Lab 2: Code Coverage Analysis using EMMA
Due: March 24, 2014 11:59 CDT

1. Objectives

- Understand the importance of Code coverage in assessing testing of software deliverables
- Be capable of using a code coverage analysis tool to determine untested segments of source code.
- Measure code coverage in Java using the ECL-Emma tool in Eclipse.
- Construct new JUnit test cases to increase code coverage.

2. Introduction

One of the hallmarks used for software testing is the concept of code coverage. While the exact relationship is often elusive, research has generally shown that the greater the code coverage during testing, the greater the reliability of the deployed software will be once the system is completed. There has been significant study of the relationship between code coverage and the resulting reliability of the source code. Garg and Del Frate indicate that there is a strong correlation between code coverage obtained during testing and software reliability, especially in larger programs. The exact extent of this relationship, however, is unknown, but the general trend is consistent, though highly non-linear, as is shown in Figure 1.

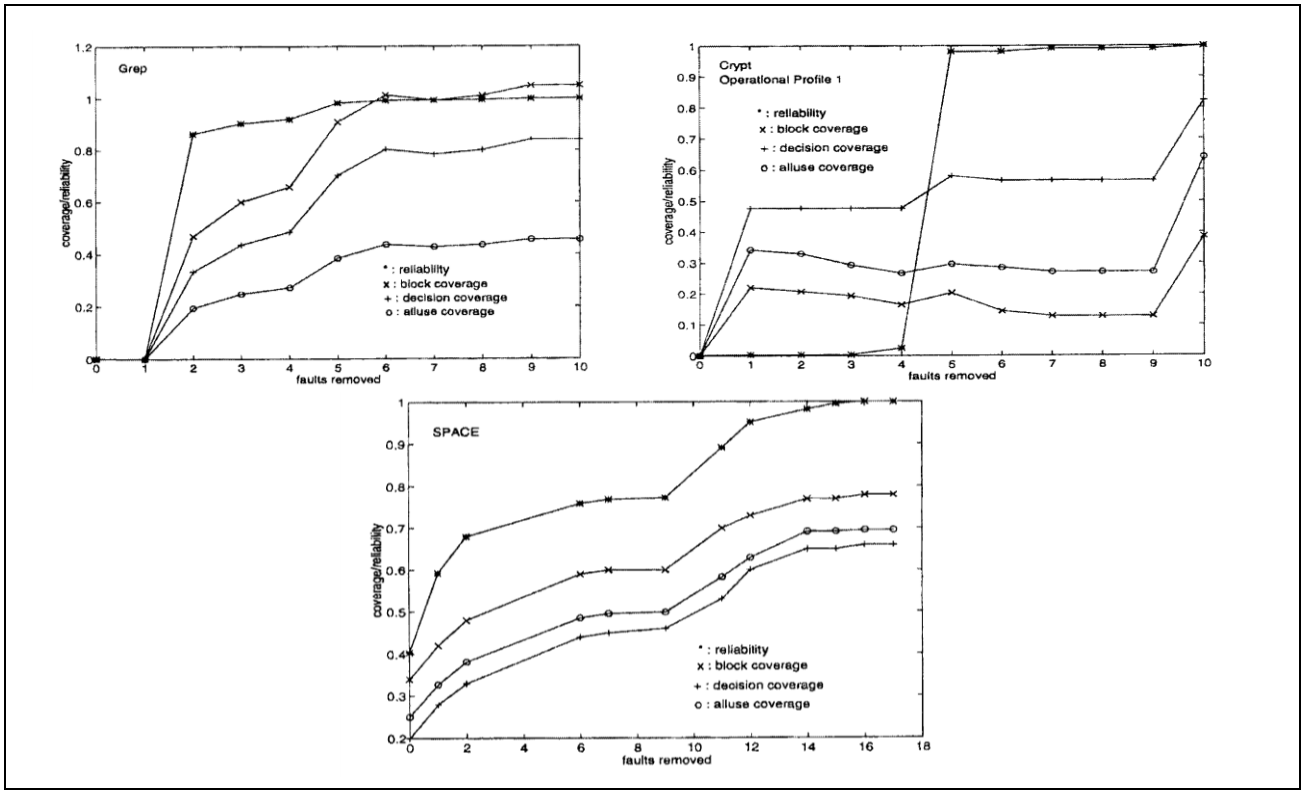

Figure 1 The relationship between software reliability and code coverage.

Marick cites some of the misuses for code coverage metrics. A certain level of code coverage is often mandated by the software development process when evaluating the effectiveness of the testing phase. This level is often varied.

Extreme Programming advocates endorse 100% method coverage in order to ensure that all methods are invoked at least once, though there are also exceptions given for small functions which are smaller than the test cases would be. Method coverage, however, is a very weak coverage measure.

Piwowarski, Ohba, and Caruso indicate that 70% statement coverage is necessary to ensure sufficient test case coverage, 50% statement coverage is insufficient to exercise the module, and beyond 70%-80% is not cost effective. Hutchins indicates that even 100% coverage is not necessarily a good indication of testing adequacy, for though more faults are discovered at 100% coverage than 90% or 95% coverage, faults can still be uncovered even if testing has reached 100% coverage.

There are two approaches to code coverage analysis. In the first case, the actual binary code is instrumented with calls to a routine which captures the method invocations. In the second case, A virtual machine executes the source code, and as part of the machine, execution logs are captured. In either case, there is a performance penalty associated with capturing the execution profile.

In the case of Emma, Emma can instrument Java classes for coverage either offline (before they are loaded) or on the fly (using an application class loader which instruments the class files).

