1

Overview

- Integration testing
  - Big bang
  - Bottom up
  - Top down
  - Sandwich
  - Continuous
- System testing
  - Functional
  - Performance
- Acceptance testing
- Examples
- Summary

2

Integration Testing

- The entire system is viewed as a collection of subsystems (sets of classes) determined during the system and object design
- Goal: Test all interfaces between subsystems and the interaction of subsystems
- The Integration testing strategy determines the order in which the subsystems are selected for testing and integration.

3

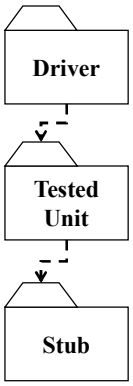
Why do we do integration testing?

- Unit tests only test the unit in isolation
- Many failures result from faults in the interaction of subsystems
- Often many Off-the-shelf components are used that cannot be unit tested
- Without integration testing the system test will be very time consuming
- Failures that are not discovered in integration testing will be discovered after the system is deployed and can be very expensive.

4

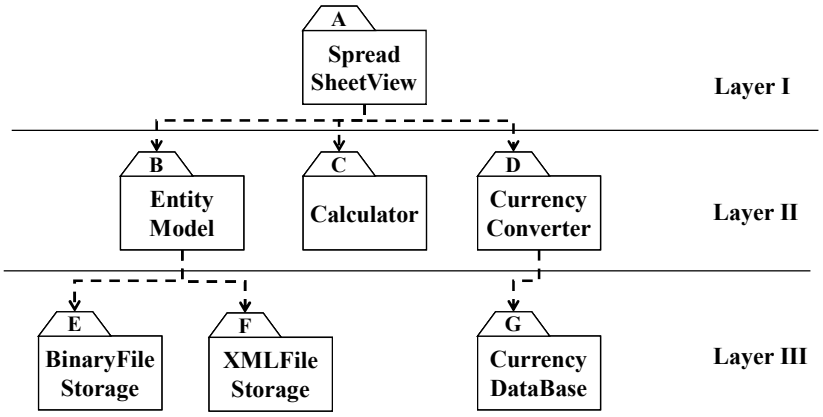
Stubs and drivers

- Driver:
  - A component, that calls the `TestedUnit`
  - Controls the test cases
- Stub:
  - A component, the `TestedUnit` depends on
  - Partial implementation
  - Returns fake values.



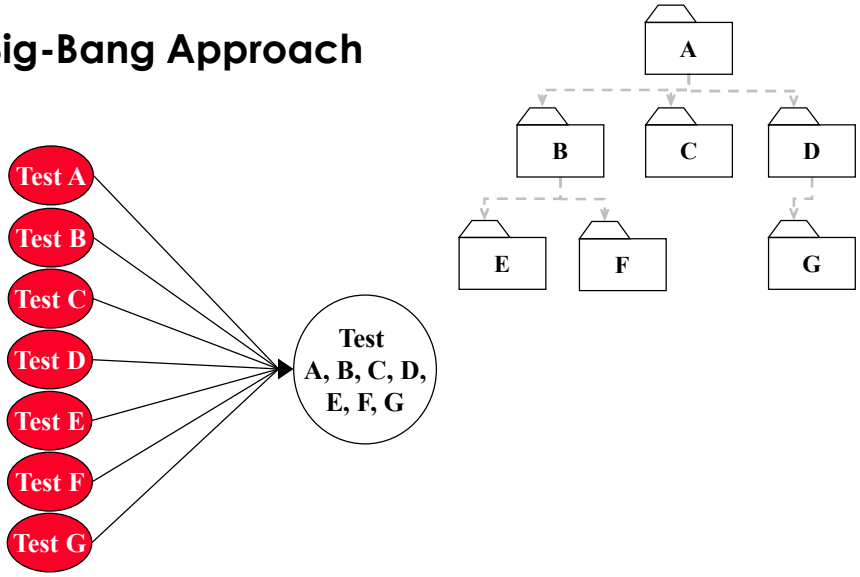
5

Example: A 3-Layer-Design (Spreadsheet)



