Volume 4, Issue 6, June 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

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*— *A good quality software product requires efficient measures to accurately monitor the internal software quality attributes. Software metrics have been widely and successfully used to measure such internal quality attributes for object-oriented software systems. Coupling is an important qualitative measure for measuring the performance of software. Coupling is defined as how much dependent a module is on other modules. Coupling defines the external complexity of a class, i.e. how dependent a class is on other classes.

For coupling measurement there are two metrics available static metrics and dynamic metrics. Static metrics measure the expected coupling behavior of object-oriented software and not the actual behavior. Whereas dynamic metrics can measure the actual coupling behavior 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 behavioral analysis of such systems. The objective of my thesis is to propose a new dynamic coupling metrics.

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

I. INTRODUCTION

software metrics proposed are used for procedural paradigm have been found inadequate for object-oriented software products mainly because of the distinguishing features of the object-oriented paradigm such as classes, encapsulation, inheritance and polymorphism[1]. The most popular and time-honoured software metrics have been loc (lines of code) and cyclomatic complexity. These measures were originally defined for procedural programs and later incorporated for object-oriented systems. The loc metric is a measure of a size of a module and cyclomatic complexity measures logical complexity of a module. Coupling between packages is the degree of interdependence between them. Theoretically each package is a stand-alone unit, but in reality packages may depend or rely on each other as either they require services from other packages or provide services to other packages. Thus, coupling between various packages cannot be completely avoided but it can only be controlled to some level. Coupling between packages is major factor that affects external quality attributes of software. The increasing importance has being placed on coupling measurement for evaluating and predicting the quality of software. [3]

Static metrics that are applied to the design/source code can only measure the expected coupling behavior of object-oriented software and not the actual behavior. The behavior 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 measure the actual coupling behavior as they are evaluated from data collected during runtime. Dynamic metrics is used to measure the internal quality attributes of object-oriented systems by working on the information gathered from the runtime behavioral analysis of such systems. These dynamic metrics are usually obtained from the execution traces of the code or from the executable models. [4] If used properly, software engineering metrics allow us to quantitatively define the degree of success or failure, for a product, a process, or a person, make meaningful and useful managerial and technical decisions, and make quantified and meaningful estimates [5].

II. ANALYSIS

Payal Khurana & Puneet Jai Kaur [1] described that software metrics measure different aspects of software complexity and therefore play an important role in analyzing and improving software quality. They provide useful information on external quality aspects of software and provide a means of estimating the effort needed for testing.

Traditional metrics for measuring software such as Lines of Code (LOC) have been found to be inadequate for analysis of object-Oriented software.

In recent years many researchers and practitioners have proposed a number of static code metrics for object-oriented software, e.g. the suite of metrics proposed by Chidamber and Kemerer. These code metrics quantify different aspects of complexity of the source code. However the ability of such static metrics to accurately predict the dynamic behavior of an application is as yet unproven. Static metrics alone may be insufficient in evaluating the dynamic behavior of an application at run time, as its behavior will be influenced by the operational environment as well as complexity of object-oriented software.

Cohesion refers to the relatedness of the elements in a module. A module that is highly cohesive is one whose elements have tight relationship among them in order to provide a single functionality of the module, whereas a low cohesive

module has some of the elements that have little relation with others, which indicates that the module may contain several unrelated functionalities. It is widely accepted higher the cohesion of a module is, more easy is to develop the module, maintain and reuse. In the object oriented paradigm, various cohesion measures for classes have been proposed. In the beginning of research on cohesion measures for classes, researchers just considered syntactic relationship between class members such as interaction between class members such as interactions between instance variables and methods. However more recent researches have been tried to identify inherent characteristics of classes which can affect the cohesiveness of classes and incorporate them into cohesion metrics.

Shweta Sharma [2] proposed that software metrics are used to check and evaluate various aspects of the complexity of a software product. Cohesion and Coupling are attributes that are considered to be the most important. Software quality measurements resulted into extensive research into software metrics and the development of software metric tools.

Many Software Metrics were proposed for object oriented systems to measure various attributes of software quality. Many different object-oriented coupling and cohesion metrics have been developed. To develop high quality software, choice of developer is always low coupled and highly cohesive design.

