



Development of Dynamic Package level Coupling Measurement

Silky Bindra*, Er. Navdeep Gupta
CSE & Haryana Engineering College,
Haryana, India

Abstract: During the last decade, the software product measurement field has known many improvements and becomes an emerging field of the software engineering. Depends on the approaches and concept such as object-oriented and structure programming concept and so on. Many kinds of measures are proposed in the literature. The static metric has been found to be limited for modern object-oriented software due to insufficient contribution of object-oriented features such as polymorphism data hiding and inheritance concept. This concept motivates to focus on the dynamic metrics as dynamic metrics have many advantages over the static metrics. Different types of the dynamic metrics were proposed for the measurement of coupling at the run-time status. New types of the dynamic software metrics were presented for the measurement of design and complexity of the object-oriented software.

Keywords: –Class level metrics, Complexity, SDLC, Hybrid Model, and Testability.

I. INTRODUCTION

1.1 SOFTWARE ENGINEERING

Software engineering is an engineering discipline that is concerned and associated with all the aspects of the software productions. Products of the software were made up of developed programs and associated documentation is required in all these products. The necessary and essential product attributes of software engineering are maintainability, efficiency, functionality, reusability and usability. The software system process contains many activities which involved in developing software products. Important activities of the software engineering are software specifications, validation or verification processes. Methods that are used organized in a ways of producing the software.

1.2 SOFTWARE METRICS

Software metrics are the units of measurement which are used to characterize the software engineering products (requirement, design, source code, testing etc.), software engineering cycle processes (analysis requirement, design, coding, testing and maintenance etc.) and software engineering professionals (the efficiency and ability of an individual tester, and the productivity and creativity of an individual designer is increased efficiently). If all these used properly then the software engineering metrics allows us to quantitatively define the degree of success or the unsuccessful, for the product, or for the person, make meaningful and useful and decisions, and after that make them quantified and meaningful properly and thus, the incorporating metrics into development plans is a very simple step towards creating better systems. LOC (lines of code) and Cyclomatic Complexity are most popular and time honoured software metrics.

1.3 COUPLING MEASUREMENT

Coupling measurement has traditionally been performed by using the static code metrics analysis due to the reason that most of existing work was done on the non-object oriented code and because dynamic code analysis is very expensive and there is difficult to be perform. For modern system, this type of focus on the static analysis concept can be problemized due to the reason that the dynamic binding exists before the advent of the object-oriented process however its usage has been increased in the past few year.

1.4 PACKAGE LEVEL METRICS

Packages are defined as re-usable components for the modern object-oriented systems. Package was commonly consisting of classes, interfaces and the sub-packages concept. To promote reuse criteria in the object-oriented systems and to make the tasks like deployment and maintenance quite easy, in that type of case packages in object-oriented systems should follow the basic principles of design i.e. maximum cohesion and the less coupling. The process of coupling between packages is defined as the degree of the interdependence between them. Each and every package is a stand-alone unit

II. RELATED WORK

S. Babu[1]. In this paper, a hybrid model in Distributed Object Oriented Software for measure the coupling dynamically is proposed. In the proposed method, there are three steps such as Instrumentation process, Post processing and Coupling measurement. In this process the instrumented JVM that has been modified to trace method calls. During this process, three trace files are created namely .prf, .clp, .svp. In the second step, the information in these file are merged. At the end of this step, the merged detailed trace of each JVM contains pointers to the merged trace files of the other JVM such that the path of every remote call from the client to the server can be uniquely identified. Finally, the coupling metrics are measured dynamically. The implementation results show that the proposed system will effectively measure the coupling metrics dynamically.

Jitender Kumar Chhabra [2] In this paper, advantages of dynamic metrics over static metrics are discussed and then a survey of the existing dynamic metrics is carried out. These metrics are characterized into different categories such as dynamic coupling metrics, dynamic cohesion metrics. Towards end of the paper, potential research challenges and opportunities in the field of dynamic metrics are identified.

Kritika [3] In this paper, the metrics available for coupling measurement belong to two major categories i.e. static and dynamic. 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 behavior. This is because the behaviour of a software application is not only influenced by the complexity but also by the operational or runtime environment of the source code. Dynamic metrics on the other hand can capture the actual coupling behaviour as they are evaluated from data collected during runtime.

Divya[4] 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.

Mohd Nazir [5] Testability has always been an elusive concept and its correct measurement or evaluation a difficult exercise. Most of the studies measure testability or more precisely the attributes that have impact on testability but at the source code level. This paper provides a roadmap to industry personnel and researchers to assess, and preferably, quantify software testability in design phase. A prescriptive framework has been proposed in order to integrate testability within the development life cycle. It may be used to benchmark software products according to their testability

Linda Badri [6] The aim of this work is to explore empirically the relationship between lack of cohesion metrics and testability of classes in object-oriented systems. We addressed testability from the perspective of unit testing. We performed an empirical analysis using data collected from two Java software systems for which JUnit test cases exist. To capture testability of classes, we used different metrics to measure some characteristics of corresponding JUnit test cases. In order to evaluate the capability of lack of cohesion metrics to predict testability, we used statistical analysis techniques using correlation and logistic regression. The performance of the predicted model was evaluated using Receiver Operating Characteristic (ROC) analysis. The achieved results provide evidence that there exist a relationship between lack of cohesion and testability.

