Chapter-6

Test Design Techniques:

6.1 The test Development Process:

Test Design
- It is the act of creating and writing test suites for testing a S/w.

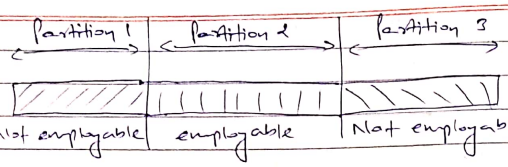
Test Design Technique:
- It helps to select a good set of tests from the total number of all possible tests for a given system.
- There are many different types of S/w testing technique each with its own strengths and weakness.
- Each individual technique is good at finding a particular types of defect and relatively poor at finding other types.
  eg: component testing is more likely to find coding logic defects rather than system design defects.

6.2 categories of test design techniques:
Two main categories:
1) Static technique
2) Dynamic technique
   i) specification based or Black Box Techniques
      - Equivalence partitioning
      - Boundary value Analysis
      - Decision table testing
      - Use case testing
      - State transition testing
   ii) Structure based techniques / white box
      - Statement testing and coverage
      - Decision Testing and coverage
      - Other structure based Techniques
   iii) Experience based Techniques.

i) Specification based Technique / BBT

a) Equivalence Partitioning
- In ~, the input space is partitioned into valid inputs and invalid inputs.
- Here we need to test only one condition for each partition.
- This is because we assume that all the conditions in one partition will be treated in the same way by the software.
- If one condition in a partition works, we assume all of the conditions in that partion will work.
- If one of the condition in a partition doesnot work then we assume that none of the conditions in that partition will work.

Example:
The following examples illustrates this technique
- In human reource application, employees are required within the range of 16-65.
- So, the valid value is between age 16-65 and there are two partitions of invalid value; one partition is below 16 and other partition is above 65.
- Therefore there are three partitions for this case
  - one valid and two invalid
- one test case can be designed for each partition resulting in three test cases.



Age less than 16 | Age between 16 and 65 | Age more than 16.

- If the input condition specify a range of values (such as 16-65), there are three partitions.
- If the input condition specify a single value (such as 50), there will be three partitions. one valid partition, one invalid partition above the valid value and one invalid partition below the valid value.
- If the input condition specify a boolean value, there will be two partitions, one valid partition and one invalid partition.
- If the input condition specify a set of valid values (such as 20, 25, 30) there will be one partition for each of the valid values and one invalid partition for an invalid value.

b) Boundary Value Analysis

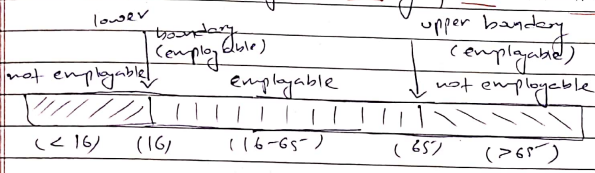
- BVA is based on testing at the boundaries between partitions.
- Here the boundaries can be either valid or invalid
- In valid boundaries, boundary values are valid value.
- In invalid boundaries, boundary values are invalid value.
- When BVA and EP are combined, then better test case can be designed.

Example:

In the above example, there are two boundaries, the minimum employable age 16 and maximum employable age 65.

These two boundaries should be accepted, and all values above 65 and below 16 must be rejected. Therefore test cases that combine the techniques of equivalence partitioning and BVA are as follows:

1) one value between 16 and 65 - valid value
2) One value at the lower boundary of 16 - valid value
3) one value at the higher value boundary of 65 - valid value
4) one value below the lower boundary of 16 - invalid value
5) one value above the higher boundary of 65 - invalid value



c) Decision Table:

- The technique of EP and BVA often deals with specific situation of inputs.
- However different combinations of input results in different actions being taken, this can be more difficult to show using EP and BVA because they are more focused on interface.
- So decision table (& state transaction testing) are used which are more focused on business logic
- It provides systematic way of stating complex business rules.
- It helps to test all possible combination of conditions / inputs

- It is always better to prioritize and test the most important conditions.

