Analyzing OpenCover Code Coverage tool in Visual Studio

Khushbu
Maharshi Dayanand University, Rohtak, India
Kamna Solanki
Maharshi Dayanand University, Rohtak, India
Sandeep Dalal
Maharshi Dayanand University, Rohtak, India

ABSTRACT: ***In software testing process, Code Coverage analysis helps by finding areas which are not exercised by set of test cases of a program or to find defects in a program which are not exercised. It ensures testing of all key functional areas is carried out effectively and includes all essential features. The coverage information is very useful for many other related activities, like unit testing, regression testing, mutation testing etc. Code Coverage Analysis tools are used for Languages like Java, C, C++, Python etc. Working on testing code of programs helps to find problem/defects in particular software. For .NET applications, the only open source code coverage tool is OpenCover. It is easy to use and powerful tool. OpenCover doesn`t complain and just ignores the arguments it doesn`t recognize. Code instrumentation is not needed for using OpenCover tool. Producing PDB files in Opencover requires building the code into debug mode and run the application using OpenCover Prompt. OpenCover tool can be used for measuring code coverage of unit tests. This paper focuses on analyzing Opencover tool cove coverage analysis tool in detail.***

Keywords— Code coverage, Code Covereage tools, open cover tool, unit testing using OpenCover tool, ReportGenerator, Limitations

1 INTRODUCTION

1.1 *Introduction*

Code coverage analysis is the process of finding the areas of a program that are not exercised by a set of test cases. To increase code coverage, it may create additional test cases. It determines quantitative and qualitative measure of code coverage. It can also identify those redundant test cases that do not contribute towards enhancing test coverage or Code Coverage. The process of code coverage analysis is automated by code coverage analyzer (CCA). These are also known as test coverage analyzers (TCA). While testing software quality, developers can use code coverage analyzers to easily determine the tested and untested parts of an application. Coverage analyzer takes two inputs: set of test cases and code to be covered and produces output as a set of redundant test cases and percentage of code coverage. Sometimes it also shows the untested part of code using color coding scheme.

In general, a code coverage analyzer provides the following:

- Code coverage report (60% to 70% of code coverage of an application is acceptable)
- Coverage report by subsystem
    - Customization
    - Colour coding for source
    - Comparing
    - test-profiles of two runs

1.2 *Limitations*

1. In a multi-dimensional concept, only one dimension is measured by coverage techniques. The same coverage may be achieved by data coverage of two different test cases where input data of one test case find an error and the other doesn't.

2. Code coverage only measures what is written in the code; coverage related to the software application can`t be is measured.

3. If an unimplemented specified function was omitted from the specification, and then it specifies only the existed or available function.

1.3 *Code Coverage Metrics*

Software Metrics is used in measurement of software products and process. They help in tracking various aspects of a software project, such as rates of finding and fixing defects, changing requirements, and growth in size & complexity of code. The metrics typically focus on the quantity

and quality of any software, according to point of view of the testing. Code coverage Metrics strongly support SPM(software project management) activities mainly test management.

2 VARIOUS CODE COVERAGE ANALYZER TOOLS

There are various code coverage analyser tools that support different languages and performs

- **JAVA** language: JCover, Emma, Gretel, Code Cover, Cobertura, Clover, Hansel, JACOCO, NoUnit, PITest, Quilt, Serenity BDD, Testwell CTC++, Parasoft JTest, Spira Team.
- **C** language: Bullseye Coverage, Testwell CTC++, Frog Logic CoCo, Intel C++ Compiler 15.0, Parasoft JTest, Spira Team, Vector Software, Visual Studio.
- **C++** language: Bullseye Coverage, Testwell CTC++, Frog Logic CoCo, Intel C++ Compiler 15.0, Parasoft JTest, Spira Team, Vector Software, Visual Studio.
- **C#** language: Testwell CTC++, Frog Logic CoCo, Spira Team.
- **.NET** language: NCover, Parasoft JTest, Spira Team, Dot Cover, Open Cover.
- **COBOL** language: Code Cover, Spira Team.
- **Python** Programming:Coverage.py, Spira Team.
- **PERL** Language: Devel::Cover.

