International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-5, June 2014

Assessing the Relation Between Quantity of Test Cases and Amount of Code Coverage

Anju Bansal

Abstract*—The software testing process can be unpredictable due to deficiencies in the measurement process, resulting in poor quality of software releases. Code coverage analysis is a vital activity in any software testing process. It provides developers with a measure of how well their source code is being exercised by the test runs. Several types of code coverage include the statement, branch and symbol coverage. The focus of this paper is to find out the relationship, if any, between the quantity of test cases and amount of code coverage. Code coverage process is automated through software known as code coverage tool. In this research, NCover code coverage tool is used to measure the above mentioned relationship. NCover is the .Net code coverage tool which shows the untested part of code. For analyzing the relationship, we have used a project "Secure Mailing System" which a web based application that sends and receives messages in a secure manner by encrypting them. Results showed that there is no relationship between number of test cases and amount of code coverage. Code coverage depends on the quality of test cases rather than the quantity. The association found in this research is an important software quality indicator capable for use in describing the software test effectiveness. The results of this research are valuable data for guidance to future research in code coverage analysis.

***Index Terms*—Code Coverage, Coverage Metrics, NCover, Software Testing**

I. INTRODUCTION

We are surrounded by software products. Everything from electric razors and washing machines to cell phones, cars and control systems for nuclear plants, contain software. We are dependent of software today and we expect them to work, that is, that they are sufficiently reliable. Hence, it is important that they keep on working without malfunctioning. One method for increasing the reliability of software is testing. *Software testing* is the evaluation of software by observing its execution.[1] " Software Testing is a process to detect the defects and minimize the risk associated with the residual defects of software". Software testing is now an essential activity in software maintenance life cycle. It is an activity used to determine and improve software quality. [2]Where development is more systematic, organizations seek measures of testing completeness and goodness to establish test completion criteria. Code coverage is one such measure. Code Coverage has been used to prioritize certain parts of a system for testing. [3]. A test case tests the response of a single method to a particular set of inputs. Test case is a combination of inputs, executing function and expected output developed to verify software module compliance with a specified requirement [4].

Manuscript published on 30 June 2014.
***** Correspondence Author (s)
Anju Bansal, Computer Science & Engineering, UIET, M.D.U, Rohtak, India.

Software testers might design multiple test cases called test suite to validate or determine that a software module is functioning correctly [5]. "Coverage is the extent that a structure has been exercised as a percentage of the items being covered. If coverage is not 100%, then more tests may be designed to test those items that were missed and therefore, increase coverage". Test coverage can help in monitoring the quality of testing, and assist in directing the test generators to create tests that cover areas that have not been tested before. [6]

II. CODE COVERAGE

Code coverage (also referred as test coverage) is employed as a method to measure how thoroughly software is tested. Code coverage is a metric of test completeness derived from inspecting the execution of the source code [7].Coverage is used by developers and vendors to indicate their confidence in the readiness of their software. Evaluating software coverage and taking proper actions lead to an improvement of software quality. It helps in evaluating the effectiveness of testing by providing data on different coverage items [8][9]. Code coverage analysis is the process of

- Find areas of a program not exercised by a set of test cases
- Create additional test cases to increase coverage
- Determine a quantitative measure of code coverage
- Identify redundant test cases that do not increase coverage. [10]

A. What Coverage is and is Not

✓ 100% Code Coverage does not say the product is bug free, it ensures product is 100% tested. If product was tested wrongly, Code coverage can't help.
✓ Code Coverage is not about white box testing. Code Coverage is generated when you test the application in any way. To analyze the code review happens at code level. It is not white box testing.
✓ Code Coverage is not through Unit Testing or Automated Testing.
✓ Code Coverage doesn't require extra testing efforts.
✓ Code Coverage is not an end game activity, the earlier the better. It gives scope to improve tests and cover more code, thereby ensuring higher quality.
✓ Code Coverage instrumented builds can't be used for Performance Testing. It will add performance overhead.

