ECE 322 Lab Assignment 4

Unit and Pairwise Testing (White-box Testing #1)

Overview

The objective of this laboratory assignment is to become familiar with the rudimentary techniques of white-box testing, specifically unit testing. Also this assignment introduces us to the pairwise test case generation tools. This lab makes use of Java, Eclipse, and the JUnit testing framework.

Introduction

White-box Testing:

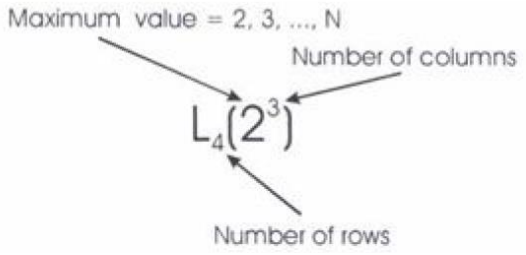
White-box testing focuses on testing the internal structure and implementation details of the application. In contrast to black-box testing, this style of testing requires the tester to have detailed knowledge of the internal structure of the software under test, usually in the form of direct access to the source code.

In this lab we will concentrate on **control flow testing** where we consider four fundamental coverage criteria: *Statement coverage, branch coverage, condition coverage,* and path coverage.

Pairwise Testing:

Instead of considering all possible combinations of input variables, pairwise testing provides an efficient way to construct a set of test cases that covers all combinations of the test data for each pair of variables. The **Orthogonal Array Test Strategy** (OATS) is a widely used technique of testing the pair-wise interactions of variables. An orthogonal array-array can be expressed in the following way:

	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1



A Library of Orthogonal Arrays can be found here: http://neilsloane.com/oadir/

Frequently, the exact orthogonal array we need for the problem at hand is not available, in this scenario we can select an orthogonal array which is slightly "too large" and modify it to suit our needs, or use approximation of orthogonal arrays to generate reduced, though perhaps not minimal, test cases.

PICT pairwise test case generation tool

Due to the popularity and effectiveness of pairwise testing, many tool have been developed to help generate pairwise test cases without the need for the proper orthogonal array to be readily available, or the correct array to be time consumingly found in a database. One such tool, developed by Microsoft, is PICT which computationally calculates approximations to orthogonal arrays which are then used to generate pairwise test cases. This simple tool uses an input file which defines the valid inputs for each variable in the system. When passed a valid input file the program outputs a list of test cases. The tool can be obtained at:
http://download.microsoft.com/download/f/5/5/f55484df-8494-48fa-8dbd-8c6f76cc014b/pict33.msi

And instruction documentation is available at:
http://www.amibugshare.com/pict/help.html#_Toc127249182 ,and
http://msdn.microsoft.com/en-us/library/cc150619.aspx

Note that the results produced from PICT are algorithmic approximations to orthogonal arrays and may be less efficient in terms of the number of test cases than using a true orthogonal array. The provided download link is only valid for windows installations (it contains an MSI file). If you wish to use this tool on a Unix system you will need to build it yourself from the source code available at: https://github.com/Microsoft/pict

Build instructions are provided at the bottom of the PICT projects readme file on Github. Use this resource to install PICT on non-windows operating systems.

Preparation:

Make sure you have Eclipse and JUnit installed on your account/computer, or an equivalent Java IDE. The code for this experiment is available on the class website and can be imported into eclipse as an existing project once extracted. You may need to download the JUnit JAR file and **add it to your project build path** in order for test cases to compile and run.

Task 1 (70marks)

The bisection method is a simple way to solve equations in the form of f(x) = 0 given an interval in which the root can be found. The source for this project contains some code which executes the bisection method to find the root of an equation. The source code is available on the class website.

First, **draw a control flow graph** for the algorithm, then generate test cases that comply with:

• Statement coverage
• Branch coverage
• Execution each loop body at least twice (enter the loop and repeat)

Submit your test cases as JUnit tests and compare results based on the above coverage criteria in your written report. Constructors and error cases are included in the requirements for code coverage. For your report you must include both:

- The test cases as **JUnit code** which can be executed against the provided lab project
- A **test case table** similar to those used in black box testing containing a meaningful
- description for each test case, the expected result, and the actual result

In addition to the requirement of code coverage, keep in mind that for full marks **your test cases are required to assert meaningful things regarding the functionality of the code** and the results of your tests. For example, a successful use of the bisection method should assert that a root value is returned from the algorithm within the given error threshold.

Provide a brief discussion on the effectiveness of these coverage criteria. Based only on the coverage criteria, how confident are you in the bug free nature of the code? Do tests fulfilling these criteria adequately cover the full functionality of the algorithm? How many tests would be required to fulfil path coverage?

As always, your test cases should be executed, and the results recorded into a **test case table** to be handed in with your report. Failed test cases must be highlighted, and the reasons for failure must be identified.

Task 2: (30 marks)

Assume we have a system with three independent variables (A, B, and C). Each variable has three possible values (0, 1, 2).

Your task is to:

- Select a proper orthogonal array from the repository provided earlier in this document
- Identify the set of test cases for this problem based on this array, reduced if necessary
- Compare the use of orthogonal arrays to randomly generated combinations

And

- Create a valid PICT input file for the problem
- Use the tool to generate test cases
- Comment on the effectiveness of this type of tool in test case generation

Compare the use of orthogonal arrays to the PICT estimation tool. There is no application under test for this component, this is a conceptual exercise.

Include the generated test cases in your lab report as well as your observations on this method. As there is no application to test, test cases need not be executed. Comment on the effectiveness of using these types of tools to generate pairwise test cases, and on the effectiveness of pairwise testing in general. What types of errors are caught by pairwise testing, what types are missed? Are the weaknesses of the method justified by the test case efficiency increase when compared to exhaustive testing? How many test cases would be required to cover all combinations of input values in the example application?

Lab report:

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

- Your write-up and any required diagrams
- The results of your test cases in table format
- Any conclusions you have drawn from these test cases regarding the applications under test.
- Your test case code as Java files to be easily executed against the provided source for validation.

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