Software Testing Methodologies

Test Case Design Techniques

Categories of Test Design Techniques

Specification based:

- Use models from which test cases can be derived Based on functional/non-functional requirement
- Also called black box

Structure based:

- Information about how the software constructed is used to derive the test cases
- Also called White box or glass box

Experience based:

- Knowledge and experience of people is used to derive test cases.
- Knowledge of software, its usage, it's environment and its distribution is required for deriving test cases

Specification Based Techniques

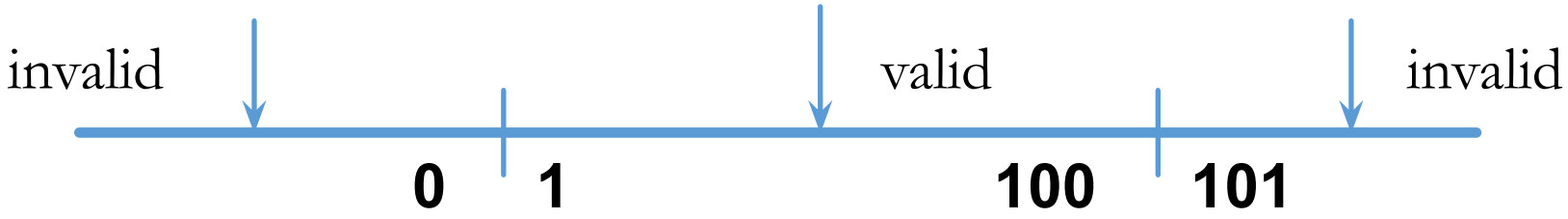
- Also called functional/behavioral/requirement based testing.
- Focus on evaluating the function
- Determine whether the combinations of input and operations produce expected results

Types of Specification Based Techniques

- Equivalence partitioning
- Boundary value analysis
- Decision table testing
- State transition testing
- Use case testing

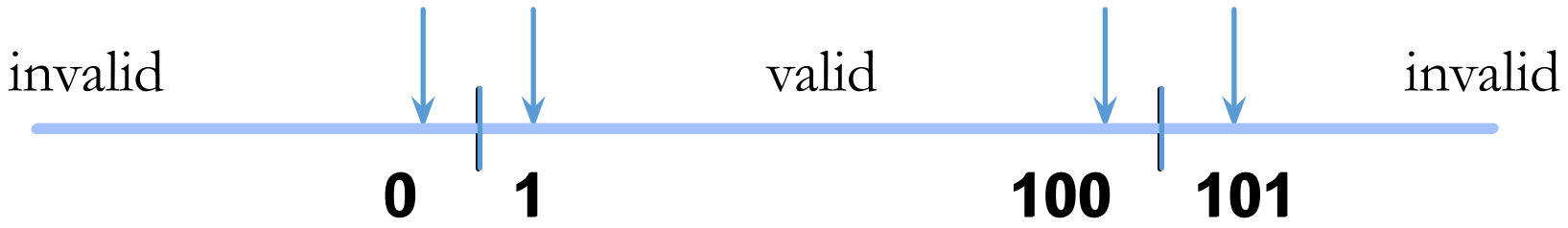
Equivalence Partitioning

- Equivalence partitioning defines set of valid and invalid sates for input conditions
- Divide the inputs and outputs into equivalence classes
- Equivalence partitioning is applicable at all levels of testing



Boundary Value Analysis

- Defect tend to lurk near boundaries
- Test values on either side of valid boundary
- BVA enhances equivalence partitioning by selecting element just on, and just beyond the border of each class
- Applies to all levels of testing

