The Study of Various Code Coverage Tools

Sneha Shelke, Sangeeta Nagpure

Deparment of Computer Engineering

K J Somaiya College of engineering

Mumbai University, Mumbai, India.

***Abstract*— Code coverage is used to describe the degree to which the source code of a program is tested. There are many code coverage testing tools available, working on different criteria providing different features. Here we have studied five code coverage tools and out of which one tool was actually evaluated for their proposed features. A comparative study is presented on the basis of the set criteria.**

***Keywords*— software testing, code coverage, code coverage tools, code coverage criteria.**

I. Introduction

Software testing is used to indicate the software quality [12]. Software testing is a process of assuring a program is bug free and also that it performs the intended functions which are error free[12]. It is used to determine and improves the quality of the software. Testing activities include obtaining the test coverage. Code coverage [10] is a way of ensuring that your tests are actually testing your code. When you run your tests you are presumably checking that you are getting the expected results.

The output of coverage measurement can be used in several ways to improve the testing process. It also gives the information to the user about the status of the verification process. It can help to find areas that are not covered.

Test coverage is used to measure how the software is tested and developers use it to indicate their confidence in the readiness of their software. “A Survey of Coverage-Based Testing Tools” studies and compares 17 coverage-based testing tools primarily focusing on, but not restricted to, coverage measurement [1]. All tools included in this survey have coverage measurement capability. This survey compares tools released before 2007 for three important coverage tool characteristics.

There are several tools in order to facilitate the software testing process, and they have different functionalities. Our objective in this paper is to study the tools with code coverage capabilities which are released after 2007.

We selected test tools with code coverage capabilities. We have selected 5 tools out of which we have evaluated 1 tool and all other tools are studied and compared based on the literatures available.

This paper organized as follows. The section II describes the overview of the coverage. Section III describes the 5 code coverage tools. In section IV we have compared these tools based on three measurement criteria: supported programming languages, coverage measurement criteria, programming instrumentation and automation. Finally section V summarizes the work.

II. Code Coverage

Code Coverage [11] is the process of finding areas of a program which are not exercised by set of test cases, which creates additional test cases to increase coverage and determines quantitative measure of code coverage. Coverage based testing tool can be applied to any stage of testing including unit, integration or system testing.

Code coverage provides quantification of coverage related test progress, it prioritize the testing by selecting those tests that has largest incremental gain in coverage. It detects redundant cases and removes those cases since these much time to execute repeatedly.

By using the code coverage testing process can be improved and cost of correcting the errors can be reduced. Some of the benefits of Code Coverage measurement

- To know whether we have enough testing in place
- To maintain the test quality over the life cycle of a project
- To know how well our tests, actually test our code
- It creates additional test cases to increase coverage
- It helps in finding areas of a program not exercised by a set of test cases
- It helps in determining a quantitative measure of code coverage, which indirectly measures the quality of the application or product.

Drawback of Code Coverage measurement:

- One drawback of code coverage measurement is that it measures coverage of what has been written, i.e. the code itself; it cannot say anything about the software that has not been written.
- If a specified function has not been implemented or a function was omitted from the specification, then structure-based techniques cannot say anything about them it only looks at a structure which is already there.

III. Code Coverage tools

A. EvoSuite

To find defects in software, one needs test cases that execute the software systematically, and oracles that assess the correctness of the observed behavior when running these test cases. EvoSuite [3] [4] is a tool that automatically generates test cases with assertions for classes written in Java code. To achieve this, EvoSuite applies a novel hybrid approach that generates and optimizes whole test suites towards satisfying a coverage criterion. For the produced test suites, EvoSuite suggests possible oracles by adding small and effective sets of assertions that concisely summarize the current behavior; these assertions allow the developer to detect deviations from expected behavior, and to capture the current behavior in order to protect against future defects breaking this behavior.

B. JavaCodeCoverage (JaCoCo)

JavaCodeCoverage [5] is a byte-code analyser tool for test coverage analysis for Java software which neither 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.

C. Automatic Robustness Coverage Analysis Tool (AURORA)

AURORA [6] [7] is a tool that provides testers with the capability of computing the extended coverage achieved by a certain test suite ts over a program p in an automated way. The tool accepts code transformations defined by means of the TXL language, and uses standard coverage measurement libraries to compute the coverage achieved by ts on p, and using the transformations it automatically computes the fragility indexes.

