SWE 434
Software Testing Levels and Types

Dr. M .A. Wadud

Office :2055

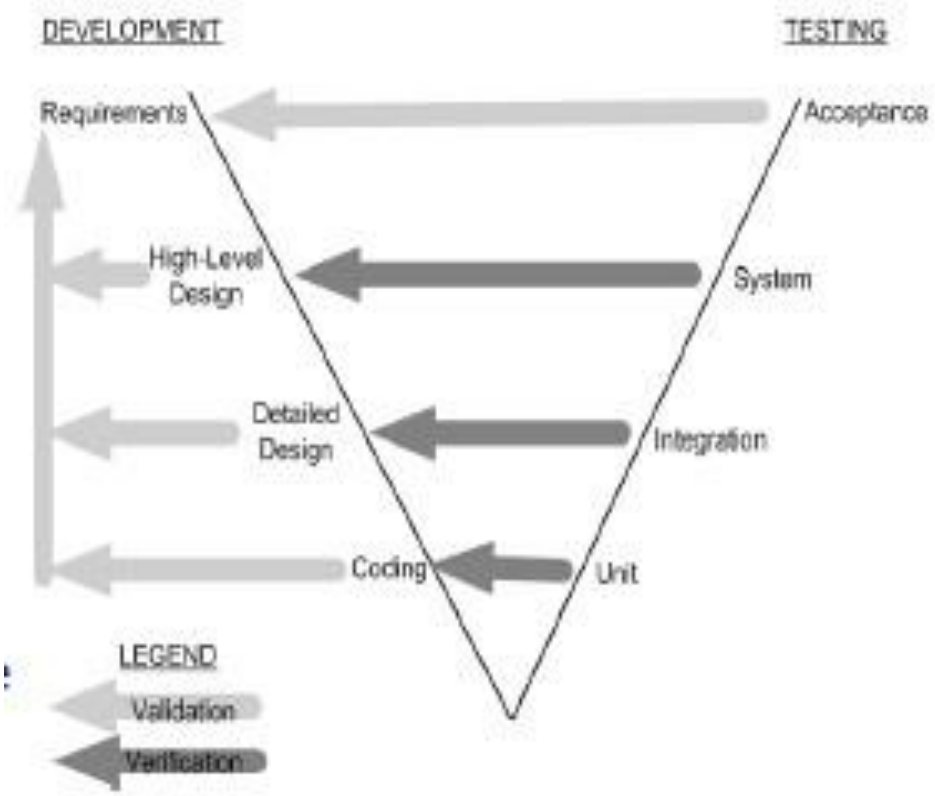
Tel :05613-444-55

E-mail: mwadud@ksu.edu.sa

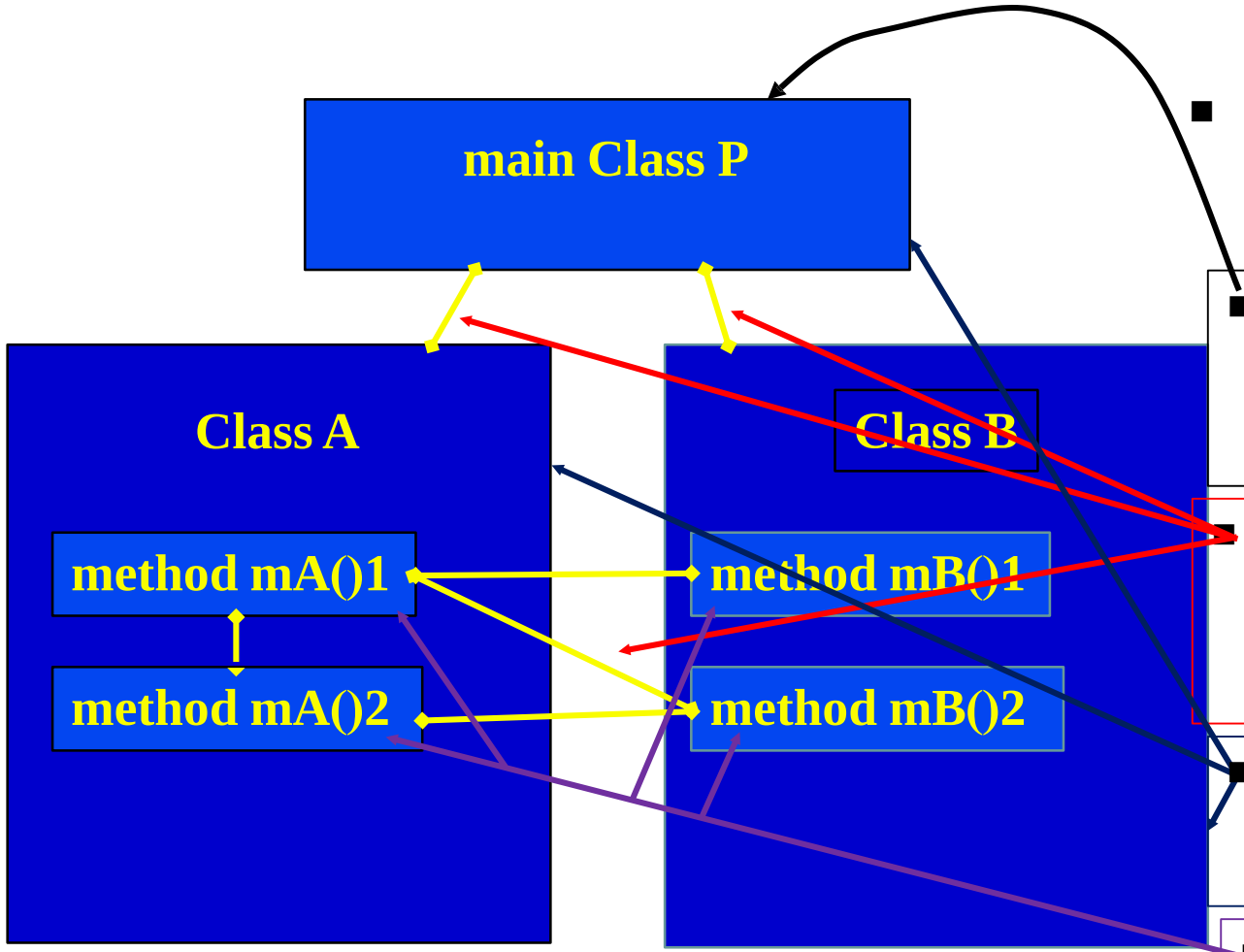
Some of the material in these slides is derived from slides produced by Prof. Some, Alan and Bob of U Ottawa, Thanks to them.

Level of Testing

- **Unit testing**
  - Individual program units, such as procedure
- **Integration testing**
  - Modules are assembled to construct larger subsystem and tested
- **System testing**
  - Includes wide spectrum of testing such as functionality, and load
- **Acceptance testing**
  - Customer's expectations from the system