Condition	Rule 1	Rule 2	Rule 3	Rule 4
New customer (15%)	T	T	f	F
Coupon (20%)	T	F	T	F
Action				
Discount (%)	35%	15%	20%	0%

fig: Decision table with combination of outcomes

d) State Transition Testing

- finite state machine means that the system can be in a finite number of different states & the transition from one state to another are determined by the rules of the machine.

- A state transition model has 4 basic parts
  1. States that the s/w may occupy.
  2. transition from one state to another.
  3. events that cause a transition.
  4. actions that result from a transition.

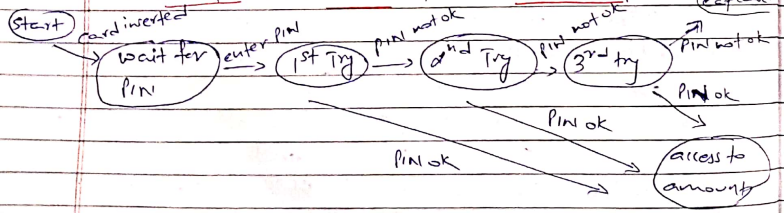


fig: State transition diagram for PIN entry

Here, we can see that in any given state, one event can cause only one action but that the same event from different state may cause a different action & a different end state.

The state diagram shows 4 states, but only four possible events (card inserted, enter PIN, PIN ok and PIN not ok)

In deriving test cases, we may start with such the possible scenario.

- first test case here would be the normal situation, where the correct PIN is entered the first time.
- A second test (to visit every state) would be to enter an incorrect PIN each time, so that the system eats card.
- A third test we can do where the PIN was incorrect the first time but ok the 2nd time and another test when the PIN was correct on the third try.

e) Use case Testing:

- It is a technique that helps us identify test cases that execise the whole system on a transaction by transaction basis from start to finish.
- Each use case describes the interactions the actor has with the system in order to achieve a specific tasks
- Actors are generally people but they may also be other systems.
- use cases are defined in terms of actor not the system describing what the actor does or what the actor sees rather than what the system expects and what the system's outputs.

- They serve as the fundamental foundation for developing test cases mostly at the system and acceptance testing levels.
- Use cases can uncover integration defects.
- Use cases must specify any preconditions that need to be met for the use case to work.
- Use cases must also specify post conditions that are observable results and a description of the final state of the system.

	Step	Description
Main Success Scenario	1	A: Insert card
A: Actor	2	S: Validates card & asks for PIN
S: System	3	A: enters PIN
	4	S: validates PIN
	5	S: Allow Access to Account
Extensions	2a	Card Not valid S: Display message & reject card
	4a	PIN not valid S: Display message & ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat Card and exit

ii) Structure Based or White Box Testing:
- Structure Based testing techniques use the internal structure of a software to derive test cases.
- They are commonly called 'White Box' or 'glass box' technique
- They can be used in all levels of testing (e.g: unit testing, integration testing, system testing and acceptance testing)

Decision coverage is stronger than statement coverage, 100% decision coverage guarantees 100% statement coverage but not vice versa.

Decision coverage = $\frac{\text{No. of all decision outcomes covered by test case}}{\text{total no. of all possible decision outcomes in the code}} \times 100\%$

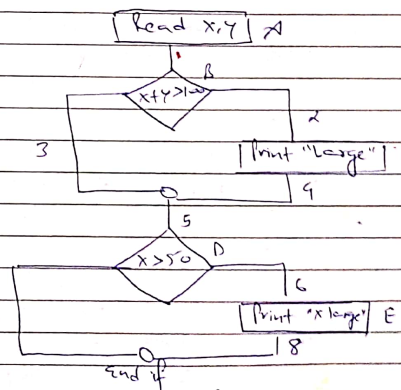
- Structure based test design techniques are a good way to help ensure more breadth of testing.
- To measure what percentage of code has been exercised by a test suite, one or more coverage criteria is used.
- A coverage criteria is usually defined as a rule or requirement a test suite needs to satisfy.

