Software Testing Methodologies

BITS Pilani

Prashant Joshi

Module 4: Agenda

innovate achieve lead

Module 4: Specification Based Testing – (2/2)

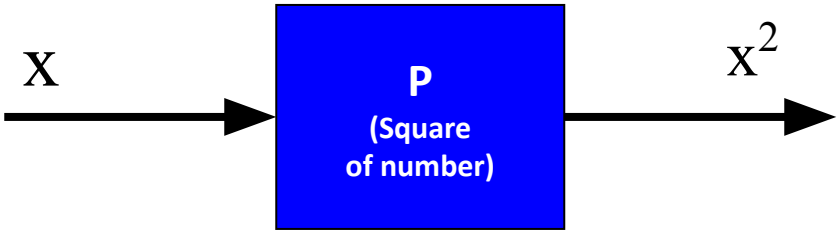
- **Topic 4.1** Domain Testing
- **Topic 4.2** Combinatorial
- **Topic 4.3** Decision Table Based Testing
- **Topic 4.4** Examples & Case Study

BITS Pilani
Pilani Campus

Topic 4.1: Domain Testing

The Concept

- A black box test technique
- Based on specifications
- Independent of implementation
- Focus
    - Functional testing
    - Behaviour
    - Input & corresponding output



Implementation may be,

A. Multiplication *(x*x)*
B. successive addition *(x+x… x times)*

Approaches & “View”

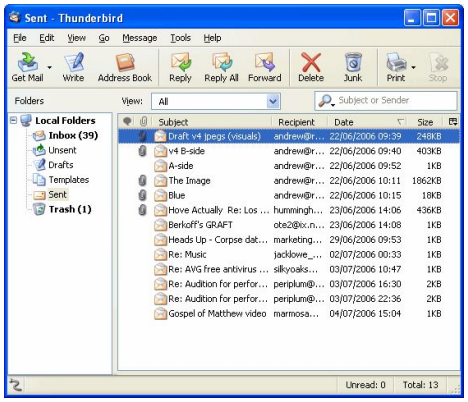
- Purpose is to uncover defects
- Demonstrate the system works (Treat this as a by-product!)
- Validate that it functions per specifications
- Works as specified – always!

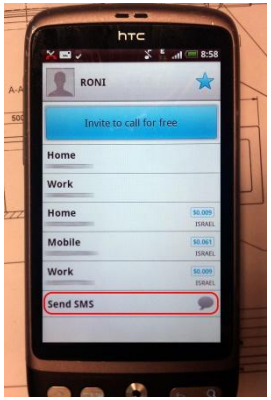
Perspectives

- Customer/Client
- Alpha/Beta User
- End User/Consumer
- Development Engineer
- Architect
- Product Manager
- Maintenance Engineer
- …

Examples

- Automated Teller Machine
- Tea/Coffee Vending Machine
- Washing Machine
- Contacts – Mobile Phone Application
- Messaging – Mobile Phone Application
- Email – Webmail/App/Client
- …





What is it?

- It is a systematisation of the Equivalence Partitioning and Boundary Value Analysis
- Introduces concept of Test Specification which allows the test engineer to take a closer look at the specifications
- Allows division of tasks for larger systems

Category Partitioning Method

- Systematic approach to generation of test cases from requirements
- Mix of manual and automated steps

- Ref: T2 Chapter 3 Section 3.5

CP Method Steps

1. Analyse Specification
2. Identify Categories
3. Partition Categories
4. Identify Constraints
5. (Re) write test specification
6. Process specification
7. Evaluate generator output
8. Generate test scripts

1: Analyse Specification

- Identify the functional unit that can be tested separately
- For larger systems, break down into subsystems, that can be tested independently

2: Identify Categories

- For each testable unit, analyse the specification and isolate inputs
- Identify objects in the environment
- Determine characteristics (category) of each parameter and environment object
  - Explicit characteristics
  - Implicit characteristics

3: Partition Categories

- Identify cases against which the functional unit must be tested (cases = choices)
- Partition categories into at least two subsets – correct values and incorrect values
  - Valid subdomain
  - Invalid subdomain

4: Identify Constraints

- Test for functional unit consists of a combination of choices for (a) parameter and (b) environment object
- Identify possible and not-possible combinations
- Thus, specify the constraints

5: (Re) write test specification

innovate achieve lead

- Based on previous steps a test specification can now be written
- Write the complete test specification
- Make use of a TSL which can help automate the process

6: Process Specification

- TSL or test specification is used to generate test frames
- Test frames are not test cases

7: Evaluate Generator Output

- Examine test frames for redundancy or missing cases
- Iterative process and steps

8: Generate Test Scripts

- Generate test cases from test frames
- Generate test scripts from test cases; typically a group of test cases

Software Testing Methodologies

BITS Pilani

Prashant Joshi

Topic 4.2: Combinatorial

Need of Combinatorial

- Software to be test in various environments
- Various combination factors
- Need for combinations of inputs
- Application
    - High Reliability needs
        - » Work on various configurations (Windows, Mac, Linux)
    - Interoperability
        - » Various Clients & Servers (MMS)

Modelling The Input

- Program P
- Input variables: X and Y
- X can take one value from {a, b, c}
- Y can take one value from {d, e, f)
- Leads to:
- 9 factor combinations (3^2)