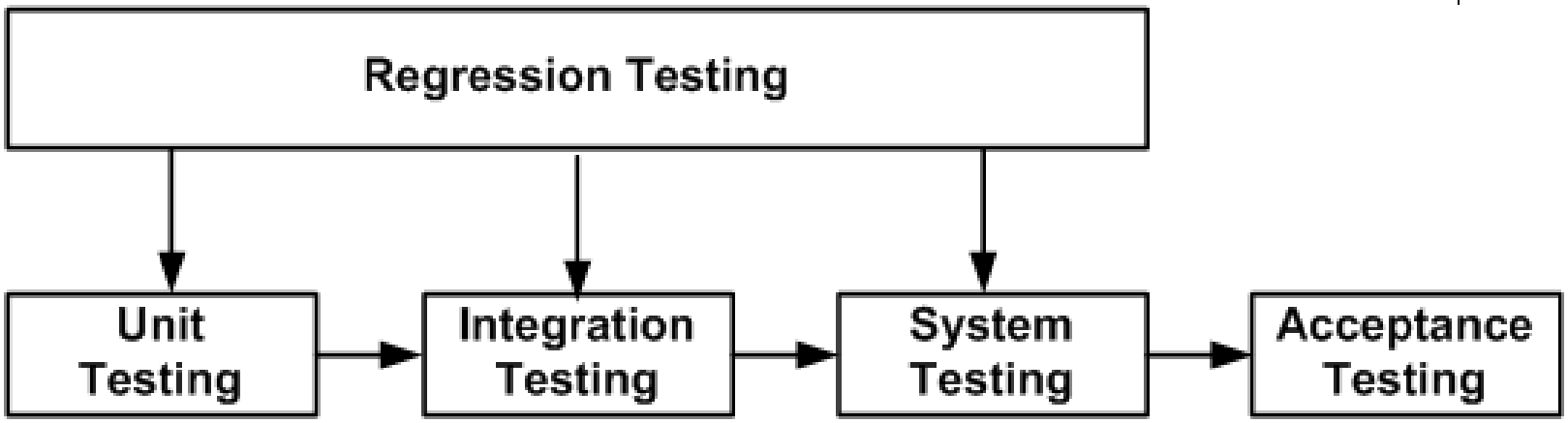


Testing at Different Levels



- Acceptance testing: Is the software acceptable to the user?
- System testing: Test the overall functionality of the system
- Integration testing: Test how modules interact with each other
- Module testing: Test each class, file, module or component
- Unit testing: Test each unit (method) individually

Levels of Testing



. New test cases are not designed

. Test are selected, prioritized and executed

. To ensure that nothing is broken in the new version of the software

Types of Testing

Types of testing ●Unit testing

- Black box testing ➡ based on specification

)Testing without having an insight into the details of underlying code(

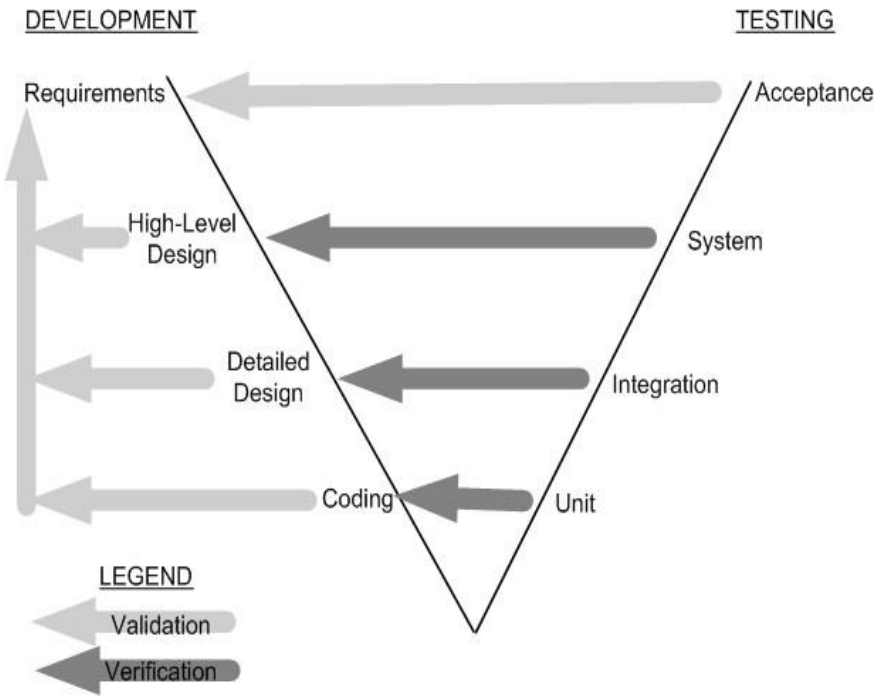
- White box testing ➡ based on internal logic
- Gray box testing ➡ based on design model


```
While(i <6){
if (y == } (2
...
{
```

- Integration testing
- System testing includes
  - Function testing
  - Performance testing
- Acceptance testing
  - Alpha testing
  - Beta testing
- Regression testing :aims to verify whether new faults are introduced

Tool support at different levels

- Unit testing
  - Tools such as JUnit
- Integration testing
  - Stubs, mocks
- System testing
  - Security, performance, load testers
- Regression testing
  - Test Management tools (e.g. defect tracking(... ,

