Indian Journal of Science and Technology, *Vol 9(32), DOI: 10.17485/ijst/2016/v9i32/100202, August 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Open Source Code Coverage Tools for Java: A Comparative Analysis

Priyanka Dhareula* and Anita Ganpati

Computer Science Department, Himachal Pradesh University, Shimla – 171005, Himachal Pradesh, India;
priyankarana.id@gmail.com, anitaganpati@gmail.com

Abstract

This research compares two code coverage tools to understand the relationship between the code coverage and regression testing, henceforth the effectiveness of the code coverage detail provided by the tools. The methodology adopted to meet the objectives follows theoretical as well as empirical approach. To achieve the empirical approach a platform was setup in eclipse IDE for Java application which was integrated with Junit to execute test cases for Java program. Two open source code coverage tools CodeCover and Eclemma were exercised respectively upon a small Java application with twenty one test cases. This execution shows that EclEmma is effective in providing the detail of individual test case. CodeCover on the other side provides combined measurement of the test cases. CodeCover provide the coverage at fine level as well as at coarser level of granularity. However regression testing demands detail coverage made by each test case which code cover fails to dispense. EclEmma generates code coverage report by providing information about individual test case. This information is most desirable when performing test case optimization in regression testing. Further coverage details of test suites given by code coverage tool will be used in proposing a hybrid regression test case optimization technique.

Keywords: CodeCover, Coverage, EclEmma, Regression Testing, Test Case

1. Introduction

The process of testing modifications made to computer programs to make sure that previous code still works with new modifications known as regression testing[1]. The quality of software code can be measured and assessed by making use of code coverage analysis for the code. Code coverage defines the measure of code executed during testing. Code coverage is useful during the regression testing, as it helps in identifying the non-covered area and augmenting appropriate test cases to increase the code coverage[1]. SUT with high code coverage signifies that it has been thoroughly tested and has a lower chance of containing software bugs than a program with low code coverage.

There are a large number of open source tools available to determine the extent of code coverage provided by test cases for Java based application. For the process of selecting appropriate code coverage tool a comparative analysis of EclEmma and Codecover is performed on a Java based application. These tools are in corporated in Eclipse IDE, where the test cases for both the tools are executed by Junit plug in for Eclipse IDE.

Code coverage analysis targets various aspects of the code, used for different purposes. The main purpose of selecting suitable code coverage tool in this study is to use the details of code coverage for SUT in regression test case optimization. Code coverage inspects the parts of code being exercise by the test cases and that needs to be improved. Code coverage identifies that part of code which has not been exercised by test cases and hence there is a need to augment the test suite. If the test cases are targeting same part of the code, it tells which test cases are redundant[2]. The code coverage information can aid in numerous activities viz. regression testing, test case selection, test case prioritization, test case minimization, etc[3]. Among various code coverage tools for Java[4], this paper has selected two open source tools for code coverage analysis to evaluate the effectiveness of test cases.

**Author for correspondence*

2. Review of Literature

In their research described CodeCover code coverage tool[1]. This paper performed a detailed investigation of versatilities of CodeCover tool for java projects. CodeCover tool executes on source code by providing synchronized, term, loop, branch and statement coverage. Code Cover is open source. The paper discussed that CodeCover can aid in test case reduction, augmentation of test cases, test case optimization in regression testing.

Java-based test coverage reporting tool called Java CodeCoverage. This tool provides test coverage details for individual as well as test suite as a whole. This tool is a byte code analyser. A noteworthy character of Java CodeCoverage is that it stores coverage detail for separate test cases, hence allowing analysis of detail coverage[7].

An approach for evaluating the features of various testing tools in order to compare them systematically and select the best one[2]. For the tool comparison they selected two code coverage tools Emma and CodeCover. The features selected by them to compare the two tools are Response Time (RT), Human-Interface Design (HID), Reporting Features (RF) and Ease of Use (EU). Their analysis concluded that CodeCover tool is more efficient than the reviewed various code coverage tools and presented the usage of information obtained by code coverage analysis[8].