Vasudha Dixit, Dr. Rajeev Vishwkarma [3] described object-oriented programming is the dominant development paradigm for software systems. In object oriented programming we provide abstraction by classes and interfaces. Classes are used to hold functional logic and an interface is used to organize source code. According to object oriented programming, the interface provides abstraction and cannot inherit from one class but can implement multiple interfaces. Complexity of source code is directly related to quality and cost of software. Coupling models are presented to measure the possible interactions between objects. High coupling between various objects increases complexity as well as cost. Low coupling is good for designing object oriented software. Inheritance introduces more interactions among classes. This will increase the complexity. This paper presents a comparison between object oriented interfaces and inheritance class diagrams. Software engineering started with a humble beginning and it has slowly come into existence. Now, software engineering provides the best solutions for the software problems in object oriented programming. Accurate measurement is a basic necessity for all engineering disciplines and software engineering is not an exception. To improve software quality, measurements are essential.

Software measurement plays an important role in finding the software quality. The concept of measurement requires appropriate measurement tools to measure, to collect, to verify and validate relevant metric data. Nowadays, many metric tools are available for software measurement.

S. P. Sreeja [4] defined OO design and development is becoming very popular in today's software development environment. OO analysis considers the problem by looking for system entities that join them. Object oriented analysis and design pays attention on objects as the primary agents involved in a computation; each class of data and related operations are collected into a single system entity.

Since there is availability of various software metrics, it is important to decide what type of metric that is best suited for the environment. Selection of metric is based on the specific needs of a software project that can be used for measuring the quality of software. The validity of these metrics needs to be checked thoroughly by using various ongoing projects at run time. There exist few OO metrics that are most popular and used frequently. Lack of Cohesion in Methods (LCOM) is one of the prevalent metrics that is used to measure the dissimilarity of methods in a class.

Schikuta [5] was the first to propose a dynamic approach to measure coupling of software systems. It was observed that the previously used measurement systems were statically oriented and provided the incomplete dynamic behavior of the system. That is why static systems give only measures for syntactical program analysis and thus they have limited ability to measure.

III. PROPOSED WORK

The conventional static metrics have been found to be inadequate for modern object-oriented software due to insufficient contribution of object-oriented features. Due to this fact we need 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.

In this paper a dynamic coupling tracer application is developed for the purpose of computation of new dynamic coupling metrics. Existing dynamic coupling metrics take into account only method-method invocations between objects of classes at run-time.

Dynamic coupling metrics will take into account all major types of relations: run-time aggregation relations, run-time inheritance relations, run-time method-attribute reference relations, run-time method-method invocation relations.

IV. TOOL USED

a. Visual Studio 2008

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows super family of operating systems, as well as web sites, web applications and web services. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silver light. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. Visual Studio supports different programming

languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists.

b. N Crunch

N crunch tool is an automated concurrent testing tool which is used to test various modules by using different types of metrics.

N crunch gives you a huge amount of useful information about your tested code such as code coverage

Features of N Crunch:

- Automatic Concurrent Testing
- Code Coverage
- Performance metrics
- Parallel Execution
- Easy Debugging

