


DA-IICT

IT 314: Software Engineering

*Requirements-based Test Generation
for
Functional Testing*

Saurabh Tiwari

1



“The program reads three integer values from an input dialog. The three values represent the lengths of the sides of a triangle. The program displays a message that states whether the triangle is scalene, isosceles, or equilateral”.

2

Methods of testing

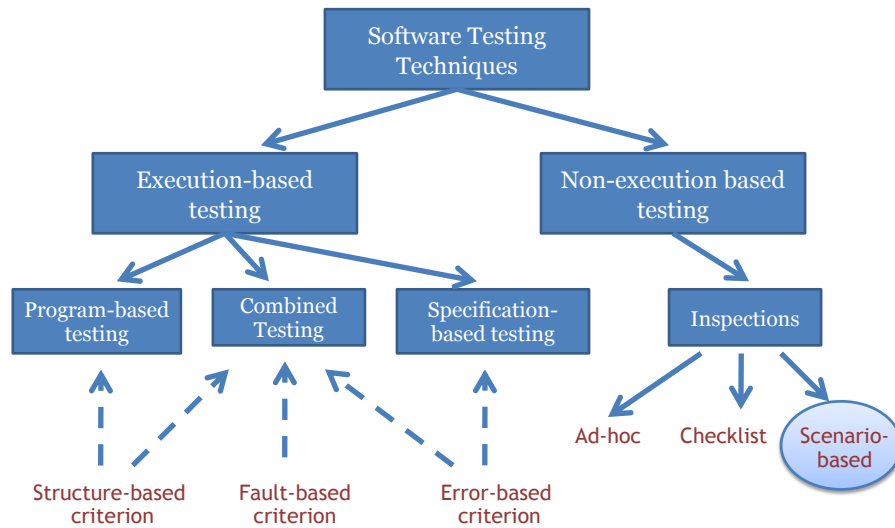
- Test to specification:
 - Black-box
 - Data-driven
 - Functional testing
 - Code is ignored: only use specification document to develop test cases
 - Test to code:
 - Glass box/White box
 - Logic driven testing
 - Ignore specification and only examine the code.
-

Static Testing

Static testing is the process of carefully and methodically reviewing the software design, architecture, or code for bugs without executing it.