B. Code Coverage Items

There are three granularity levels of test coverage items, i.e. fine grain, medium-grain and coarse-grain.

1. Fine grain level consists of Statement, Basic Block, Branch, Condition, Condition / Decision coverage, Modified Condition MCDC, and LCSAJ (Linear Code Sequence and Jump) items.
2. Medium-grain level consists of Method or Function, Class, Package and Design items.
3. Coarse-grain level consists of requirements. [11]

C. *Coverage and Software Quality Relation*

Several researchers [12][13][14]have studied the relation between coverage and finding faults (software quality) by trying to quantify the relation. The mathematical models [12] and the experimental results [13], [15] show that linear increase in code coverage results exponential improvement in the software quality, although 100% coverage can't guaranty absence of defects. Some other researchers were more careful with defining the relation between the coverage and the software quality (effectiveness of test). They support the mentioned claim that linear relation between coverage and software quality does not exist, by their experiments, but that a nonlinear relation exist [16].

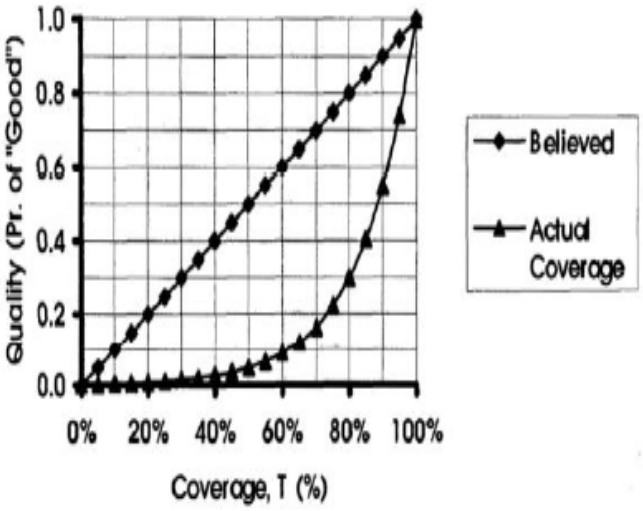

Fig. 1 Ratio between Coverage and Software Quality [17]

III. Code Coverage Analyzer

A code coverage analyzer automates the process of coverage analysis. When it is time to test software quality, developers can use coverage analyzers, to easily see which parts of an application have been tested and which haven't. The code coverage tool quantifies the test coverage produced while the tester executes the designed test suite on a software module. The coverage of test is measured in terms of percentage of code covered in a module that can range in value from 0 to 100. [24]

A. *Features:*

In general, a code coverage tool provides the following:

- ✓ It provides overall code coverage report (a coverage of 60% to 70% of an application can be considered acceptable)
- ✓ It allows developers to analyze selected application modules or entire applications. Developers can analyze the entire application, a subset, or break the covered modules into different components and use the code coverage tool to obtain coverage information about each individual module.
- ✓ It uses color to distinguish between covered code, uncovered basic blocks, uncovered functions, and partially covered code.
- ✓ The code coverage tool can be used to compare the test-profiles of a primary and secondary run. This feature helps developers find portions of code not exercised by tests in the primary run that are exercised in the second run.

There are a number of commercial and free code coverage tools. Some companies also make built in tools according to their requirements. Variety of tools is available to perform code coverage analysis. Following table provide a list of some Code Coverage Analyzers with supported languages.

Table 1: Tools with supported languages

Tools	*Java*	*C/C++*	*Other*
JavaCodeCoverage	X		
JFeature	X		
JCover	X		
Cobertura	X		
Emma	X		
Clover	X		.Net
Quilt	X		
CodeCover	X		COBOL
Jester	X		
GroboCodeCoverage	X		
Hansel	X		
Gretel	X		
BullseyeCoverage		X	
NCover			.Net
Testwell CTC++		X	
eXVantages	X	X	
OCCF	X	X	
JBlanket	X		

IV. Coverage Metrics

To measure how well the program is exercised by a test suite, coverage metrics are used. There exists a number of coverage metrics. Following are descriptions of some types of coverage metrics:

A. *Statement Coverage:*

This metric is defined as the percentage of executable statements in a component that have been exercised by a test case suite. [18]. Achieving higher statement coverage is correlated with the probability of detecting more defects [19][16] and increasing reliability of software [20].