2.1 *Related work*

Asaf, S., E. Marcus, et al., have pointed out adopting a selective coverage analysis approach rather than a detailed one or do selective code coverage (based on partition operations, selection and projection). A program is implemented this can be applied to the software development process. Extreme coverage (method) means, In Unit testing ,all nontrivial remaining implemented methods are also 100% exercised where 100% of all unit tests must pass. In code coverage Extreme coverage has been measured by JBlanket (tool). Code coverage Looks at how much of our code is `used` or exercised`. They look at tools that can help us with code coverage, name NCover and NCoverExplorer tools [6]. Author proposed an approach to measure code coverage for both online and offline dynamic analysis tools. We then pick online tools including ABM, Anubis, Copper Droid, Trecedroid, as well as offline tools including standard Android emulator, Droid Box, and Droid Scope [19]. During process of projects all projects perform well in both code coverage and test suite effectiveness, with the exception of one project in which the test suite effectiveness drops drastically. This drop shows that all projects are at risk of low test suite effectiveness, by not using mutation testing techniques [23]. Therefore, an interesting direction of research would be to develop effective test case prioritization approaches for effective testing. The need of the hour is to focus on prioritization of test cases, whose effectiveness ultimately determines the software quality.

2.2 *.NET supported code coverage tools*

- **Visual Studio** In the .NET system by default visual studio code coverage has some features such as it reports coverage percentage at various granularities (e.g., class, assembly, etc.). One can select all of their tests or subsets of them. it lets you actually visualize the coverage with a option of paint IDE when you look at your code. It support maintenance and usage as it comes from Microsoft, and support if you invest in its usage.
- **DotCover** It get the features of Detailed Reporting, a cool “hotspots” view that call out risky methods, Both Visual Studio and CI integration of coverage measurements, IDE painting, Navigate to covering tests.
- **NCover** It provides extremely detailed information, includes supporting measurements of coverage and integrating it for the whole team. It provides centralized and detailed data about coverage. User support and extensive documentation. 32 bit and 64 bit processors support; plus memory consumption optimization. and IDE painting.
- **OpenCover** It supports .NET2 and above(only on windows) open source tool and works with both 32 bit and 64 bit processors.
- **NCrunch** In real-time, as you type it constantly runs your tests. There is no need to run your tests or even compile, to get feedback on whether your code changes are breaking anything.

NDepend This tool imports the data from coverage tool doesn`t directly measure code

coverage, and from imported data allows you to do some really powerful things with it.

PartCover :- This appears to have been created in response to NCover going commercial and though actively used it has limitations e.g. 32-bit only. The original repository is no longer being maintained by its original developers and is now being maintained on github. Now it is not further developing and moved to or replaced by OpenCover.

Coverage.eye: - Originated at Microsoft and was available on gotdotnet, the repository/sample has since gone and no full mirrors appear to exist.

*Difference between opencover and ncover :- NCover reports higher code coverage percentage than OpenCover.

III. Opencover tool:-

Open cover is a free open source code coverage tool for .net 2.0 and above running on the .NET platform. It supports sequence coverage, branch coverage and has a cover by test facility. Though OpenCover is command line only, a rich HTML UI of the results can be visualized using ReportGenerator.