V. RESULTS AND DISCUSSION

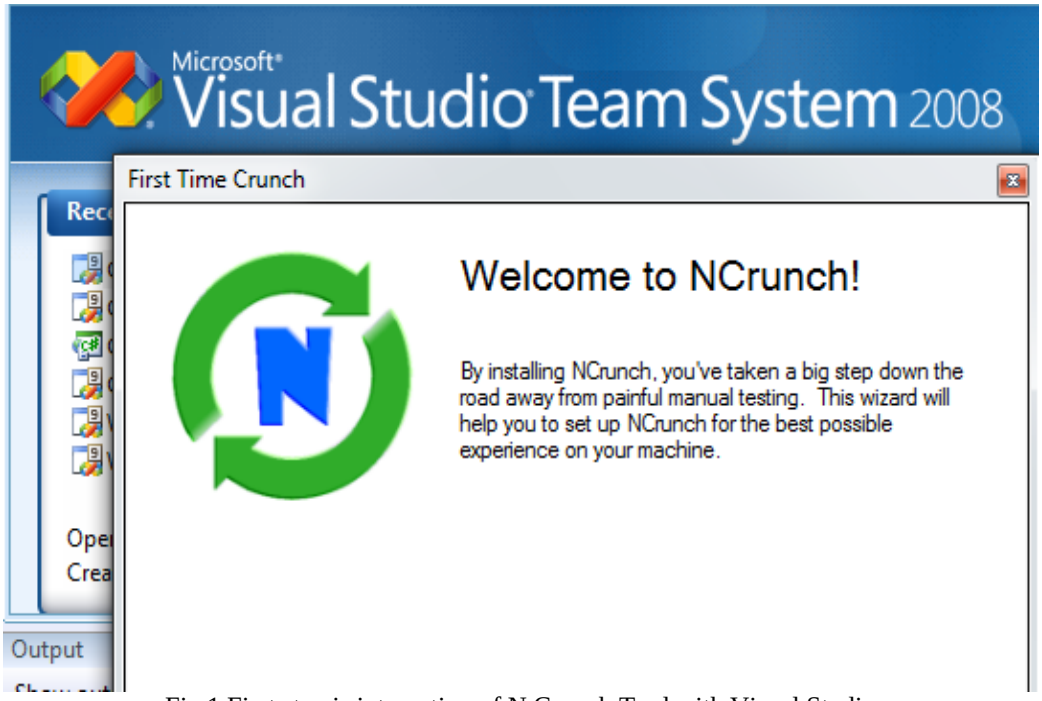

Fig 1.First step is integration of N Crunch Tool with Visual Studio

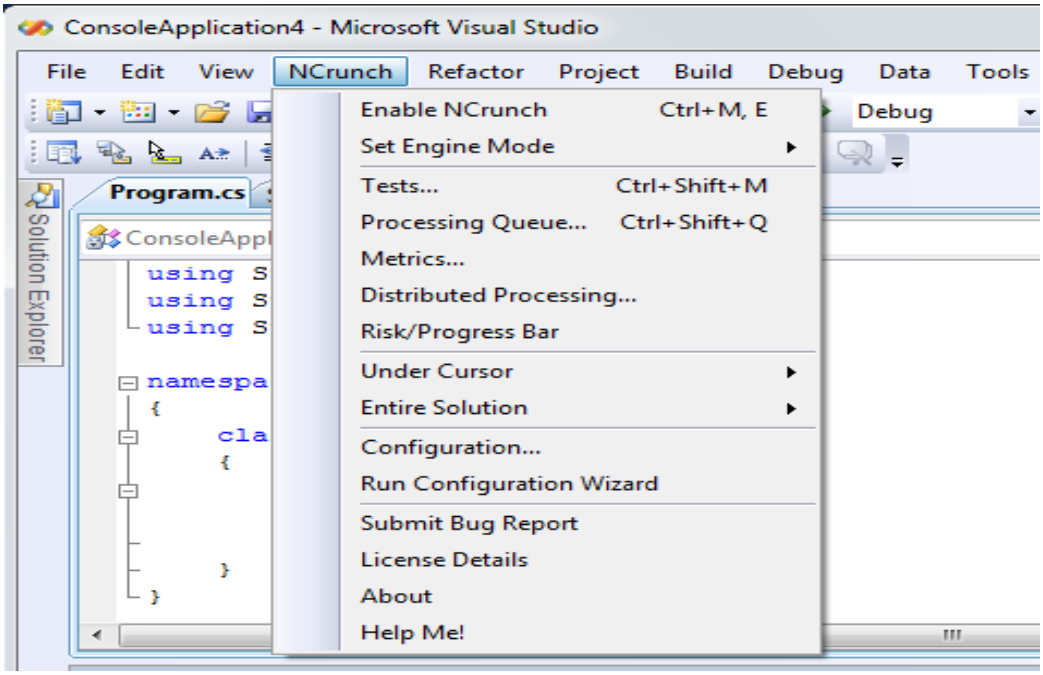

Fig2. Enable NCrunch from NCrunch menu with the application

