Volume 3, Issue 9, September 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Tracer Application for Automatic Concurrent Testing Using Ncrunch

Kritika
HCTM, Kaithal
India

Vivek Bhasin
CSE Department, KUK
India

Abstract: Designing and developing a good quality software product requires efficient measures to accurately monitor the internal software quality attributes, such as coupling, cohesion, size and complexity, throughout the course of a software development life cycle. Over the years, software metrics have been widely and successfully used to measure such internal quality attributes for object-oriented software systems. 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 quantifies the external complexity of a class, i.e. how dependent a class is on other classes. A low level of coupling is desired among the classes of an object-oriented application. The theory behind this measure contributes to the external quality attributes of a software application, such as its maintainability, reusability, testability, fault-tolerance etc. 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.

I. Introduction

Designing and developing a good quality software product requires efficient measures throughout the life cycle of software development to accurately monitor the internal software quality attributes, such as coupling, cohesion, complexity and size. Past many years, software metrics have been widely and successfully used to measure such internal quality attributes for object-oriented software systems. 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 quantifies the external complexity of a class, i.e. how dependent a class is on other classes. A low level of coupling is desired among the classes of an object-oriented application. The theory behind this measure contributes to the external quality attributes of a software application, such as its, usability, reusability, fault tolerance, maintainability etc

The metrics available for coupling measurement belong to two major categories i.e. static and dynamic. Static metrics that are applied to the design or source code can only measure the expected coupling behaviour of object-oriented software and not the actual behaviour. 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. Dynamic metrics can be defined as the metrics used to measure the internal quality attributes of object-oriented systems by working on the information gathered from the runtime behavioural analysis of such systems. There are just a few dynamic coupling metrics proposed till date.

1.1 DYNAMIC COUPLING METRICS

The term *coupling* means interaction-based coupling when viewed from a static context. But it is not so when viewed from a dynamic/runtime context. As coupling is affected by the object-oriented features like inheritance and dynamic binding at runtime, 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, dynamic binding etc.). We call this combination *dynamic coupling* or *coupling at runtime*. Dynamic coupling measures were introduced as a refinement to existing coupling measures due to gaps in addressing dynamic binding, polymorphism, and the presence of unused code by static structural coupling measures.

II. Literature Survey

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

Briand *et al.* [2] 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 [3][4], Response For Class (RFC) and RFC∞ [3][4], [8] Efferent Coupling (Ce), Afferent Coupling (Ca) [8], Coupling Factor (COF) [2], Message Passing Coupling (MPC) [6], Data Abstraction Coupling (DAC) and DAC1 [7], [7] Information-flow-based Coupling (ICP), Briand *et al.* suite (IFCAIC, ACAIC, OCAIC, FCAEC, etc) [1]. Most of these metrics are based on method invocations and attribute references.

Chidamber and Kemerer [3] 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 [4]. They concluded tht when there is high coupling it leads to high complexity. Defination of CBO is later revised to include coupling due to inheritance [4].

Hitz and Montazeri [5] introduced two distinct levels of coupling, class-level and object-level coupling.

Schikuta [10] 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 behavior of the system. This is because static systems give only measures for syntactical program analysis and thus have limited ability. They also insisted that the dynamic measures should always be used in combination with their static counterparts.

Briand *et al.* [1] studied a vast amount of available literature on coupling in object-oriented systems and concluded that all the metrics at that time measured class level coupling evaluated from static code.

Paques and Delcambre [9] presented an approach to evaluate the dynamic coupling at analysis phase. They proposed Dynamic 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.

Yacoub *et al.* [11] defined a set of object-level dynamic coupling metrics designed to evaluate the change-proneness of a design.