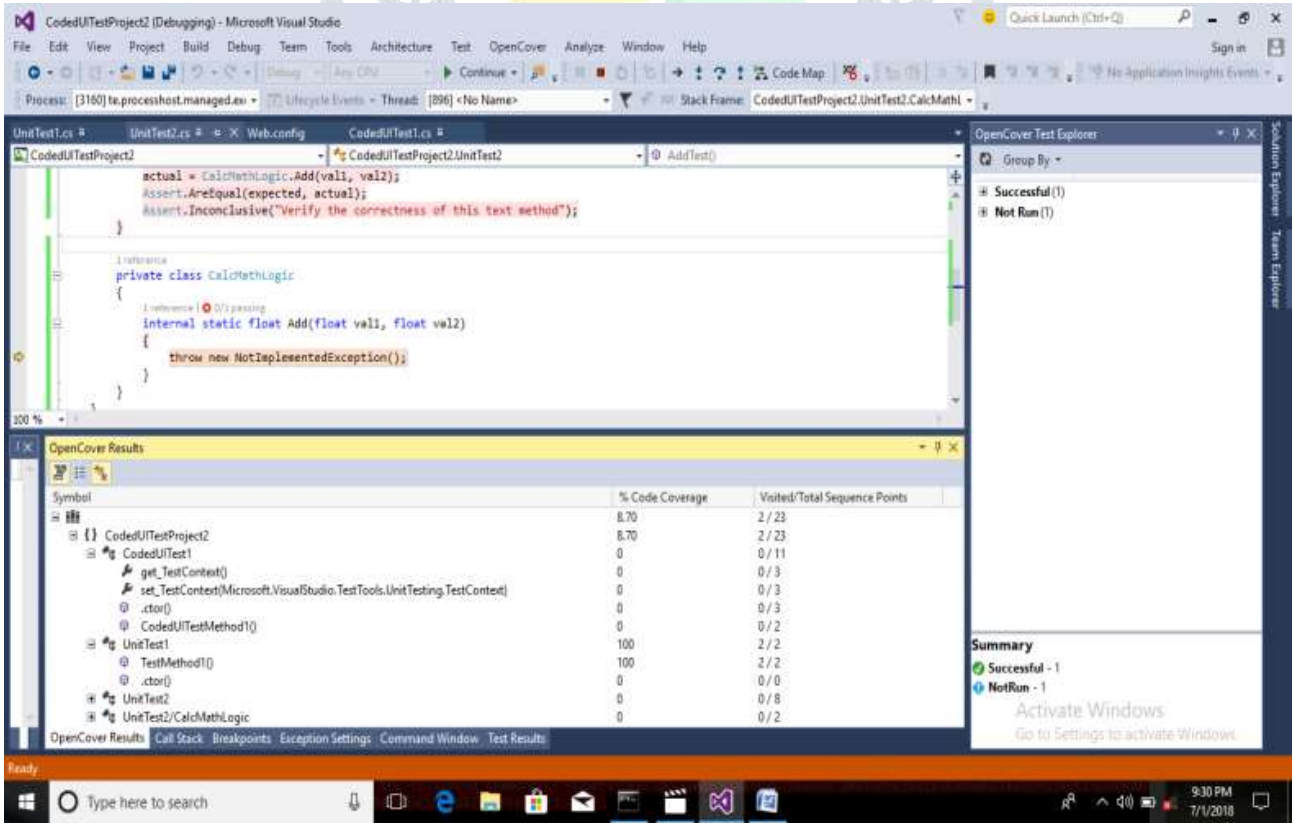

Fig. 1: Code Coverage with Coloring In Visual Studio

After debugging:-

It provides the feature of debugging where code coverage can be increased and debugging is done by selecting function name right click on it and select debugging. It will create this particular result:-

OpenCover is currently the only actively developed and maintained open-sourced tool of its type for the .NET platform.

OpenCoverUI:-

In this paper we have taken three different programs and created test cases and their comparison is done here.

First Program is about taking two values adding them and comparing that result is equal to the third value or not if compared show message addition is correct otherwise addition is not correct using if-else statement. We have created two three test cases here: one is CodedUITest , second is UnitTest1, and third is UnitTest2. Their results are as follows:-