D. Dynamic Code Coverage (DCC)

Dynamic Code Coverage [9] is an easy-to-use tool that indicates which source code is exercised during one or more executions of a program. This information is invaluable in determining how thoroughly a test suite exercises a program.

E. Open Code Coverage Framework (OCCF)

There are many programming languages and coverage criteria exist, and every coverage measurement tools support programming language and the coverage criteria. So many tools exist and they have various programming languages that lead to difference between the existing tools. To overcome the diversity of existing tool, a novel approach for measuring the coverage for multiple programming languages called open code coverage framework.[8] [10] [13]

IV. Coverage Measurement Criteria

All tools support coverage measurement capability [11]. It consists of supported languages, program instrumentation, coverage measurement, automation. We have studied 5 code coverage tools out of these we have evaluated java code coverage tool and all other tools studied based on the available literature.

A. Supported languages

Every tool supports programming languages. Some of them support only java, some of them support only C/C++, some of them support both java and C/C++, some of them supports FORTRAN, C#,.NET. Table I shows a list of tools and the languages they support.

The selection of supported languages reflects each company's target industries. EvoSuite [3] [4] is a tool that automatically generates test cases with assertions for classes written in Java code. Java Code Coverage [5] is a byte-code analyser tool for test coverage analysis for Java software which requires neither the language grammar nor the source code.
To overcome the diversity of existing tools, a novel framework developed for consistently and flexibly measuring the coverage supporting multiple programming languages, called Open Code Coverage Framework (OCCF) [10]

TABLE I
SUPPORTED LANGUAGE

Tool Name	C/C++	Java	Other
EvoSuite	--	✓	--
JaCoCo	--	✓	--
AURORA	--	✓	--
DCC	✓	--	--
OCCF	✓	✓	✓

B. Program Instrumentation

Coverage-testing tools capture coverage information by monitoring program execution. Execution is monitored by inserting probes into the program before or during its execution. A probe is typically a few lines of code that, when executed, generate a record or event that indicates that program execution has passed through the point where the probe is located. There are two kinds of overhead associated with instrumenting a program with probes:

- The off-line overhead :
  It cannot be used source code is not available. They are most efficient in terms of compilation time but less portable.

- The run time overhead :
  The tools which are provided for system software or embedded software, they tend to focus on reducing the run time overhead, so their tools can be usable in real time environment.

The EVOSUITE [4] tool implements the approach presented for generating JUnit test suites for Java code. EVOSUITE works on the byte-code level and collects all necessary information for the test cluster from the byte-code via Java Reflection. This means that it does not require the source code of the SUT and in principle is also applicable to other languages that compile to Java byte-code.

OCCF [10] inserts instrumentation code into source code. The abstract syntax trees of source code for most programming languages have similar structure. Thus OCCF provides a reusable common code to insert instrumentation code through AST's by utilizing the similarities.

TABLE II
PROGRAM INSTRUMENTATION

Tool Name	Source-Code Instrumentation	Byte-Code Instrumentation
EvoSuite	--	✓
JaCoCo	--	✓
AURORA	--	✓
DCC	✓	--
OCCF	✓	--

C. *Coverage Measurement Criteria*

There are varieties of coverage measurement criteria [11] such as statement or line coverage, decision coverage, block coverage, function/method coverage. The statement coverage [11] is also known as line coverage or segment coverage. The statement coverage covers only the true conditions**.** Through statement coverage we can identify the statements executed and where the code is not executed because of blockage. Decision coverage [11] is also known as branch coverage or all-edges coverage. It covers both the true and false conditions unlikely the statement coverage. Condition coverage reports the true or false outcome of each condition. Condition coverage measures the conditions independently of each other. Functional coverage [11] is a measure of which design features have been exercised by the tests. Functional coverage is tied to the design intent and is sometimes called "specification coverage," while code coverage measures the design implementation.

