An Evaluation of Test Coverage Tools in Software Testing

Dr Muhammad Shahid
Universiti Teknologi Malaysia
32 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Suhaimi Ibrahim
Universiti Teknologi Malaysia
140 PUBLICATIONS **1,128** CITATIONS
SEE PROFILE

2011 International Conference on Telecommunication Technology and Applications
Proc .of CSIT vol.5 (2011) © (2011) IACSIT Press, Singapore

An Evaluation of Test Coverage Tools in Software Testing

Muhammad Shahid[1], Suhaimi Ibrahim

Advanced Informatics School (AIS), Universiti Teknologi Malaysia
International Campus, Jalan Semarak, Kuala Lumpur, Malaysia
smuhammad4@live.utm.my, suhaimiibrahim@utm.my

Abstract: Test Coverage is an important indicator of software quality and an essential part of software maintenance. It helps in evaluating the effectiveness of testing by providing data on different coverage items. Automated testing tools can be used to enhance the maintainability, testability and stability of the software. This paper aims to provide an evaluation of current test coverage tools in software testing. Management and test managers require an appropriate tool for the software under test. We hope, evaluation identified here will help to select the efficient and effective tool.

Keywords: Code Coverage, Coverage Measurement, Coverage based Testing Tools, Software Testing, Test Coverage

1. INTRODUCTION

Software testing is considered now as an essential activity in software maintenance life cycle. It is a practice often used to determine and improve software quality. Where development is more systematic, organizations seek measures of testing completeness and goodness to establish test completion criteria. Code coverage is one such measure [1]. “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”[2]. 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 [3].

In recent years, Telecom industry has grown beyond the expectations of everyone with very sophisticated and diversified service offerings. Service providers and operators have to work hard with the highest efficiency and effectiveness in this age of competition. Software testing tools can help these to achieve this goal. As there are different tools suitable for different systems and industry, we are focusing on tools that measure test coverage either code based or requirement based or both.

The rest of this paper is organized as follows. Section 2 describes the overview of test coverage and test coverage process. Section 3 reports the tools for test coverage measurement and analysis. Section 4 provides evaluation of the tools based on a set of criteria. Threats to validity are discussed in section 5. Finally section 6 presents an analysis table and concludes this paper.

2. TEST COVERAGE

Coverage is a quality assurance metric which determines how thoroughly a test suite exercises a given program [4]. Coverage-based testing can be applied to any stage of testing including unit, integration or system testing. The output of coverage measurement can be used in several ways to improve the verification process. The coverage tool can detect illegal events that occur and help find bugs in the design. It can help to

[1] Tel.: + 60 16 2242627; fax: +60 3 26930933

E-mail address: ssheikh_85@hotmail.com find holes in the testing, i.e. areas that are not covered [2]. Test coverage also helps in Regression testing, test case prioritization, test suite augmentation and test suite minimization.

2.1 Test Coverage Process

The test/code coverage analysis process is generally divided into three tasks: code instrumentation, data gathering, and coverage analysis. Code instrumentation consists of inserting some additional code to measure coverage results. Instrumentation can be done at the source level in a separate pre-processing phase or at runtime by instrumenting byte code. Data gathering consists of storing coverage data collected during test runtime. Coverage data analysis consists of analysing the collected results and providing test strategy recommendations in order to reduce, to feed or to modify the relevant test suite.

3. TEST COVERAGE TOOLS

This section presents state-of-art coverage based tools that perform code coverage analysis. There are many test coverage tools that are available in literature and internet, commercially or lab version. Following are some test coverage tools.

3.1 JavaCodeCoverage

JavaCodeCoverage [5] is an open source bytecode analyzer tool for test coverage analysis for Java software which requires neither the language grammar nor the source code. An important aspect of JavaCodeCoverage is that it stores the coverage information for individual test case thereby facilitating detailed coverage analysis. Another important aspect of JavaCodeCoverage is that it records all vital code-elements and test coverage information in open source database software MySQL.

3.2 JFeature

JFeature is an open source feature/requirement coverage tool that facilitates focusing on requirements as code is developed. It lets leverage from standard development practices to get more insight into the requirements covered by the code [6]. It is a plug-in for the Eclipse IDE and also allows user to import requirements and match them to JUnit test cases within Java application.

3.3 JCover

JCover [7] is a code coverage analyzer for Java programs. It provides a mechanism to generate statistical information on the coverage of an application during a test run. It can be used to calculate the percentage of code that was executed, percentage not executed, what sources were not used in files and so on. JCover supports statement and branch coverage.

