IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727,
PP 37-42
www.iosrjournals.org

A Framework for Dynamic Software Analysis & Application Performance Monitoring

Dr. Ashish Oberoi[1], Pallavi[2]
[1](Cse, / M.M Engineering College, India)
[2](Cse, / M.M Engineering College, India)

Abstract : *The objective of this research paper is to analyze software product that requires efficient measures to accurately monitor the internal software quality, based on modular design. During the course of software development life cycle, it involves defect avoidance rather than defect removal. Software metrics have been widely used to measure internal quality attributes such as coupling and cohesion for object-oriented software systems. The metrics available for coupling measurement is divided into two major categories i) Static metrics and ii) Dynamic metrics. Static metrics can only measure the expected coupling behavior of object-oriented software and not the actual behavior. Dynamic metrics can capture the actual coupling behavior as they are evaluated from data collected during runtime. Results are shown by using N-Crunch technique to measure software performance, dependency cycles between components to achieve higher code maintainability. Lines of Code, Cyclomatic Complexity, Coupling, Nesting Depth and Rank etc.*

Keywords: *Metrics, Object Oriented, Performance Monitoring, Software, Testing.*

I. INTRODUCTION

Software metrics enable us to make meaningful estimates for software products and help us in taking managerial and technical decisions. Conventional static metrics have been found to be inadequate for modern object-oriented software. It motivates us to focus on dynamic metrics in place of traditional static metrics . Coupling has been defined as one of the most basic qualitative measures for measuring the performance of software during design or implementation phase. It is normally defined as the degree of interdependency among modules. Coupling measures the external complexity of a class . The metrics suite for object-oriented design is partly evaluated by applying principles of measurement theory. Using the object coupling measure as an example, failing to establish a sound empirical relation system may lead to deficiencies of software metrics[3] .

II. RELATED STUDY

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).

Chidamber and Kemerer [2] introduced the first coupling metric for object-oriented systems. In their metric suite, they defined CBO (Coupling between Objects) .

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 dunamically.

Schikuta [4] proposed 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.

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

S S. Babu and R.M.S. Parvathi et al. [6] defined coupling measurement has traditionally been performed using static code.

III. PROPOSED METHODOLOGY

The two new steps that is Collaborative process for requirement engineering and Mutation testing is discussed in detail in this chapter. Collaborative process is introduced for validating rapidly changing requirements. A white box technique that is Mutation testing is introduced to prevent human errors in the software The steps for proposed Cleanroom Software Engineering is shown in figure 1.

