CSIS0403/COMP3403
Implementation, Testing, and Maintenance of Software Systems
Department of Computer Science
The University of Hong Kong

Assignment 1 – Practical Black Box Testing and Test Automation

Part 1

Description

The Environmental Protection Department (EPD) is developing a new air pollution index in line with EU standards. The new index is based principally on the measured concentration of Respirable Suspended Particulates (RSP), a very common pollutant in HK. The RSP concentration is used to calculate a raw index value. An adjustment is then applied to transform the raw value into a final reported value. The adjustment reflects the level of RSP or, for lower RSP, the combined effect of three additional pollutants: SO_2, NO_2 and O_3. Thus, all four pollutants are represented in the index.

The EPD has developed a Java class containing a method to calculate the index. The method takes the measured concentrations of each of the four pollutants as arguments and returns the corresponding value of the pollution index. You are required to perform black box testing on this method. The class is a re-implementation of a prototype which, while accurate, was not developed to high standards of quality. The original prototype will serve as a gold standard oracle for your testing.

The number of parameters and sub-domains is small and the specification can be captured in the form of a decision table or cause-effect graph. You will apply a combination of Decision Table Testing and Streamlined Domain Testing. In addition, the availability of an oracle provides an opportunity to apply Random Testing techniques. Your test strategy incorporates both approaches.

In Part 1 of this assignment, you will prepare a set of test cases to test the method and will create a Java test harness to automate testing. Your harness will also have the ability to perform random testing on the method.

After Part 1, you will be supplied with the implementation under test (IUT) and an oracle, both in the form of class files in a JAR file. In Part 2 you will execute your tests and document your findings.

Background

When the application is deployed, the four measured pollutant concentrations will be obtained from measuring instruments of known precision and known effective ranges. The four arguments, their valid ranges, and their precision are as follows (note that three ranges are over half-open intervals):

Pollutant	Units	Range	Precision	Example
RSP	$\mu g/m^3$	[0, 300]	0.1	132.4
SO_2	mg/m^3	[0, 3)	0.001	1.376
NO_2	mg/m^3	[0, 1.5)	0.001	1.499
O_3	mg/m^3	[0, 1.5)	0.001	0.428

The intervals in the range column indicate the input ranges over which measurements are considered valid. The overall implementation of the system guarantees that the method you are testing will never be exposed to argument values lying outside the valid ranges shown. You can trust that guarantee.

The method will calculate and return an int in the range [0, 5000] representing the value of the air pollution index for the given arguments. The index value is determined using one of three different methods of calculation depending on the concentration of pollutants. The method of calculation is selected according to the following rules.

Condition	Method of calculation
[RSP] exceeds 100 $\mu g/m^3$	Hazardous RSP (HRSP)
$[SO_2] + 4*[NO_2] + 2*[O_3]^2 > 3$ mg/m^3	Significant additional pollutants (SAP)
All other cases	Minor additional pollutants (MAP)

The requirements specification states that the rules should be applied in the order given in the table and the first one to fire determines the method of calculation. For example, if the measured concentrations are [RSP]: 50, $[SO_2]$: 1.5, $[NO_2]$: 0.375 and $[O_3]$: 0.1 then the "Significant additional pollutants" method is used to calculate the index.

JAR Details

In Part 2 you will be supplied with a JAR file named `IndexEngine.jar` containing a package `C3403.A1`. In the package are an interface, `IndexEngine`, and two classes that implement that interface: `IUTEngine` and `OracleEngine`, representing the IUT and oracle implementations respectively.

The interface declares a single method with a return type of `int`. This is the method you will test. Its signature is:

```
calculateIndex(double, double, double, double)
```

Tasks

1. Develop test cases

   a) Construct the Decision Table or Cause-Effect Graph – for this simple problem, either is sufficient. Any format is acceptable as long as each variant is expressed clearly and labelled. It will help us understand your test cases.

   Your Actions will not specify calculated results since you are using an oracle in this assignment, but they should state which method of calculation you expect to be applied for each variant.