We will be using the EclEmma plug-in for Eclipse which uses the EMMA code coverage tool to collect metrics about the code coverage obtained when you execute your program. The Emma tool is an open-source code coverage analysis tool for Java.

3. Procedure

1. Install the EclEmma tool in Eclipse using the following instructions:[1]
    a. From your Eclipse menu select Help → Install New Software
    b. Click add for the site.
    c. Enter EclEmma as the name and enter http://update.eclemma.org/ for the site URL:
    d. Press Finish.
    e. Follow the steps in the installation wizard.
2. Now that you have EclEmma installed, download the project from the course website. In the project is an implementation for a US tax calculator as well as a set of test cases for the tax calculator. Using the Coverage menu, run your test cases and take a look at the source code. Is everything green, or are there lines colored “yellow” or “red”.

[1] Or, if these instructions will not work, follow the instructions at http://www.eclemma.org/.

3. Switch into the "Coverage" view in eclipse. Right click on the tool and export an html report of your testing. Which methods have been tested and to what coverage? Which have not been tested? Based on this, was the testing any good?
4. Now download a modified code base from the website. This is a slightly different implementation of the tool which has loop unrolling and a few other changes. Using the same test cases, how does your code coverage change?
5. Now that these two projects have been analyzed, write new test cases to improve the code coverage for the implementations. Can you write a set of test cases that will achieve 100% coverage for both implementations? Why or why not?

4. Lab Deliverables

Each individual will be responsible for submitting one report with the following contents:

1. Introduction
   a. What are you trying to accomplish with this lab? This section shall be written IN YOUR OWN WORDS. DO NOT copy directly from the assignment.
2. Coverage Analysis:
   a. This is where you supply the reports generated by the ECLEmma tool. Don't worry too much about beautifying the html reports. Simply paste them in some readable format. (You may be able to directly import the html into Word.)
3. Added tests.
   a. In part 5, you were asked to write a new JUnit test to cover something which your previous tests did not. What did you add and why, and how did adding this test impact coverage?
4. Questions:
   a. Answer the questions, in narrative format, posed in the procedures regarding easier / harder testing and whether on code base has better / worse coverage versus another.
5. Things gone right / Things gone wrong
   a. This section shall discuss the things which went correctly with this experiment as well as the things which posed problems during this lab.
6. Conclusions
   a. What have you learned with this experience? What improvements can be made in this experience in the future?

This material should be submitted as a single pdf file named SE2832_Lab2Report.pdf.

Additionally, submit your modified test cases as a separate java file.

If you have any questions, consult your instructor.

Appendix A: The Tax Law

Taxes are complex, but for this assignment, they have been simplified.

The first thing of importance is the person's filing status. A return may indicate that a person is single, head of the household, married filing jointly, married filing separately, or a qualifying widow. While you will not need to check the conditions that would quality a person to file using one of these categories, you will need to use the categories to test your program.

First off you will need to test whether a person needs to file. Depending on the filing status, a person is required to file an income tax return with the IRS if the criterions of table 1 are met.

Table 1. **2008 Filing Requirements Chart for Most Taxpayers (From IRS Publication 501)**

IF your filing status is...	**AND at the end of 2008 you were…**	**THEN file a return if your gross income was at least…**
Single	Under 65	$8950
	65 or older	$10300
Head of household	Under 65	$11500
	65 or older	$12850
Married, filing jointly	Under 65 (both spouses)	$17900
	65 or older (one spouse)	$18950
	65 or older (both spouses)	$20000
Married, filing separately	Any age	$3500
Qualifying widow(er)	Under 65	$14400
	65 or older	$15450

The constructors for the Tax Calculator are responsible for accepting the name of the filer, the type of filer, and the filer's age, along with the spouse's age if the filing type is any form of married.

Income tax is based upon the taxable income of the filer. The taxable income is determined by taking the gross income and subtracting the standard deduction from this amount. The standard deduction amount depends on the filing status and the age of the filer. This information is provided in Table 2.

Table II: Standard Deduction

Filing Status	**Base Standard Deduction**
Single	$5450
Married, filing separately	$5450
Head of household	$8000
Married, filing jointly	$10900
Qualifying widow(er)	$10900

In addition to the base standard deductions, the standard deductions are increased by $1050 for each person on the return who is age 65 or older. For example, a couple who is filing jointly and has the ages of 72 and 71 would have a standard deduction of $13000.

The amount of tax is calculated based upon a sliding scale. For example, a single person making less than 8025 would simply pay 10% on their income. However, if they made more than $32550, they would pay 10% on the first $8025, 15% on the amount between $8025 and $32550, and 25% on the remaining income. B This is detailed in Appendix A.

5. Examples

Jill, age 19, is in college and works part time as a waitress. Each year, she makes a total of $7500.00. Because her income is so low, she is not required by law to file a return. Her taxable income, if she did file, would be $2050.00, and she would be responsible for a total of $205.00 in taxes. While she is not required to file a return, it may be advantageous, as if more than $205.00 has been withheld from her paychecks, she may be able to receive a refund.

Jill's boyfriend Mark, also 19, works as a pizza delivery man. Each year, he makes a total of $11,000.00. He must file a return, as his income is greater than $8950.00. His standard deduction is $5450.00, making his taxable income $11,000.00-$5450.00=$5550.00. His tax due is 10% of his taxable income, yielding a net tax rate of 5.045%.

Oil tycoon, age 95, John D. Rockerfeller earns 5,000,000.00 a year and is single. He must file a return. Because of his age, his standard deduction is $6500.00, yielding taxable income of $4,993,500.00. On his, he pays a total tax of $1,726,321.75, yielding a net tax rate of 34.526%.