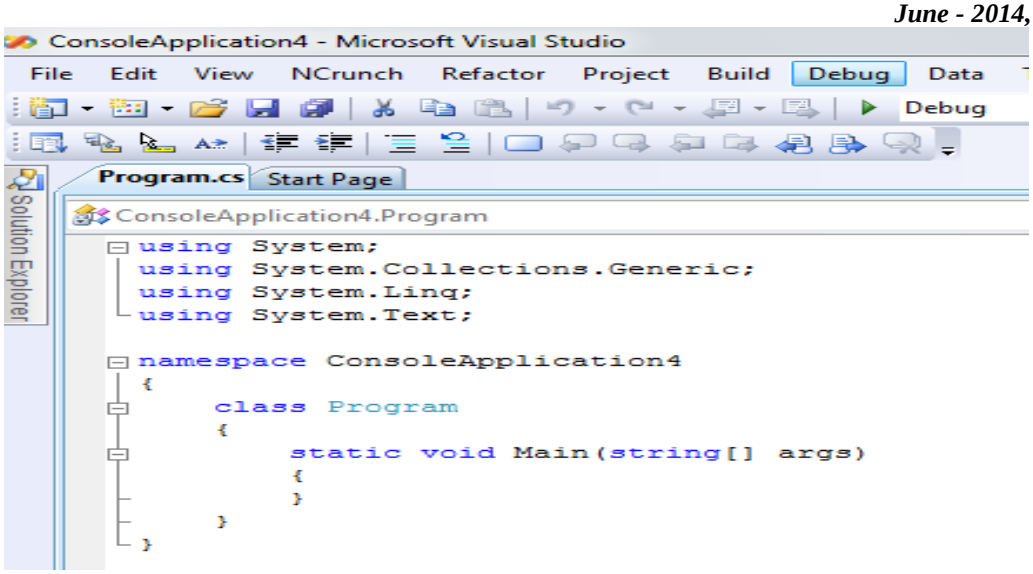

Fig3. Submit the code to be tested by the NCrunch Tool

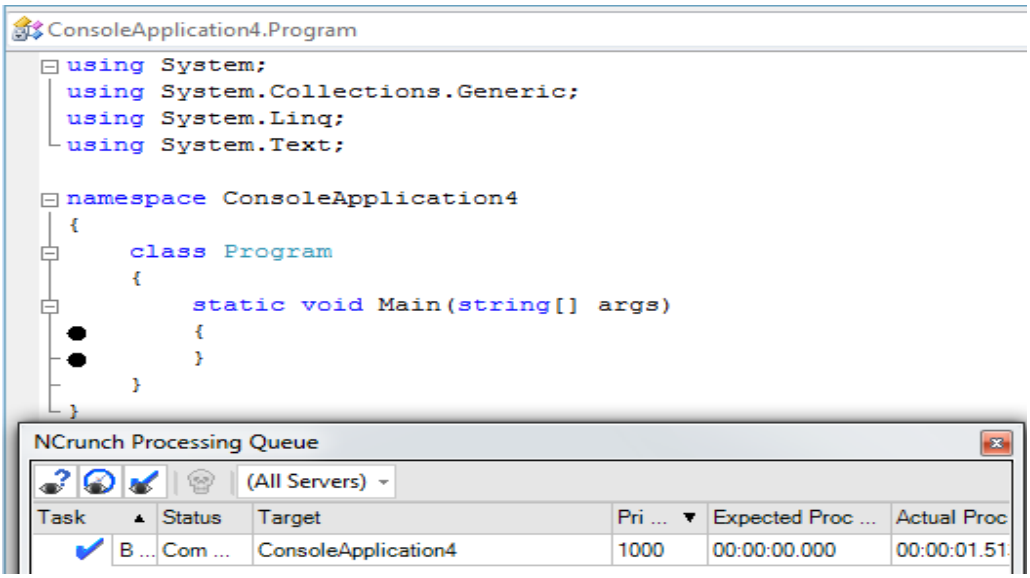

Fig4. N Crunch Processing Queue shows the result after testing that is successful. Code window shows code without errors.