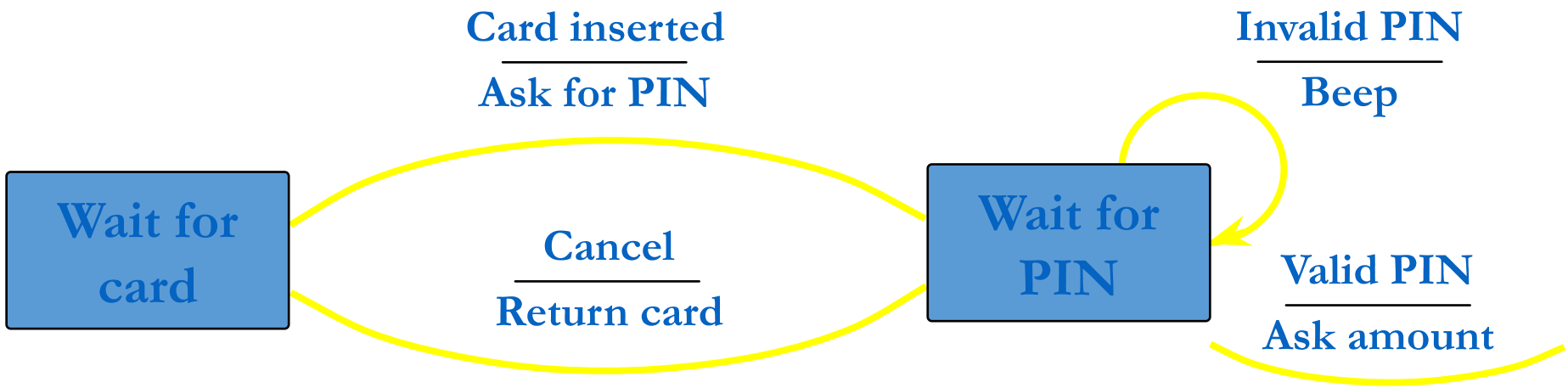


Decision Table Testing

- A table containing combinations of true and false for all input conditions and resulting actions for each
- Each column corresponds to a business rule that defines a unique combination of conditions that result in execution of the actions associated with that rule
- Requirement that contain logical conditions can be captured
- At least on test case is written per column

State Transition Analysis

- State transition diagram indicates how the system behaves as a consequence of external events
- It serves as the basis for behavioral modeling



Use Case Testing

- Use case defines goal oriented set of interactions between external actors and system under consideration.
- Use cases capture who (Actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals.
- It describes the process flows through the system based on it actual likely use.

Structure Based Techniques

- Also called logic driven techniques
- Use specific knowledge of programming code to examine outputs and assumes that the tester knows the path of logic in a unit or program
- Carried out during the coding an unit testing phase
- Evaluate that all aspects of the structure have been tested and that the structure is sound

Statement Testing

Statement:

- An entity in programming language which is typically the smallest indivisible unit of execution.

- A statement which when compiled is translated into object code and which will be executed procedurally when program is running and may perform an action on data.

Statement Coverage:

- In this the test case is executed in such a way that every statement of the code is executed at least once.

Branch/Decision Testing

Branch:

- A basic block that can be selected for execution based on a program construct in which one of two or more alternative program paths are available.

Branch / Decision Coverage:

- Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once i.e. every branch (decision) taken each way, true and false.

- It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.

Path Testing

Path Coverage:

- In this the test case is executed in such a way that every path is executed at least once.

- All possible control paths taken, including all loop paths taken zero, once, and multiple (ideally, maximum) items in path coverage technique, the test cases are prepared based on the logical complexity measure of a procedural design.

Other Structure-Based Techniques

- There are stronger levels of structural coverage beyond decision coverage, for example, condition coverage and multiple condition coverage

- The concept of coverage can also be applied at other test levels For example, at the integration level the percentage of modules, components or classes that have been exercised by a test case suite could be expressed as module, component or class coverage

