Comparison of Code Coverage Analysis Tools: A Review

Dr. Neetu Dabas, Mrs. Kamna Solanki
Maharshi Dayanand University, Rohtak, India.

ABSTRACT

Code coverage is a promising measure of test effectiveness. There are a large number of coverage analyzers or coverage tools that perform coverage analysis. The output of coverage measurement can be used in several ways to improve the testing process. The coverage can help to find holes i.e. areas that are not covered by test cases. Code coverage also helps in Regression testing, test case prioritization, test suite augmentation and test suite minimization. This paper aims to provide a comparison of code coverage analysis tools. It is still a challenge for test managers and developers to identify the appropriate code coverage tooling solution for the system or component under test. This comparison identified here may help to select the efficient and effective tool.

Keywords: Code Coverage, Coverage Metrics, Code Coverage Analyzer, JCover, NCover.

1. INTRODUCTION

Software testing is now an essential activity in software maintenance life cycle. It is an activity used to determine and improve software quality. In software testing, software metrics enable the appropriate quantitative information, to support us in the decision-making on the most efficient and appropriate testing tools for our programs. Where development is more systematic, organizations seek measures of testing completeness and goodness to establish test completion criteria. Code coverage is one such measure. As there are various code coverage tools suitable for different applications, we are focusing here on eight most commonly used tools.

The remaining part of this paper is organized as follows. Section 2 describes the code coverage in testing. It provides the definition of code coverage, code coverage analyzer, and its importance and limitations of code coverage analysis. It is followed by section 3 which describes various coverage metrics supported by code coverage tools. Section 4 provides the survey of code coverage tools and section 5 provides the evaluation of these tools based on a set of criteria. Finally section 6 concludes the paper.

2. CODE COVERAGE

Code coverage (also referred as test coverage) is employed as a method to measure how thoroughly software is tested. 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 [1],[4]. Code coverage analysis is sometimes called test coverage analysis. Likewise, a coverage analyzer is sometimes called a coverage monitor or coverage tool.
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.

2.1 Why use a Code Coverage Analyzer?

Without continual change and improvement, software is unable to continue meeting its users' needs. But without comprehensive automated test-suites, such change can also expose users to very undesirable risks: the risk that critical features no longer function as before, the risk of data loss and the risk of a system crash. Agile software development methodologies are helping improve the quality of software. Test-driven development, where tests are written to test features before those features are added, ensures that every piece of software is coupled with a test-suite.

No matter how good such methodologies are, and how diligently they're followed in an organization, it isn't possible to ensure that software testing is as comprehensive as it could be. That's where tools can help.

2.2 Code Coverage analyzer (CCA)

A code coverage analyzer automates the process of coverage analysis. These are also known as test coverage analyzers (TCA). 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.

In figure 1 coverage analyzer is shown by the cone which takes two inputs: set of test cases and code to be covered and produce 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.

Features

In general, a code coverage analyzer provides the following:

- Overall code coverage report (a coverage of 60% to 70% of an application can be considered acceptable)
- Coverage report by subsystem
- Customization
- Color coding for source
- Comparing test-profiles of two runs
Some specifics tools may have other features such as program prioritization for testing, assistance in debugging, and automatic generation of test cases. The quality of a code coverage tool is determined (among other things) by its coverage measurement capabilities. Guiding users in achieving high code coverage in an effective way is one of the most important features a good code coverage tool can provide. There are a number of commercial and free code coverage tools. Some companies also make built in tools according to their requirements [3].

2.3 Limitations:

Coverage techniques measure only one dimension of a multi-dimensional concept. Two different test cases may achieve exactly the same coverage but the input data of one may find an error that the input data of the other doesn’t.

- 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.

3. COVERAGE METRICS

Software metrics is defined as the current state of art in the measurement of software products and process [7]. There exists various type of software metrics such as procedural metrics, object oriented metrics and coverage metrics. Among all metrics coverage metrics plays an important role in selecting best test cases. It describes the fraction code that is covered by test cases. In this way more faults of code will be discovered by testers not by users. To measure how well the program is exercised by a test suite, coverage metrics are used. There exists a number of coverage metrics in literature [5,6]. Following are descriptions of some types of coverage metrics.