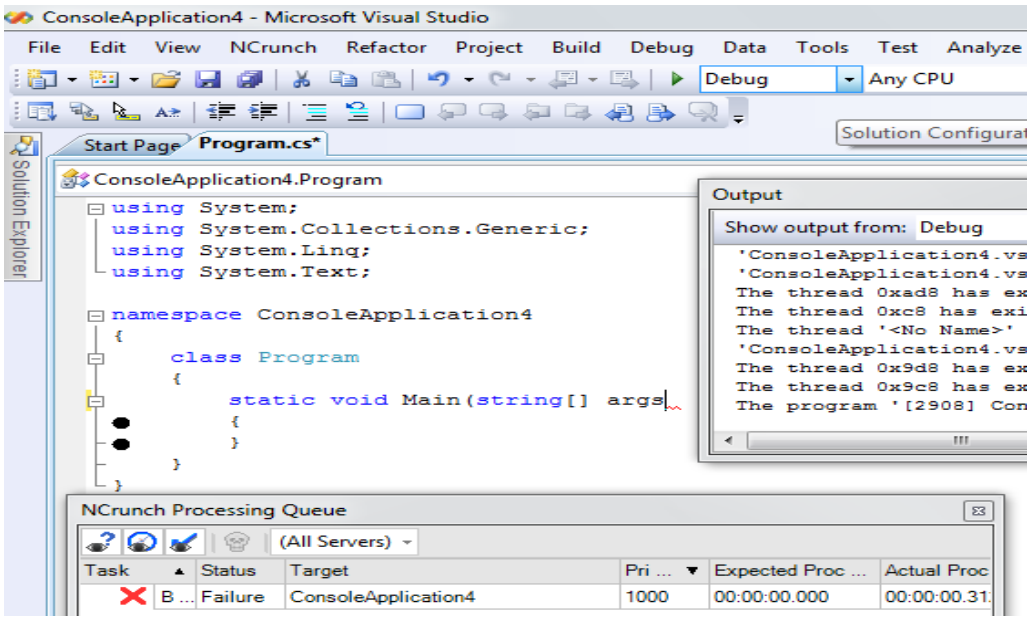

Fig5. Here the code window shows the code with errors. Black Circles indicates the code is tested and built. N Crunch Processing Queue shows the result after testing that is failure.

VI. CONCLUSION

N crunch gives us huge amount of useful information about our tested code such as code coverage. It tells us about which type of testing can be performed on code such as unit testing or integration testing. N Crunch provides us great advantage for our entire solution. No matter where we are in our source code, N Crunch will be able to analyse problems quickly. Full code coverage metrics are also available for our entire solution. N Crunch provides efficient method of testing our code and it reduces overhead of testing modules.

REFERENCES

[1] Payal Khurana, Dynamic Metrics At Design Level, Volume 2, pp. 449-454.
[2] Shweta Sharma, Coupling and cohesion metrics in object oriented environment, Volume 4.
[3] Vasudha Dixit, Static and Dynamic coupling and cohesion measures in object oriented programming, Volume 2, pp: 472-77
[4] Erik Arisholm , Lionel C. Briand and Audun Foyen, "Dynamic Coupling Measurement for Object-Oriented Software", IEEE Transactions on Software Engineering, vol. 30, no. 8, pp. 491-506, August 2004. International Journal of Information and Telecommunication Technology Vol. 1, Issue 1, 2010, (0976-5972).
[5] S.p. Sreeja, different approaches of determining cohesion based object-oriented metrics, Volume 4, pp. 30-38.
[6] 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 (ICSE'97), 1997, Boston, USA, pp. 412–421.
[7] L. C. Briand, J. Daly, V. Porter and J. Wust, "A Comprehensive Empirical Validation of Product Measures", Vol. 4
[8] L.C. Briand, J.W. Daly, and J.K. Wust, "A Unified Framework for Coupling Measurement in ,Object-Oriented Systems", IEEE Transactions on Software Engineering, vol. 25, no. 1, pp. 91–121, Jan 1999.
[9] M. Cartwright and M. Shepperd, "An Empirical Investigation of an Object-Oriented Software System", Department of Computing, Bournemouth University, UK, Technical Report, 1997.
[10] S. R. Chidamber, and C. F. Kemerer, "Towards a Metrics Suite for Object-Oriented Design"1991,SIGPLAN Notices, Vol. 26, no.\11, pp. 197–211.
[11] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object-Oriented Design", IEEE Transactions Software Engg., vol. 20, pp. 476-493, 1994.
[12] M. Choi and S. Lee, "A Dynamic Coupling for Reusable and Efficient Software System", In Proceedings of the 5th ACIS international Conference on Software Engineering Research, Management & Applications, 2007, pp. 720-726. SERA. IEEE Computer Society, Washington DC.
[13] P. Coad, and E. Yourdon, Object-Oriented Design, Prentice Hall, Yourdon Press, 1991.
[14] B. Dufour, K. Driesen, L. Hendren, and C. Verbrugge, "Dynamic Metrics for Java", In Proceedings of the ACM SIGPLAN'03 Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '03), 2003, Anaheim, California, USA.
[15] K. El Emam, "Object-Oriented metrics: A review of theory and practice", In Advances in Software Engineering: Topics in Comprehension, Evolution, and Evaluation, O. Tanir and H.Erdogmus, Eds., pp. 23-50, Spinger-Verla
[16] N. Gupta and P. Rao, "Program Execution-Based Module Cohesion Measurement", In Proceedings 16th IEEE International Conference on Automated Software Engineering (ASE'01), 2001, pp. 144-157.