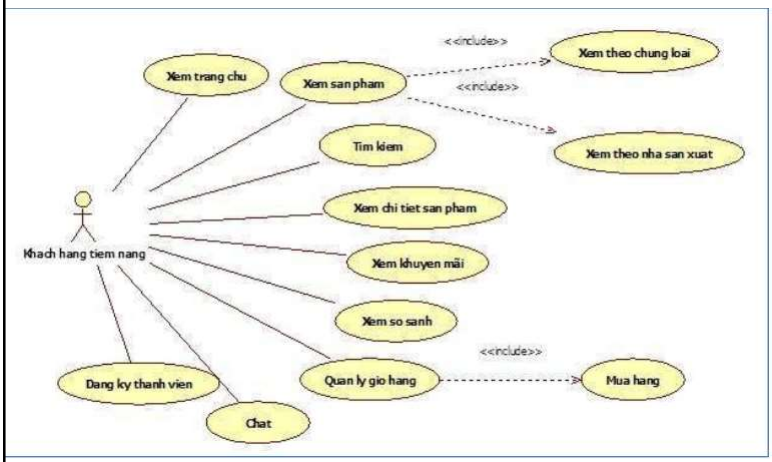
6

Big-Bang Approach



7

Big bang Example



1. Test xem trang chu
2. Test xem san pham
3. Test Tim kiem
4. Test xem chi tiet SP
5. ……..
6. Test xem theo chung loai
7. Test xem theo nha SX
8. Test mua hang

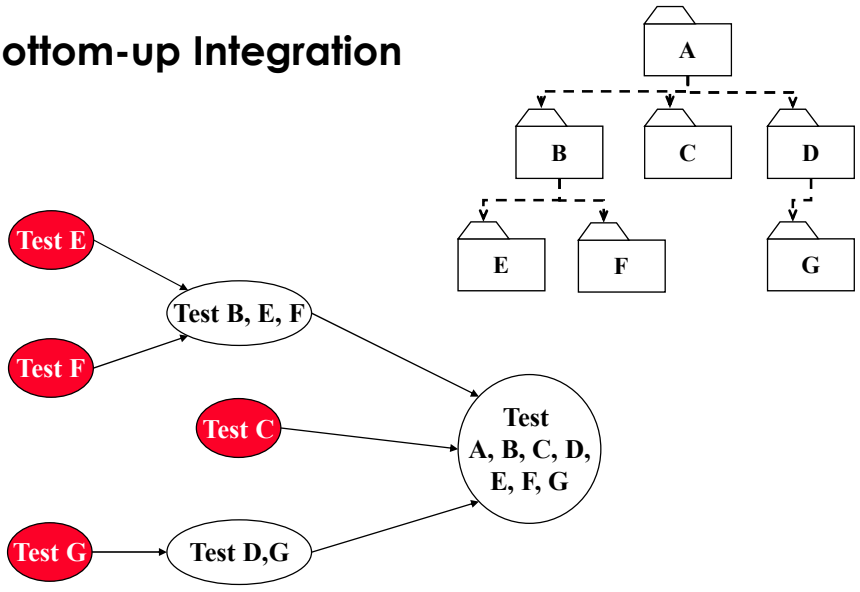
8

Bottom-up Testing Strategy

- The subsystems in the lowest layer of the call hierarchy are tested individually
- Then the next subsystems are tested that call the previously tested subsystems
- This is repeated until all subsystems are included
- Drivers are needed.