Paramvir Singh[7] This paper, as a part of our ongoing research on dynamic coupling and dynamic inheritance object-oriented metrics, proposes a new set of class level dynamic metrics to measure coupling at runtime. It characterizes the ability of such metrics to estimate the external quality attributes of a software design and to compare the coupling assessments achieved from static and dynamic analysis. Initial investigations towards finding any possible relation between dynamic and static metric analysis results are also presented. An empirical study is conducted that involves applying the proposed metrics to assess the quality of a java-based object-oriented sample

III. PROPOSED WORK

Software metrics are used to measure software engineering products (design, source code etc.), processes (analysis, design, coding, testing etc.) and professionals (efficiency or productivity of an individual designer).

- The common approach used for software development is the OO (object-oriented) approach and mainly two kinds of software metrics exist for object-oriented software - static software metrics and dynamic software metrics.
- The static software metrics are obtained from static analysis of the software, whereas dynamic software metrics are computed on the basis of data collected during execution of the software
- Here some new dynamic software metrics are presented for the measurement of design and complexity of object-oriented software. The research work is based on Package level metrics.
- Packages are re-usable components for modern object-oriented systems. A package usually consists of classes, interfaces and sub-packages. To promote reuse in object-oriented systems and to make deployment and maintenance tasks easy, packages in object-oriented systems should follow the basic principles of design i.e. maximum cohesion and minimum coupling. . The coupling between packages is the degree of interdependence between them. Every package is a stand-alone unit

IV. RESULTS & DISCUSSIONS

The result is implemented in visual studio and it has been analysed using N-depend tool.

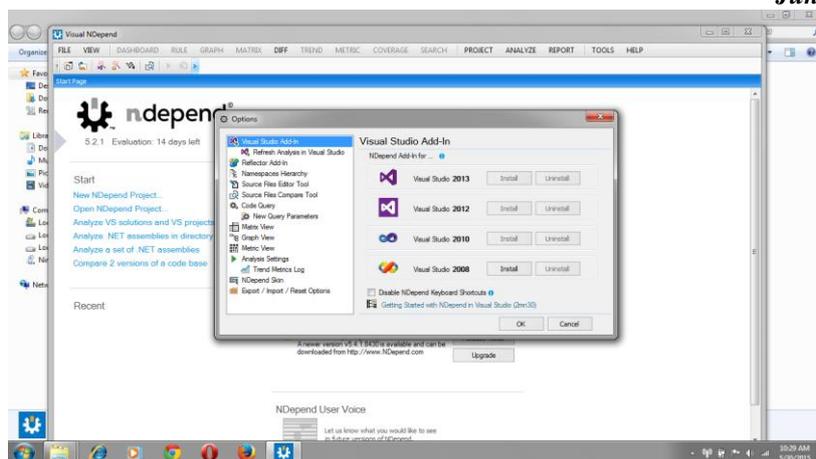


Figure 1: Adding N-depend Add – in for visual studio.

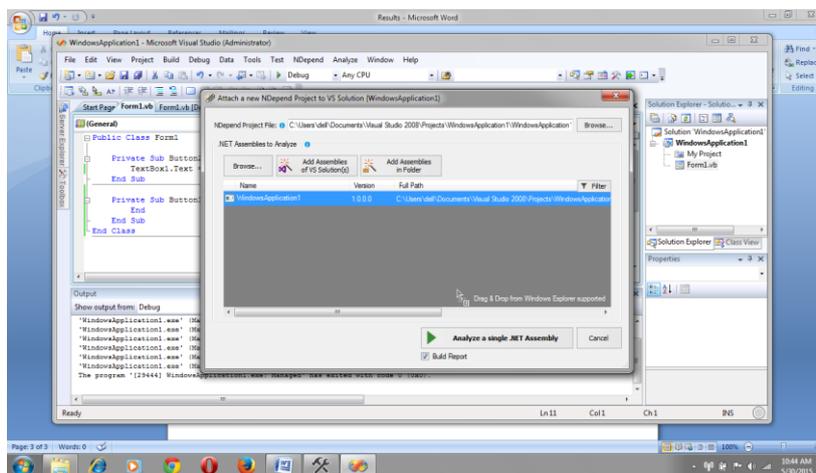


Figure 2: Attaching a new N-depend project to VS Solution

Table 1: Displaying the rules violated in project by tracer application

NAME	MATCHES	ELEMENTS	GROUP
Class with no descendant should be sealed if possible	1	TYPES	OBJECT ORIENTED DESIGN
A stateless class or structure might be turned into a static type	1	TYPES	OBJECT ORIENTED DESIGN
Non-static classes should be instantiated or turned to static	3	TYPES	OBJECT ORIENTED DESIGN
Types with disposable instance fields must be disposable	1	TYPES	DESIGN
Classes that are candidate to be turned into structures	1	TYPES	DESIGN
Nested types should not be visible	3	TYPES	DESIGN
Instances size shouldn't be too big	1	TYPES	DESIGN
Types that could have a lower visibility	3	TYPES	VISIBILITY
Types that could be declared as private, nested in a parent type	4	TYPES	VISIBILITY
Avoid namespaces with few types	2	NAMESPACES	DESIGN
Methods should be declared static if possible	6	METHODS	OBJECT ORIENTED DESIGN
Empty static constructor can be discarded	3	METHODS	DESIGN
Potentially dead Methods	6	METHODS	DEAD CODE