Coverage = $\frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$

There are number of coverage criteria:
i) Statement Coverage
ii) Decision (Branch) coverage
iii) ~~Path~~ other Coverage.

for example:

```
Read X, Y
If X+Y>100 THEN
Print "Large"
END IF
IF X>50 THEN
Print "X Large"
END IF
```



Nodes (▭ & ◇) represent statement of code
edges (—) represent links between nodes

It is the assignment of executable statements that has been exercised by a test case suite. / percentage of

a) Statement coverage:

- Statement coverage is a white box testing technique where all the statements of the source code are executed atleast once.
- To calculate statement coverage, find out the shortest number of paths following which all the nodes will be covered.

In the above example, in case of "Yes", while traversing through each statement of code and the traversing path ~~(A-B)~~ (A1-B2-C4-5-D6-E8) all the nodes are covered.

So, by traveling through only one path all the nodes (A, B, C, D and E) are covered.

∴ Statement coverage (SC) = 100 %.

Statement coverage = (no. of executable statements covered by test case / total no. of all executable statements in the code under test) × 100%

b) Branch Coverage / Decision

- Branch coverage is a white Box testing technique which ensures that every possible branch from each decision point in the code is executed atleast once.
- To calculate branch coverage find out the minimum number of paths which ensures covering all of all edges.

In the above examples, in case of traversing through a 'Yes' decision path (A1-B2-C4-5-D6-E8), maximum number of edges (1, 2, 4, 5, 6 and 8) are covered but edges 3 and 7 are left out. To cover these edges, we have to follow (A1-B3-5-D7). So by travelling through two paths (Yes, Yes) all the edges (1, 2, 3, 4, 5, 6, 7, 8) are covered.

Branch Coverage / Decision coverage (BC) = 2

Decision coverage: assignment of % of decisions that have been covered / exercised by a test suite.

c) Path coverage:

Path coverage is a white box testing technique which ensures covering all the paths from begining to end.

In the above example, All the possible paths are:

A1-B3-5-D7

A1-B2-C4-5-D6-E8

A1-B2-C4-5-D7

A1-B3-5-D6-E8

path coverage (PC) = 4

c) Other structure-Based testing

A1-B2-C4-5-D6-E8 ⇒ condition coverage and multiple condition coverage.

iii) Experience Based Testing technique.

- Here, people's knowledge, skill and background are of prime importance to the test conditions and test cases.
- The experience of both technical and business people is required, as they bring different perspectives to the test analysis and design process.
- Because of the previous experience with similar systems they may have an idea as what could go wrong which is very useful for testing.
- Experienced-based techniques go together with specification based and structure-based techniques and are also used when there is no specification or if the specification is inadequate or out of date.
- This may be the only type of technique used for low-risk systems, but this approach may be particularly useful under extreme time pressure.

Error Guessing:

- The error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.

- The success of error guessing is very much dependent on the skill of the tester as good testers know where the defects are mostly like to occur

Inf

6.6 Choosing Test Techniques:

Q How to choose which testing technique is best?

- Each technique is good in its own way in finding out the certain kind of defect, and not as good for finding out the other kind of defects.
- So, how to choose which testing techniques is best decision will be based on a number of factors, both internal and external.

Internal factors:

1) Development model used in developing the system will govern which testing technique can be used.

2) How much testers know about the system and about testing techniques will influence the choice of testing technique.

3) knowledge of similar kind of (defects / projects) will be very helpful in choosing techniques.

4) The testing objectives decides how much detailly the testing should be carried out and helps to select the appropriate testing technique.

5) The existence of documentation and the content and style of documentation influences the selection of testing technique.

6) The lifecycle model used (either sequential or incremental) influence the selection of testing technique

External factors:

1) Risk assesment: The greater the risk, the greater the need for more thorough and more formal testing.

2) Customer and contractual requirements:
   - Sometimes contracts specify particular testing techniques to use.

3) Type of system used:
   - The type of system (embedded, financial, graphical etc) will influence the choices of techniques.

4) Regulatory requirements:
   - Some industries have regulatory standards or guidelines that govern the testing techniques uses.

5) Time and Budget of the project:
   How much time and budget is available for testing guides the selection of testing techniques to be used