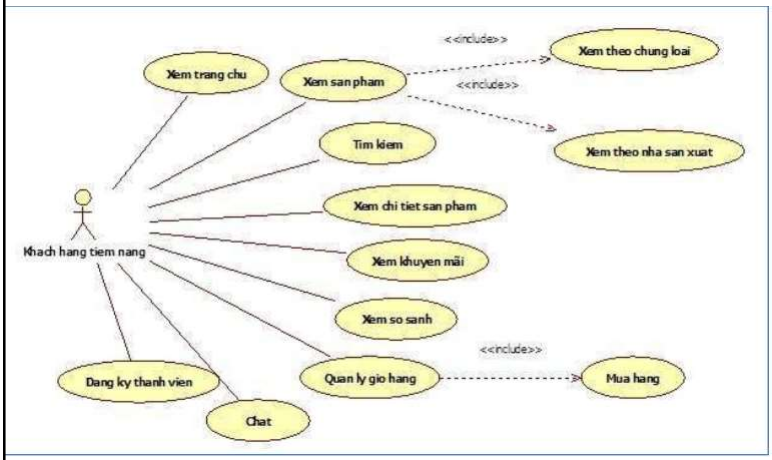
9

Bottom-up Integration



10

Bottom-up Example



1. **Test xem trang chu**
2. **Test xem theo nha SX**
3. **Test xem theo chung loai**
4. **Test xem san pham**
5. **Test mua hang**
6. **Test Quan ly gio hang**
7. **Test xem chi tiet SP**
8. **……..**

11

Pros and Cons of Bottom-Up Integration Testing

- Con:
  - Tests the most important subsystem (user interface) last
  - Drivers needed
- Pro
  - No stubs needed
  - Useful for integration testing of the following systems
    - Object-oriented systems
    - Real-time systems
    - Systems with strict performance requirements.

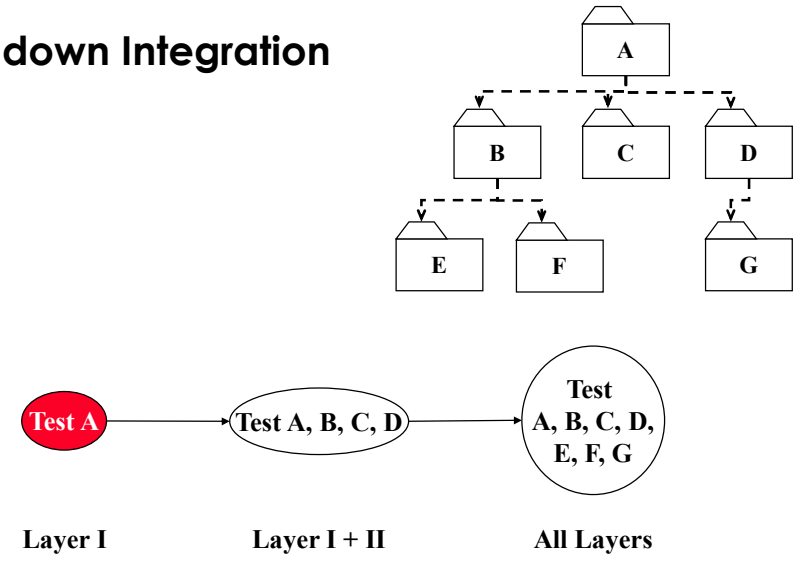
12

Top-down Testing Strategy

- Test the top layer or the controlling subsystem first
- Then combine all the subsystems that are called by the tested subsystems and test the resulting collection of subsystems
- Do this until all subsystems are incorporated into the test
- Stubs are needed to do the testing.

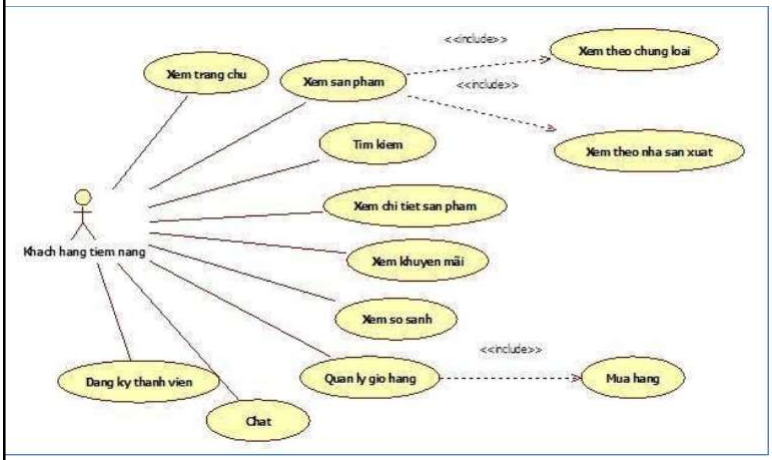
13

Top-down Integration



14

Top-down Example



1. **Test xem trang chu**
2. **Test xem san pham**
3. **Test xem theo nha SX**
4. **Test xem theo chung loai**
5. **Test Quan ly gio hang**
6. **Test mua hang**
7. **Test xem chi tiet SP**
8. **……..**

15

Pros and Cons of Top-down Integration Testing

Pro

- Test cases can be defined in terms of the functionality of the system (functional requirements)
- No drivers needed

Cons

- Writing stubs is difficult: Stubs must allow all possible conditions to be tested.
- Large number of stubs may be required, especially if the lowest level of the system contains many methods.
- Some interfaces are not tested separately.

