0.15	24.65	3.47	0.10	39.56
XML API	6.06	0.24	987	5.87	0.43	24.34	6.79	0.23	57.87
Random Problems									
Random50	2.83	0.04	7.78	3.64	0.11	33.85	3.88	0.07	43.12
Random100	4.67	0.12	24.61	4.82	0.15	32.42	4.98	0.17	40.99
Random150	6.65	0.13	12.45	10.11	0.22	76.53	10.57	0.18	84.16

maximum improvement is 38.6%. If we compare the three coupling strength scheme, the additive coupling strength scheme and relative coupling strength scheme perform better than the binary coupling strength in most of the problem instances. Now, if we compare the additive coupling strength scheme and relative coupling strength scheme, the Table shows that the relative coupling strength scheme perform better than the additive coupling strength scheme with all problem instance.

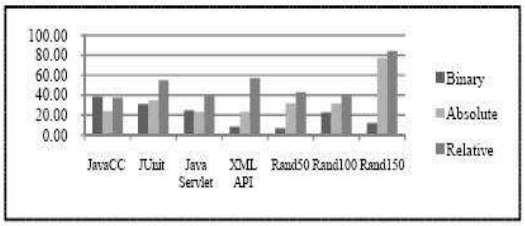

Figure 2. Percentage Improvements in MQ Values with Various Coupling Scheme

The Figure 2 summarizes the results of binary coupling, additive coupling and relative coupling scheme. Basically, it shows the percentage improvement in MQ values of all three coupling strength scheme. The horizontal axis shows the problem instances and the vertical axis sows the percentage improvement in the MQ values

8. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a search-based multi-objective optimization technique to improve the modular structure of an object-oriented software system. The approach is evaluated over eight software systems where the class coupling is determined using various coupling strength scheme namely binary coupling, additive coupling and relative coupling strength scheme. The experimentation results show that the approach with each coupling strength scheme, improves the MQ value of each problem instance. However, the proposed approach with additive coupling and relative coupling schemes perform better compared to binary coupling scheme over maximum problem instances. On the other hand, the relative coupling scheme performs better in comparison of additive coupling scheme over all the problem instances. Hence, the results conclude that the performance of the multi-objective optimization approach can be improved with the application of suitable coupling scheme. The

future works includes the formulation and evaluation of more coupling schemes with multi-objective optimization techniques for improving the object-oriented software structures.