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

B. Branch Coverage:

This metric is defined as "The percentage of branches in a component that have been exercised by a test suite." The simplest example is an if-instruction which has a "then" branch and an "else" branch.

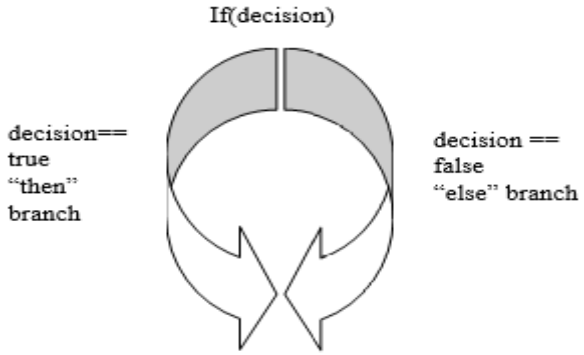

Fig. 2 A decision in an if-statement causes two branches

C. Loop Coverage:

This metric is defined as "The percentage of loops in a component that have been exercised by a test suite." This metric reports whether you executed each loop body zero times, exactly once, and more than once (consecutively). For do-while loops loop coverage reports whether you executed the body exactly once or more than once. The valuable aspect of this metric is determining whether while-loops and for-loops execute more than once, information not reported by other metrics.

D. Decision Coverage:

The definition of decision coverage is:

- ✓ Every point of entry and exit in the program has been invoked at least once, and
- ✓ Every control statement (i.e. branch point) in the program has taken on all possible outcomes (i.e. branches) at least once, and
- ✓ Every non-constant Boolean expression in the program has evaluated to both a true and a false result. [21]

E. Multiple Condition Coverage:

This criterion requires that all combinations of conditions inside each decision are tested. [22]

F. Function Coverage:

This metric is defined as "The percentage of functions in a component that have been exercised by a test suite."

G. Path Coverage:

This metric is defined as "The percentage of paths in a component that have been exercised by a test suite."
A path is a sequence of branches taken during execution of a test case for the test object. Path coverage measures how many of the possible paths are executed during the tests. [23]

H. Entry/Exit Coverage:

This metric is defined as "The percentage of call and return of the function in a component that have been exercised by a test suite."

I. Requirements Coverage:

This metric is defined as "The percentage of requirements in a component that have been covered by a test case suite."

V. PROJECT DESCRIPTION

The project used in this analysis is Secure Mailing System. This is a web based application developed in ASP.net and SQL server for sending and receiving messages and documents in a secure manner by encrypting them using RSA algorithm.