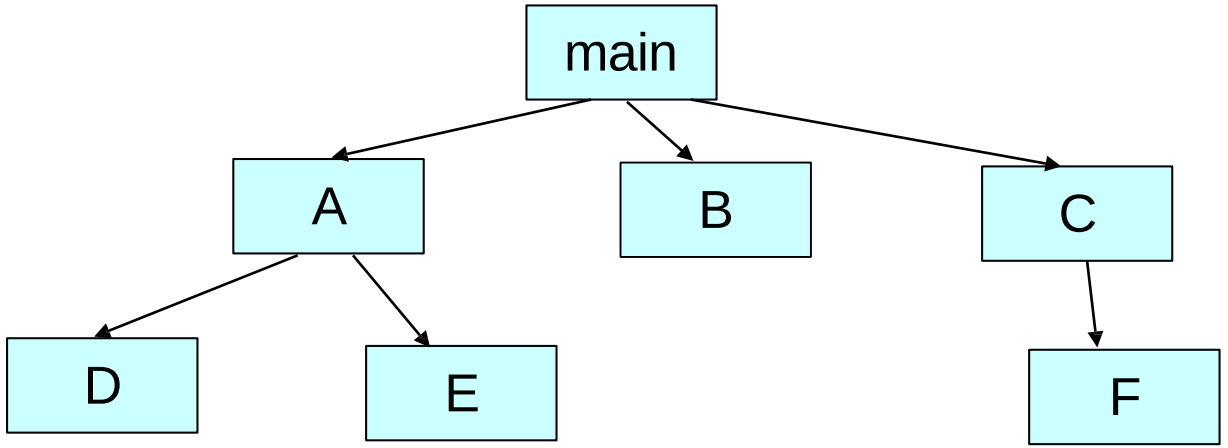


Integration Testing

- Objective of *system integration testing (SIT(*
  - to build a “working” version of the system
  - putting modules together
  - ensuring that the additional modules work as expected without disturbing the functionalities of the modules already put together
- *Integration testing is said to be complete when*
  - the system is fully integrated together
  - all the test cases have been executed
  - all the severe and moderated defects found have been fixed

Integration testing

- Objective: ensure assembled modules, that work fine in isolation, work together
  - Attempt to find interface faults
- Need a module call graph.



Types of Testing

- Functional or Black-box testing
  - Deriving tests from external descriptions of the software, including specifications, requirements, and design
  - the component-under-test (CUT) is a black box
  - we ignore the internal behavior of the CUT and focus exclusively on its interface. For Example,
    - Equivalence partition: sets of equivalent in/valid inputs
    - Boundary-value testing (proceeds from previous one(
    - Cause-effect graph, performance and stress testing, etc.
- Structural or White-box testing
  - Deriving tests from the source code internals of the software, specifically including branches, individual conditions, and statements
  - the CUT is an open or white box
  - we use our knowledge of the internal workings of the CUT to ensure its internal behavior is correct
    - Statement coverage, decision coverage, path coverage
- Grey-box testing Coverage
  - combine black-box testing and white-box testing
  - focus on message in/out CUT and interaction between CUTs
  - intra- and inter- CUT testing, unit test and integration test
    - C++ Data Scenarios [Turner & Robson, [1993
    - Modifier Sequences [Parrish et al., [1994

System Testing

- Performed after the software has been assembled.
- Test of entire system, as customer would see it.
- High-order testing criteria should be expressed in the specification in a measurable way.
- Check if system satisfies requirements for:
  - Functionality
  - Reliability
  - Recovery
  - Multitasking
  - Device and Configuration
  - Security
  - Compatibility
  - Stress
  - Performance
  - Serviceability
  - Ease/Correctness of installation

System Testing

- Acceptance Tests
  - System tests carried out by customers or under customers' supervision
  - Verifies if the system works according to the customers' expectations
- Common Types of Acceptance Tests
  - Alpha testing: end user testing performed on a system that may have incomplete features, within the development environment
    - Performed by an in-house testing panel including end-users.
  - Beta testing: an end user testing performed within the user environment.