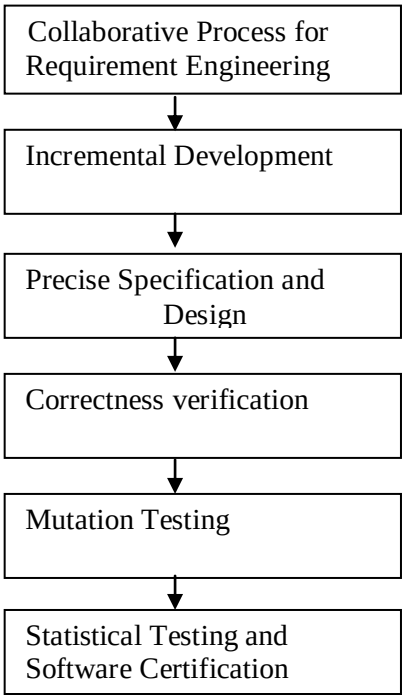

Fig.1. Proposed Technique for Cleanroom Software Engineering

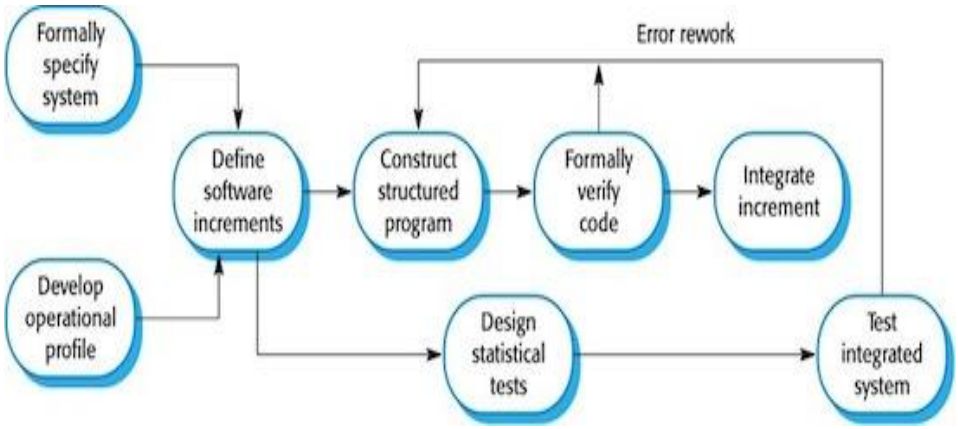

Fig.2. The Cleanroom Process

There are large number of object-oriented metrics that have been proposed in the literature for assessing 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 may be useful in envisaging 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 object-oriented software attributes. Many metrics have been proposed in the last several years to measure class cohesion in object-oriented systems. Class cohesion is defined as the degree of relatedness between different members of a class. Object-oriented analysis and design methods endorse a

modular design by creating high cohesive classes. However, improper assignment of responsibilities in the design phase can lead to low cohesive classes with unrelated members .

The insight provided by testing is valuable during different phases such as design, coding, testing and quality assurance. Testability generally used to detect a fault. Therefore improving software testability is an important objective in order to reduce the number defects that result from poorly designed software. It is an foreseeable fact that testability information is useful that may be complementary to testing. Higher test coverage may be achieved by transforming a system more testable for the same effort. Achieving testability is mainly a matter of separation of concerns, coupling between classes, subsystems, and cohesion . The term coupling means interaction-based coupling when viewed from a static context. But it is not so when viewed from a runtime context.

Distributed systems with service oriented components have more heterogeneous networking. Traditional coupling measures had been taken into account only "static" coupling. They lack for "dynamic" coupling due to polymorphism which 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 use 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 dynamically. The implementation level results show that the proposed system will effectively measure the coupling metrics dynamically .

IV. PROPOSED WORK

A few new dynamic metrics are proposed for the measurement of coupling at run-time. Moreover, different approaches for the dynamic analysis of softwares 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 efficient 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 such as run-time aggregation relations, run-time inheritance relations, run-time method-attribute reference relations and run-time method-method invocation relations.

V. TOOL USED

i) 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.

ii) N Crunc: 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 large amount of useful information about tested code such as code coverage features of N Crunch:
a) Automatic Concurrent Testing
b) Code Coverage
c) Performance metrics
d) Parallel Execution
e) Easy DebuggingAn acknowledgement section may be presented after the conclusion, if desired.

