FIT5171 Tutorial 4

Black box / functional testing
White box / structural testing

Saranya Alagarsamy
Faculty of Information Technology
Monash University

Course outline

1. Overview of unit & Fundamentals of software testing
2. Test Planning & Test-driven development
3. Discrete mathematics refresher
4. Black box / functional testing
5. White box / structural testing
6. Component testing
7. Software quality & metrics
8. System testing
9. Object-oriented testing
10. Software verification and validation techniques
11. Revisiting the fundamentals of testing

A **flight recorder** is an electronic recording device placed in an aircraft for the purpose of facilitating the investigation of aviation accidents and incidents.

The device is often be referred to as a

“black box”

What is Black Box Testing

Black box testing involves testing a system with no prior knowledge of its internal workings.

A tester provides an input, and observes the output generated by the system under test.

Types Of Black Box Testing

Functional Testing

For example, checking that it is possible to log in using correct user credentials, and not possible to log in using wrong credentials.

Non-Functional Testing

Black box testing can check additional aspects of the software, beyond features and functionality. A non-functional test does not check “if” the software can perform a specific action but “how” it performs that action. Eg. Stress Test in TV, multiple key press.

Regression Testing

Black Box Testing Techniques

Equivalence Partitioning

Testers can divide possible inputs into groups or “partitions”

Boundary Value Analysis

Testers can identify that a system has a special response around a specific boundary value.

Decision Table Testing

Many systems provide outputs based on a set of conditions.

State Transition Testing

Responses are generated when the system transitions from one state to another.

Error Guessing

Testing for common mistakes developers make when building similar systems.

What is White Box Testing?

White box testing is a form of application testing that provides the tester with complete knowledge of the application being tested, including access to source code and design documents.

This in-depth visibility makes it possible for white box testing to identify issues that are invisible to gray and black box testing.

What Does White Box Testing Focus On?

White box testing takes advantage of extensive knowledge of an application’s internals to develop highly-targeted test cases

- Path Checking
- Output Validation
- Loop Testing
- Data Flow Testing

Types Of White Box Testing

White box testing can be performed for a few different purposes. The three types of white box testing are:

- ***Unit Testing***
- ***Integration Testing***
- ***Regression Testing***

White Box Testing Techniques

One of the main advantages of white box testing is that it makes it possible to ensure that every aspect of an application is tested. To achieve full code coverage, white box testing can use the following techniques:

Statement Coverage: Statement coverage testing ensures that every line of code within an application is tested by at least one test case.

Branch Coverage: Conditional statements create branches within an application's execution code as different inputs can follow different execution paths.

Path Coverage: An execution path describes the sequence of instructions that can be executed from when an application starts to where it terminates.

In Lecture 4 we discussed Boundary Value Testing (BVT) and the (minimum) number of test cases for the “normal” (non-robust) version of BVT for n variables (4n + 1). In this question, work out a formula for the number of test cases for each of the following cases and briefly explain why.

(a) The robust BVT (with additional values min− and max+ for each variable).

(b) Weak normal equivalence class testing.

(c) Weak robust equivalence class testing

The BVT (with additional values min− and max+ for each variable).

For each variable, we need to test its minimum and maximum values, which gives us 2n test cases.

The total number of test cases required for the robust version of BVT is:

2n (minimum and maximum values) + 4n + 1 (normal values)

6n + 1

Weak normal equivalence class testing.

Once the equivalence classes are defined, we then select test cases from each equivalence class. In weak normal equivalence class testing, we select test cases that are "typical" or "normal" for each equivalence class.

We might define the following equivalence classes:

Age: Under 18, 18-65, over 65

Income: Under $30,000, $30,000-$100,000, over $100,000

For example, we might select the following test cases:

Age: 30 (18-65 class)

Income: $50,000 ($30,000-$100,000 class)

Let C_x denote the equivalence classes of valid values for variable x
Then the number of test cases, for x_i ranging over all variables.

