Volume 4, Issue 4, April 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Review Paper on Static and Dynamic Analysis of Object Oriented Systems

Divya Sharma[*]
M.tech Scholar,CSE Deptt,HEC
India

Pooja Narula
Asst Professor,CSE Deptt,HEC
India

Abstract— The purpose of this research paper is to study a software product that requires efficient measures to accurately monitor the internal software quality, such as coupling and cohesion, throughout the course of software development life cycle. Software metrics have been widely and successfully used to measure such internal quality attributes for object-oriented software systems. Coupling is defined as one of the most basic qualitative measures for measuring the performance of software at design or implementation phase. It is normally defined as the degree of interdependency among modules. The metrics available for coupling measurement is divided into two major categories i.e. static metrics and dynamic metrics. Static metrics that are applied to the design/source code can only measure the expected coupling behaviour of object-oriented software and not the actual behaviour. Dynamic metrics on the other hand can capture the actual coupling behaviour as they are evaluated from data collected during runtime.

Keywords— Software Metrics, Coupling Measurement, Static Metrics, Dynamic Metrics, software quality

I. INTRODUCTION

Software metrics help us to make meaningful estimates for software products and guide us in taking managerial and technical decisions. Conventional static metrics have been found to be inadequate for modern object-oriented software due to the presence of object-oriented features such as polymorphism & dynamic binding, inheritance and unused code. It motivates us to focus on dynamic metrics in place of traditional static metrics [1]. Coupling has been defined as one of the most basic qualitative measures for measuring the performance of software at design or implementation phase. It is normally defined as the degree of interdependency among modules. Coupling measures the external complexity of a class, i.e. how much dependent a class is on other classes [2]. The metrics suite for object-oriented design is partly evaluated by applying principles of measurement theory. Using the object coupling measure (CBO) as an example, failing to establish a sound empirical relation system can lead to deficiencies of software metrics [3]. A large number of object-oriented (OO) metrics have been proposed in the literature. They are used to assess different software attributes. Software metrics can be calculated automatically from source code. The assessment of large software systems can be performed quickly at a low cost. Software metrics can be helpful in predicting software quality attributes and supporting various software engineering activities. Empirical validation of software metrics is therefore important to ensure their practical relevance. Cohesion is considered as one of most important OO software attributes. Many metrics have been proposed in the last several years to measure class cohesion in object-oriented systems (OOS). Class cohesion is defined as the degree of relatedness between members of a class. In Object oriented system, a class should represent a single logical concept, and not to be a collection of different features. OO analysis and design methods promote a modular design by creating high cohesive classes. However, improper assignment of responsibilities in the design phase can produce low cohesive classes with unrelated members [4]. The process of Software Engineering evolves with a unique issue of testability. It is a software attribute that measures the complexity and effort required for testing software. The insight provided by testing is valuable during design, coding, testing and quality assurance. Testability suggests testing intensity, and provides the degree of difficulty which will be incurred during testing of a particular location to detect a fault. Improving software testability is an important objective in order to reduce the number defects that result from poorly designed software. It is an inevitable fact that testability information is useful that may be complementary to testing. Higher test coverage may be achieved by making a system more testable for the same effort. Achieving testability is mainly a matter of separation of concerns, coupling between classes, subsystems, and cohesion [5]. The term coupling means interaction-based coupling when viewed from a static context. But it is not so when viewed from a dynamic or runtime context. Coupling is affected by the object-oriented features like inheritance and dynamic binding dynamically; interaction-based coupling is bound to show the impact of such features in the dynamic analysis results. Thus from a dynamic context, the type of coupling must be viewed as interaction-based coupling reflecting the effects of such dynamic features (i.e. inheritance, polymorphism). We call this combination as dynamic coupling or coupling at runtime. Dynamic coupling metrics was introduced as refinement to existing coupling metrics due to gaps in addressing dynamic binding, polymorphism, and unused code by static structural coupling measures[6]. Software metrics are increasingly playing an important role in the planning and control of software development projects. Coupling metrics have important applications in software development and maintenance phase. Existing work on software metrics is mainly focused on centralized systems, while in distributed systems, mainly in service-oriented systems, is limited. Distributed systems with service oriented components have more heterogeneous networking and execution environment. Traditional coupling measures had taken into account only "static" coupling. They do not account for "dynamic" coupling due to polymorphism and may underestimate the complexity of software and misjudge the need for testing and debugging. This is resulted in poor predictive accuracy of quality models in distributed Object Oriented systems that utilize static coupling measurements. In order to overcome these issues, we present a hybrid model in Distributed Object Oriented Software for measuring the coupling at run time. There are three steps such as Instrumentation process, Post process and coupling measurement process. In the instrumentation process the instrumented JVM has been modified to trace method calls. During this process, three major trace files are created named .prf, .clp and .svp. In the second step, the information present in these file are merged. At the end of second step, the merged detailed trace of each JVM contains pointers to the merged trace files of other JVM such that the path of every remote call from the client to server is identified uniquely. Finally, the coupling metrics is measured at run time. The implementation level results show that the proposed system will effectively measure the coupling metrics dynamically [7].The best way to define and measure coupling is through structural and static code analysis. However, because of object oriented concepts like polymorphism, dynamic binding and unused code in commercial software, the resulting coupling measures are impractical as they do not properly reflect the actual coupling taking place among classes dynamically. When we use static analysis to measure coupling it is difficult and sometimes impossible to determine what actual methods can be invoked from a client class if those methods are already overridden in the subclasses of server [8].