Functional and Performance Testing

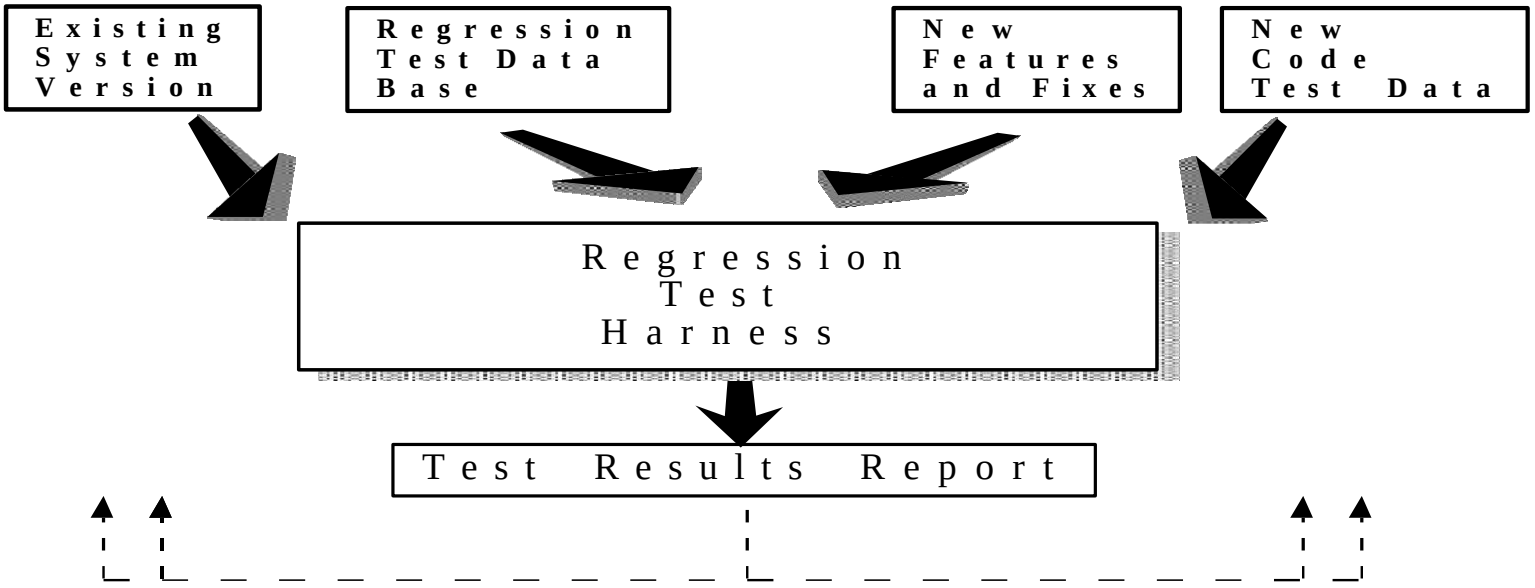
- Functional Testing
  - Ensure that the system supports its functional requirements.
  - Test cases derived from statement of requirements.
    - traditional form
    - use cases
- Performance/Stress/Load Testing
  - Performance Testing – evaluate compliance to specified performance requirement
    - throughput
    - response time
    - memory utilization
    - input/output rates

Stress Testing

Tests that are at the limit of the software's expected input domain

- Very large numeric values (or very small(
- Very long strings, empty strings
- Null references
- Very large files
- Many users making requests at the same time
- Invalid values

System (Regression) Testing



System testing tests that a system has not 'regressed' due to changes.

Involves at least a few thousand tests (or many more).

Test Tools at each level

- **Unit testing**
  - Junit
- **Integration testing**
  - Mock
- **System testing**
  - Security, performance, load testers
- **Acceptance testing**
  - Test Management tools (e.g. defect tracking(... ,

Types of Testing Tools
Test Planning and Management

- Create/maintain test plans
  - integrate with project plan
- Maintain links to Requirements/Specification
  - generate Requirements Test Matrix
- Reports and Metrics on test case execution
- Tracking of history/status of test cases
  - defect tracking

Types of Testing Tools
Test Design & Implementation

- Automatic creation of test cases
  - Based on test design approaches
    - graph based
    - data flow analysis
    - logic based
    - ...
  - Very few concrete usable tools
- Random test data generator
- Stubs/Mocks

Types of Testing Tools
Test Execution

- Test Drivers and Execution Frameworks
  - Run test scripts and report result
  - e.g. JUnit
- Runtime test execution assistance
  - memory leak checkers
  - comparators

Types of Testing Tools
Test Performance assessment

- Analysis of the effectiveness of test cases for extend of system covered
  - Coverage analyzers
  - report on various levels of coverage
- Analysis of the effectiveness of test cases for bug detection
  - mutation testing

Types of Testing Tools
Specialized testing

- Security testing tools
  - password crackers
  - vulnerability scanners
  - packet crafters
  - ...
- Performance / Load testing tools
  - performance monitors
  - load generators
  - ...