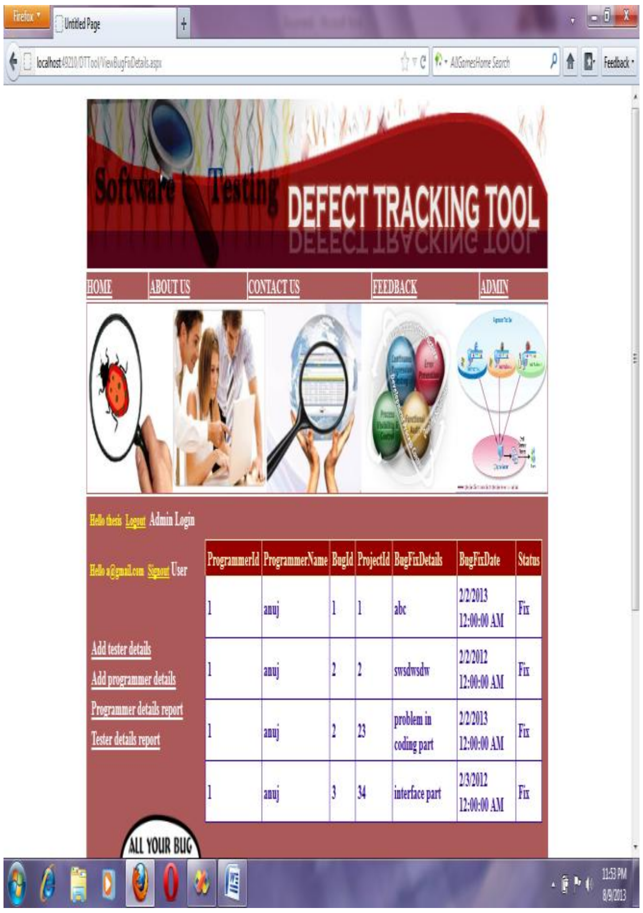
III. Objectives

The conventional static metrics have been found to be inadequate for modern object-oriented software due to insufficient contribution of object-oriented features such as polymorphism, dynamic binding, inheritance and unused code towards their measurement. This fact motivates to focus on dynamic metrics in addition to the traditional static metrics. A few new dynamic metrics are proposed for the measurement coupling at run-time. Moreover, different approaches for the dynamic analysis of programs required for collection of run-time data for the measurement of dynamic metrics are compared. AOP approach is used for the computation of coupling metrics. Dynamic coupling measurement has also caught little attention of researchers. Existing dynamic coupling metrics take into account only method-method invocations between objects of classes at run-time.

1. To propose new dynamic coupling metrics which take into account all major types of relations (a) run-time aggregation relations
 (b) run-time inheritance relations
 (c) run-time method-attribute reference relations
 (d) run-time method-method invocation relations, between the objects of different classes during measurement.
2. To develop a dynamic coupling tracer application for the purpose of computation of the proposed metrics. With the help of this tracer application, the proposed metrics will be validated in .NET framework

IV. Results & Discussion

A defect tracking system is a application designed in Visual studio 2008 and is successfully integrated with N-Crunch . It is very valuable and allow users to easily remove bugs. All issues related to bugs are studied while fixing the bug. A defect is a kind of ticket which is opened normally by a testing team when they find any bug. Here we have two logins. One is admin and other is user. The main work of admin is to report the problem, means to say admin can forward all the bug report to the appropriate user. Once we login as admin then we add tester details ,which include tester name , tester email id, tester password, tester qualification, tester experience. After filling all details we submit this. Tester is that person which performs testing with suitable test cases, also apply testing on all modules. In admin we add programmer details also, which include programmer name, programmer email id, programmer password, programmer qualification, programmer experience.