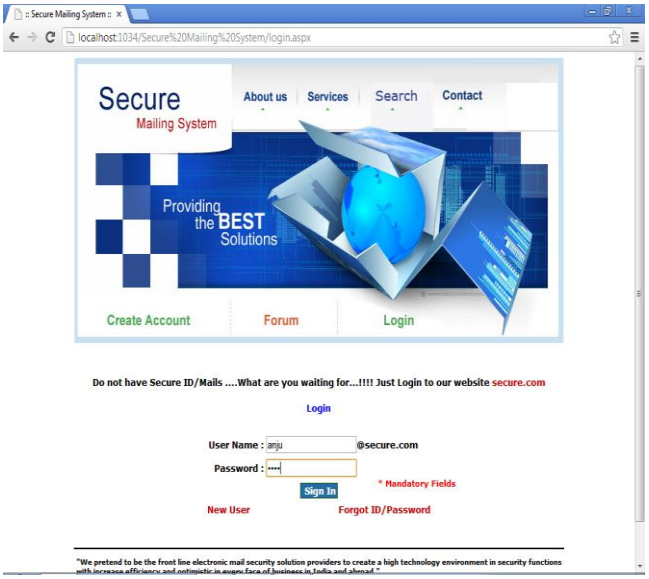

Fig. 3 Project Snapshot

VI. HYPOTHESIS TESTING

In this thesis we assumed following hypothesis:
There is a strong direct correlation between the number of test cases and amount of code covered. To test the above mentioned hypothesis, we have NCover code coverage analyzer. As we run the test cases manually, these are recorded in Web Test Recorder window positioned in the left side as shown in fig. 4 below:

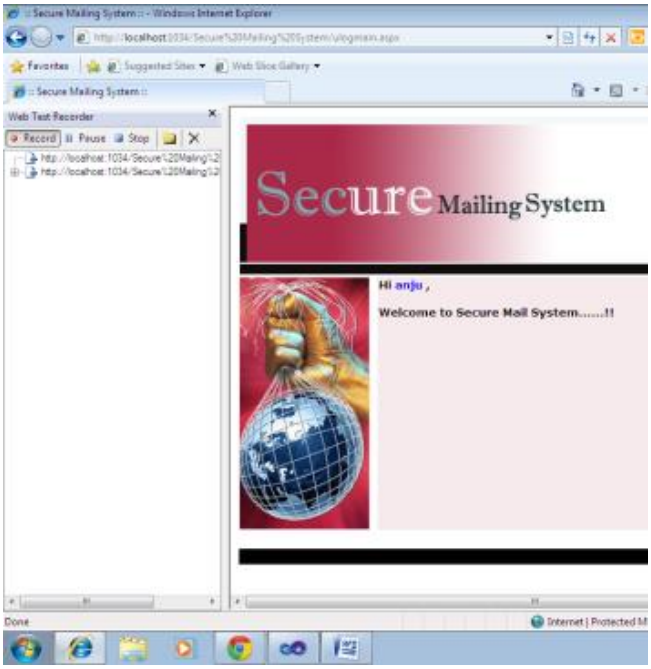

Fig. 4 Running Test Cases

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

When recording is stopped, NCover collects coverage and shows in form of graphs.

Fig. 5 Collecting Coverage

VII. RESULTS

Following our measurements with NCover, we have noticed that when we applied seven test cases then branch coverage is 33.3% and sequence point coverage is 30.7% (as shown in fig. 6). After applying 15 test cases there is increase in coverage i.e. branch coverage is 39.8% and sequence point coverage is 42.7% (fig.7). However, if we increase no. of test cases i.e. increase test cases to 18 then coverage remains same as before (fig.8). This shows that increase in number of test cases do not increase coverage. It is the quality of test cases that matters not the quantity of test cases.

Applying 7 test cases:

Fig. 6 Result after 7 test cases

After Applying 15 test cases:

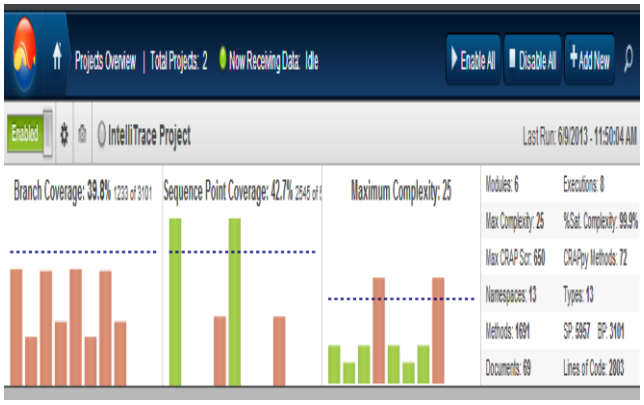

Fig. 7 Result after 15 test cases

After Applying 18 test cases

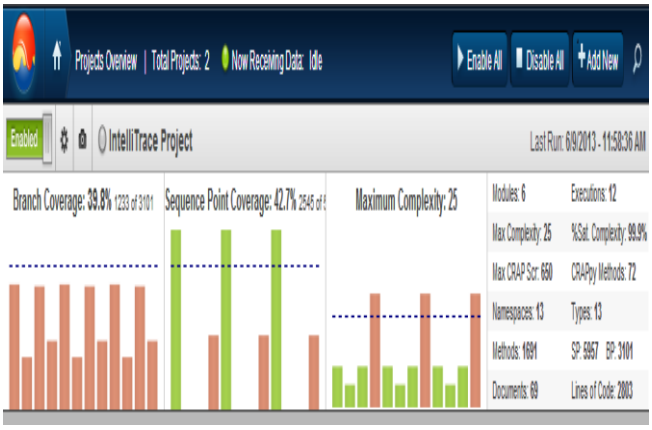

Fig. 8 Result after 18 test cases

After applying 25 test cases:

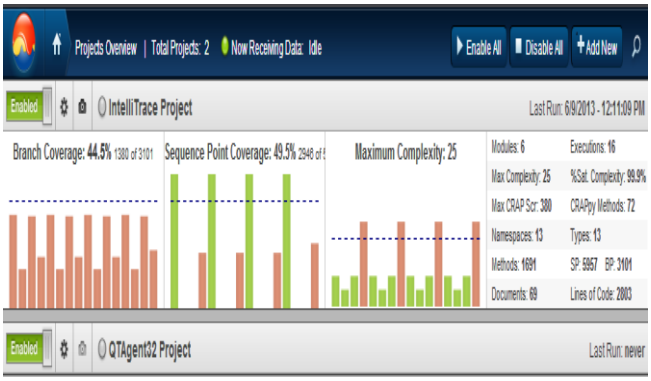

Fig. 9 Result after 25 test cases

After applying 30 test cases:

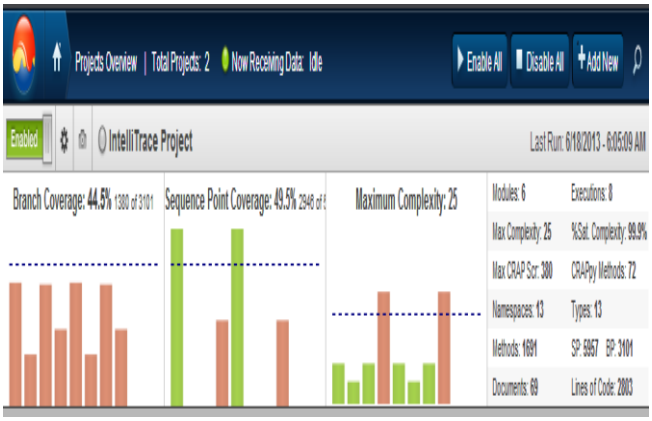

Fig.10 Result after 30 test cases

We have proven that our hypothesis doesn't hold. The experiment results show that code coverage does not depend on the size of the test suite instead depends on the effectiveness of test suite.

VIII. CONCLUSION

Code coverage tools are great when used along with other test activity. The dead code can be found earlier in the product development cycle and removed promptly in the same release. The purpose of this quantitative research was to assess the relationship between the quantity of test cases and amount of code coverage.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The results of this research contributed to a foundation for a better understanding of coverage testing theory as well as of software process improvement. The research has shown evidence that there is no relationship between number of test cases and amount of code coverage. Code coverage does not depend on quantity of test cases; it actually depends on quality of test cases. Using this as an indicator for test quality will establish a roadmap for improving future software testing process.

