Week #5

Test Design Techniques

Dr. Vidhyacharan Bhaskar

Agenda

1. Design a good Test Case
2. Positive Test Cases and Negative Test Cases
3. Specific testing techniques
   - Equivalence Partitioning (EP)
   - Boundary Value Analysis (BVA)
   - Cause-effect graphing & Decision Table
   - Control-flow/Coverage Testing (Basis Paths Cyclomatic Complexity)

3. What is a good code?
4. Test Coverage

Design a good Test Case

Test Case

A test case is usually a single step, or occasionally a sequence of steps, to test the correct behavior/functionality, features of an application. An expected result or expected outcome is usually given.

A test case contains defined test conditions, the inputs, and the expected outputs or the expected behavior of the test object.

Sample: 5_Test-Cases-for-OrangeHRM-SoftwareTestingHelp.xls

(Appended with this file)

Design a good Test Case

- Avoid having test cases dependent upon each other (i.e. having preconditions of another test case passing).
- Write each test case so that it can reveal one type of fault.

Positive and Negative Test Cases

Valid Test Case (Positive test case)

A positive test case is when the test is designed to return what is expected according to the requirement.

Invalid Test Case (Negative test case)

A negative test case is when the test is designed to determine the response of the product outside of what is defined.

Negative testing is testing which attempts to shows that application or module does not do anything which is not supposed to do.

Positive and Negative Test Cases

- **Example**

Requirement:

The field shall accept numbers between 1 and 1,000.

(these tests assume that you submit the value to the application and all other input values are acceptable)

Positive and Negative Test Cases

Positive tests:

- enter 1
- enter 1,000
- (using equivalence classes, use 10 numbers between 1 and 1,000)

Positive and Negative Test Cases

Negative tests: (attempt to)

- enter 0
- enter 1001
- enter -
- enter -1
- enter 10000000
- enter 1 000 (notice the space)
- enter 1,000 (notice the comma)
- enter 10.0 (notice the decimal point)
- enter 10,0 (notice the comma not quite the same as the one above)

Positive and Negative Test Cases

Negative tests: (attempt to) (contd..)

- enter a
- enter A
- enter z
- enter Z
- enter o
- enter !
- enter $
- enter $100 (other currency symbols as well)
- enter your name
- leave the field blank (could be a positive test)
- enter the path to a text file containing only the number 10

Positive and Negative Test Cases

Case Study:

Positive Test cases for Email Address

Negative Test cases for Email Address

Positive and Negative Test Cases

1. Can any form of input to the program cause division by zero? Get creative!
2. What if the input type is wrong? (You are expecting an integer, they input a float. You are expecting a character, you get an integer.)
3. What if the customer takes an illogical path through your functionality?
4. What if mandatory fields are not entered?
5. What if the program is aborted abruptly or input or output devices are unplugged?

Equivalence Partitioning (EP)

- A Test design technique that divide input test data into partitions
- Once you have identified these partitions, you choose test cases from each partition. Test cases are designed to cover each partition at least once.
- To start, choose a typical value somewhere in the middle of (or well into) each of these two ranges.
- Equivalence Partitioning is a strategy that can be used to reduce the number of test cases that need to be developed.
- Equivalence partitioning uses fewest test cases to cover maximum requirements.

Equivalence Partitioning (EP)

- Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:

  1) One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be same. So one test case for valid input data should be sufficient.

  2) Input data class with all values below lower limit. i.e. any value below 1, as a invalid input data test case.

  3) Input data with any value greater than 1000 to represent third invalid input class.

Equivalence Partitioning (EP)

- EP Example – DATE (1 to 31)
- EP Example – Username (6 to 10 characters)
- EP Example – Age (18 to 80 Years except 60 to 65)

Boundary Value Analysis (BVA)

- Programmers often make mistakes on the boundaries of the equivalence classes/input domain. As a result, we need to focus testing at these boundaries.
- Boundary value is defined as a data value that corresponds to a minimum or maximum input, internal, or output value specified for a system or component.