In[5] discovered thirty one tools for code coverage. They found four tools that aid in branch coverage. They further chose one tool instrumenting byte code and two tools instrumenting the source code. From their study they found that each tool details the branch coverage differently.

In[3,4] performed experiments in controlled environment to measure the difference between the details of code coverage provided by various tools. This research used line, statement, branch and method coverage metrics. Their results showed that different code coverage tools give different results for mainly branch and method metrics.

An exploratory study of regression test case selection techniques. There study found that selection of test cases is primarily focused on the coverage of test case followed by change identification capability and test case fault detection[6].

3. Objectives

The main objective of this research is to compare the code coverage tools for Java program. However, the specific objectives of the study are:

- To have an understanding of the relationship between the code coverage and regression testing.
- To find the coverage level of granularity (fine granularity or coarse granularity) of test cases by CodeCover and EclEmma.
- To analyse the effectiveness of the code coverage tools on the basis of code coverage information and execution time taken for code coverage.

4. Research Methodology

To achieve the objectives of the study theoretical as well as empirical approach has been adopted. In the theoretical approach many research papers, books, online website were referred to get a thorough understanding of code coverage tools used for java and the environment setup required to perform the analysis. For the empirical approach, a platform was setup in Eclipse IDE[15] for Java developers which were integrated with Junit[14] to execute test cases for Java program. Small Java application[12] was used as shown in Table 1. The details of classes, functions, statements and the number of test cases used are given in Table 1.

- Person class contains person's name and maximum number of books that this person borrows at a particular time.
- Book Class contains title, author and person who borrow the book.
- MyLibrary contains list of books and list of people who borrows them.

CodeCover[10] and EclEmma were integrated into eclipse and exercised upon the SUT one after the other to get the code coverage[11]. The results and comparative analysis for both the tools is discussed in the next section.

5. Results and Analysis

Firstly, CodeCover was activated for Java SUT. Once the execution of the application is finished, the cover-

Table 1. Test cases of java application used for code coverage

Classes	Functions	Statements	Test Cases
MyLibrary	16	372	7
Person	6	38	8
Book	7	59	6

age details are displayed in Coverage view as shown in Figure 1.

The coverage view of CodeCoverdisplays the coverage measurement of the active test cases as shown in Figure 1. In the coverage view each row displays the coverage measurement of the corresponding element. Elements are java classes, functions and the statements of the classes.

Figure 2 displays the snapshot of execution time reported by CodeCover in running twenty one test cases. These test cases took 0.021 seconds to exercise upon SUT.

Secondly, EclEmma was enabled for SUT to get the code coverage of twenty one test cases. Initially only one test case was exercised on SUT to examine its area of coverage. Figure 3 displays the coverage detail of one of the test case named testBook(). Figure 3 shows that test-

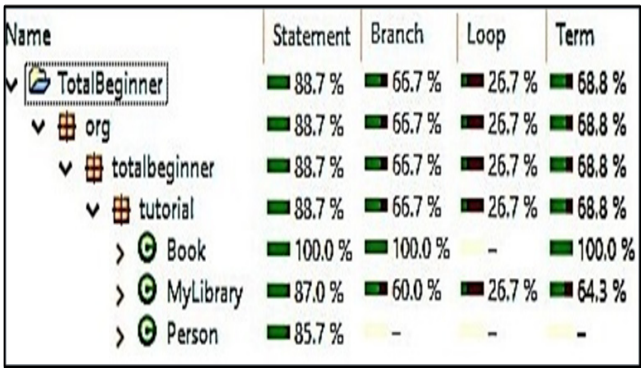

Figure 1. Coverage view of Codecover for twenty one test cases.