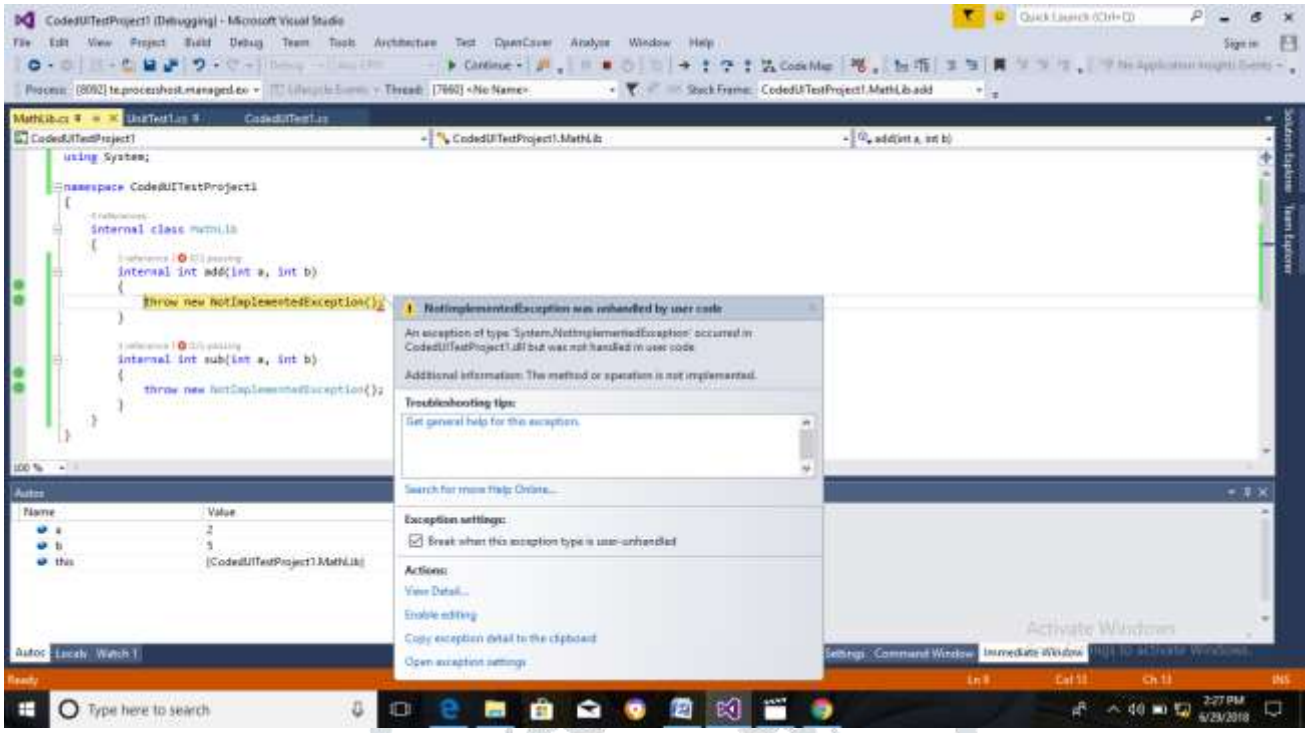

Figure 2:-Showing code coverage debugging using OpenCover tool

Second Program includes three numbers where addition of two numbers is equal to third number here we use the concept of exception handling and if-else statement.

Three test cases are generated here: first CodedUITest, second UnitTest1, and third UnitTest2. Their results are shown here:-