Boundary Value Analysis (BVA)

1. If input conditions have a range from a to b (such as a=100 to b=300), create test cases:

- immediately below a (99)
- at a (100)
- immediately above a (101)
- immediately below b (299)
- at b (300)
- immediately above b (301)

2. If input conditions specify a number of values that are allowed, test these limits.

Boundary Value Analysis (BVA)

- BVA Example – DATE (1 to 31)
- BVA Example – Username (6 to 10 characters)
- BVA Example – Age (18 to 80 Years except 60 to 65)

Boundary Value Analysis (BVA)

- Run the equivalence class test cases first. If the program does not work for the simplest case (smack in the middle of an equivalence class), it probably will not work for the boundaries either.
- If you run a boundary test first, you will probably go run the general case (equivalence class test) before investigating the problem. So, instead just run the simple case first.

Cause-effect graphing & Decision Table

Cause-effect graph

The logical relationships between the causes and their effects in a component or a system.

- It must be possible to find the causes and effects from the specification.
- Every cause is described as a condition that consists of input conditions.
- The condition are connected with logical operators (AND, OR, NOT)

http://www.softwaretestinghelp.com/cause-and-effect-graph-test-case-writing-technique/

Cause-effect graphing & Decision Table

- **Example**

Cause-effect graph analysis for a ATM Conditions

- The bankcard is valid
- The PIN must be correctly entered
- The maximum number of PIN inputs is there
- There is money in the machine, and in the account

Cause-effect graphing & Decision Table

- Cause-effect graph analysis for a ATM

  Actions

  – Reject card
  – Ask for another PIN input
  – “Eat” the card
  – Ask for an alternate dollar amount
  – Pay the requested amount of money

Text book 5.1 pp. 135 - 138 (Figure 5-7, Table 5-10)

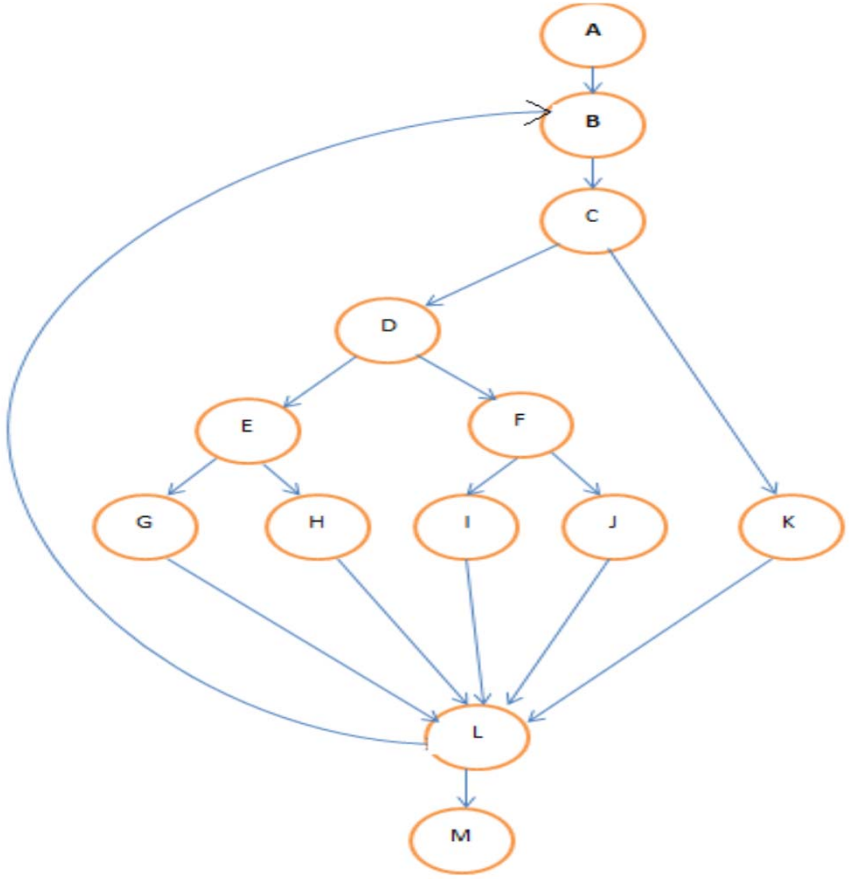
Cause-effect graphing and Decision Table

- Decision Table

Condition / Cause	TC1	TC2	TC3	TC4	TC5
Bankcard is valid	N	Y	Y	Y	Y
PIN is correct		N	N	Y	Y
3 incorrect PIN		N (exit)	Y		
Money available				N	Y
Effect / Action					
Reject card	Y	N	N	N	N
Ask new PIN	N	Y	N	N	N
Eat card	N	N	Y	N	N
Ask new amount	N	N	N	Y	N
Pay money	N	N	N	N	Y