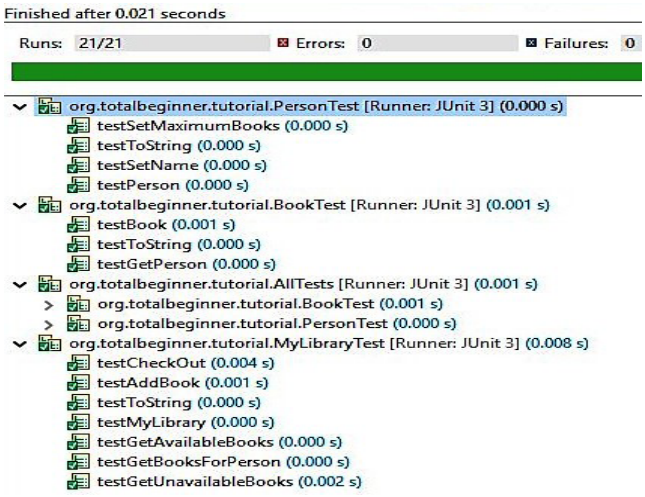

Figure 2. Snapshot of the time required for running twenty one test cases in Codecover.

Book() test case gives 2.1% coverage to the SUT. But the big question is what part of SUT is covered by testBook() test case. Figure 3 displays that there are nine instructions in the Book(string) function and the test case testBook() covers all the nine instructions, hence giving 100% coverage to the function Book(String).

It is evident from the snapshot of Figure 3 that EclEmma is an effective tool in determining the coverage at finer level of granularity. To determine whether EclEmma gives the coverage details at coarser level, again we refer to Figure 3 under the src folder the Book class shows 15.3% coverage by the testBook() test case. Therefore, it is evident that out of three classes this test case covered only Book class with 15.3% coverage. Book class contains seven functions out of which testBook() covered only one function Book(String) giving it 100% coverage. Therefore, from these details we can state that EclEmma proves to be efficient tool in providing the details of individual test cases at coarser and finer level of granularity.

Figure 4 shows testBook() test case took 0.008 seconds to exercise on SUT given by EclEmma.

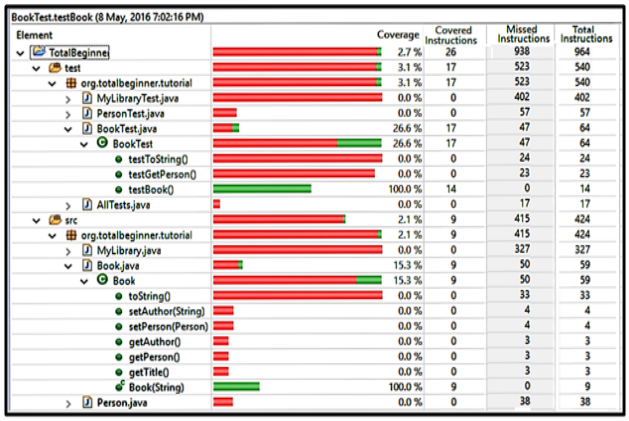

Figure 3. EclEmma coverge report of testBook() test case for SUT

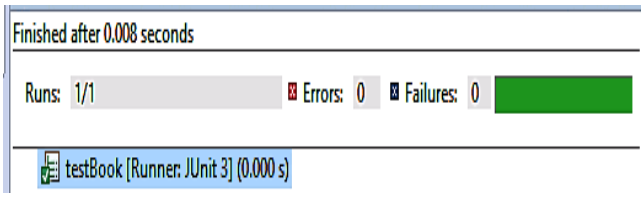

Figure 4. Snapshot of the time required for running testBook() test cases in EclEmma.

Figure 5 shows the screen shot of all the test cases run in EclEmma. Total twenty one test cases were executed on SUT. The percentage of code coverage detail provided by EclEmma for SUT is 63.4%. This information shows that we need to augment our test suite to maximize the coverage, which will result in better quality software.