3.4 Cobertura

Cobertura is a free open source Java tool that calculates the percentage of code accessed by tests. It can be used to identify which parts of Java program are lacking test coverage [8]. It can be executed from ant or from the command line.

3.5 Emma

It is an open-source tool for measuring and reporting code coverage for Java [9]. It can instrument classes for coverage either offline (before they are loaded) or on the fly (using an instrumenting application class loader). Supported coverage types are class, method, line and basic block.

3.6 Clover

Clover is available as either an Eclipse or IDEA plug-ins or using ANT script [10]. It supports statement, method, class, and package coverage. This tool provides accurate, configurable coverage analysis. Coverage reporting is in XML, HTML, or via a Swing GUI. It is a low cost coverage tool.

3.7 Quilt

Quilt [11] *is* a Java software development tool that measures coverage, the extent to which unit testing exercises the software under test. It is optimized for use with the JUnit unit test package, the Ant Java build facility, and the Maven project management toolkit. Quilt intercepts code as it is being loaded and alters it. It doesn't work at the source code level. It manipulates compiled classes and their bytecode, the machine code of the Java Virtual Machine, the JVM.

3.8 Code Cover

Code cover [12] is a free testing tool for Java programmers. It is fully integrated into Eclipse and performs source instrumentation for coverage measurement especially for condition coverage. It helps to increase test quality.

3.9 InsECT

InsECT (Instrumentation Execution Coverage Tool), is a system developed in Java to obtain coverage information for Java programs [13]. It instruments Java class files at the bytecode level. The aim of InsECT is to provide detailed coverage information about Java programs.

3.10 Jester

Jester [14] is for Java code and JUnits tests. It finds code in the software that is not covered by tests. Jester's approach is called mutation testing or automated error seeding.

3.11 GroboCodeCoverage

GroboCodeCoverage [15] is a 100% pure Java implementation of a code coverage tool. It uses Jakarta's BCEL platform to post-compile class files to add logging statements for tracking coverage. It helps to discover what parts of your code have not executed during unit tests.

3.12 Hansel

It is an extension to JUnit. Hansel [16] gives very useful information that how much of the code which a test is supposed to test is covered? It deals with branch coverage of the class.

3.13 JBlanket

JBlanket [17] is a tool for assessing and improving method coverage of unit test case. It is meant for both stand alone and client server programs.

3.14 Coverlipse

Coverlipse [18] is an Eclipse plug-in for code coverage visualization. The coverage results are shown after a JUnit test run. It supports branch, block and all-uses coverage.

3.15 Gretel

Gretel [19] is an open source test coverage monitoring tool for Java programs. It implements residual test coverage measurement. Gretel can re-instrument the program and remove instrumentation for those parts that have already been executed. Currently it provides statement coverage monitoring.

3.16 BullseyeCoverage

It is a C and C++ code coverage analyzer tool that tells how much of source code was tested [20]. It pinpoints areas that need attention to be reviewed. Supported coverage types are function and condition/decision. BullseyeCoverage supports the widest range of platforms of any code coverage analyzer including Windows and Linux.

3.17 NCover

NCover [21] is an open source code coverage tool for .NET platform. It provides a very powerful and flexible tool set which can integrate into build process and help to deliver higher quality code. It tells about how many times each line of code was executed during a particular run of the application. It supports method and class coverage.

3.18 Testwell CTC++

Testwell CTC++ [22] is a powerful instrumentation-based test coverage and dynamic analysis tool for C and C++ code. It shows the coverage all the way to the modified condition/decision coverage (MC/DC) level as required by DO-178B projects. The tool is light but still contains all the essential "must" features of an industry strength testing tool.

3.19 TestCocoon

TestCocoon (formerly CoverageMeter) [23] is a complete code coverage tool for C/C++ and C# programs available under Apple® Mac OS X, Linux™ or Microsoft® Windows. It analyzes the performance of a software validation and permits to measure the performance and optimizes the testing process of a C/C++ or C# applications. It reduces the amount of tests by finding redundant tests and finds dead code.

3.20 eXVantage

It is a tool suite for code coverage testing, debugging and performance profiling [24]. It supports Java and C/C++ platforms. eXVantage uses source code instrumentation for C/C++ and bytecode instrumentation for Java. It analyzes the program in such a way that it can select the least number of probes to be inserted into the target program, that's why it has the highest off-line instrumentation overhead.