More Reading

- To have an more idea about testing please

read the following tutorial: Thanks to the owner

of the tutorial (both pages:(

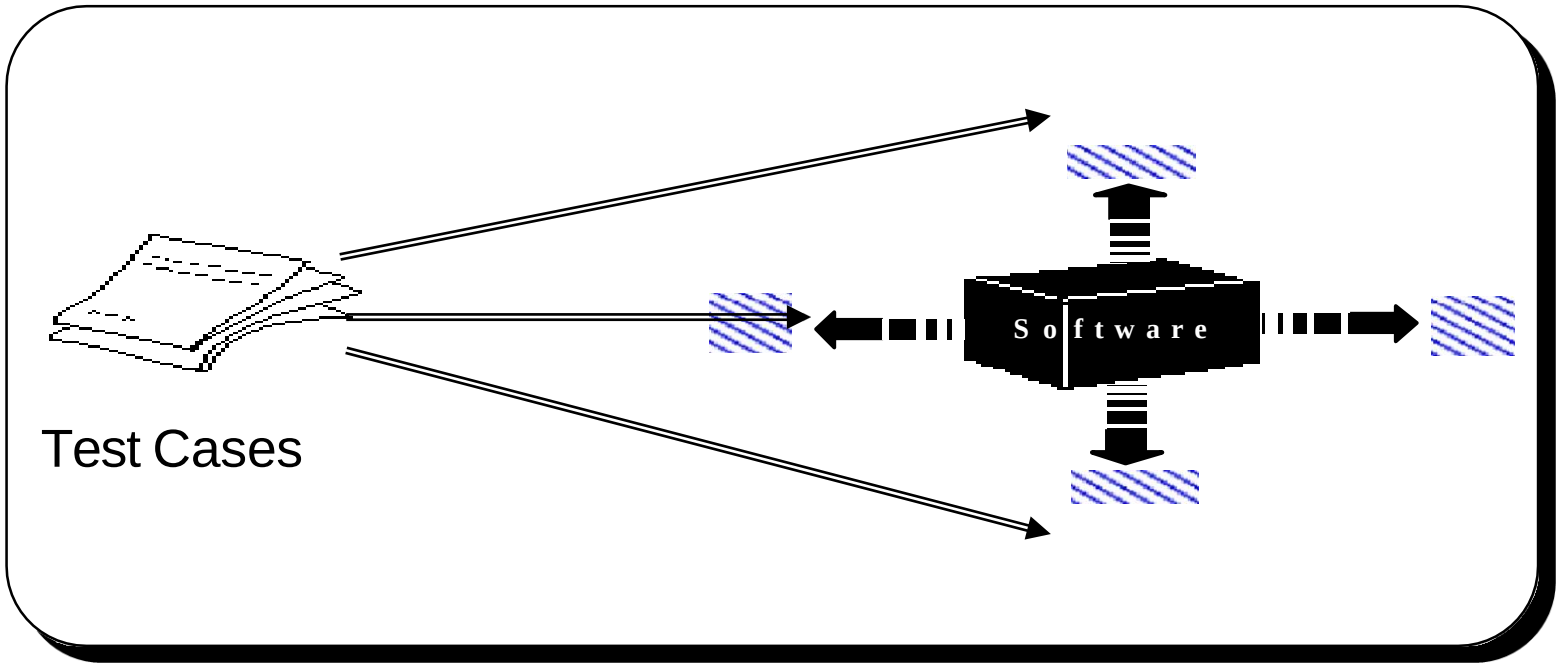
1. http://en.wikipedia.org/wiki/Software_testing (You will get basic idea about testing(

2. http://www.exforsys.com/tutorials/testing/testing-types/1.html

Readings: Ron Patton ch 3, 4, 15, 16 & Daniel Galin ch 9.1 and finally ch 1 from Ammann & Offut

Black Box Test Design

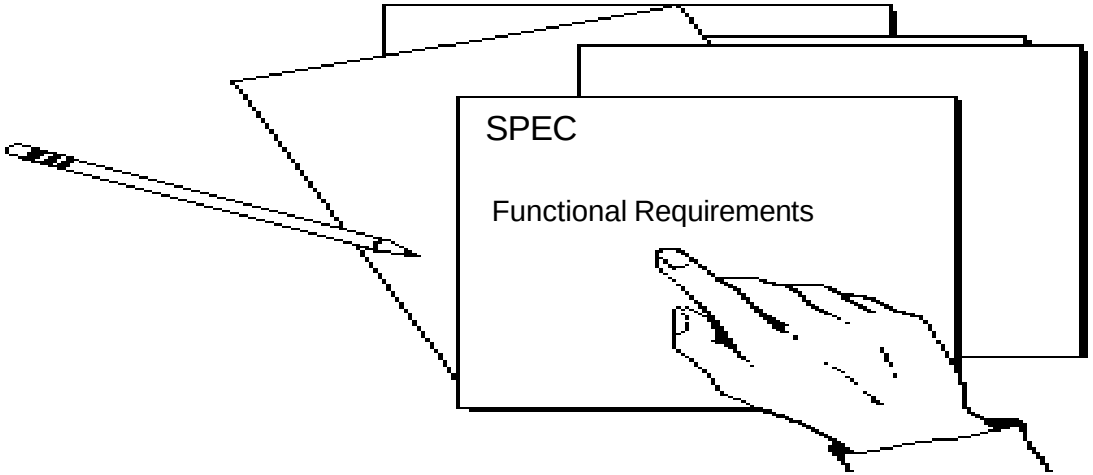
Tests are not concerned with program structure.



