ADVANCED SOFTWARE DEVELOPMENT METHODOLOGIES

24-11-2020

Software Testing Strategy & Environment

What is Software Testing ?

Testing is a verification and validation activity that is performed by executing program code.

“Paying more attention to running tests than to designing them is a classic mistake”-Brian Marick

“Testing is the process of executing a program with the intent of finding errors” - Glen Myers

Lessons

- Many errors are made in the early phases
- These errors are discovered late
- Repairing those errors is costly
- $\Rightarrow$ It pays off to start testing real early

Terminologies

- Mistake- Any action that later shows up as an incorrect result during program execution.
- Error- Result of the mistake committed
- Failure-Incorrect behaviour exhibited by the program during its execution.
- Test Scenario- It is an abstract test case which only identifies the aspects of the program that are to be tested without identifying input, state and output.
- Test Script- encoding of a test case as a short program

Cont..

- Test Suite- set of all test that have been designed by a tester to test a given program
- Testability- Implementation of requirement conforms to it in both functionality and performance
- Equivalent Faults- Two or more bugs that result in the system failing in the same failure mode.

Objectives of SW Testing

- The main objective of SW testing is to derive a set of tests to find uncovering error in software.

- Testing helps in assessing the quality and reliability of software.

What testing cannot do ?

- *Show the absence of errors*

How then to proceed?

- Exhaustive testing most often is not feasible
- Random statistical testing does not work either if you want to find errors
- Therefore, we look for systematic ways to proceed during testing

Verification vs. validation

- Verification:

  "Are we building the product right".

  - The software should conform to its specification.
- Validation:

  "Are we building the right product".

  - The software should do what the user really requires.
- Is a whole life-cycle process - V & V must be applied at each stage in the software process.
- Has two principal objectives
  - The discovery of defects in a system;
  - The assessment of whether or not the system is useful and useable in an operational situation.

V& V goals

- Verification and validation should establish confidence that the software is fit for purpose.
- This does NOT mean completely free of defects.
- Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed.

V & V confidence

- Depends on system's purpose, user expectations and marketing environment
  - Software function
    - The level of confidence depends on how critical the software is to an organisation.
  - User expectations
    - Users may have low expectations of certain kinds of software.
  - Marketing environment
    - Getting a product to market early may be more important than finding defects in the program.

Levels of Testing

Type of Testing	Performed By
Low-level testing	
Unit (module) testing	Programmer
integration testing	Development team
High-level testing	
Function testing	Independent Test Group
System testing	Independent Test Group
Acceptance testing	Customer

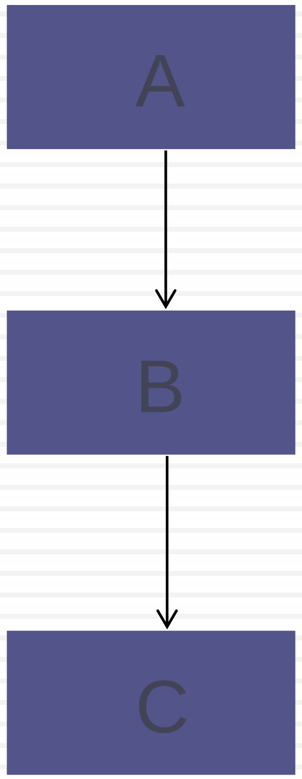
Unit Testing

- done on individual modules
- test module w.r.t module specification
- largely white-box oriented
- mostly done by programmer
- Unit testing of several modules can be done in parallel
- requires *stubs* and *drivers*

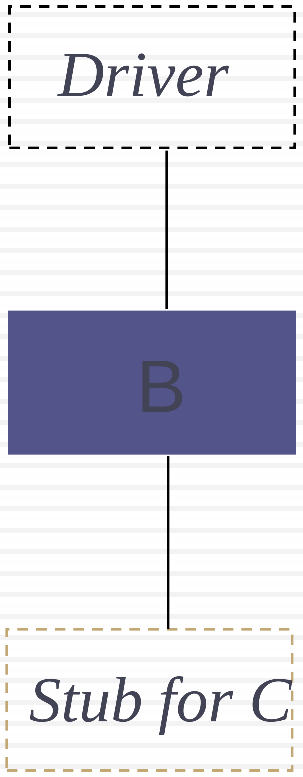
What are Stubs, Drivers ?