3.21 OCCF

OCCF (Open Code Coverage Framework) [25] supports multiple programming languages. Sample tools can be produced for C, Java and other languages using OCCF. The researchers developed a tool that can measure four coverage criteria. They reduced costs by reusing common code, and obtained consistent measurements by supporting multiple languages, flexible measurements through expanding features, and complete measurements by inserting measurement code into the source code.

3.22 JAZZ

JAZZ [26] is a structural testing tool. It does branch, node, and def-use coverage and implements a GUI, test planners, dynamic instrumentation, and a test analyzer. Jazz is incorporated in Eclipse and Jikes for the Intel x86. Instrumentation is dynamically inserted on demand as the program executes. Instrumentation is also deleted at the time it is no longer needed.

4. EVALUATION OF THE TOOLS

This section describes a comparative evaluation of various test coverage tools. All the tools included in this article support coverage measurement. We have excluded some tools like, InsECT, Coverlipse, JBlanket because they look like dead and there is no activity in these for many years. The evaluation criteria consist of language support, instrumentation, Coverage Measurement, GUI and Reporting.

4.1 Language Support

Most of the test coverage tools support Java and C/C++ programming languages. There are some tools that work for both and some for JavaScript, FORTRAN, C# and, .NET platforms. Table 1 shows a list of the tools with their language support.

4.2 Instrumentation

Test coverage tools use code instructions or probes that are inserted in the code to monitor the specific part of the program. Probes are inserted before or during the execution. These lines of code generate a record during program execution. This process is called instrumentation. Instrumentation can be divided into two types; source code instrumentation and bytecode instrumentation. Source code instrumentation technique can be used only when source code is available. Table 2 shows tools with their instrumentation techniques.

4.3 Coverage Measurement

Coverage measurement is an important criteria for the tools. The effectiveness of coverage measurement and analysis depends upon which coverage items have been used by a tool. The common coverage items used include; Statement, Basic Block, Branch, Path, Conditions, Loop, Condition/ Decision, Modified, Condition/Decision (MCDC), Method or Function, Class, Package, Design and Requirement. Some other coverage types are Data Flow and Control Flow coverage, c-use and p-use coverage. Not all coverage types have been used by vendors. Table 4 shows tools with coverage measurements. We can notice that JFeature is the only tool that covers requirements.

Table 1 Tools With Supported Language

Tools	Java	C/C++	Other
JavaCodeCoverage	×		
JFeature	×		
JCover	×		
Cobertura	×		
Emma	×		
Clover	×		.Net
Quilt	×		
Code Cover	×		COBOL
Jester	×		
GroboCodeCoverage	×		
Hansel	×		
Gretel	×		
BullseyeCoverage		×	
NCover			.Net
Testwell CTC++		×	
TestCocoon		×	C#
eXVantage	×	×	
OCCF	×	×	×
JAZZ	×		

Table 2 Tools With Instrumentation

Tools	Source Code Instrumentation	Byte Code Instrumentation	Other
JavaCodeCoverage		×	
JFeature	×		
JCover	×		
Cobertura		×	
Emma		×	
Clover	×		
Quilt		×	
Code Cover	×		
Jester			×
GroboCodeCoverage		×	
Hansel			×
Gretel	×		
BullseyeCoverage	×		
NCover	×		
Testwell CTC++	×		
TestCocoon	×		
eXVantage	×	×	
OCCF	×		
JAZZ			×

Table 3 Coverage Measurement Level

Tools	Stateme nt/ Block	Branch / Decision	Method/ Function	Class	Require ment
JavaCodeCoverage	×	×	×		
JFeature			×		×
JCover	×	×	×	×	
Cobertura	×	×			
Emma	×		×	×	
Clover	×	×	×	×	
Quilt	×	×			
Code Cover	×	×			
Jester					
GroboCodeCoverage				×	
Hansel		×			
Gretel	×				
BullseyeCoverage		×	×		
NCover			×	×	
Testwell CTC++		×			
TestCocoon					
eXVantage	×	×	×		
OCCF	×	×			
JAZZ		×			

4.4 GUI and Reporting

Graphical User Interface (GUI) and reporting is also an important feature for comparing the tools. Some tools have both GUI and batch mode version to fulfill the needs of different users. Tools like, JavaCodeCoverage, JFeature, Bullseye, Clover, Cobertura, Emma, eXVantage, JCover, Gretel, Code Cover have GUI support.

Most of the tools especially commercially developed, produce automatic summary reports, some of which are graph based and some file based. Clover is one of the best tools for report generation in our view. Its HTML reports show green and red bars to demonstrate coverage percentage in addition to the numbers. There are XML and PDF format reports. Clover also generates executive and historical reports to show detailed overview of coverage. Emma has a fast report generation facility in HTML, XML and text reports. NCover generates many types of beautiful reports. Cobertura and JCover have XML reports. Bullseye has CSV and HTML report facility. Code Cover generates HTML reports.