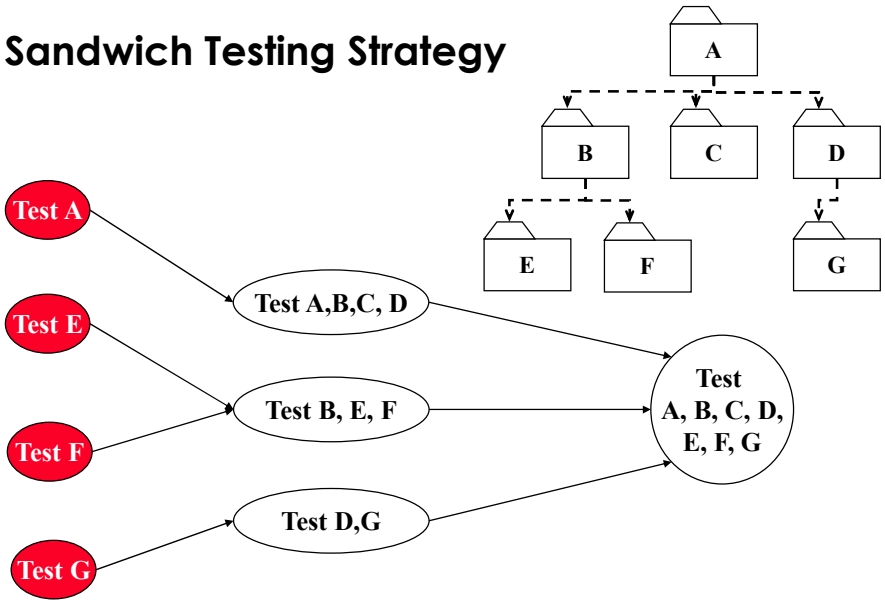
16

Sandwich Testing Strategy

- Combines top-down strategy with bottom-up strategy
- The system is viewed as having three layers
  - A target layer in the middle
  - A layer above the target
  - A layer below the target
- Testing converges at the target layer.

17

Sandwich Testing Strategy



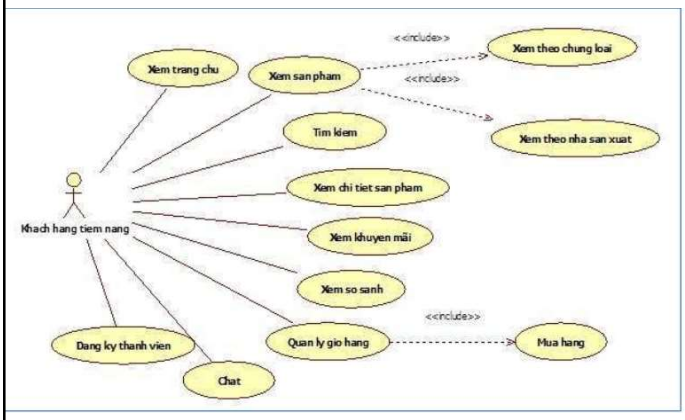
18

Pros and Cons of Sandwich Testing

- Top and Bottom Layer Tests can be done in parallel
- Problem: Does not test the individual subsystems and their interfaces thoroughly before integration
- Solution: Modified sandwich testing strategy

19

Sandwich Example



1. Test xem trang chu
2. Test xem theo nha SX
3. Test xem theo chung loai
4. Test xem san pham
5. Test mua hang
6. Test Quan ly gio hang
7. Test xem theo nha SX
8. Test xem theo chung loai
9. Test Quan Ly gio hang
10. Test Mua hang
11. Test xem chi tiet SP
12. ……..

