ECE 322 Lab Assignment 6

Regression and Mutation Testing

Overview

The objective of this laboratory assignment is to become familiar with regression and mutation testing techniques. This lab makes use of a mutation testing tool called PiTest.

Introduction

The following section will provide a brief overview of mutation testing and regression testing, serving as an introduction to these topics for the purposes of this lab assignment.

Mutation testing:

Mutation testing uses the concept of automatic code mutation to determine how well unit tests are able to detect errors in the application. This type of testing changes the existing source code by changing equality operators, mathematical operators, etc. The existing unit tests are then run against the mutated source to see how many mutations are caught. If a previously passing test fails as the result of source mutation we say that mutant was *killed*.

For this lab session we will be using the Pi Test mutation tool which is available at http://pitest.org/. For additional information on the use of Pi Test refer to the resource document available on the course website.

Regression Testing:

Regression testing is concerned with re-testing a system after updates (such as bug fixes, or new features) have been added to the application. The intention of regression testing is to show that the changes to the code have not introduced any new bugs in the system or impacted the system negatively.

The simplest form of regression testing involves simply running all existing test cases after the modifications have been introduced. In some cases however, this can be excessive and take a long time when the system is very large. An alternative approach is to selectively re-run test cases based on where changes to the system were made. A final essential aspect of regression testing is to update and create test cases for new functionality in the system, and to change existing test cases where the requirements of the application have changed.

Preparation:

Ensure that you have all required resources for this lab session downloaded and ready for use on your account. It may also be a good idea to **read the laboratory resource document** sections relevant to this lab, and to check PiTest documentation.

Source code for this lab is available on the class website.

Lab Experiment

Task 1: (40 marks)

The provided lab files include an implementation of a simple array helper library which will be used for the mutation testing portion of this lab assignment. Your task for this lab is in two parts:

First:

- Create test cases fulfilling the requirements of statement coverage
- Create test cases fulfilling the requirements of branch coverage
- As with all unit testing your test cases must make meaningful assertions regarding the functionality of the array library
- Ensure that your test cases cover (and test!) the basic functionality of the library

Second:

- Create mutations using the PiTest library, and run your tests against them (via PiTest interface).
- Record the results of your testing (number of mutants killed and total mutants etc.). If there are surviving mutants, improve your test cases so that all, or almost all mutants are killed.
- Save your mutation testing report and submit it with your lab code

Submit the results of your mutation testing and comment on the effectiveness of the different testing strategies employed with regards to finding and killing different types of mutants. Was mutation testing effective in exposing holes in your test cases? Did you find any errors in the provided code? Did you kill all the mutants? If not, why not?

Provide a discussion on the use of mutation testing. Do you think it is an effective method of evaluating test case quality? What types of functionality is is appropriate for? For what types of functionality is it not very useful? **Demonstrate your understanding of the concept** and how it could be used in a real world setting.

Task 2: **(40 marks)**

In this part of the lab we will be applying the concept of regression testing to a simple maths library which you are in charge of testing before the release of a new version. This package contains a number of simple mathematical functions which you will need to test:

Your task for this task is in two parts:

First:

- Prepare and implement test cases for the existing functions in the math package and run your tests against the provided code. As always these must be meaningful, comprehensive unit tests.
- **Report** any errors you have found in the source code with your test cases (Do not fix them yourself at this point).
- Record your first set of next cases in a test case table indicating failed test cases. Make note of what caused these failures.

Next:

- Replace the math library class file in your project with the contents of the file labelled *commit.java* which can also be found in the provided code. This simulates the next addition to the library from the development team in which they claim to have fixed the errors you found previously, and they have also added some additional functionality.
- Some functions have been added to the library. Adjust your test suite to cover the added functionality following the same coverage requirements as part 1.
- Test the new version of the software and report your findings
- Resolve and errors you find in the new version and report these changes in your report
- Record the results of your testing in a test case table and submit both tables with your report

Hint: There may be hidden changes between versions which cause unexpected failures, the use of a diff tool can sometimes be useful in determining exactly where changes to the code were made.

Include in your report where you found failures, the cause of these failures, and whether or not they were resolved by the new commit from the development team.

Provide a brief discussion on the merits of regression testing, and why it may be important in a real world setting.

Lab report:

Must be typed, no handwritten versions will be accepted. Follow the general format as included in the lab guideline. Submit all code via email. Your report should include:

- Your write-up
- The results of your test cases in tabular format
- Any conclusions you have drawn from these test cases regarding the applications under test, and your discussion on the testing methods used in this lab assignment.
- Any changes you needed to make to the provided source to make your tests pass. Any modified source code should be submitted with your report.
- Your test cases as Java files to be run against your provided changed code for validation

Clearly indicate all failed test cases, and explain the cause of these failures.