Control-flow/Coverage Testing

- Control Flow Based Testing



Different Possibilities

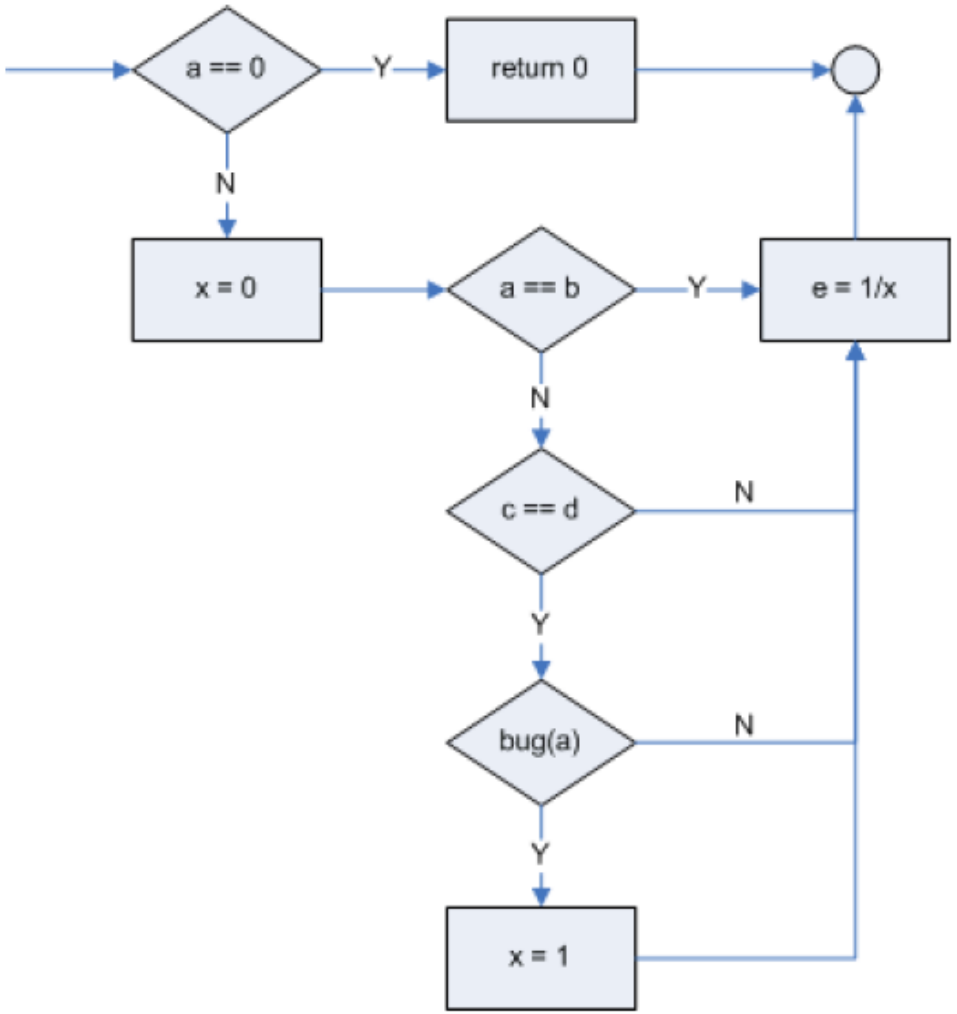
ABCDEGLM, ABCDEHLM,
ABCDFILM, ABCDFJLM, ABCKLM,
ABCDEGLBCDEGLM,
ABCDEGLBCDEHLM,
ABCDEGLBCDFILM,
ABCDEGLBCDFJLM,
ABCDEGLBCKLM,
. . .

5 x 5 = 25

Control-flow/Coverage Testing

```
1    int foo (int a, int b, int c, int d, float e)  {
2          float e;
3          if (a == 0)  {
4               return 0;
5          }
6          int x = 0;
7          if ((a==b) OR ((c == d) AND bug(a) )) {
8               x=1;
9          }
10          e = 1/x;
11    return e;
12     }
```