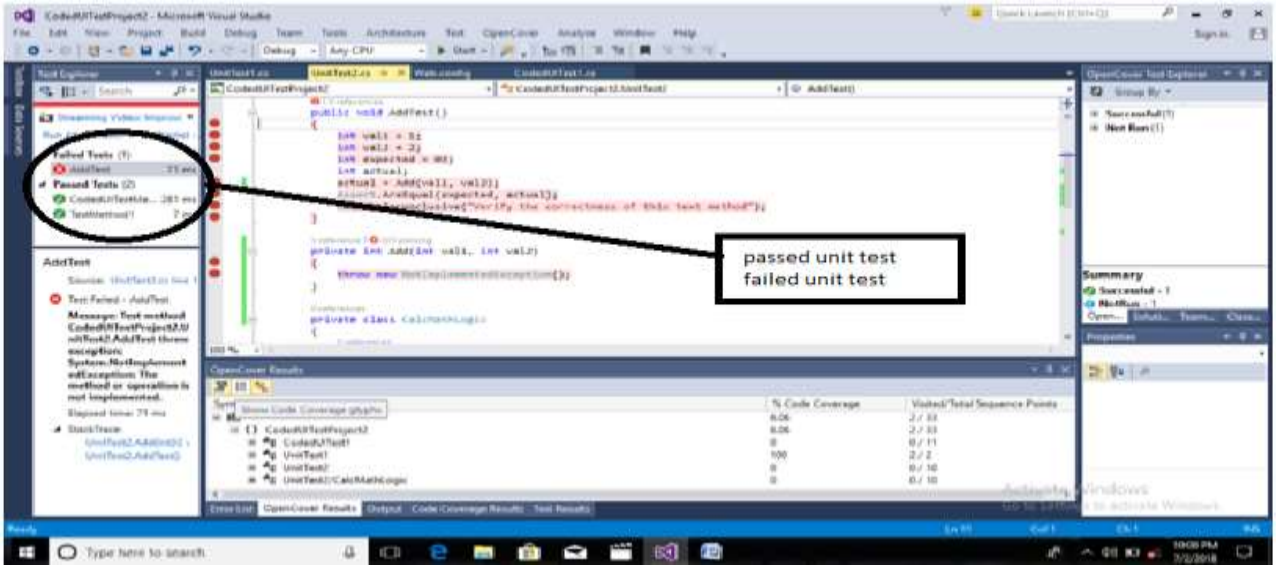

Figure3: showing CodedUITest and UnitTests results using OpenCover tool in visual studio for another program

In above figure we have shown code coverage results as well as one test case is failed here and two test cases are passed here is shown using circle and also code coverage colouring is here.

Third is a bank account application program, where it includes customers debit, credit, balance details.

Code Coverage includes four unit test cases related to debit, credit, debit-credit, balance details. Here we get different code coverage by taking different values. Here we get redundant test cases also which don't increase code coverage. Results of this application are shown below:-