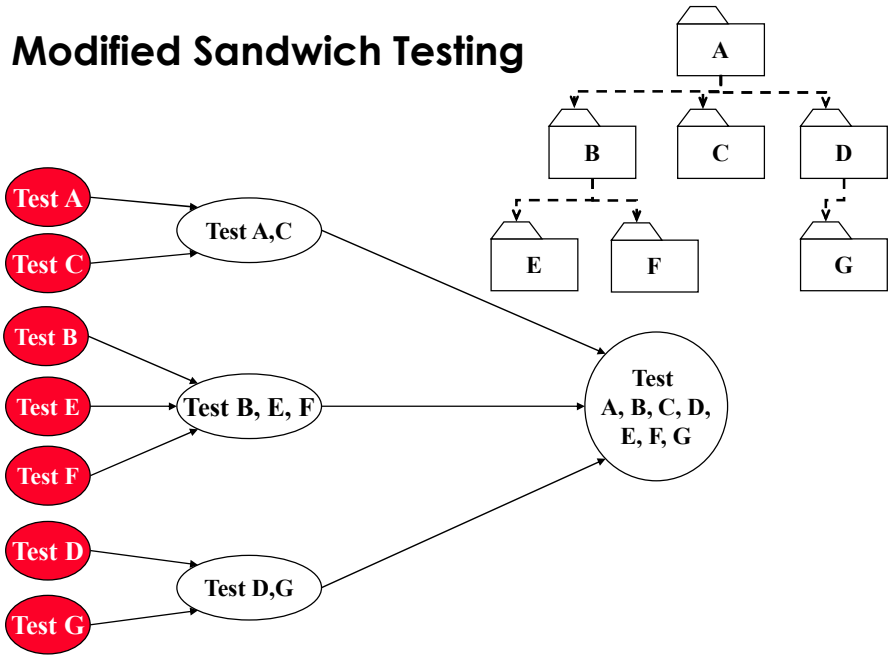
20

Modified Sandwich Testing Strategy

- Test in parallel:
  - Middle layer with drivers and stubs
  - Top layer with stubs
  - Bottom layer with drivers
- Test in parallel:
  - Top layer accessing middle layer (top layer replaces drivers)
  - Bottom accessed by middle layer (bottom layer replaces stubs).

21

Modified Sandwich Testing



22

Continuous Testing

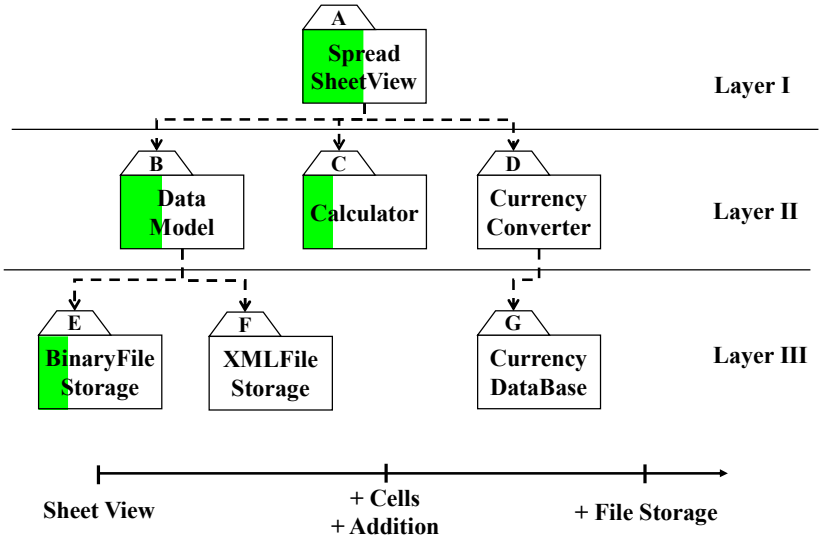
- Continuous build:
  - Build from day one
  - Test from day one
  - Integrate from day one

  ⇒ System is always runnable

- Requires integrated tool support:
  - Continuous build server
  - Automated tests with high coverage
  - Tool supported refactoring
  - Software configuration management
  - Issue tracking.