We have summarized all the tools with coverage criteria in Table 4.

Table 4 Analysis

Tool	**Language Support**		**Instrumentation**		**Coverage Measurement**				GUI Support	Reports
	Java	C/C++	Source Code Instrumentation	Byte Code Instrumentation	Statement/ Block	Branch / Decision	Method/ Function	Class		
JavaCodeCoverage	×			×	×	×	×		×	×
JFeature	×		×				×	Req.	×	×
JCover	×		×		×	×	×	×	×	×
Cobertura	×			×	×	×			×	×
Emma	×			×	×		×	×	×	×
Clover	×		×		×	×	×	×	×	×
Quilt	×			×	×	×			No	No
Code Cover	×		×		×	×			×	×
Jester	×			Other				×	×	
GroboCodeCoverage	×			×				×	No	Simple
Hansel	×			Other		×			No	No
Gretel	×		×		×		×		No	Line Table
BullseyeCoverage		×	×			×	×		×	×
NCover		.Net	×					×	×	×
Testwell CTC++		×	×			×				×
TestCocoon		×	×				×		×	×
eXVantage	×	×	×	×	×	×			×	×
OCCF	×	×	×		×	×			NA	NA
JAZZ	×			Other		×			NA	NA

5. THREATS TO VALIDITY

We have collected all the information about test coverage tools from websites and literature. There is a possibility that some papers and websites may be missed, causing exclusion of some tools or properties of included tools.

6. CONCLUSION

This paper evaluates 19 test coverage tools. We have compared five features: language support, instrumentation, coverage measurement, GUI and reporting. In our opinion, these are the best criteria to test the coverage tools. Table 4 summarizes our analysis. Each tool has some strong and weak points. Users and developers can select the tool according to their need. We hope our work will help in more usage and selection of tools.

7. REFERENCES

[1] Zhu, H., Hall, P.A.V. and May, J.H.R. (1997) Software unit test coverage and adequacy. ACM Comput. Surv., 29, 366–427

[2] ISTQB, “International Software Testing Qualification Board” version 2.0, 2007 www.istqb.org

[3] Grinwald, R., Harel, E., Orgad, M., Ur S, Ziv, A., “User Defined Coverage – A tool Supported Methodology for Design Verification”, IBM Research Lab, Haifa Dac, San Francisco, CA USA, 158-163, 1998.

[4] M. Kessis, Y. Ledru, G. Vandome. “Experiences in Coverage Testing of a Java Middleware”, in Proceedings SEM 2005, Lisbon, Portugal. ACM, pp. 3945, 2005.

[5] R. Lingampally, A. Gupta, P. Jalote. "A Multipurpose Code Coverage Tool for Java," In Proceedings of the 40th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, 261b, 2007.

[6] JFeature, http://www.technobuff.net/webapp/product/showProduct.do?name=jfeature

[7] JCover, http://www.mmsindia.com/JCover.html

[8] Cobertura, http://cobertura.sourceforge.net/

[9] Emma, http://emma.sourceforge.net/

[10] Clover, http://www.atlassian.com/software/clover/

[11] Quilt, http://www.codecoverage.com

[12] Code Cover, www.codecover.org

[13] InsECT, http://www.codecoveragetools.com/index.php/coverage-process/code-coverage-tools-java.html

[14] Jester, http://jester.sourceforge.net/

[15] Grobo,CodeCoverage, http://groboutils.sourceforge.net/codecoverage/index.html

[16] Hansel, http://hansel.sourceforge.net/

[17] JBlanket, http://csdl.ics.hawaii.edu/Tools/JBlanket/

[18] Coverlipse, http://coverlipse.sf.net

[19] Gretel, http://www.cs.uoregon.edu/research/perpetual/dasada/software/Gretel/docs/index.html

[20] BullseyeCoverage, www.bullseye.com

[21] NCover, www.ncover.com

[22] Testwell CTC++, www.testwell.fi

[23] TestCocoon, http://www.testcocoon.org/

[24] eXVantage, http://www.research.avayalabs.com/user/jjli/exvantage.htm

[25] OCCF, http://sourceforge.jp/projects/codecoverage/

[26] Misurda, J., Clause, J. A., Reed, J. L., Childers, B. R., Soffa, M. L. (2005), In the Proceedings of 27th International Conference on Software Engineering, ICSE 2005