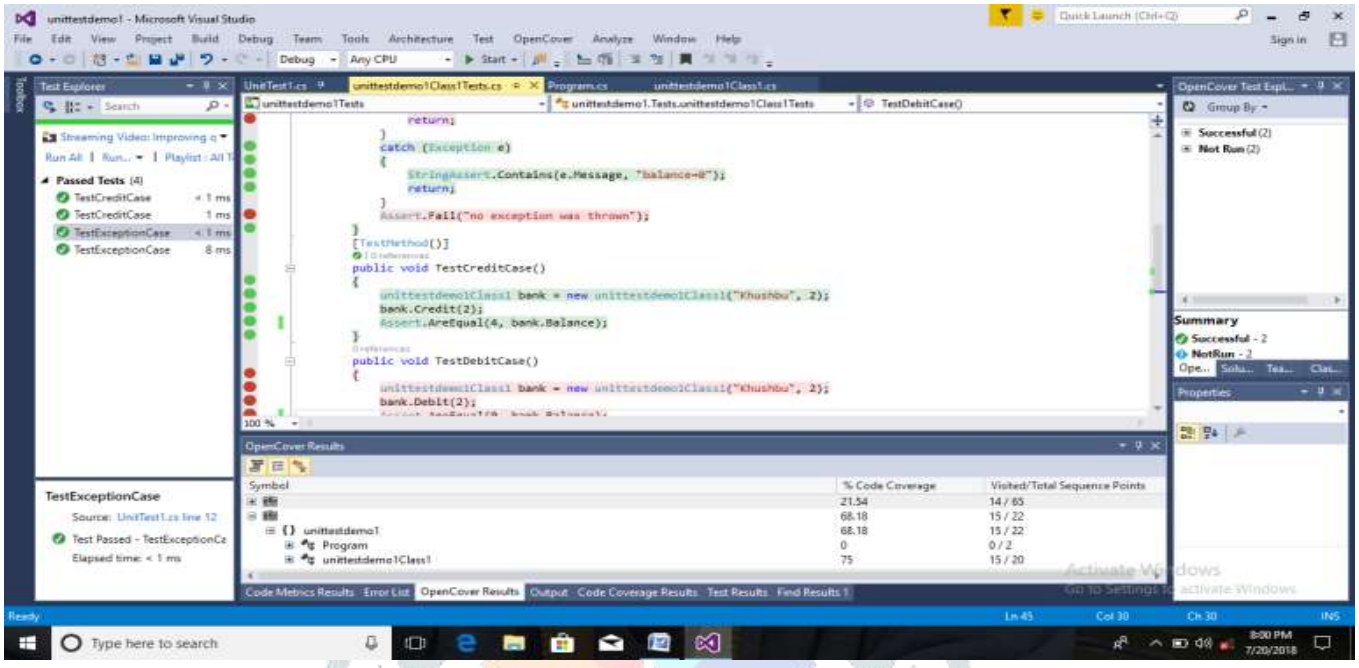

Figure 4: showing UnitTest results using OpenCover tool in visual studio for another program

In this third application code coverage result figure we have shown four passed unit test cases.

Here code coverage colouring is shown with green and red dot. Green dot means covered data and red dot means uncovered data. This applications Code metrics results:-

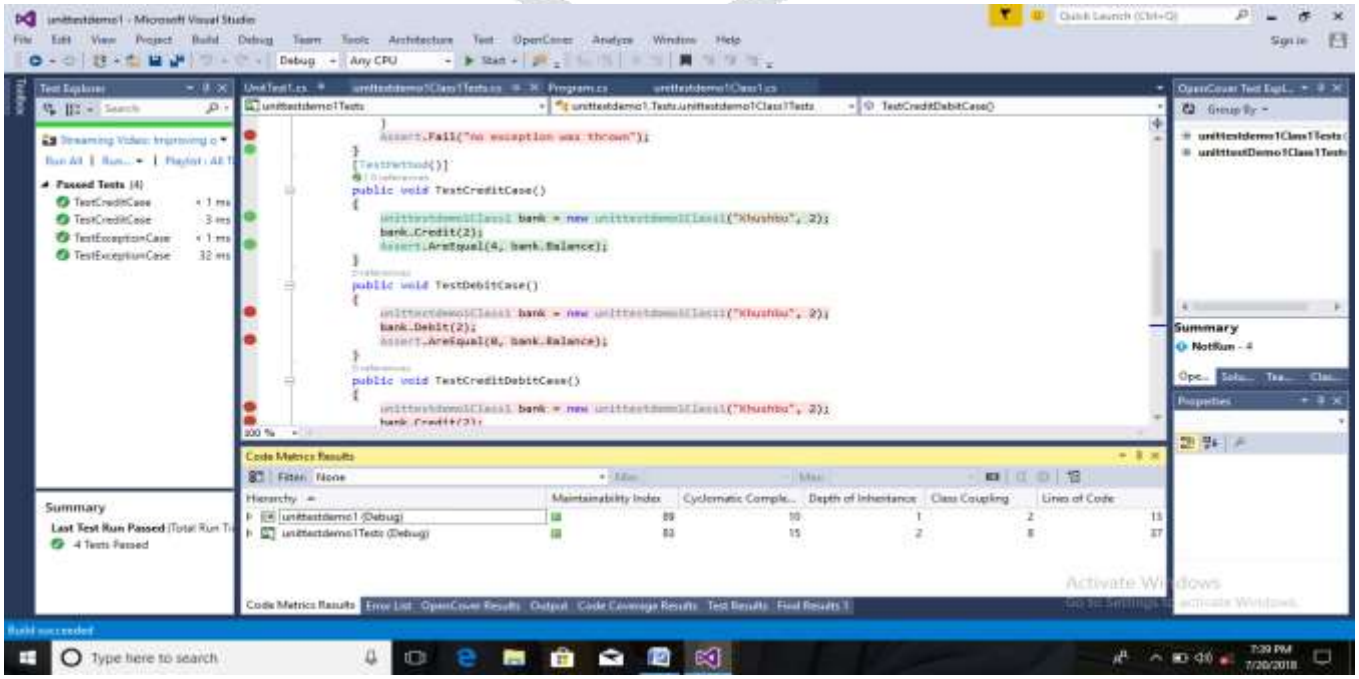

Figure 5: visual studio unit test cases results for a bank account debit credit calculation/balance: Code Metrics Results