23

Continuous Testing Strategy



24

Steps in Integration Testing

1. Based on the integration strategy, *select a component* to be tested. Unit test all the classes in the component.
2. Put selected component together; do any *preliminary fix-up* necessary to make the integration test operational (drivers, stubs)
3. Test functional requirements*:* Define test cases that exercise all uses cases with the selected component
4. Test subsystem decomposition*:* Define test cases that exercise all dependencies
5. Test non-functional requirements: Execute *performance tests*
6. *Keep records* of the test cases and testing activities.
7. Repeat steps 1 to 7 until the full system is tested.

The primary *goal of integration testing is to identify failures* with the (current) component *configuration*.

25

System Testing

- Functional Testing
  - Validates functional requirements
- Performance Testing
  - Validates non-functional requirements
- Acceptance Testing
  - Validates clients expectations

26

Functional Testing

Goal: Test functionality of system

- Test cases are designed from the requirements analysis document (better: user manual) and centered around requirements and key functions (use cases)
- The system is treated as black box
- Unit test cases can be reused, but new test cases have to be developed as well.

27

Performance Testing

Goal: Try to violate non-functional requirements

- Test how the system behaves when overloaded.
  - Can bottlenecks be identified? (First candidates for redesign in the next iteration)
- Try unusual orders of execution
  - Call a receive() before send()
- Check the system's response to large volumes of data
  - If the system is supposed to handle 1000 items, try it with 1001 items.
- What is the amount of time spent in different use cases?
  - Are typical cases executed in a timely fashion?

28

Types of Performance Testing

- Stress Testing
  - Stress limits of system
- Volume testing
  - Test what happens if large amounts of data are handled
- Configuration testing
  - Test the various software and hardware configurations
- Compatibility test
  - Test backward compatibility with existing systems
- Timing testing
  - Evaluate response times and time to perform a function
- Security testing
  - Try to violate security requirements
- Environmental test
  - Test tolerances for heat, humidity, motion
- Quality testing
  - Test reliability, maintain-ability & availability
- Recovery testing
  - Test system's response to presence of errors or loss of data
- Human factors testing
  - Test with end users.

29

Acceptance Testing

- Goal: Demonstrate system is ready for operational use
  - Choice of tests is made by client
  - Many tests can be taken from integration testing
  - Acceptance test is performed by the client, not by the developer.
- Alpha test:
  - Client uses the software at the developer's environment.
  - Software used in a controlled setting, with the developer always ready to fix bugs.
- Beta test:
  - Conducted at client's environment (developer is not present)
  - Software gets a realistic workout in target environ-ment