Control-flow/Coverage Testing



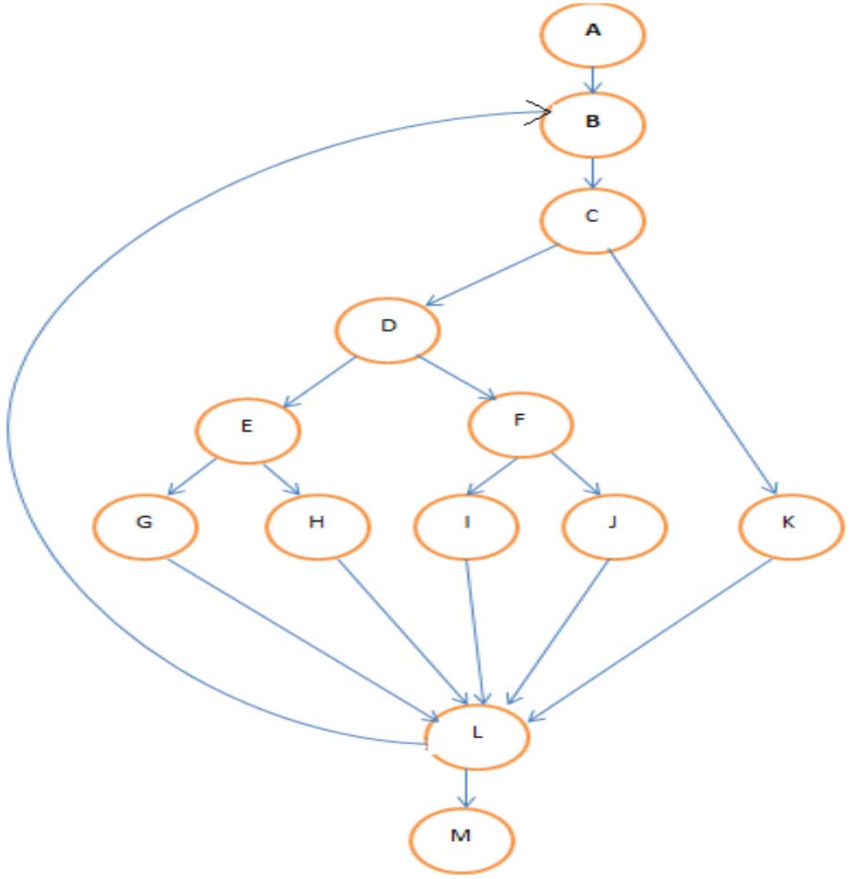
Control flow/Coverage Testing

Cyclomatic Complexity is computed using the control flow graph of the program:

The nodes of the graph correspond to indivisible groups of commands of a program, and a directed edge connects two nodes if the second command might be executed immediately after the first command.

Fundamentals of Testing

- Control Flow Based Testing



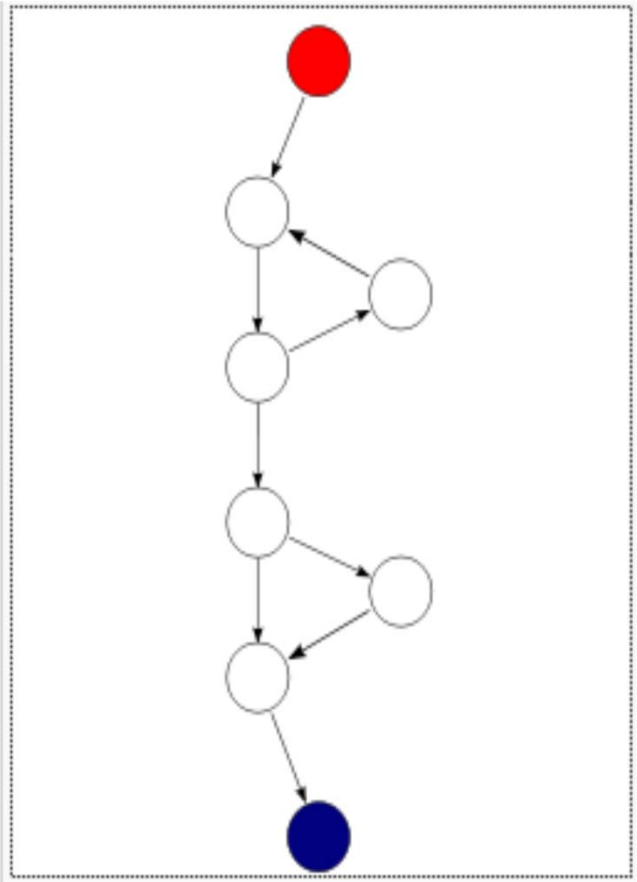
Different Possibilities

ABCDEGLM, ABCDEHLM,
ABCDFILM, ABCDFJLM, ABCKLM,
ABCDEGLBCDEGLM,
ABCDEGLBCDEHLM,
ABCDEGLBCDFILM,
ABCDEGLBCDFJLM,
ABCDEGLBCKLM,
. . .