In the above figure we have shown code metrics results of bank account application. But we have calculated metrics for all three programs and as well as for a simple console application. Which clearly show that OpenCover tool includes

more coverage and it has some features that simple unit testing don`t have.

So there's OpenCover in a nutshell. On one hand it doesn't provide the same instant gratification that tools like NCrunch are dealing, but it is easy to integrate into your Continuous Integration pipeline. With tools like Coveralls you can even fail the build if code coverage drops a predetermined threshold.

Their results are shown by using table below:-

Table 1: showing code metrics results of one simple console application and three different programs

Metrics	Console applicaton 1	1st Program	2nd Program	3rd Program
Maintainability Index	**100**	**89**	**65**	**83**
Cyclomatic Complexity	**2**	**10**	**3**	**15**
Depth of Inheritance	**1**	**1**	**1**	**2**
Class Couping	**0**	**2**	**11**	**8**
Lines of Code	**1**	**13**	**15**	**37**

In the above code metrics results basic metrics are taken and their results are calculated. Metrics are Maintainability Index, Cyclomatic Complexity, Depth of Inheritance, Class Coupling and Lines of code. Some other metrics are also their but we have chosen visual studio and it provide us feature of calculating these metrics. As comparison shows coverage is high in a simple console application but as we know for a code better metrics calculation is done in bank account application here 37 line of code is covered.

After coverage is done we get a table of results as shown below. All three programs coverage results are shown here and showing total sequence points which is calculated by using OpenCover tool.

Table 2: showing code coverage results of three different Program using OpenCover

Test Cases	% Code Coverage	Total Sequence Points
1st Program		
CodedUITest	8.07%	2/23
Unit Test1	100	2/2
UnitTest2	0	0/2
2nd Program		
CodedUITest	6.06%	2/33
UnitTest1	100	2/2
UnitTest2	0	0/10
3rd Program		
UnitTest1	21.54%	14/65
UnitTest2	20%	13/65
UnitTest3	41.34%	13/65

UnitTest4	46.15%	30/65

V. LIMITATIONS:-

- OpenCover is a nutshell. On one hand it doesn't provide the same instant gratification that tools like NCrunch are dealing, but it is easy to integrate into your Continuous Integration pipeline.
- Difference between opencover and ncover:- comparing .NET test coverage tools NCover and OpenCover. It seems that NCover reports higher code coverage percentage than OpenCover.
- OpenCover, which generates some amazing code coverage metrics. Only trouble is, they're all in XML - Better suited to be read by a build server. That's where ReportGenerator comes in, which "converts XML reports generated by OpenCover, PartCover, Visual Studio or NCover into human readable reports in various formats".
- OpenCover doesn't complain – it just ignores the arguments it doesn't recognize.
- OpenCover still not have .Net Core support.

VI. CONCLUSION:-

This paper, studied about different code coverage analysis tools which used for finding defects in a program or software. These tools can be used in broad ways, and different tools used for different suitable languages selected by user which make it broad. It is found that purpose of selecting particular tool according to particular software is fulfilled. For .NET applications the only open source code coverage tool is OpenCover. It is a nutshell. On one hand it doesn't provide the same instant gratification that tools like NCrunch are dealing, but it is easy to integrate into your Continuous Integration pipeline. With tools like Coveralls you can even fail the build if code coverage drops below a predetermined threshold.

