Chapter 3 Testing Techniques

SE4930 Software Testing

Testing Techniques

- “I am for those means which will give the greatest good to the greatest number” – Abraham Lincoln

Topics covered

- General Testing techniques
- Functional Testing techniques (to meet functional requirements)
- Nonfunctional Testing techniques (to meet non functional requirements)

General Testing Techniques

- Positive and negative testing
- White box and black box testing
- Experienced based testing or error guessing
- Automated software testing

Positive and Negative Testing

- Positive testing
  - Verify that a system conforms to its stated requirement]
- Negative testing
  - Demonstrate that a system does Not do what it is not supposed to do

Positive and Negative Testing

- A Compact Disk(CD) Player can be in one of three status: Standby, On or Playing
- When in Standby mode, the CD Player can be turned on by pressing the Standby button once (an indicator light turns from red to green to show the CD Player is On)
- When the CD Player is On, it can return to Standby mode by pressing the Standby button once (an indicator light turns from green to red to show the CD Player is in Standby mode)
- When the CD Player is on, Pressing the Play button causes the currently loaded CD to play. Pressing the Stop button when the CD Player is playing a CD causes the CD Player to stop playing the disk

Positive and Negative Testing

- Examples of positive tests could include:
- Verifying that with the CD Player in Standby mode, pressing the Standby button causes the CD Player to turn on and the indicator light changes from red to green
- Verifying that with the CD Player in the On state, pressing the Standby button causes the state of the CD Player to change to Standby and the indicator light changes from green to red
-

Positive and Negative Testing

- Examples of negative tests could include:
- Investigating what happens if the CD Player is playing a CD and the Standby button pressed
- Investigating what happens if the CD Player is On an the Play button is pressed without CD in the CD Player

Positive and Negative Testing

Positive Testing	Negative Testing
Requirements specification document	Not documented or poorly documented
	Often ended technique

General Testing Techniques

- Positive and negative testing
- White box and black box testing
- Experienced based testing or error guessing
- Automated software testing

White Box and Black Box Testing

White Box and Black Box Testing

- Rely on the level of knowledge the test analyst has of the internal structure of the software

```
public int max(int a , int b){
    int max;
    if( a > b ) {
        max = a;
    }else{
        max = b;
    }
}
```

General Testing Techniques

- Positive and negative testing
- White box and black box testing
- Experienced based testing or error guessing
- Automated software testing

Experienced Based Testing

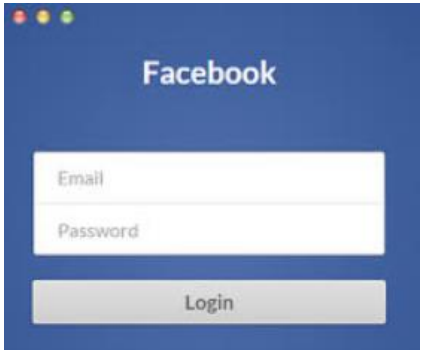
- Experienced based testing is not in itself a testing techniques, but rather a skill that can be applied to all of the other testing techniques to produce more effective tests

  - Knowledge about the AUT
  - Knowledge of the results of any earlier testing phases
  - Experience of testing similar or related systems
  - Knowledge of typical implementation errors
  - General testing rules of thumb or heuristics

Experienced Based Testing

- Needs imagination and thinking about how to break a system

- Having previous experience is an asset
  - E.g., working with link list
    1. Add node
    2. Delete first node and add a node as a first one
    3. Delete last node and add a last node
    4. Delete all nodes and then add them again



General Testing Techniques

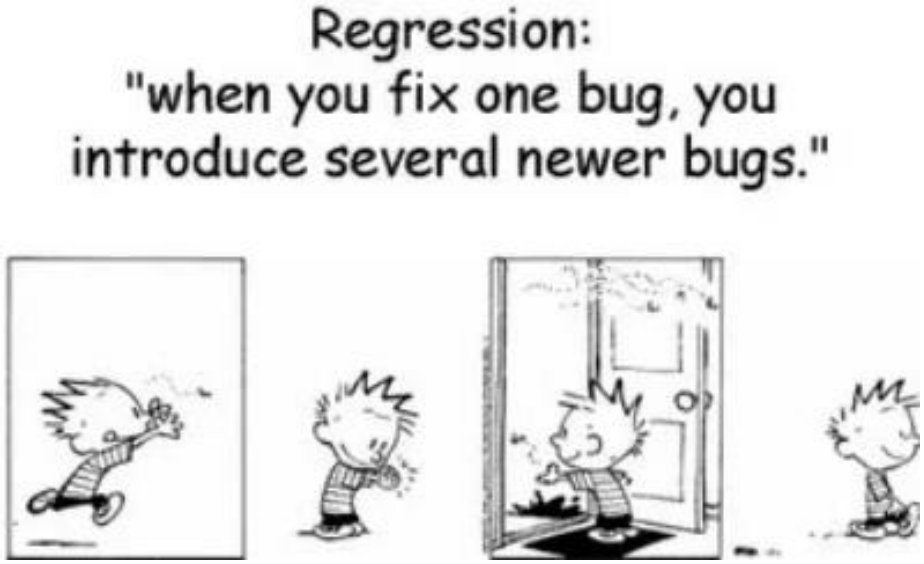
- Positive and negative testing
- White box and black box testing
- Experienced based testing or error guessing
- Automated software testing