Hello thesis Logout Admin Login

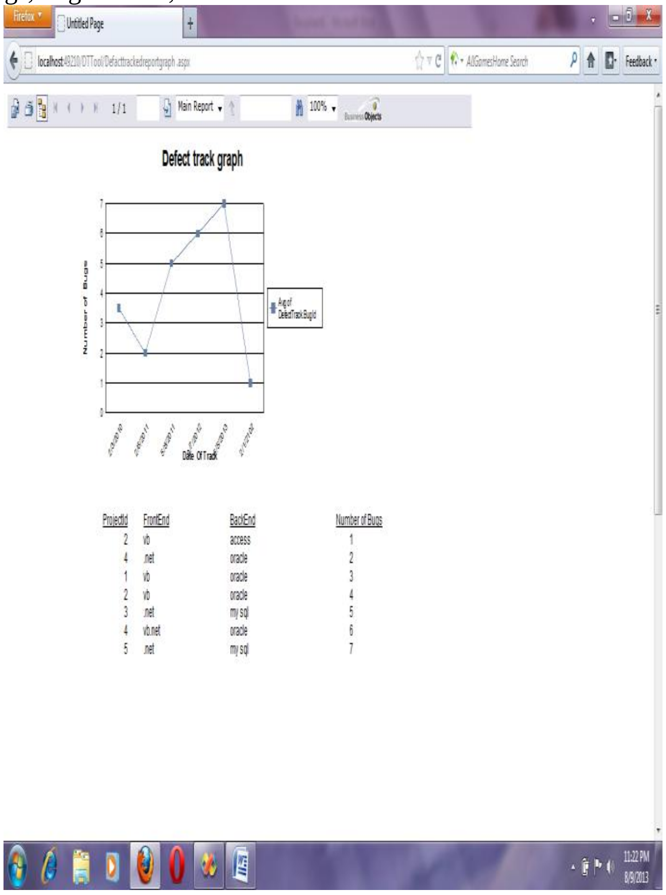
Hello a@gmail.com Signout User

Add tester details
Add programmer details
Programmer details report
Tester details report

ProgrammerId	ProgrammerName	BugId	ProjectId	BugFixDetails	BugFixDate	Status
1	anuj	1	1	abc	2/2/2013 12:00:00 AM	Fix
1	anuj	2	2	swsdwsdw	2/2/2012 12:00:00 AM	Fix
1	anuj	2	23	problem in coding part	2/2/2013 12:00:00 AM	Fix
1	anuj	3	34	interface part	2/3/2012 12:00:00 AM	Fix

Figure 4.1 Bug Fixed Details

Figure 4.1 shows details of bugs, bug fix date, and also status.



ProjectId	FrontEnd	BackEnd	Number of Bugs
2	vb	access	1
4	.net	oracle	2
1	vb	oracle	3
2	vb	oracle	4
3	.net	my sql	5
4	vb.net	oracle	6
5	.net	my sql	7

Figure 4.2 Defect Track Graph

Above graph is between number of bugs and date of track, details of front end and backend used and also details of number of bugs found.

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 covervage and performance metric, inline in your IDE while you type.
Full code coverage matrix are also available for your entire solution. No matter where your source code you will be able to analyse your problem quickly

References

[1] L. C. Briand, P. Devanbu and W. L. Melo, "An Investigation into Coupling Measures for C++", In *Proceedings of 19th International Conference on Software Engineering*, 1997, Boston, USA

[2] "A Unified Framework for Coupling Measurement in Object-Oriented Systems" L.C. Briand, J.W. Daly, and J.K. Wust,, *IEEE* vol. 25, no. 1, pp. 91–121, Jan 1999

[3] "Towards a Metrics Suite for Object-Oriented Design", S. R. Chidamber, and C. F. Kemerer, *(OOPSLA' 91)*, 1991, SIGPLAN Notices, Vol. 26, no.11, pp. 197–211.

[4] "A Metrics Suite for Object-Oriented Design" S. R. Chidamber and C. F. Kemerer, *IEEE Transactions Software Engg.*, pp. 476-493, 1999 vol. 20,

[5] "Measuring Coupling and Cohesion in Object-Oriented Systems" M. Hitz and B. Montazeri,, In *Proceedings International Symposium on Applied Corporate Computing*, 1995, Monterrey, Mexico.

[6] "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow" Y. S. Lee, B. S. Liang, S. F. Wu, and F. J. Wang,, In *Proceedings International Conference Software Quality*, 1995, Maribor, Slovenia.

[7] "Object-oriented metrics that predict maintainability" Wei Li and Sallie Henry, *Journal of Systems and Software*, vol. 23, no. 2, pp. 111-122, Nov. 1993.

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

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

[10] "Dynamic Coupling Metrices" E. Schikuta,, 1993.

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