OpenCover don't provide us the feature of complete code coverage as shown by comparison. As code increases coverage increases its quality increases. But for a big data code we require another tool like NCover. So, there are still many aspects such as can`t get 100% error free software is not fulfilled by taking simple steps or using a tool that can work in all environments and provide a software according to user requirement. Still changes are happening there in our developing world. Comparison of tools on the behalf of features is very difficult. This research will add to the software testing body of knowledge and will help the testing community to achieve significant cost and time saving during coverage analysis by reducing the cost of maintenance activity and maintenance effort. Effective coverage will also ensure the quality of the modified software.

REFERENCES:

[1] Sigal Asaf, Eitan Marcus, and Avi Zivs, "Defining coverage views to improve functional coverage analysis", pp. 41-44, 2004.

[2] Joy M. Agustin, "JBlanket: Support for Extreme Coverage in Java Unit Testing",2005.

[3] Mehdi Kessis, Yves. Ledru, G. Vandome, "Experiences in Coverage Testing of a Java Middleware", pp. 39-45, 2005.

[4] Marco Lormans and Arie van Deursen, "Reconstructing Requirements Coverage Views from Design and Test using Traceability Recovery via lsi", pp. 37-42, 2005.

[5] J. Jenny Li, "Prioritize code for testing to improve code coverage of complex software", pp. 10, 2005.

[6] Craig Murphy, "Automated Code Coverage and Unit Tests", 2006.

[7] Michael W. Whalen,Mats P. E. Heimdahl, Ajitha Rajan, and Steven P. Miller, "Coverage metrics for requirements-based testing.", pp. 25-35, 2006.

[8] Smith, Ben, Williams, Laurie Ann, "A Survey on Code Coverage as a Stopping Criterion for Unit Testing.", 2008.

[9] Shanmuga Priya, Ram Raj, Askarunisa A., "Measuring The Effectiveness Of Open Coverage Based Testing Tools", Vol. 5, No.5, pages 499-514, 2009.

[10] Qian Yang, J. Jenny Li, David M. Weiss, "A Survey of Coverage-Based Testing Tools", volume 52 (5): pp. 589-597, 2009.

[11] Rauf, A., S. Anwar, "Automated GUI Test Coverage Analysis Using GA", pp. 1057-1062, 2010.

[12] Faizah Omar, S. I., "A Software Traceability Approach to Support Test Coverage Analysis." 2010.

[13] Shahid Muhammad, Ibrahim Suhaimi, "An Evaluation of Test Coverage Tools in Software Testing", vol.5, pp 216-222, 2011.

[14] Muhammad Shahid, Suhaimi Ibrahim and MohdNaz'riMahrin, "A Study on Test Coverage in Software Testing", 2011.

[15] E Kajo-Mece ,MegiTartari , "An Evaluation of Java Code Coverage Testing Tools", pp. 72-75 , 2012.

[16] Neetu Dabas, Mrs. Kamna Solanki, "Comparison of Code Coverage Analysis Tools: A Review", 2013.

[17] Anju Bansal and Kamna solanki,"A Review on Code Coverage Analysis Tools", In CiiT International Journal of Software Engineering and Technology, 2013.

[18] Jones, J.A. and Harrold Examining the Effectiveness of Testing Coverage Tools: An Empirical Study", 2014.

[19] Chun-Ying Haung,Ching-Hsiang Chiu,Chih-Hung Lin etc "Code Coverage Measurement for Android Dynamic Analysis Tools", 2015.

[20] Parveen Singh kochhar, FerdianThung and David Lo,"Code coverage and test suite effectiveness: Empirical study with real bugs in large systems", 2015.

[21] David Tengeri, LaszioVidacs, Arpad Beszedeset, "Relating Code Coverage, Mutation Score and Test Suite Reducibility to Defect Density", 2016.

[22] Ivan Kreslin, "Automated Code Coverage Analysis", 2016.

[23] Oskar Alfsson,"Analysis of Mutation testing and code coverage during progress of projects", 2017.

[24] Leo Ufimtsev, "Java Code Coverage in Eclipse", 2017.

[25] Erik Dietrich, "A Guide to Code Coverage Tools for C#", 2017.