Tests focus on the features and program behavior.

Black Box Test Design

Test plans help validate the formal written software requirements which are called:



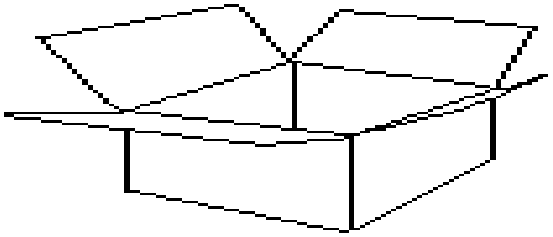
Functional Requirements are reviewed and validated before design begins. When validated, functional requirements become a permanent section in the designer documentation.

Deriving Black Box Test Plans is an essential part of detecting errors and omissions in functional requirements (before errors are mistakenly "frozen" into the requirements(

Functional Requirements

White Box Test Design

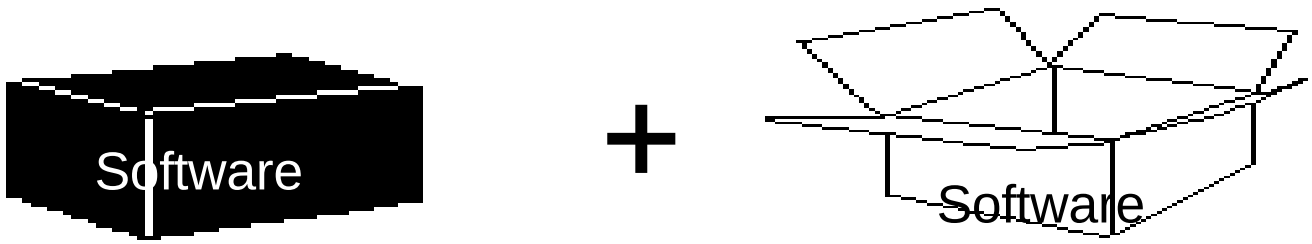
Structural testing focuses directly on the internal logic



to verify that the code is executing as intended.

Comprehensive Testing/Grey Box Testing

White box test design is most appropriate at the completion of black box test execution.



Designers should first test using black box test design methods to test the software features, and then measure the level of coverage obtained by the black box tests to detect gaps in exercising the code.

The reason for this is to maximize code stability before starting white box testing--a more efficient allocation of resources!

Testing Automation

- Use of automated tools is essential
  - provide speed, efficiency, accuracy, precision, etc
  - allow repeatibility (regression testing(
- Types of tools:
  - viewers and monitors (e.g. code coverage tool, debugger(
  - drivers and stubs
  - stress and load tools
  - analysis tools (e.g. file comparison, screen capture and comparison(
  - random testing tools (monkeys(
  - defect tracking

When to automate testing

- Automated testing is especially beneficial if the tests need to be re-executed “quickly”
  - Frequent recompiles
  - Large number of tests
  - Using an agile development process
- An automated test can be duplicated to create many instances for capacity / stress testing.

When NOT to automate

- Initial functional testing
- Automated testing is more likely to find bugs introduced by changes to code or the execution environment, rather than in new functionality.
- Automated test scripts may not be ready for first software release.
- Situations requiring human judgment to determine if system is functioning correctly.

What do we need to do automated testing?

- Test script
  - Test Case Specification
    - Actions to send to system under test (SUT.(
    - Responses expected from SUT.
  - How to determine whether a test was successful or not?
- Test execution system
  - Mechanism to read test script, and connect test case to SUT.
  - Directed by a test controller.