Automated Software Testing

- Not a testing technique
- Provides significant gains in productivity for the testing task.
- Regression testing
- Not be appropriate for all testing tasks.
- Selenium

Automated Software Testing

- Regression Testing



Automated Software Testing

- Example of where automated testing tools may be particularly appropriate

- Testing of applications that have a rapid build and release cycle
- Testing of applications that have to be delivered across diverse platforms
- Testing of applications with complex GUI and/or client server architectures
- Testing of applications that require thorough, rigorous, repeatable testing
- Where there is a need to reduce testing timescales and effort
- Where there is a need to do more thorough testing in the same timescales

Quiz

- 1. White box testing or Black box testing

- 2. Design test case by using experienced based testing (error gussing)

Functional Testing Techniques

- To confirm that the application under test (AUT) meets its functional requirements

- Equivalence partitioning
- Boundary value analysis
- Static testing

Functional Testing Techniques

- To confirm that the application under test (AUT) meets its functional requirements

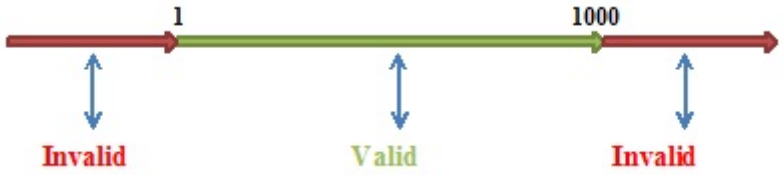
- Equivalence partitioning
- Boundary value analysis
- Static testing

Equivalence partitioning

- Inputs and outputs from the AUT can be grouped or partitioned into coherent groups or classes
- Testing on instance of the class is equivalent to testing all of them

- Testing for input box accepting numbers from 1 to 1000
  - Invalid partition : … ~ 0
  - Valid partition : 1 ~ 1000
  - Invalid partition : 1000 ~ …

- Pick a single value from each partition



Equivalence partitioning

- The Testing problem
  - The specification for a software system for validating expenses claims for hotel accommodation includes the following requirements
  - There is an upper limit of $90 for accommodation expense claims
  - Any claims above $90 should be rejected and cause an error message to be displayed
  - All expense amounts should be greater than $0 and an error message should be displayed if this is not the case

Equivalence partitioning

- Test case table

Test Case ID	Hotel Charge	Partition	Output
1	50	0< charge <=90	OK
2	-25	<= 0	Error Message
3	99	> 90	Error Message

Equivalence partitioning

- Height of an object
- Name of a city
- File name argument to Unix *lpr* command
  - (The **lpr** command is used to submit print jobs in Linux)

Equivalence partitioning

- Height of an object
  - Valid: {x|x > 0}, Invalid: {x| x <= 0}, Invalid: {x| x is not a number}
- Name of a city
  - Valid: {x| x is alphabetic}, Invalid: {x|x is alphanumeric}, Invalid: {x| everything else}
- File name argument to Unix *lpr* command
  - Valid: {x| an existing file}, Invalid: {x| everything else}
- Usually number of invalid sets are greater than number of valid sets

Equivalence partitioning

- If an input condition specifies a *range*, one valid and two invalid equivalence classes are defined
  - Height of an object , 100 ~ 160

- If an input condition specifies *a member of a set*, one valid and one invalid equivalence class are defined
  - Name of a city,
  - Vaild :{ Winston-salem, Greensboro,..}, InValid{x| everything else}

- If an input condition is *Boolean*, one valid and one invalid class are defined