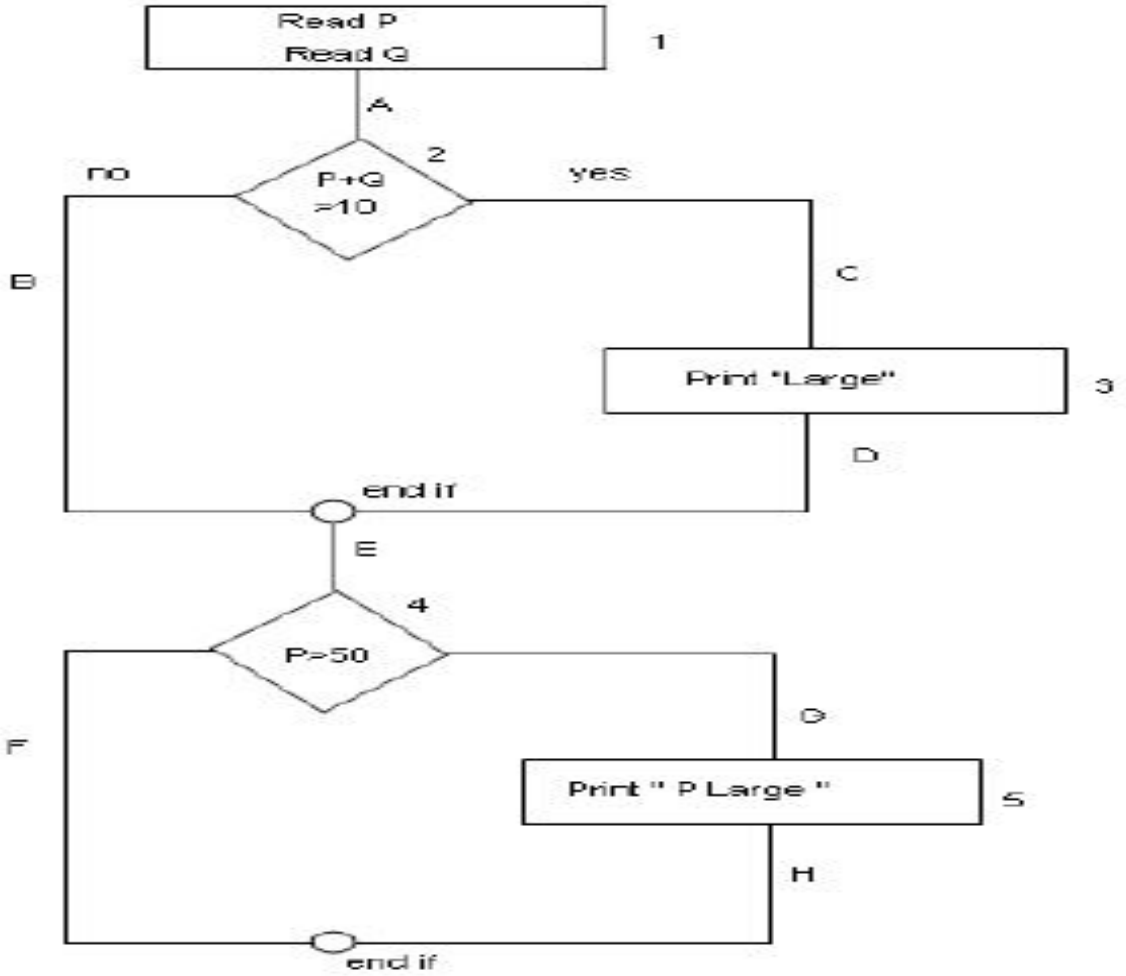
How to Calculate SC, BC and PC?

- Nodes represent entries, exits, decisions and each statement of code.
- Edges represent non-branching and branching links between nodes.

Example- Code

```
Read P
Read Q
IF P+Q > 100 THEN
Print “Large”
ENDIF
If P > 50 THEN
Print “P Large”
ENDIF
```

Example- Flowchart



How to Calculate Statement Coverage?

- To calculate Statement Coverage, find out the shortest number of paths following which all the nodes will be covered.

- Here by traversing through path 1A-2C-3D-E-4G-5H all the nodes are covered. So by travelling through only one path all the nodes 12345 are covered.

- So the Statement coverage in this case is 1.

How to Calculate Branch Coverage?

- To calculate Branch Coverage, find out the minimum number of paths which will ensure covering of all the edges. In this case there is no single path which will ensure coverage of all the edges at one go.