Figure 6 shows that it took 0.019 seconds to run twenty one test cases reported by EclEmma. Whereas in case of CodeCover twenty one test cases gave code coverage of 88.7% at statement level in 0.021 seconds.

The graph in Figure 7 shows difference of code coverage between CodeCover and EclEmma. CodeCover give 88.7% coverage in comparison to EclEmma, which gives only 63.4% code coverage. EclEmma is effective for this study as it provides the details of individual test case. In CodeCover the combined measurement of the test cases is given. It is not identifiable as to which test case covered which part of the code. Though CodeCover gives the coverage at fine level of granularity as well as at coarse level of granularity. But regression testing demands the individual detail coverage made by each test case which CodeCover fails to deliberate. EclEmma[9,11] generates code coverage reports by providing information about individual test case. This information is most desirable when performing test case optimization in regression testing[6].

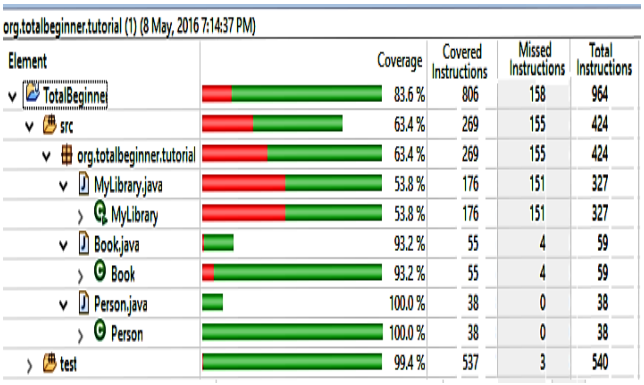

Figure 5. EclEmma reporting the coverage of twenty one test cases on SUT.

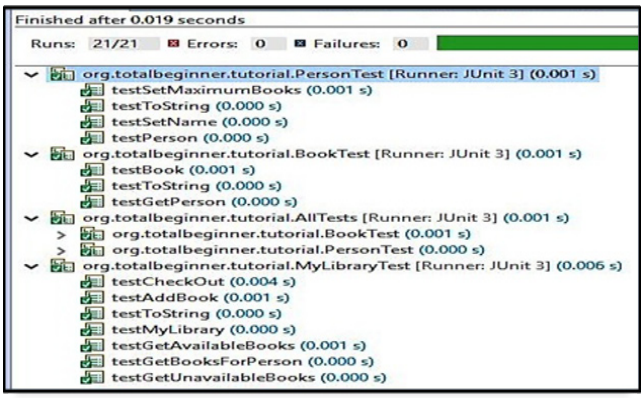

Figure 6. EclEmma reporting the total time required to execute twenty one test cases on SUT.

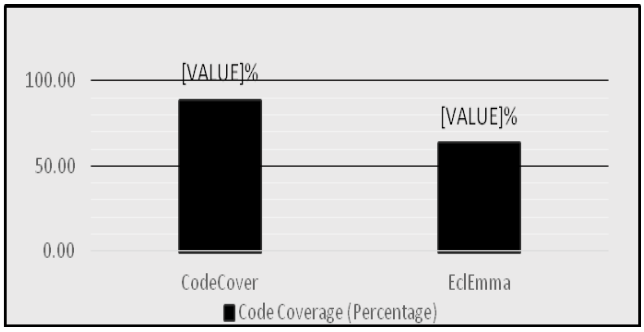

Figure 7. Comparison of tools on the basis of code coverage.