REFERENCES

[1] Ammann, P. & Offutt, J., Introduction to Software Testing, Cambridge University Press. ISBN: 978-0-521-88038-1, 2008
[2] Antonia Bertolino, Software Testing Research: Achievement, Challenges, Dreams, IEEE Computer Society, 2007.
[3] M. Gittens, K. Romanufa, D. Godwin, and J. Racicot, "All code coverage is not created equal: a case study in prioritized code coverage," Conference of the Center for Advanced Studies on Collaborative research, pp. 1-15, Toronto, Ontario, Canada, 2006.
[4] Gupta A. & Bhatia R., Testing Functional Requirements using B-model specifications. ACM SIGSOFT Software Engineering Notes, pp 1-7, 2010
[5] Taipale O., Observation on Software testing practice. Lappeenranta University of Technology, Lappeenranta, Finland, 2010
[6] ISTQB, "International Software Testing Qualification Board" version 2.0, 2007 www.istqb.org
[7] Baird, K.C. Ruby by example: Concepts and code.San Fransisco, 2007
[8] E Kajo-Mece , Megi Tartari , "An Evaluation of Java Code Coverage Testing Tools",pp 72-75 , 2012.
[9] Shahid Muhammad, Ibrahim Suhaimi, "An Evaluation of Test Coverage Tools in Software Testing", International Conference on Telecommunication Technology and Applications, Proc .of CSIT vol.5, Singapore, pp 216-222, 2011.
[10] Cornett S., Minimum Acceptable Code Coverage. Retrieved from http://www.bullseye.com/minimum.html
[11] Shahid Muhammad, Ibrahim Suhaimi, Harihudin Selamat, Test Coverage Measurement and Analysis on the Basis of Software Traceability Approaches, International Journal of Information and Electronics Engineering, Vol. 1, No. 2, September 2011
[12] Ohba M., Caruso J., Piwowarski P., Coverage measurement experience during function test. Los Alamitos, CA, USA, 1993
[13] Mercer T.W., Mucha M.R., Williams J.P., Code coverage, what does it mean in terms of quality? Philadelphia, 2001
[14] Slonim J., Bauer M., Ye J., Software Reliability Assurance in early Development Phase: A Case Study in an Industrial Setting. Aspen, CO, USA, 1996
[15] Monica Hutchins, Herb Foster, Tarak Goradia, and Thomas Ostrand, Railway applications Communication, signaling and processing systems Software for railway control and protection systems. Los Alamitos, CA, USA, June 2011
[16] Namin A.S. , Andrews J.H., The influence of size and coverage on test suite effectiveness. New York, 2009
[17] Sarabi M., Evaluation of Structural Testing Effectiveness in Industrial Model-driven Software Development, Malardalen University ,Sweden, 2012
[18] Park S., Hussain I.,Taneja K., Hossain B.M., Grechanik M., Chen Fu, Qing Xie, CarFast: Achieving Higher Statement Coverage Faster, November 2012
[19] Y. W. Kim. Efficient use of code coverage in large-scale software development. In CASCON '03, pages 145–155, 2003
[20] Malaiya, Y.K., Li, M.N., Bieman, J.M. and Karcich, R. (Software reliability growth with test coverage. IEEE Trans. Reliab., 51, 420–426,2002.
[21] John Joseph Chilenski and Steven P. Miller, "Applicability of Modified Condition/Decision Coverage to Software Testing", Software Engineering Journal, September 1994, Vol. 9, No. 5, pp.193-2000.
[22] Panday A., Singh M.K., Gupta M., Ali N., Test Case Redundancy Detection and Removal Using Code Coverage Analysis, MIT International Journal of Computer Science & Information Technology Vol. 3, No. 1, pp. 6–10, Jan. 2013.
[23] Nirpal P.B. and Kale K.V., Comparison of Software Test Data for Automatic Path Coverage Using Genetic Algorithm, International Journal of Computer Science & Engineering Technology, Vol. 2, No. 2, 2011
[24] Waldschmidt, P. NCover Complete: The Code Coverage Analyzer Retrieved from: http://www.ncover.com/

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)