TABLE III
COVERAGE MEASUREMENT CRITERIA

Tool name	Statement/ Line	Branch/ Decision	Method/ Function
Evosuite	--	✓	--
JaCoCo	✓	✓	✓
AURORA	✓	--	--
DCC	✓	✓	✓
OCCF	✓	✓	--

V. *Automation*

Automation Testing [14] is used to re-run the test scenarios that were performed manually, quickly and repeatedly. Automation of testing process includes number of steps such as test case generation, test execution and creation of test oracles It increases the test coverage; improve accuracy, saves time and money in comparison to manual testing. Automated test generation tends to be linked with code coverage, i.e. the goal of generating test automatically can easily be linked to the goal of increasing coverage. EvoSuite is a tool that automatically generates test cases with assertions for classes written in Java code

VI. Conclusions

We have studied 5 code coverage-based testing tools. Our study includes the comparison of three features: Code coverage measurement, Coverage criteria, Automation

Out of these we have evaluated java code coverage tool and all other tools are evaluated basis on the paper. Java code coverage tool effectively used for bug place identification as well as condition/decision coverage evaluated both as true and false. All other tool studied on the basis of paper and summarize in the following table.

TABLE IV
SUMMARY OF CODE COVERAGE TOOL

	Evosuite	JaCoCo	AURORA	DCC	OCCF
Supported languages	Java	Java	C	Java	C,C++,java, python, JavaScript, ruby, Lua
No. Coverage criteria	1	4	3	4	4
Instrumentation	Byte code	Byte code	--	Byte code	Source code
Automation	Yes	Yes	No	Yes	No

Acknowledgment

"No matter how big or small an endeavor is, we do nothing in vacuum" I do it with supporting roles of many others who have helped us directly or indirectly. This paper will not be complete without thanking to people who encouraged us and gave many suggestions and updating with patience and tolerance.
It is my privilege to acknowledge with deep sense of gratitude to my Guide Mrs. Sangeeta Nagpure, for her valuable suggestions and guidance. She took interest in checking the minute details of the paper and guided me throughout the same.
I express my deepest thanks and gratitude to Prof. Jyothi M Rao and all the staff members for their constant encouragement and support. I am sincerely thankful to our HOD Prof. H N Bharathi.
Also thankful to my colleagues and friends, who backed my interest and helped me out during our times of trouble. I also thank all the unseen people on the internet through which me able to gather the necessary ideas and concepts required for my paper.

REFERENCES

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

[2] Williams, B. S. a. L. (2008). "A Survey on Code Coverage as a Stopping Criterion for Unit Testing.", Technical report (North Carolina State University. Dept. of Computer Science), TR-2008-22.

[3] G. Fraser and A. Arcuri, "Evolutionary Generation of Whole Test Suites," Proc. 11th Int'l Conf. Quality Software, pp. 31-40, 2011.

[4] G. Fraser and A. Arcuri, "Evosuite: Automatic Test Suite Generation for Object-Oriented Software," Proc. 19th ACM SIGSOFT Symp. and the 13th European Conf. Foundations of SoftwarEng., 2011

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

[6] Angelo Gargantini, Marco Guarnieri and Eros MagriAURORA: AUtomaticRObustnesscoveRage Analysis Tool in 6th IEEE International Conference on Software Testing, Verification and Validation - Testing Tools Track (ICST 2013)

[7] Angelo Gargantini, Marco Guarnieri, Eros MagriExtending Coverage Criteria by Evaluating their Robustness to Code Structure Changes in 23rd International Conference on Testing Software and Systems (ICTSS 2012 - Acceptance rate: 33%)

[8] Kazunori Sakamoto, et al.,"A Framework for Measuring TestCoverage Supporting Multiple Programming Languages",First Software Engineering Postgraduates Workshop (SEPoW 2009; In conjunction with APSEC 2009), 2009. Sakamoto

[9] Dynamic Code Coverage for Sun Solaris, Linux, and HP UX @ http://www.dynamic-memory.com/

[10] K., H. Washizaki, et al. (2010). "Open Code Coverage Framework: A Consistent and FlexibleFramework for Measuring Test Coverage Supporting Multiple Programming Languages**"**, In the 10thInternational Conference on Quality Software, QSIC,2010, pp. 262-269

[11] http://en.wikipedia.org/wiki/Code_coverage

[12] http://en.wikipedia.org/wiki/Software_testing

[13] https://nuget.org/packages/OpenCodeCoverageFramework

[14] http://en.wikipedia.org/wiki/Test_automation