VI. RESULTS AND DISCUSSION

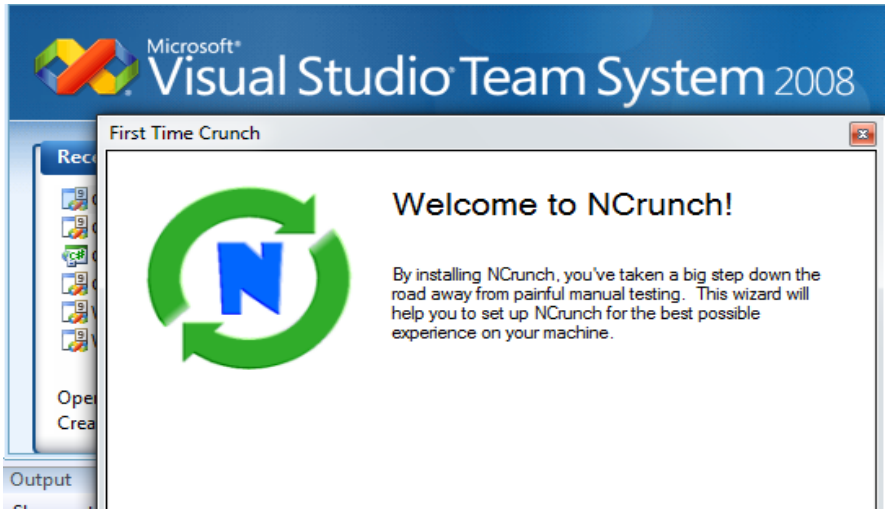

Fig.3. First step is integration of N Crunch Tool with Visual Studio

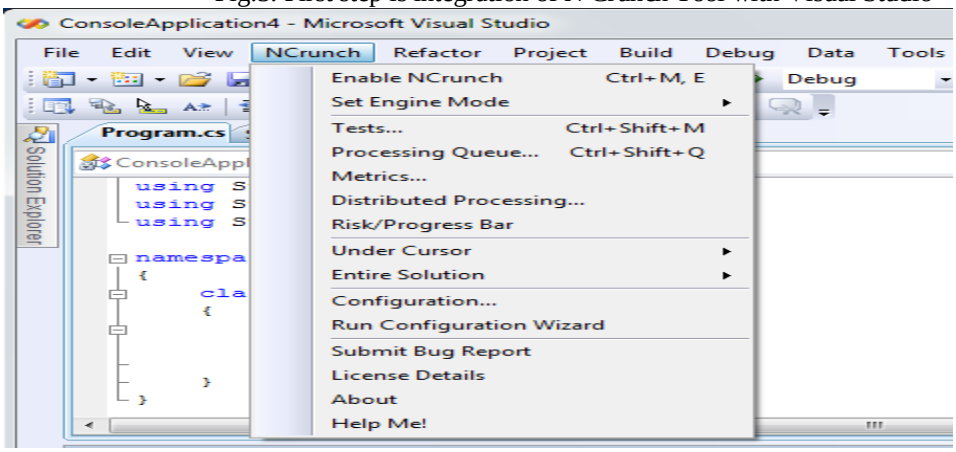

Fig 4: Enable NCrunch from NCrunch Menu with the Application

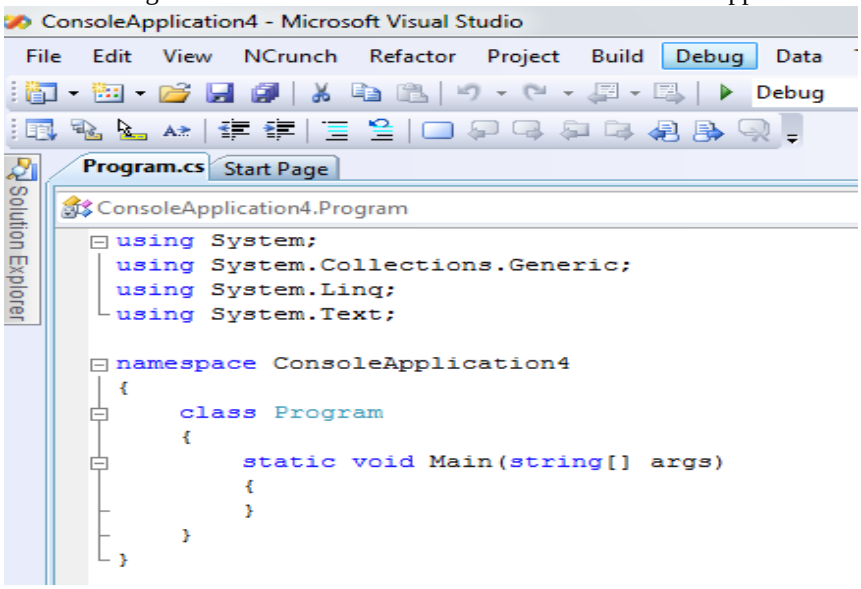

Fig.5. Submit the code to be tested by the NCrunch Tool

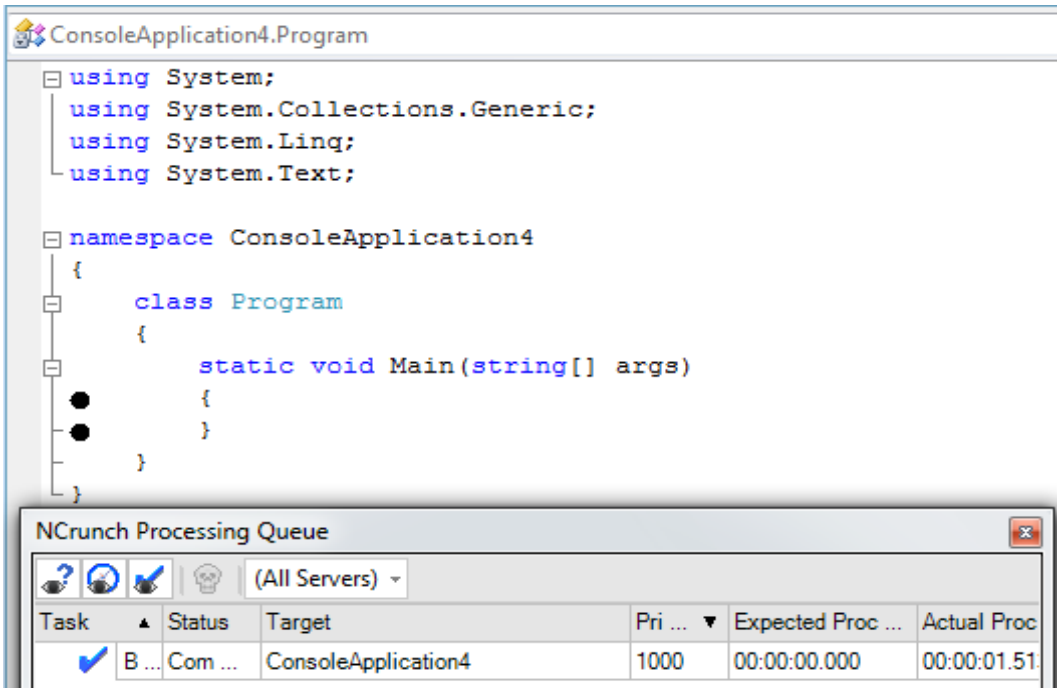

Fig.6. N Crunch Processing Queue Shows the Result after Testing is Successful.

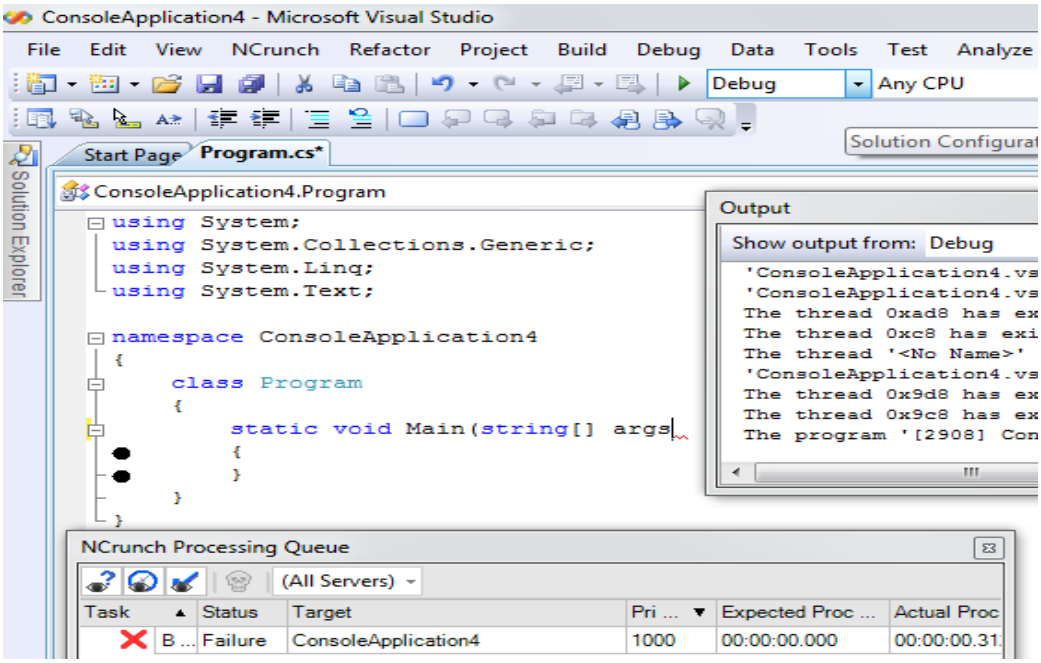

Fig.7. Final Processing

VII. CONCLUSION

From study, it has been found that AOP approach provides a balanced method for the dynamic analysis of programs and softwares. In addition, AOP approach is far easier to implement and at the same time an efficient one for dynamic analysis without any side effects. N crunch gives us huge amount of valuable information about tested code using code coverage. N Crunch provides great improvement for our entire solution. No matter where we are in our source code, N Crunch is efficient enough to analyze problems quickly. Full code coverage metrics are also available for entire solution. Further N Crunch reduces overhead of testing modules.

REFERENCES

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

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

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

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

[5] Sherif M. Yacoub, T. Robinson, and H. H. Ammar(1999), "Dynamic Metrics for Object Oriented Designs", *In Proceedings of the 6th International Symposium on Software Metrics*

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

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

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

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

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

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