Methods that could have a lower visibility	2	METHODS	VISIBILITY
Property Getters should be immutable	3	METHODS	PURITY - IMMUTABILITY - SIDE-EFFECTS
Don't assign static fields from instance methods	1	FIELDS	OBJECT ORIENTED DESIGN
Fields that could have a lower visibility	1	FIELDS	VISIBILITY
Fields should be declared as private	1	FIELDS	VISIBILITY
Fields should be marked as Read Only when possible	2	FIELDS	PURITY - IMMUTABILITY - SIDE-EFFECTS
Avoid static fields with a mutable field type	1	FIELDS	PURITY - IMMUTABILITY - SIDE-EFFECTS
Instance fields should be prefixed with a 'm_'	4	FIELDS	NAMING CONVENTIONS
Static fields should be prefixed with a 's_'	14	FIELDS	NAMING CONVENTIONS
Mark assemblies with CLS Compliant	1	ASSEMBLIES	SYSTEM

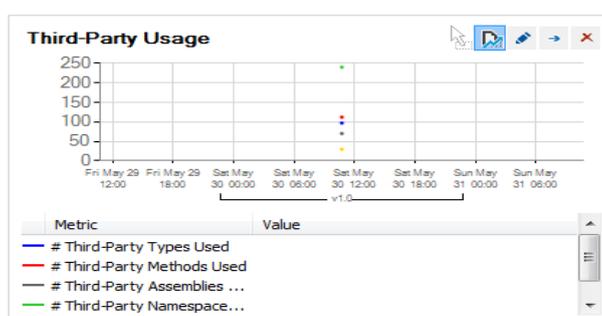


Figure 3: change in Lines of Code in project

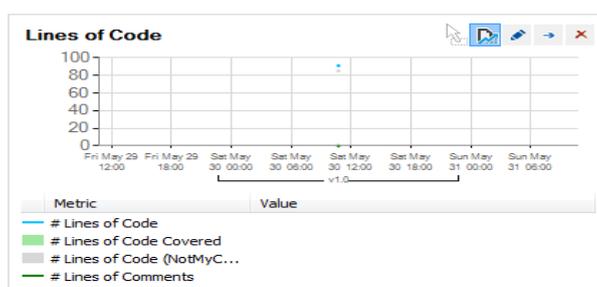


Figure 4: change in Lines of Code in project

V. CONCLUSION

Dynamic analysis of software can be performed in many ways: - using profilers, from dynamic models and using aspect-Oriented programming (AOP). Some other less popular techniques like, pre-processor based approach, method-wrappers based approach and hybrid approach can also be used for this purpose. From study, it is found that AOP approach provides a balanced method for the dynamic analysis of programs. In addition, this approach is easier to implement and at the same time an efficient technique for dynamic analysis without any side effects. N crunch tool is an automated concurrent testing tool is successfully studied .N crunch gives you a huge amount of useful information your tested code such as code as code coverage and performance metric, inline in your IDE while you type. Full code coverage matrix is also available for your entire solution. No matter where your source codes you will be able to analyze your problem quickly.

REFERENCES

- [1] S.Babu and Dr. R.M.S Parvathi (2012) "Development of dynamic coupling measurement of distributed object oriented software based on trace events", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1.

- [2] Paramvir Singh1 and Hardeep Singh (2010) "Class-level Dynamic Coupling Metrics for Static and Dynamic Analysis of Object-Oriented Systems", International Journal of Information and Telecommunication Technology, IJITT, Vol. 1, Issue 1.
- [3] Jitender kumar chhabra and varun gupta (2010) "a survey of dynamic software metrics", journal of computer science and technology, 1016-1029.
- [4] Kritika (2013) "Review Paper on Static and Dynamic Analysis of Object Oriented Systems International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 9.
- [5] Divya Sharma (2014) "Review Paper on Static and Dynamic Analysis of Object Oriented Systems", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 4.
- [6] Linda Badri, Mourad Badri & Fadel Toure (2011) "An Empirical Analysis of Lack of Cohesion Metrics for Predicting Testability of Classes", International Journal of Software Engineering and Its Applications Vol. 5 No. 2.
- [7] Mohd Nazir, Dr. Raees A. Khan, Dr. K. Mustafa (2010) "Testability Estimation Framework", International Journal of Computer Applications (0975 – 8887) Volume 2 – No.5.
- [8] Ramanath Subramanyam and M.S. Krishnan (2003) "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects", IEEE transaction on software engineering, VOL. 29, NO. 4.
- [10] 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.