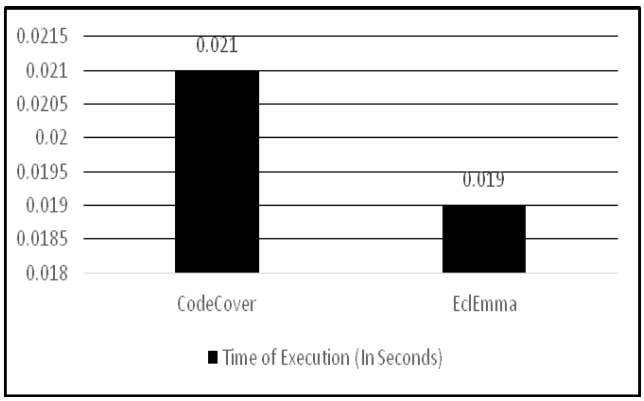

Figure 8. Comparison of tools on the basis of time of execution.

It is evident from graph in Figure 8 that CodeCover exercised twenty one test cases in 0.021 seconds in comparison to EclEmma with 0.019 seconds. Therefore, it is stated that Eclemma is most desirable for this study as it executes the test suite in lesser time as compared to CodeCover code coverage tool.

6. Conclusion And Future Work

The quality of SUT can be determined by the extent to which it has been tested. Regression testing can be effective if the designed test cases provide maximum code coverage to SUT. It is impossible to do exhaustive testing but those test cases can be chosen that provide

maximum code coverage. Therefore, the effectiveness of the test case can be determined by the level of code coverage for SUT. It is evident from the above results that EclEmma is an effective tool in comparison to CodeCover as it can give the details of individual test cases which is required for the optimization of test cases in regression testing. In CodeCover the combined measurement of the test cases is available which does not solve the purpose of examining the coverage of individual test case. Therefore, CodeCover does not aid in determining the effectiveness of individual test case. This further will not aid in regression test cases optimization. For the future work more number of coverage tools with more number of applications will be analysed to get the coverage detail which in turn will help in the regression test case optimization. The coverage details acquired for the test suite by the code coverage tool will be used in proposing a hybrid regression test case optimization technique.

7. References

1. Abhinandan HP, Nandini SS. CodeCover: A coverage tool for java projects. Proceedings of International Conference on ERCICA; 2013. p. 414-21.
2. Kajo-Mece E, Tartari M. An evaluation of java code coverage testing tools. BCI (Local); 2012. p. 72-5.
3. Alemerien K. Evaluation of software testing coverage tools: An empirical study [Doctoral dissertation]. North Dakota State University.
4. Alemerien K, Magel K. Examining the effectiveness of testing coverage tools: An empirical study. International Journal of Software Engineering and its Applications. 2014; 8(5):139-62.
5. Li N, Meng X, Offutt J, Deng L. Is byte code instrumentation as good as source code instrumentation? An empirical study with industrial tools (experience report). IEEE 24th International Symposium Software Reliability Engineering (ISSRE); 2013 Nov 4. p. 380-9.
6. Dhareula P, Ganpati A. Prevalent criteria's in regression test case selection techniques: An exploratory study. IEEE International Conference on Green Computing and Internet of Things (ICGCIoT); 2015 Oct 8. p. 871-6.
7. Lingampally R, Gupta A, Jalote P. A multipurpose code coverage tool for Java. IEEE 40th Annual Hawaii International Conference on System Sciences (HICSS); 2007 Jan. p. 261b.
8. Pathy S, Panda S, Baboo S. A review on code coverage analysis. IJCSET. 2015; 6(10):580-7.
9. Laurie W, Ben S, Sarah H. Test coverage with EclEmma. Available from: http://agile.csc.ncsu.edu/SEMaterials/tutorials/eclemma/
10. CodeCover. Available from: http://codecover.org/documentation/install.html
11. Java Code Coverage for Eclipse. Available from: http://eclemma.org/
12. Mark D. Eclipse and Java: Free Video Tutorials. Available from: http://eclipsetutorial.sourceforge.net/Total_Beginner_Companion_Document.pdf
13. Open Source Code Coverage Tools in Java. Available from: http://java-source.net/open-source/code-coverage
14. Junit. Available from: http://junit.org/junit4
15. Eclipse. Available from: http://www.eclipse.org/downloads