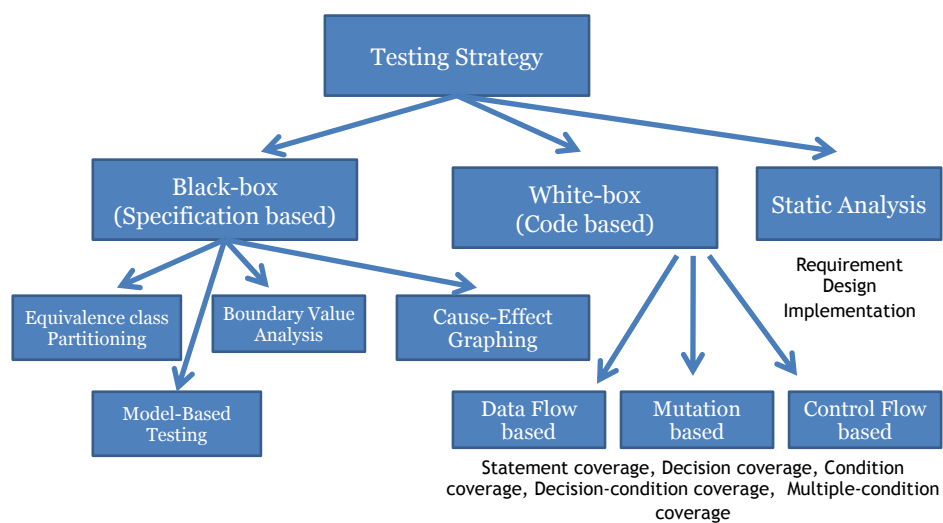


Testing Technique Hierarchies



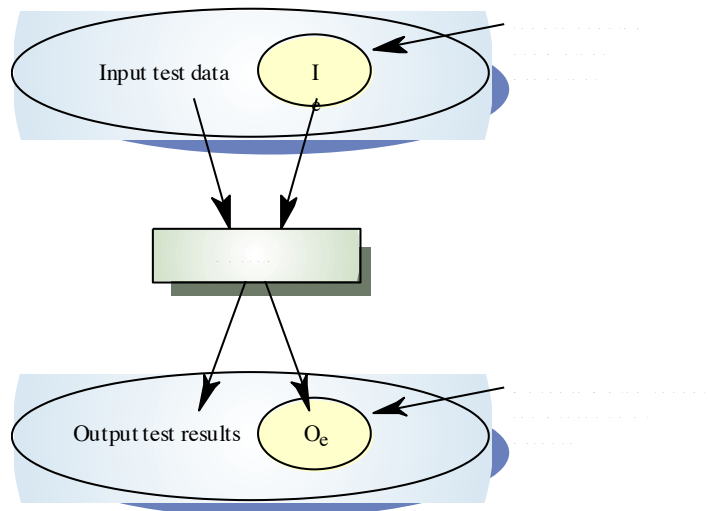
5

Testing Strategy



6

Black-Box : Test to Specification



7

Equivalence Class Partitioning

- Partition the program input domain into equivalence classes (according to the specifications)
- The rationale is that test of a representative value of each class is equivalent to a test of any other value of the same class.
- Identify valid as well as invalid equivalence classes
- One test case from each equivalence class

8

A Set of Guidelines

- If an input condition specifies a range of values (e.g., “the item count can be from 1 to 999”), identify one valid equivalence class (1<item count<999) and two invalid classes (item count <1 and item count >999)
- If an input condition specifies the number of values (e.g., “one through six owners can be listed for the automobile”), identify one valid equivalence class and two invalid equivalence classes (no owners and more than six owners)
- If an input condition specifies a set of input values and there is a reason to believe that the program handles each differently (“type of vehicle must be BUS, TRUCK, TAXICAB, PASSENGER, or MOTORCYCLE”), identify a valid equivalence class for each and one invalid equivalence class (e.g., “TRAILER”)
- If an input condition specifies a “must be” situation such as “first character of the identifier must be a letter”, identify one valid equivalence class (it is a letter) and one invalid equivalence class (it is not a letter)

9

A Summary: Set of Guidelines

Input Condition	Valid Eq. Classes	Invalid Eq. Classes
Range of values (1-200)	One valid (value within the range)	two invalid
Number “N” valid values	One valid	Two invalid (none, more than N)
Set of input values each handled differently by the program (e.g. A, B, C, ... (n))	One valid Eq. class for each value (total n)	One (e.g. any value not in valid input set)
Must be condition (e.g., id name must begin with a letter)	One (e.g., it is a letter)	One (e.g., it is not a letter)

10

Example

- If there is any reason to believe that the program does not handle elements in an equivalence class identically, split the equivalence class into smaller equivalence classes.

Example:

Input condition	$0 \leq x \leq \text{Max}$
Valid Equivalence Classes	$0 \leq x \leq \text{Max}$
Invalid Equivalence Classes	$x < 0; x > \text{Max}$

3 Test Cases

11

Example

Example of Equivalence class partitioning

- A text field permits only numeric characters
- Length must be 6-10 characters long

Partition according to the requirement should be like this:

0 1 2 3 4 5 | 6 7 8 9 10 | 11 12 13 14

Invalid | Valid | Invalid

12

Identifying Test cases

- Assign a unique number to each equivalence class
- Until all valid equivalence classes have been covered by test cases, write a new test case covering as many of the uncovered valid equivalence classes as possible.
- Each invalid equivalence class cover by a separate test case.

13

Identifying Test cases

- A individual test case could not cover invalid cases.