Basis Paths: 5 x 5=25
Cyclomatic Complexity:
17 - 13 + (2*1) = 6

Control flow/Coverage Testing

Cyclomatic Complexity



M = E – N + 2P or
v(G) = e –n +2

V(G) = cyclomatic number of the graph G

e = number of edges in the control flow graph

n = number of nodes in the control flow graph

See text book p.101 also

What is a good code?

- A good code is code that works.
- The good code must not contain the defect or bug and is readable by other developers and easily maintainable.
- Organizations have coding standards all developers should follow, and also every programmer and software engineer has different ideas about what is best and what are too many or too few rules.
- We need to keep in mind that excessive use of rules can decrease both productivity and creativity. Peer reviews and code analysis tools can be used to check for problems and enforce standards.

Test Coverage

- How much testing is enough?
- When we will say the product is good to go?
- How we can assure the quality of the product is good?
- When we will have to stop testing?
- Is the 100% test coverage is important?
- Can we achieve 100% test coverage with in timeline?
- When we will say testing is sufficient?
- How do we measure the software testing completeness?

Test Coverage

- **Requirement coverage** that mean how many requirements are get validated? Can all requirements trace the respective test cases? Do all the test cases have passed?
- **Defect Coverage** in need to be known how many defect we found? How many defects has fixed?

Test Coverage

- **Code coverage** is all about evaluating the test cases against the actual code. We also do the code coverage by doing statement coverage, branch coverage, condition coverage, method coverage.
- **Test Case Coverage** a measure of how many of the product requirements are being tested by the defined test cases. It is the testers' job to define their test cases based on the requirements and the goal should be 100% coverage or close to that.

Test coverage Formula= (Total number of test cases executed/ Total number of test cases planned) x 100

Test definitions

- **Load test** - Performed to determine a system's behavior under both normal and anticipated peak load conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation.
- **Stress test** - When the load placed on the system is raised beyond normal usage patterns, in order to test the system's response at unusually high or peak loads, it is known as stress test.
- **Performance test** – It is a kind of physical test covering a wide range of Engineering or function evaluations where a material or a product is not specified by component specifications, but on final measurable performance characteristics.

Test definitions (contd..)

- **Volume test** - Volume testing refers to testing a software application with a certain amount of data. This amount can, in generic terms, be the database size or it could also be the size of an interface file that is the subject of volume testing.
- **Stability test** – In software testing, it is an attempt to determine if an application will crash. In pharmaceutical field, it is an attempt to know how well a product retains its quality over the life span of the product.
- **Documentation test** – Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results'. Documentation is as important to a product's success as the product itself. If the documentation is poor, non-existent, or wrong, it reflects on the quality of the product and the vendor.

Test definitions (contd..)

- **Web test –** Name given to software testing that is based on web applications. Complete testing of a web-based system before the system is revealed to the public.
- **Partial test** – It is a test on executable software which is only partially complete.
- **Interface test** – The main interfaces are:
  a. Web server and application server interface
  b. Application server and Database server interface.

  Check if all the interactions between these servers are executed properly. If database or web server returns any error message for any query by application server, then application server should catch and display these error messages appropriately to users.

Exercises

1. Read Text Book Chapter 5
2. Try to find answers of Chapter 5
3. Read about

   White BoxTesting

   Black BoxTesting

and try to find answers to Chapter Questions

Self-check Questions

1. What methodologies do you used to develop test cases?
2. What is a Test Case? What does it include?
3. What is Negative Testing?
4. What is difference between test effectiveness and test efficiency?
5. How to Estimate Testing effort?

Homework-3B

Read the following pseudo code:

```
if (input is in AllowedCharacterSet)
   if (input is a number)
         if (input >= 0)
              put input into positiveNumberList
         else
               put input into negativeNumberList
   else
         if (input is an alphabet)
               put input into alphabetList
         else
               put input into symbolList
else
   exception("Illegal character")
```

A. Draw a flow diagram that depicts the pseudo code. Label each node in the diagram with a unique alphabet.
B. What is the cyclomatic number of the program?
C. Identify each independent execution path in this program.