That is, the maximum number of equivalence classes for all variables.

$$max(\#C_{x_i})$$

where $\#C_{x_i}$ denotes the number of equivalence classes for variable x_i.

Weak robust equivalence class testing

In addition to testing "typical" or "normal" values for each equivalence class, we also test values that are outside the range of valid inputs or that violate other constraints or requirements.

For example, continuing with the previous example we might define the following equivalence classes:

Age: Under 18, 18-65, over 65

Income: Under $30,000, $30,000-$100,000, over $100,000

Values that are outside the range of valid inputs or that violate other constraints. For example, we might select the following test cases:

Age: 16 (under 18 class), 75 (over 65 class)

Income: $20,000 (under $30,000 class), $150,000 (over $100,000 class)

We assume the same settings as in the previous question.

Let I_{x_i} denote the equivalence classes of invalid values for variable x_i . Let n denote the number of variables. the number of tests for x_i ranging over all variables.

$$max(\#C_{x_i}) + \sum_{i=1}^{n}(\#I_{x_i})$$

Basically, we include also the total number of test cases in the invalid areas for each variable.

In the last lecture we showed a triangle example to demonstrate test case generation for BVT (slide 11). In the example each of the three variables a, b and c is the length of a side from the range [1, 200].

Come up with test cases for weak normal equivalence class testing that cover the same expected outputs (isosceles, equilateral, scalene, not a triangle).

Requirement

R1 = {(a, b, c) | the triangle with sides a, b and c is isosceles}

R2 = {(a, b, c) | the triangle with sides a, b and c is equilateral}

R3 = {(a, b, c) | the triangle with sides a, b and c is scalene}

R4 = {(a, b, c) | sides a, b and c do not form a triangle}

Equilateral triangles:

Test case 1: a=10, b=10, c=10 (all sides are equal)

Isosceles triangles:

Test case 4: a=50, b=50, c=80 (two sides are equal)

Scalene triangles:

Test case 7: a=30, b=40, c=50 (all sides are different)

Not a triangle:

Test case 10: a=1, b=2, c=4 (sum of two sides is less than the third)

Decision table for testing.

		Rules								
Conditions	c1: a,b,c form a triangle?	F	T	T	T	T	T	T	T	T
	c2: $a = b$?	-	T	T	T	T	F	F	F	F
	c3: $a = c$?	-	T	T	F	F	T	T	F	F
	c4: $b = c$?	-	T	F	T	F	T	F	T	F

Under the tutorial resources, you will find a pdf document which includes the NextDate method, which, given a day, a month, and a year, returns the date of the following day.

(a) Complete the decision table on slide 37 for NextDate by filling in the missing conditions and associated actions.

Decision table - extended example

		Rules						
		1	2	3	4	5	6	7
Conditions	c1: day in	D1, D2, D3	D4	D5	D1,D2,D3,D4	D5	…	D5
	c2: month in	M1	M1	M1	M2	M2	…	M3
	c3: year in	-	-	-	-	-	…	-
Actions	a1: impossible			X			…	
	a2: increment day	X			X		…	
	a3: reset day		X			X	…	X
	a4: increment month		X			X	…	
	a5: reset month						…	X
	a6: increment year						…	X

How many test cases are needed to completely cover the entire decision table?

In general, the number of test cases needed to cover a decision table with n conditions and m actions

2*n + m.

Number of test cases = 8+5 = 13

For the NextDate method, assuming the year variable ranges over [1812, 2016], how many test cases are needed for strong, normal boundary value testing?

Since BVT doesn’t understand the semantics of variables, the boundary values for the variables are:

day: 1 and 31,

month: 1 and 12;

year: 1812 and 2016

For strong, normal boundary value testing, each variable can take on (min, min+, nom, max-, max) values (normal), and all variables are free to take any of the above values (strong). Hence, the total number of test cases is 5^n for n variables.

For our example, it is $5^3 = 125$ test cases.

Compare and comment on the effort and effectiveness of BVT and decision table testing.