For example, if the specification is

“enter book type (HARDCOVER, SOFTCOVER, or LOOSE)” and amount (1-999)”

The test case, XYZ, 0, expressing two error conditions (invalid book type and amount) will probably not exercise the check for the amount.

(Program may say “XYZ is UNKNOWN BOOK TYPE”, and not bother to examine the remainder of the input.

14

Question?

Consider that *wordCount* method takes a word *w* and a filename *f* as input and returns the number of occurrences of *w* in the text contained in the file named *f*. An exception is raised if there is no file with name *f*. Using the partitioning method, we obtain the following equivalence classes.

Equivalence Class	W	F
E1	Non-null	Exists, not empty
E2	Non-null	Does not exist
E3	Non-null	Exists, empty
E4	Null	Exists, not empty
E5	Null	Does not exist
E6	Null	Exists, empty

15

Question?

Consider an application that requires two integer inputs *x* and *y*. Each of these inputs is expected to lie in the following ranges:

$$3 \leq x \leq 7 \text{ and } 5 \leq y \leq 9$$

E1: $x < 3$ E2: $3 \leq x \leq 7$ E3: $x > 7$ (y ignored)
 E4: $y < 5$ E5: $5 \leq y \leq 9$ E6: $y > 9$ (x ignored) *Six equivalent Classes*

E1: $x < 3, y < 5$
 E2: $x < 3, 5 \leq y \leq 9$
 E3: $x < 3, y > 9$
 E4: $3 \leq x \leq 7, y < 5$
 E5: $3 \leq x \leq 7, 5 \leq y \leq 9$
 E6: $3 \leq x \leq 7, y > 9$
 E7: $x > 7, y < 5$
 E8: $x > 7, 5 \leq y \leq 9$
 E9: $x > 7, y > 9$ *Nine equivalent Classes*
 "X" "x" "Y"

16

Boundary-Value Analysis

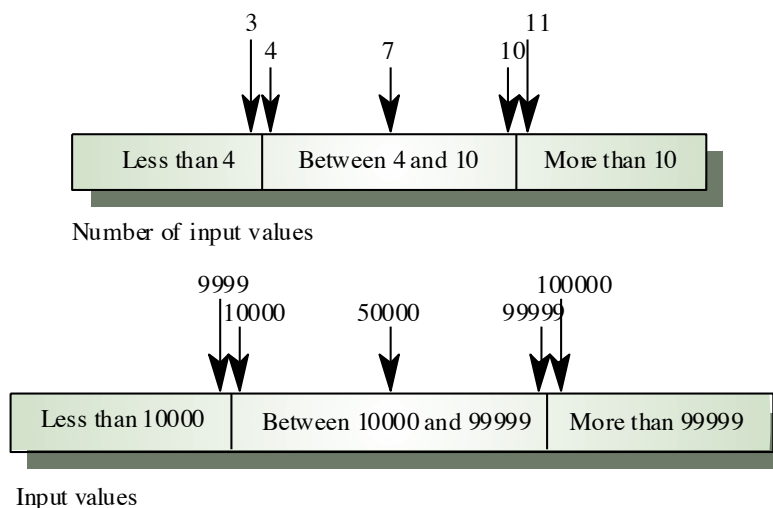
Boundary value analysis is a test selection technique that targets faults in applications at the boundaries of equivalence classes.

- Generally combined with Equivalence Class Partitioning
- Design test cases that exercise values that lie at the boundaries of an input equivalence class.
- Also identify output equivalence classes, and write test cases to generate o/p at the boundaries of the output equivalence classes.