III. LITERATURE SURVEY

There has been a lot of research done on structural and object-oriented coupling metrics over the years.

Briand *et al.* [1] described many of such metrics in their work on unified framework for coupling measurement. Some of the famous static coupling metrics are Coupling Between Objects (CBO) and CBO1, Response For Class (RFC) and RFC∞, Efferent Coupling (Ce), Afferent Coupling (Ca), Coupling Factor (COF), Message Passing Coupling (MPC), Data Abstraction Coupling (DAC) and DAC1, Information-flow-based Coupling (ICP), Briand *et al.* suite (IFCAIC, ACAIC, OCAIC, FCAEC, etc) . Most of these metrics are based on method invocations and attribute references.

Chidamber and Kemerer [2] introduced the first coupling metric for object-oriented systems. In their metric suite, they defined CBO (Coupling between Objects) metric as number of non-inheritance related couples with other classes. They concluded that higher coupling leads to higher complexity. They later revised their definition of CBO to include coupling due to inheritance.

Paques and Delcambre [3] presented an approach to evaluate the dynamic coupling at analysis phase. They proposed Dynamic or run time Clustering Mechanism (DCM) that works by tracking hot spots for dynamic coupling at analysis phase. They stated that dynamic coupling was based on the frequency with which classes interact at runtime.

Schikuta [4] was the first to propose a dynamic approach to measure coupling of software systems. It was concluded that the conventionally used measurement systems were statically oriented and provided the incomplete dynamic behaviour of the system. This is because of static systems that give measures only for syntactic program analysis and thus they have small ability. They insisted that the dynamic measures should always be used in combination with their static counterparts.

Yacoub *et al.* [5] defined a set of object-level dynamic coupling metrics designed to evaluate the change-proneness of a design.

S. Babu and R.M.S. Parvathi et al. [6] described distributed systems have become increasingly common as more and more organizations use networks to enhance communication and increase performance. Examples of these systems range from the Internet to small workstations in a local area network within a building, to processors within a single multiprocessor. In a distributed object-oriented application, classes can run on a separate computer within a network system. So, they should be distributed efficiently among different nodes. A distributed OO application consists mainly of a set of interacting objects; each one runs on a separate computer within a network system. There have been a large number of projects related to the use of object-oriented approaches for the design of problem solving environments for complex applications in various scientific fields. The object model and distributed technologies are being amalgamated. The advantage is obvious: the complexity and dependencies of the entities can make use of the object model in a distributed system to break down the intensive design process into efficient constructs. Many of the concepts of object-oriented programming are currently finding widespread application in loosely coupled distributed systems.

S. Babu and R.M.S. Parvathi et al. [7] Defined coupling measurement has traditionally been performed using static code analysis, because most of the previous work was done on non-object oriented code and because dynamic code analysis is more expensive and hard to perform. We refer to this type of coupling as dynamic or run time coupling.

IV. CONCLUSION

Dynamic analysis of software can be performed in many ways: using profilers, using aspect-oriented programming (AOP) & from dynamic models. Some other less popular techniques like method-wrappers, pre-processor based approach, and hybrid approach can also be used for this purpose. It is found that AOP approach provides a balanced method for the dynamic analysis of programs. In addition, AOP approach is easier to implement and at the same time an efficient technique for dynamic analysis without any side effects.

REFERENCES

[1] L.C. Briand, V. R. Basili and W. L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751–761, October 1996.

[2] S. R. Chidamber, and C. F. Kemerer, "Towards a Metrics Suite for Object-Oriented Design", In *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications, (OOPSLA' 91)*, 1991, SIGPLAN Notices, Vol. 26, no.11, pp. 197–211.

[3] H. Paques and L. Delcambre, "A Mechanism for Assessing Class Interactions Using Dynamic Coupling During the Analysis Phase", In *Proceedings of XVIII Brazilian Symposium on Software Engineering - SBES'99*, 1999, Florianopolis - Santa Catarina – Brasil.

[4] E. Schikuta, "Dynamic Coupling Metrics", 1993.

[5] Sherif M. Yacoub, T. Robinson, and H. H. Ammar, "Dynamic Metrics for Object Oriented Designs", In *Proceedings of the 6th International Symposium on Software Metrics*, 1999, pp.50-58, November 04-06

[6] S. Babu and R.M.S. Parvathi, "Design Dynamic Coupling Measurement of Distributed Object Oriented Software Using Trace Events", 2011.

[7] S. Babu and R.M.S. Parvathi, "Development of Dynamic Coupling Measurement of Distributed Object Oriented Software Based on Trace Events", Vol.3, No.1, January 2012.

[8] Linda Badri, Mourad Badri & Fadel Toure, "An Empirical Analysis of Lack of Cohesion Metrics for Predicting Testability of Classes", Vol. 5 No. 2, April, 2011.

[9] Mohd Nazir, Dr. Raees A. Khan, & Dr. K. Mustafa, "An Empirical Analysis of Lack of Cohesion Metrics for Predicting Testability of Classes", *Volume 2 – No.5, June 2010*.

[10] Paramvir Singh and Hardeep Singh, "Class-level Dynamic Coupling Metrics for Static and Dynamic Analysis of Object-Oriented Systems", Vol. 1, Issue 1, 2010.

[11] Jitender Kumar Chhabra and Varun Gupta, "A Survey of Dynamic Software Metrics", 2010.

[12] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics*, Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[13] R. Martin, "OO Design Quality Metrics - An Analysis of Dependencies", In *Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA'94*. 1994