REFERENCES

1. V Rajilich. Software Evolution and Maintenance, in *FOSE, Proceedings of the on Future of Software Engineering*, May 2014.
2. R D Banker and SA Slaughter. The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement, in *Information Systems Research*, 11:0219-0240, 2000.
3. B S Mitchell, S Mancoridis. On the Automatic modularization of Software Systems Using the Bunch Tool, in *IEEE Transactions on Software Engineering*, 32(3):193-208, 2006.
4. D O B Mrcio, F A Farzat, G H Travassos. Learning from Optimization: A Case study with Apache Ant, *Information Software Technololgy*, 2014,
5. B P Lientz, E B Swanson. Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations, *Addison-Wesley*, 1980.
6. M Fowler. Refactoring: Improving the Design of Existing Programs, *Addison-Wesley*, 1999.
7. W F Opdyke. Refactoring: A Program Restructuring Aid in Designing Object-Oriented Application Frameworks, *University of Illinois at Urbana-Champaign*, 1992.
8. N Anquetil, T C Lethbridge. Experiments with Clustering as a Softwarere Modularization Method. Working Conference onReverse Engineering, in *IEEE CS Press,* pages 235255, 1999.
9. D P Darcy, C F Kemerer, S A Slaughter, and J E Tomayko. The Structural Complexity of Software: An Experimental Test, in *IEEE Transactions on Software Engineering*, 31(11):982-995, Nov 2005.
10. E Arisholm. Empirical Assessment of the Impact of Structural Properties on the Changeability of Object-Oriented Software, in*Information and Software Technology,* 48:1046-1055, 2006.
11. V R Gibson and J A Senn. System Structure and Software Maintenance Performance,*Comm. ACM,*vol. 32, pages 347-358, 1989.
12. S Mancoridis, B S Mitchell, C Rorres, Y F Chen and E R Gansner. Using Automatic Clustering to Produce High-Level System Organizations of Source Code, *Proc. Intl Workshop Program Comprehension*, pages 45-53, 1998.
13. M Harman, P McMinn, J Souza, and S Yoo. Search based Software Engineering: Techniques, Taxonomy, Tutorial, in*Empirical software engineering and verification: LASER Springer*, pp. 159, 2012.
14. Mancoridis, B S Mitchell, C Rorres, Y F Chen, and E R Gansner. Bunch:Recovery and Maintenance of Software System Structures, *Proceedings of the IEEE International Conference on Software Maintenance*, pp 50-59, 1999.
15. B S Mitchell, S Mancoridis. On the Automatic modularization of Software Systems Using the Bunch Tool, *IEEE Trans. on Software Eng*, 32(3):193-208, 2006.
16. K Mahdavi, M Harman, and R M Hierons. A Multiple Hill Climbing Approach to Software-Module Clustering, *Proceedings of IEEE International Conference on Software Maintenance,* pages 315-324, 2003.
17. K Praditwong, M Harman, X Yao. Software Module Clustering as a Multi-Objective Search Problem, *IEEE Transactions on Software Engineering,* 37(2):264-282, 2011
18. U Erdemira, F Buzluca. A Learning-Based Module Extraction Method for Object-Oriented Systems, *The Journal of Systems and Software*, pages 156177, 2014.
19. K Deb, A Pratap, S Agarwal, T Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computing*, 6(2):182-197, April 2002.
20. D Hutchens, R Basili. System Structure Analysis: Clustering with Data Bindings, *IEEE Transactions on Software Eng*, vol. 11, pages 749-757, 1985.
21. T A Wiggerts. Using Clustering Algorithms in Legacy Systems Remodulariza-tion, *Working Conference on Reverse Engineering,IEEE,* pages 3343, 2000.
22. V Tzerpos, R C Holt. MoJo: A Distance Metric for Software Clustering, in *Proceedings of the 6th Working Conference on Reverse Engineering, Atlanta,GA, USA*, pages 187193, 1999.
23. V Tzerpos, R C Holt. On the Stability of Software Clustering Algorithms, *International Workshop on Program Comprehension,* IEEE, pages 211218, 2000.

24. M Harman, B F Jones. Search Based Software Engineering, *Information and Software Technology,* 43(14):833839, Dec. 2001.
25. D Doval, S Mancoridis, B Mitchell. Automatic Clustering of Software Systems using a Genetic Algorithm, *Proc. Of the International Conference on Software Tools and Engineering Practice,* 1999.
26. M Harman, R Hierons, M Proctor. A New Representation and Crossover Operator for Search-Based Optimization of Software Modularization, in *Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers,* pages 1351–1358, July 2002.
27. B Khan, S Sohail, M Y Javed. Evolution Strategy Based Automated Software Clustering Approach, *Advanced Software Engineering and Its Applications, ASEA* , pages.27–34, 13-15 Dec. 2008.
28. M Barros. An Analysis of the Effects of Composite Objectives in Multiobjective Software Module Clustering, in *Proceedings of the fourteenth international conference on Genetic and evolutionary GECCC*, pages 1205-1212.
29. L C Briand, J W Daly, J K Wust. A Unified Framework for Coupling Measurement in Object-Oriented Systems,*IEEE Transactions on Software Engineering,* 25(1):91-121, 1999.
30. A B Abreu. Coupling-Guided Cluster Analysis Approach to Reengineer the Modularity of Object-Oriented Systems, *CSMR Proceedings of the Conference on Software Maintenance and Reengineering, IEEE Computer Society Washington, DC, USA*, 2000.
31. http://www.dependency-analyzer.org/
32. http://www.stan4j.com
33. http://www.structure101.com/