30

Testing has many activities

Establish the test objectives

Design the test cases

Write the test cases

Test the test cases

Execute the tests

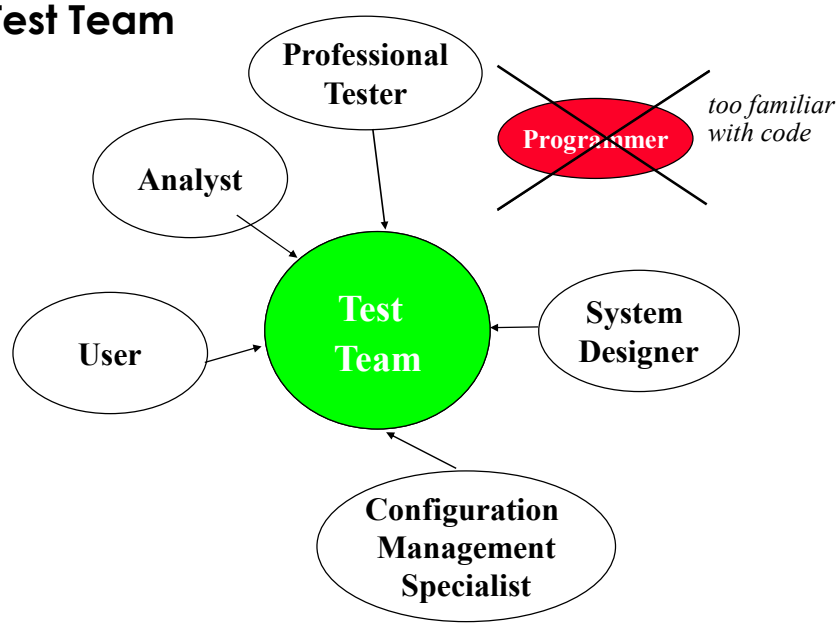
Evaluate the test results

Change the system

Do regression testing

31

Test Team



32

The 4 Testing Steps

1. Select what has to be tested
   - Analysis: Completeness of requirements
   - Design: Cohesion
   - Implementation: Source code
2. Decide how the testing is done
   - Review or code inspection
   - Proofs (Design by Contract)
   - Black-box, white box,
   - Select integration testing strategy (big bang, bottom up, top down, sandwich)
3. Develop test cases
   - A test case is a set of test data or situations that will be used to exercise the unit (class, subsystem, system) being tested or about the attribute being measured
4. Create the test oracle
   - An oracle contains the predicted results for a set of test cases
   - The test oracle has to be written down before the actual testing takes place.

33

Guidance for Test Case Selection

- Use *analysis knowledge* about functional requirements (black-box testing):
  - Use cases
  - Expected input data
  - Invalid input data
- Use *design knowledge* about system structure, algorithms, data structures (white-box testing):
  - Control structures
    - Test branches, loops, ...
  - Data structures
    - Test records fields, arrays, ...

- Use *implementation knowledge* about algorithms and datastructures:
  - Force a division by zero
  - If the upper bound of an array is 10, then use 11 as index.

34

Examples

- Web site with constrains
  - Over 100 user login to web site
  - Response time < 1ms
  - Connect SQL database
  - Authenticate user: user name/password
  - Access home page after login
  - High security

35

Functional testing examples

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
F01	Check Customer Login with valid Data	1.Go to site http://demo.guru99.com 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = guru99 Password = pass99	User should Login into home page	As Expected	Pass
F02	Check Customer Login with invalid Data	1.Go to site http://demo.guru99.com 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = guru99 Password = glass99	User should not Login into home page	As Expected	Pass

36

Performance test case examples

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
P01	Check user limited	1.Open web site http://demo.guru99.com with 51th times 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = guru99 Password = pass99	User should Login into home page	Non As Expected	Fail
P02	Check time response	1.Go to site http://demo.guru99.com with 51th times 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = guru99 Password = glass99	T<1ms	Not As Expected	Fail
P03	Check SQL Injection (security)	1.Go to site http://demo.guru99.com with 51th times 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = <JavaScript alert='Okie'> Password = <JavaScript alert='Okie'>	No user like this	SQL injecttion → data fail	Fail
P04	Check web page request (security)	1.Go to site http://demo.guru99.com/home.aspx	No data	No access	Access home page	Fail

37

Acceptant test case example

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
A01	Check browser	1.Open Firefox 2.Go to site http://demo.guru99.com 3.Enter UserId 4.Enter Password 5.Click Submit 6.Open IE 7.Go to site http://demo.guru99.com 8.Enter UserId 9.Enter Password 10.Click Submit	Userid = guru99 Password = pass99	1. Firefox: ok 2. IE: ok	Firefox: Ok IE: not ok	Fail
A02	Check invalid user	1.Go to site http://demo.guru99.com 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = user 1 Password = glass99	User can not Login into web page	As Expected	Pass

38

Test case guide

- Should
  - **Test Cases need to be simple and transparent**
  - **Create Test Case with End User in Mind**
  - **Avoid test case repetition.**
  - **Do not Assume**
  - **Using test case tools**
  - **Ensure 100% Coverage**
  - **Repeatable and self-standing**
  - **Peer Review.**

39

Scenario 1

- System testing
- 1 system includes:
  - 2 web server
  - 2 SQL database server
  - 4 switch
  - 3 LAN
- Constraints
  - Redundancy (dự phòng)
  - Cluster (gom cụm)
  - Security (bảo mật)
  - Available (sẵn sàng)

40

Scenario 2

- System testing
- 1 system includes:
  - 2 mail server
  - 2 Domain controller
  - 2 DHCP server
  - 4 switch
  - 5 LAN
  - 2 Access point
- Constraints
  - Redundancy (dự phòng)
  - Cluster (gom cụm)
  - Security (bảo mật)
  - Available (sẵn sàng)

41

Summary

- Testing is still a black art, but many rules and heuristics are available
- Testing consists of
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance testing
- Design patterns can be used for integration testing
- Testing has its own lifecycle

42