- Stub
  - dummy module which simulates the function of a module called by a given module under test
- Driver
  - a module which transmits test cases in the form of input arguments to the given module under test and either prints or interprets the results produced by it

eg. module call hierarchy



eg. to unit test B in isolation



Black-box vs. White-Box Testing

- Black box testing can detect errors such as
  - -incorrect functions or missing functions
  - -interface errors
  - -errors in data structure or external data base access
  - -behavioral or performance errors
  - -Initialization and termination errors

- White box testing can detect errors such as
  - logic errors, design errors

  It cannot detect whether the program is performing its expected functions, missing functionality.

Integration Testing

- tests a group of modules, or a subsystem
- test subsystem structure w.r.t design, subsystem functions
- focuses on module interfaces
- largely structure-dependent
- done by one/group of developers

Integration Test Approaches

- Non-incremental (Big-Bang integration)
  - unit test each module independently
  - combine all the modules to form the system in one step, and test the combination
- Incremental
  - instead of testing each module in isolation, the next module to be tested is first combined with the set of modules that have already been tested
  - testing approaches:- Top-down, Bottom-up

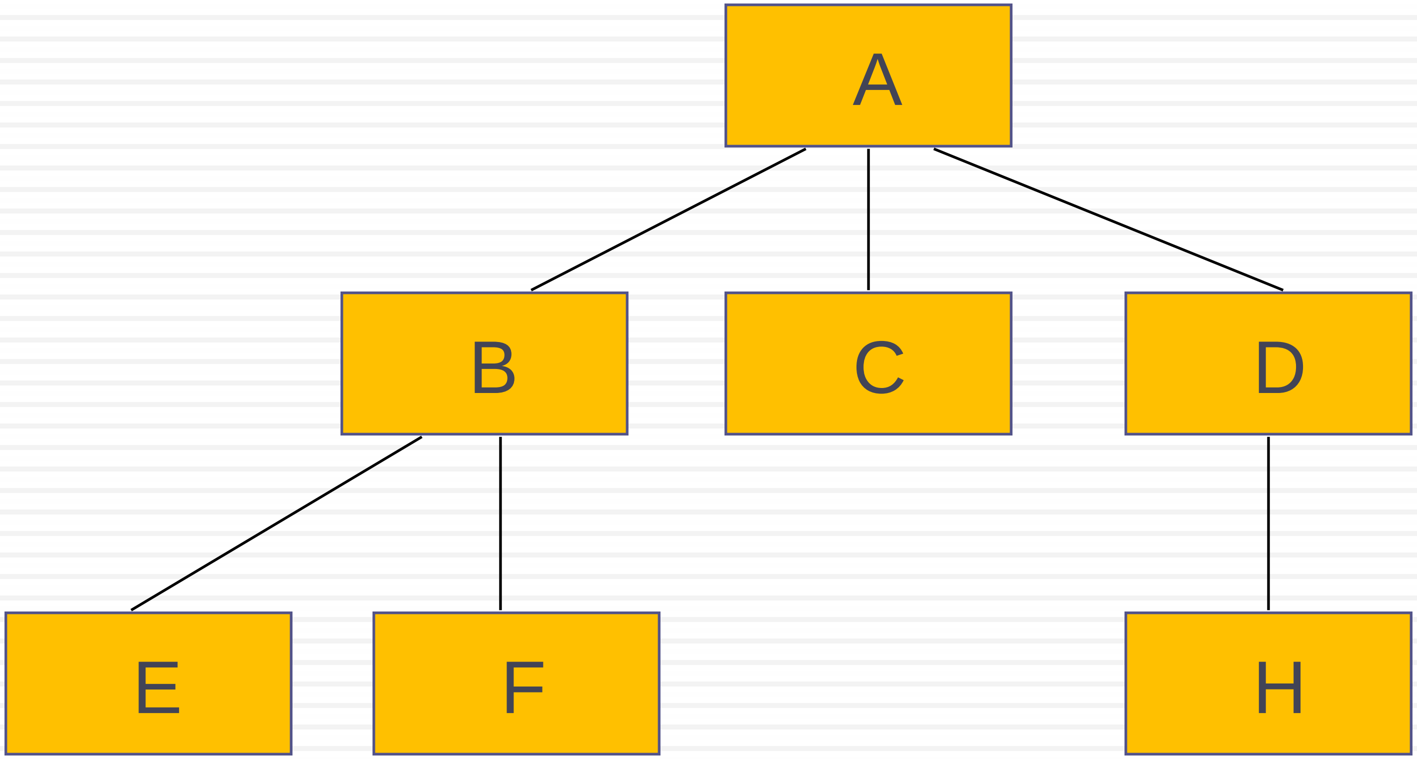
Comparison

Non-Incremental	Incremental
requires more stubs,drivers	requires less stubs, drivers
module interfacing errors detected late	module interfacing errors detected early
debugging errors is difficult	debugging errors is easier
	results in more thorough testing of modules

Top-down Integration

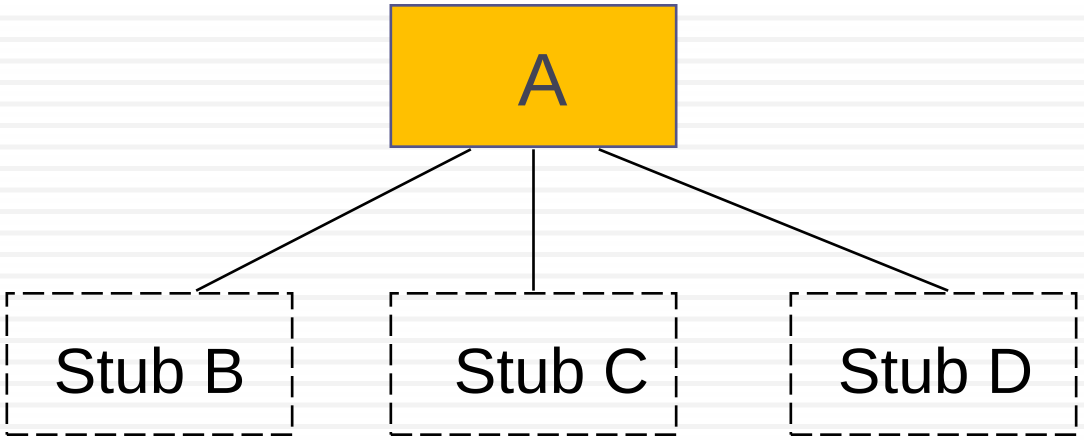
- Begin with the top module in the module call hierarchy
- Stub modules are produced
  - Stubs are often complicated
- The next module to be tested is any module with at least one previously tested super ordinate (calling) module
- After a module has been tested, one of its stubs is replaced by an actual module (the next one to be tested) and its required stubs