Example: input condition $0 \leq x \leq \text{max}$

- Test for values : 0, 1, x, max-1, max (valid inputs)
- -1, max+1 (invalid inputs)

17



18

Triangle Problem

```
final int EQUILATERAL = 0;
final int ISOSCELES = 1;
final int SCALENE = 2;
final int INVALID = 3;
int triangle(int a, int b, int c)
{
    if (a >= b+c || b >= a+c || c >= a+b)
        return(INVALID);
    if (a == b && b == c)
        return(EQUILATERAL);
    if (a == b || a == c || b == c)
        return(ISOSCELES);
    return(SCALENE);
}
```

For each input values to represent a triangle, they must be integers greater than 0 where sum of two is greater than the third.

If we are defining, equivalence partitioning,
Condition: sum of two of the integers < third

Two Possible test cases: *Invalid (3-4-5) and Valid (1-2-4)*

19

Suppose, if an expression in the program were codes as $A+B \geq C$ instead of $A+B > C$, then program would tell us that 1-2-3 represents a valid scalene triangle.

???Error???

Hence, the important difference between the equivalence partitioning and boundary-value analysis is that BVA explores situations on and around the edges of the equivalence partitions.

20

Example 1

- Suppose you have very important tool at office, accepts valid User Name and Password field to work on that tool, and accepts minimum 8 characters and maximum 12 characters. Valid range 8-12, Invalid range 7 or less than 7 and Invalid range 13 or more than 13.

Invalid Partition	Valid Partition	Invalid Partition
Less than 8	8 - 12	More than 12

Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.

21

- Test Cases 1: Consider password length less than 8.
- Test Cases 2: Consider password of length exactly 8.
- Test Cases 3: Consider password of length between 9 and 11.
- Test Cases 4: Consider password of length exactly 12.
- Test Cases 5: Consider password of length more than 12.

22

Example 2

- Test cases for the application whose input box accepts numbers between 1 to 1000.
- Valid range 1-1000, Invalid range 0 and Invalid range 1001 or more.



Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.

Test Cases 1: Consider test data exactly as the input boundaries of input domain i.e. values 1 and 1000.

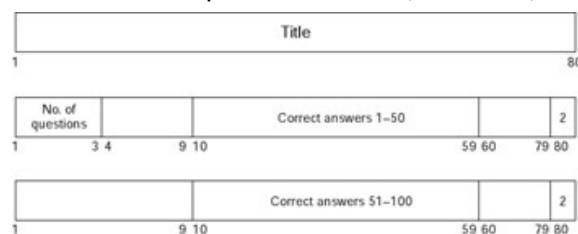
Test Cases 2: Consider test data with values just below the extreme edges of input domains i.e. values 0 and 999.

Test Cases 3: Consider test data with values just above the extreme edges of input domain i.e. values 2 and 1001.

23

Assignment - 01

MTEST is a program that grades multiple-choice examinations. The input is a data file named OCR, with multiple records that are 80 characters long. Per the file specification, the first record is a title used as a title on each output report. The next set of records describes the correct answers on the exam. These records contain a "2" as the last character. In the first record of this set, the number of questions is listed in columns 1-3 (a value of 1-999). Columns 10-59 contain the correct answers for questions 1-50 (any character is valid as an answer). Subsequent records contain, in columns 10-59, the correct answers for questions 51-100, 100-150, and so on.



24

The third set of records describes the answers of each student; each of these records contains a “3” in column 80. For each student, the first record contains the student’s name or number in columns 1-9 (any characters); columns 10-59 contain the student’s answers for questions 1-50. If the test has more than 50 questions, subsequent records for the student contain answers 51-100, 101-150, and so on, in columns 10-59. The maximum number of students is 200.

Student Identifier	Correct answers 1–50		3
1 9 10		59 60	79 80

	Correct answers 51–100		3
1 9 10		59 60	79 80

Student Identifier	Correct answers 1–50		3
1 9 10		59 60	79 80

25

The four output records are:

- A report, sorted by student identifier, showing each student's grade (percentage of answers correct) and rank.
- A similar report, but sorted by grade.
- A report indicating the mean, median, and standard deviation of the grades.
- A report, ordered by question number, showing the percentage of students answering each question correctly.

26