Functional Testing Techniques

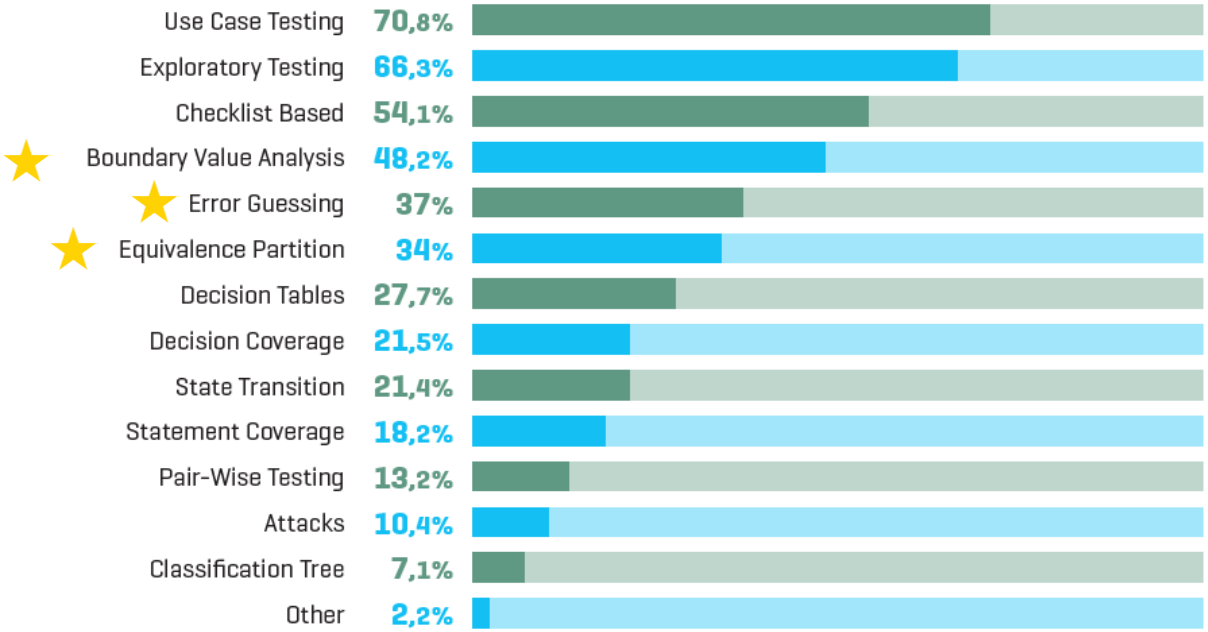
- To confirm that the application under test (AUT) meets its functional requirements

- Equivalence partitioning
- Boundary value analysis
- Static testing

Boundary Value Analysis

Which are the most adopted test techniques?

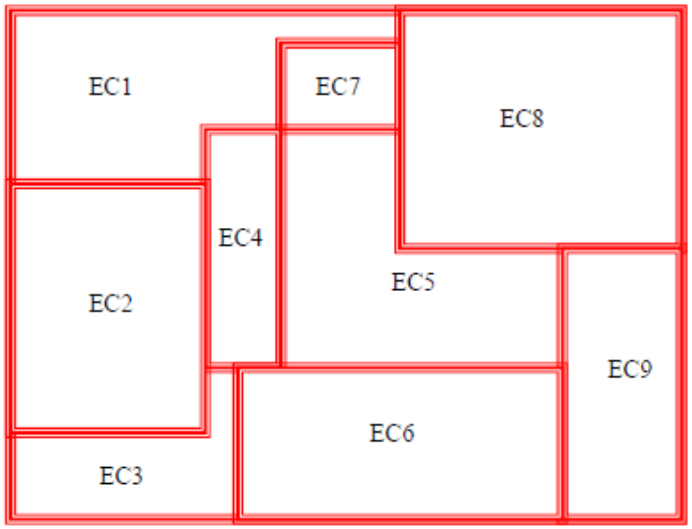
(Multiple answers were allowed.)



Use cases and exploratory testing are more extensively applied than traditional black-box techniques (like BVA or equivalence classes); coverage-based white box techniques are used by no more than 20% of the sample.

Boundary Value Analysis

- Equivalence partition deals with selecting representative values from within the class, Boundary value analysis focuses on testing values form the boundary of the class



Input Domain Space

Boundary Value Analysis

- The Testing problem
  - The specification for a software system for validating expenses claims for hotel accommodation includes the following requirements
  - There is an upper limit of $90 for accommodation expense claims
  - Any claims above $90 should be rejected and cause an error message to be displayed
  - All expense amounts should be greater than $0 and an error message should be displayed if this is not the case

Boundary Value Analysis

- Test case table

Test Case ID	Hotel Charge	Boundary tested	Output
1	-1		Error Message
2	0	0	Error Message
3	1		OK
4	89		OK
5	90	90	OK
6	91		Error Message

Functional Testing Techniques

- To confirm that the application under test (AUT) meets its functional requirements

- Equivalence partitioning
- Boundary value analysis
- Static testing

Static testing

- Static testing does not involve executing or running the AUT

- Code review and inspection
- Code walk-through
- Static analysis tools (syntax checker)
- Complexity estimating tools