Example: Module Hierarchy



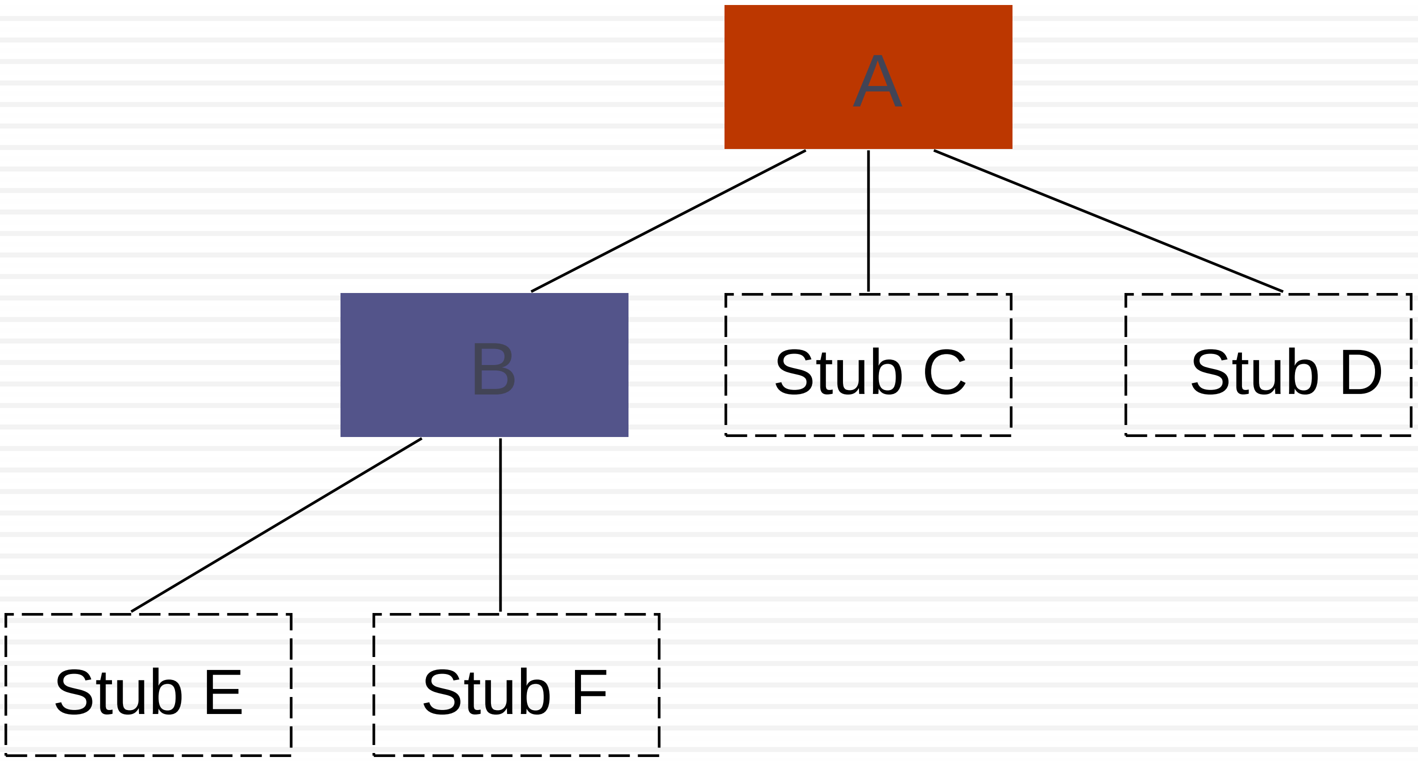
Top-down Integration Testing

Example:



Top-down Integration Testing

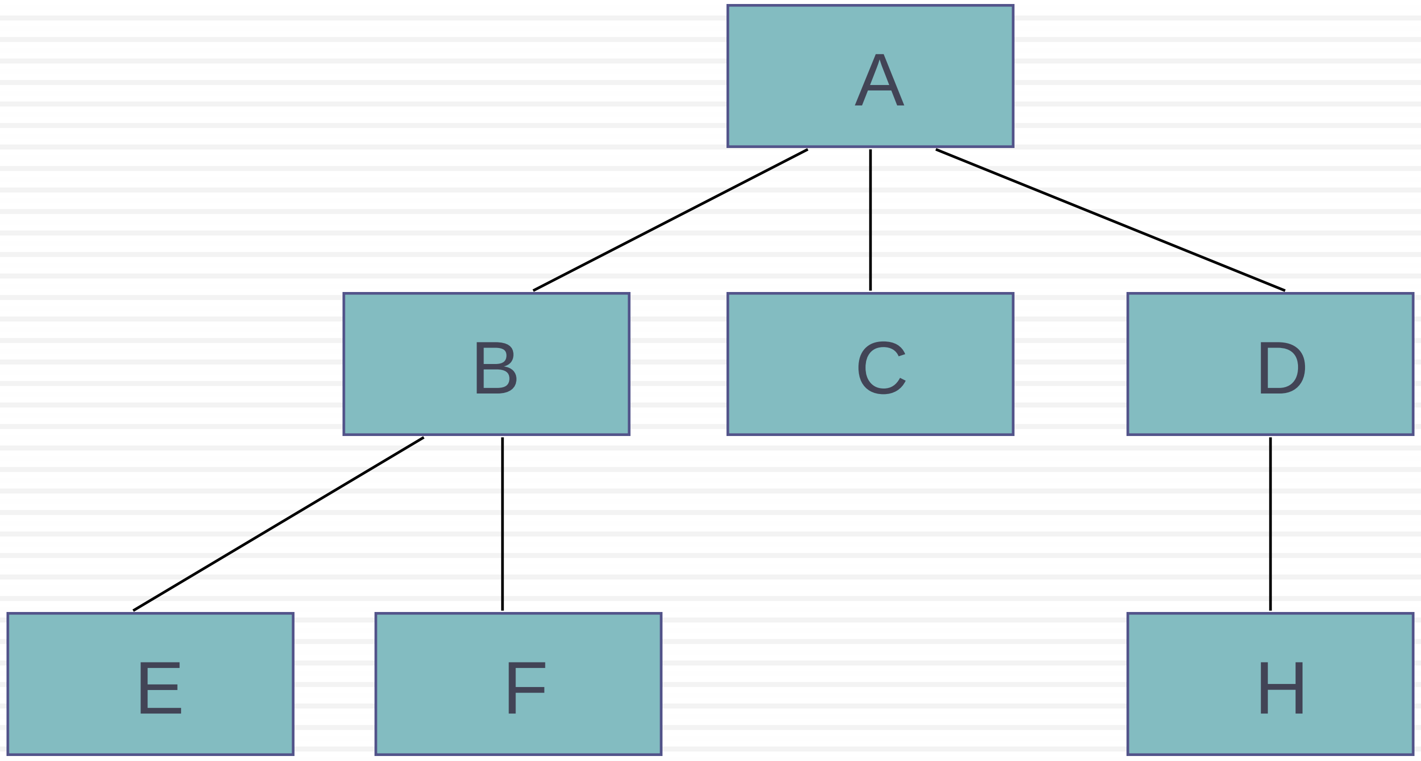
Example:



Bottom-Up Integration

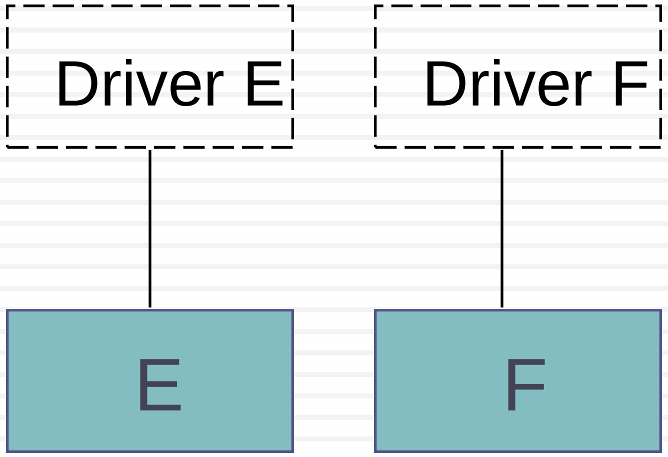
- Begin with the terminal modules (those that do not call other modules) of the modules call hierarchy
- A driver module is produced for every module
- The next module to be tested is any module whose subordinate modules (the modules it calls) have all been tested
- After a module has been tested, its driver is replaced by an actual module (the next one to be tested) and its driver