- By following paths 1A-2C-3D-E-4G-5H, maximum numbers of edges (A, C, D, E, G and H) are covered but edges B and F are left. To covers these edges we can follow 1A-2B-E-4F. By the combining the above two paths we can ensure of travelling through all the paths.

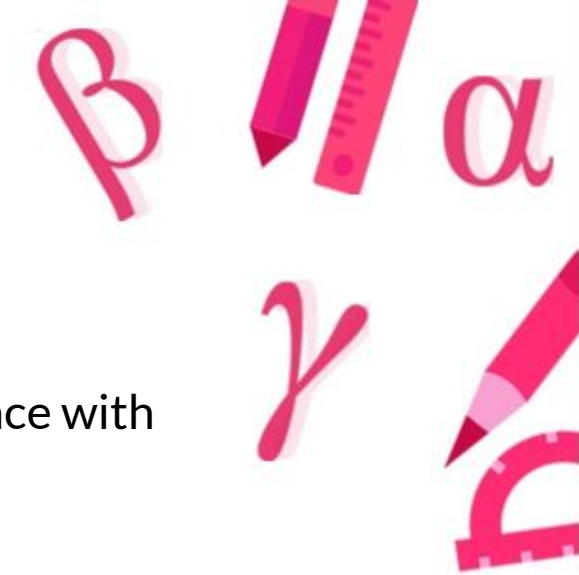
- Hence Branch Coverage is 2. The aim is to cover all possible true/false decisions.

How to Calculate Path Coverage?

- Path Coverage ensures covering of all the paths from start to end.
- All possible paths are-
  - 1A-2B-E-4F
  - 1A-2B-E-4G-5H
  - 1A-2C-3D-E-4G-5H
  - 1A-2C-3D-E-4F
- So path coverage is 4.

Memorize these…

- 100% Path coverage will imply 100% Statement coverage.
- 100% Branch/Decision coverage will imply 100% Statement coverage.
- 100% Path coverage will imply 100% Branch/Decision coverage.
- Branch coverage and Decision coverage are same.



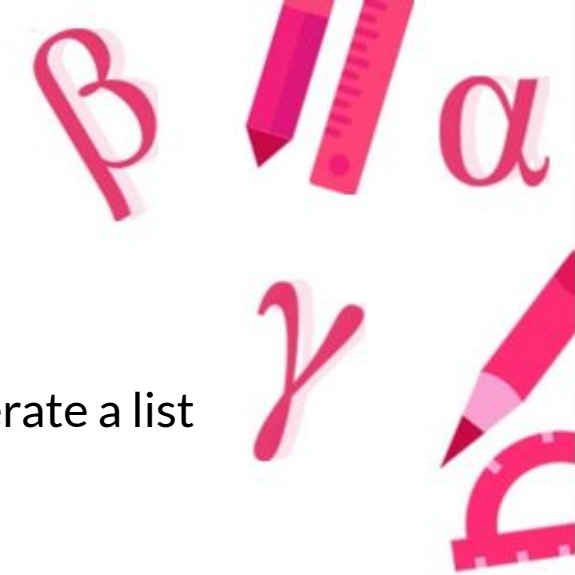
Experience Based Techniques

- Tests are derived from tester's skill and intuition and their experience with similar applications and technologies.
- Used to augment systematic techniques to identify special tests not easily captured by formal techniques.
- May yield widely varying degrees of effectiveness, depending on tester's experience.

Exploratory Testing

- Concurrent test design, test execution, test logging and learning, based on tester's charter containing test objectives, and carried out within time boxes.

- An approach that is most useful where there are few or inadequate specifications and severe time pressure, or in order to augment or complement other, more formal testing.



Error Guessing

- A structured approach to the error guessing technique is to enumerate a list of possible errors and to design tests that attack these errors.

- Error guessing by an experienced tester is probably the single most effective method of designing tests that uncover bug.

- A well placed error guess can show a bug that one could easily miss by many of the other test case design techniques. Conversely in the wrong hands error guessing can be waste of time.

Thankyou!