- For large number of variables and values of each variable these combinations can be very large
- If we image one test case per combination; we will have a huge number of test cases

Model the input

- Fault Model (Interaction Faults)
    - Two or more variables play a role
    - One or more specific values play a role
- Unique combinations
    - Latin Squares
    - Pairwise Testing
    - Orthogonal Array

In this module we focus on the use of combinatorial alone; Focus on the reduction and specific techniques for optimization will be discussed in later Module

Software Testing Methodologies

BITS Pilani

Prashant Joshi

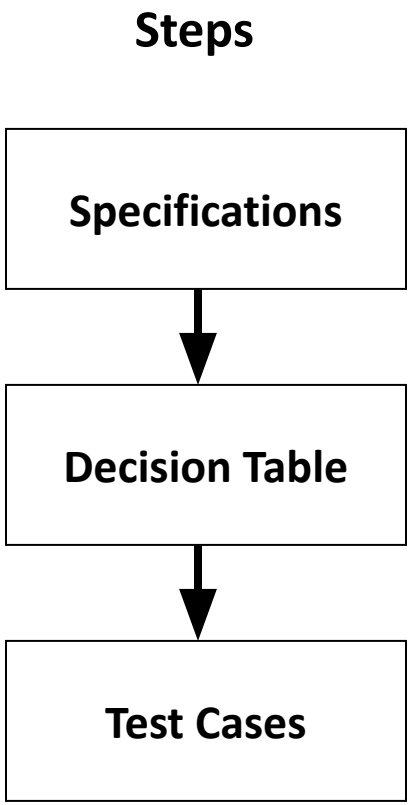
innovate achieve lead

BITS Pilani

Pilani Campus

Topic 4.3: Decision Table Based Testing

Decision Tables



- One of the most systematic approaches of designing Test Cases
- Decision Tables are rigorous – enforce logical rigor
- Related Method – Cause Effect Graphing
- Ideal for situations with number of combinations of actions are taken under varying sets of condition
- Structure of describing rules

DT Technique

Stub	Rule 1	Rule 2	Rule 3, 4	Rule 5	Rule 6	Rule 7, 8
C1	T	T	T	F	F	F
C2	T	T	F	T	T	F
C3	T	F	---	T	F	---
A1	X	X		X		
A2	X				X	
A3		X		X		
A4			X			X

T: True
F: False
---: Don't Care
<blank>: Not applicable
X: Action takes place

Notion of completeness

- N conditions □ 2^N rules
- Actions can be defined by user

DT - Example

Component Specification: Input of 2 characters such that

1. The 1st character must be A or B.
2. The 2nd character must be a digit.
3. If the 1st character is A or B and the 2nd character is a digit the file is updated.
4. If the 1st character is incorrect, message X12 is displayed.
5. If the 2nd character is not a digit, message X13 is displayed.

Develop test cases using Decision table technique

DT Example Solution

Stub	R1	R2	R3	R4	R5	R6	R7	R8
C1: 1st Char is A	T	T	T	T	F	F	F	F
C2: 1st Char is B	T	T	F	F	T	T	F	F
C3: 2nd char is Digit	T	F	T	F	T	F	T	F
A1: File is updated			X		X			
A2: Message X:12							X	X
A3: Message X:13				X		X		X
A4: Impossible	X	X						

Each Rule one Test Case (minimum)

Test Case #	Input	Rule
Test Case 1	A5	R3
Test Case 2	AC	R4
Test Case 3	B5	R5
Test Case 4	BC	R6
Test Case 5	C5	R7
Test Case 6	CD	R8

Note that it may not be possible to execute or even design test cases for the impossible action rule. But to begin with never ignore these conditions. Design them and then evaluate if they can be included in the Test Suite.

Software Testing Methodologies

BITS Pilani

Prashant Joshi

innovate achieve lead

BITS Pilani

Pilani Campus

Topic 4.4: Examples & Case Study

Test Techniques

- Equivalence Class
- Boundary Value Analysis
- Domain Partitioning
- Combinatorial
- Decision Table

Examples – Solve them

innovate achieve lead

TV Remote Control Software

- Software is designed for the remote control which has typical controls and options. Using DT design and develop test cases

A web based shopping cart checkout logic

- By number of items
- By weight of items
- By value of items

Examples – Solve them

An insurance renewal premium

- By Age
- By Number of claims
- By Value of claims

Folder and File Name generation for a Digital Camera

- Use of a Serial number
- Use of Date
- Use of Date+time

Browsers on Operating Systems

- Browsers (Firefox, Chrome, Safari and MyBrowser) are to be tested on various operating systems (Windows, Linux) and on various platforms like PC, Mobile. On Mobile phones the operating systems to be considered are (iOS, WP8, Android)

- Come up with a strategy and test cases

School Attendance System

- An attendance sub-system is developed which is part of a School Management System. At the beginning of the year the Class teacher creates the roster for the class by pulling in the details from Master Data Base. The fields that teacher uses are Full Name and Class/Division. Over that she builds a local database with entry of Nickname, Months of attendance, Attendance, Reasons for absence in case of absence (Sick, Informed Leave, School program, Competitions). Following reports are generated at the end of the week:
  - Classes attendance report
  - Student attendance report
  - Absenteeism report
  - Student absenteeism report
- Design a set of test cases to test such a sub-system

Software Testing Methodologies

BITS Pilani

Prashant Joshi