   Note that the specification provided above, in Background, is not yet in the form required for decision table testing.

   b) Apply the 1x1 Streamlined Domain Testing strategy to construct test cases for each variant in order to determine whether there are boundary defects in the implementation. Follow the example we worked through in class to construct a table for each of your variants. Each table will show the test cases needed for that variant.

   c) You will find that several test cases are duplicated across variant test sets. This is inevitable when we are probing boundaries of adjacent sub-domains. When you encounter such a test case, flag it as a duplicate and reference the variant set that contains the original, as shown in class. You will run such test cases only once. Try to improve the efficiency of your testing by flagging

all such duplicates. Label the test cases you will run so that we know which will be included in your final test set and which have been removed as duplicates.

Indicate the total number of test cases in your final test set.

2. Prepare for testing

a) Write a Java test harness to automate your testing. It will read cases from file, invoke the `calculateIndex()` methods of the IUT and oracle, and determine pass/fail status.

b) Prepare a file containing your test cases for execution by your harness.

c) Prepare a short Level Test Procedure Specification for your test cases describing the steps needed to execute them. Base your document on Section 12.2 of IEEE Std 829-2008 (see Appendix).

3. Prepare for Random Testing

a) Enhance your test harness to enable you to perform automated random testing of the IUT.

Deliverables for Part 1

1. The Decision Table or Cause-Effect Graph. [Task 1 (a)]
2. A set of concrete test cases, documented by variant in table form. [Task 1 (b) and (c)]
3. A test procedure specification. [Task 2 (c)]
4. Source code for your test harness. [Task 2 (a) and Task 3 (a)]

Deadline: Wednesday March 9. 11:55 PM

The IUT and oracle for Part 2 will be released after the deadline of Part 1.

Appendix: <Extract from IEEE Std 829-2008>

12. Level Test Procedure (LTPr)

The purpose of an LTPr is to specify the steps for executing a set of test cases or, more generally, the steps used to exercise a software product or software-based system item in order to evaluate a set of features.

Details on the content for each topic are contained in 12.1 through 12.3. A full example of the LTPr outline is shown in the boxed text.

Level Test Procedure Outline (full example)

1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
1.4. Relationship to other procedures

2. Details
2.1. Inputs, outputs, and special requirements
2.2. Ordered description of the steps to be taken to execute the test cases

3. General
3.1. Glossary
3.2. Document change procedures and history

<Section 12.1 omitted>

12.2 (LTPr Section 2) Details of the Level Test Procedure

Introduce the following subordinate sections. This section includes the inputs and outputs as well as the ordered description of the test steps required to execute each test case.

12.2.1 (LTPr Section 2.1) Inputs, outputs, and special requirements

Identify all that is needed to execute the tests, including but not limited to test cases, databases, automated tools, and external and/or third-party systems.

Identify any special requirements that are necessary for the execution of this procedure. These may include prerequisite procedures, special skill requirements, and special environmental requirements.

12.2.2 (LTPr Section 2.2) Ordered description of the steps to be taken by each participant

Include the activities below (as applicable) for each procedure; there may be one or multiple procedures in one Level Test Procedure document. Include the degree to which the procedure steps can
be varied and the process for determining the allowable degree of variation (if variance is allowed).

- Log: List any tools or methods for logging (the results of test execution, any anomalies observed, and any other events pertinent to the test).
- Setup: Provide the sequence of actions necessary to prepare for execution of the procedure.
- Start: Provide the actions necessary to begin execution of the procedure.
- Proceed: Provide any actions necessary during execution of the procedure.
- Measurement: Describe how the test measurements will be made.
- Shut down: Describe the actions necessary to temporarily suspend testing, when unscheduled events dictate.
- Restart: Describe any procedural restart points and the actions necessary to restart the procedure at each of these points.
- Stop: Provide the actions necessary to bring execution to an orderly halt.
- Wrap-up: Provide the actions necessary after the execution of the procedure has been completed (including termination of logging).
- Contingencies: Provide the actions necessary to deal with anomalies that may occur during execution.

<Section 12.3 omitted>