<table>
<thead>
<tr>
<th>Coverage Metrics</th>
<th>Indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement Coverage</td>
<td>The percentage of executable statements in a component that have been exercised by a test suite [5].</td>
</tr>
<tr>
<td>Path Coverage</td>
<td>The percentage of paths in a component that have been exercised by a test suite[5].</td>
</tr>
<tr>
<td>Branch Coverage</td>
<td>The percentage of branches in a component that have been exercised by a test suite [5].</td>
</tr>
<tr>
<td>Requirement Coverage</td>
<td>The percentage of requirements in a component that have been covered by a test case suite [5].</td>
</tr>
<tr>
<td>Loop Coverage</td>
<td>The percentage of loops in a component that have been exercised by a test suite [5].</td>
</tr>
<tr>
<td>Decision Coverage</td>
<td>The percentage of Boolean expressions in a component that have been exercised by a test suite [2].</td>
</tr>
<tr>
<td>Condition Coverage</td>
<td>The percentage of decisions in a component that have been exercised by a test suite [5].</td>
</tr>
<tr>
<td>Function Coverage</td>
<td>The percentage of functions in a component that have been exercised by a test suite [5].</td>
</tr>
<tr>
<td>Entry/Exit Coverage</td>
<td>The percentage of call and return of the function in a component that have been exercised by a test suite [5].</td>
</tr>
</tbody>
</table>
4. **SURVEY OF CODE COVERAGE TOOLS**

L. Shanmuga [4] selected four software code coverage tools viz. JCover, Emma, Gretel and Code Cover and considered various sorting programs like bubble sort, quick sort etc. They concludes that out of these four tools CodeCover tool is comparatively better in calculating the coverage metrics and helps in the selection of best test cases based on coverage for regression testing of Java programs.

Muhammad and Suhaimi [3] provide an evaluation of various test coverage tools in software testing. The evaluation criteria consist of language support, instrumentation, Coverage Measurement, GUI and Reporting. Management and test managers require an appropriate tool for the software under test. Following is a brief description of some code coverage tools:

4.1 **JCover**

JCover 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. A commercial JCover plug-in is available for Eclipse 3 [14].

4.2 **Emma**

It is an open-source tool for measuring and reporting code coverage for Java. It can instrument classes for coverage either offline (before they are loaded) or on the fly (using an instrumenting application class loader). It has a unique feature combination: support for large-scale enterprise software development while keeping individual developer's work fast and iterative. It detects dead code and verifies which parts of an application are actually exercised by test suite. It supports class, method, line and basic block coverage [12].

4.3 **Gretel**

Gretel 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 [13].

4.4 **Code Cover**

Code cover is an extensible open source 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 and helps to develop new test cases. It uses correlation matrix to find redundant test cases and optimize test suite [11].

4.5 **NCover**

NCover 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 [15].
4.6 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. It can be executed from ant or from the command line [10].

4.7 Clover
Clover is available as either an Eclipse or IDEA plug-ins or using ANT script. 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 [9].

4.8 BullseyeCoverage
It is a C and C++ code coverage analyzer tool that tells how much of source code was tested. 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 [8].

5. COMPARISON OF TOOLS
This section describes a comparative evaluation of eight code coverage analysis tools based on few important criteria such as type, coverage measures, memory space etc.

<table>
<thead>
<tr>
<th>Table 2: Comparison of Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Coverage Measures Supported</td>
</tr>
<tr>
<td>Memory Space</td>
</tr>
<tr>
<td>Graphical Representation</td>
</tr>
<tr>
<td>HTML Support</td>
</tr>
<tr>
<td>Reports</td>
</tr>
<tr>
<td>Supported Languages</td>
</tr>
</tbody>
</table>
6. CONCLUSION

This paper evaluates eight code coverage tools. Comparison has been made based on some criteria such as language support, instrumentation, graphical representation, reporting etc. Table 2 summarizes the result. Each tool has some strong point as well as some weak point. Users and developers can select the tool according to their need. This comparison will help in more efficient selection of tools.

REFERENCES