Example: Module Hierarchy



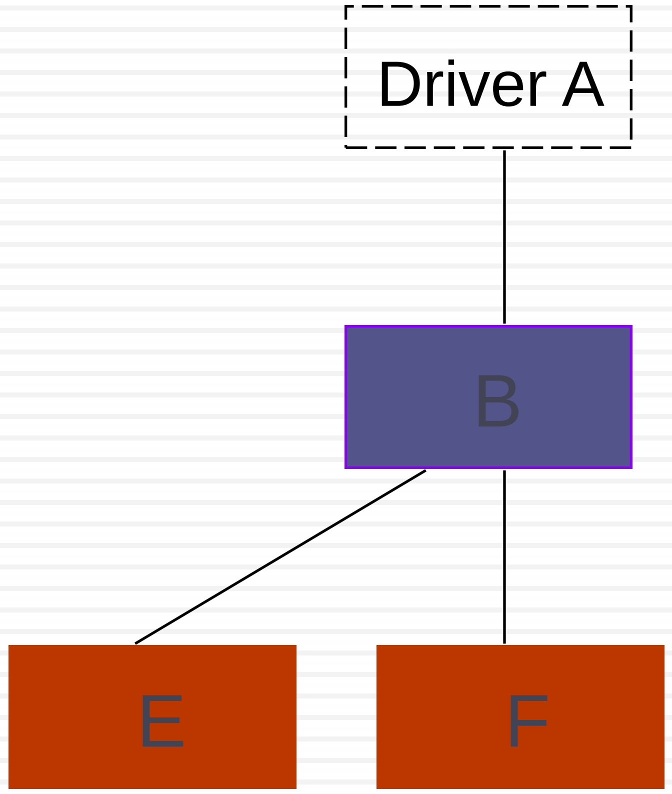
Bottom-Up Integration Testing

Example:



Bottom-Up Integration Testing

Example:



Comparison

Top-down Integration

- Advantage
  - a skeletal version of the program can exist early
- Disadvantage
  - required stubs could be expensive

Bottom-up Integration

- Disadvantage
  - the program as a whole does not exist until the last module is dded

No clear winner

- *Effective alternative* -- use Hybrid of bottom-up and top-down
  - prioritize the integration of modules based on risk
  - highest risk functions are integration tested earlier than modules with low risk functions

System Testing

- Different from Function testing
- Process of attempting to demonstrate that the program or system does not meet its original requirements and objectives as stated in the requirements specification
- Test cases derived from
  - requirements specification
  - system objectives, user documentation

Types of System Tests

- Volume testing
  - to determine whether the program can handle the required volumes of data, requests, etc.
- Load/Stress testing
  - to identify peak load conditions at which the program will fail to handle required processing loads within required time spans
- Usability (human factors) testing
  - to identify discrepancies between the user interfaces of a product and the human engineering requirements of its potential users.
- Security Testing
  - to show that the program's security requirements can be subverted

Types of System Tests

- Performance testing
  - to determine whether the program meets its performance requirements (eg. response times, throughput rates, etc.)
- Recovery testing
  - to determine whether the system or program meets its requirements for recovery after a failure
- Installability testing
  - to identify ways in which the installation procedures lead to incorrect results
- Configuration Testing
  - to determine whether the program operates properly when the software or hardware is configured in a required manner

Types of System Tests

- Compatibility/conversion testing
  - to determine whether the compatibility objectives of the program have been met and whether the conversion procedures work
- Reliability/availability testing
  - to determine whether the system meets its reliability and availability requirements
- Resource usage testing
  - to determine whether the program uses resources (memory, disk space, etc.) at levels which exceed requirements

Acceptance Testing

- performed by the Customer or End user
- compare the software to its initial requirements and needs of its end users

Alpha and Beta Testing

Tests performed on a SW Product before its released to a wide user community.

- Alpha testing
  - conducted at the developer’s site by a User
  - tests conducted in a controlled environment
- Beta testing
  - conducted at one or more User sites by the end user of the SW
  - it is a “live” use of the SW in an environment over which the developer has no control

Regression Testing

- Re-run of previous tests to ensure that SW already tested has not regressed to an earlier error level after making changes to the SW.

Performance Testing

- Stress testing
- Volume testing
- Configuration testing
- Compatibility testing
- Regression testing
- Recovery testing
- Maintenance testing
- Documentation testing
- Usability testing

Error Seeding

- n/N=s/S
- N→ Total Number of defects in the system
- n→ defects found by testing
- S→ Total number of seeded defects
- s→ defects found in testing

Defects still remaining in the program after testing

$$N-n = n * (S-1)/s$$

SW Test Documentation

- Test Plan
- Test design specification
- Test cases specification
- Test procedure specification
- Test incident reports, test logs
- Test summary report

SW Test Documentation

- Test design specification
  - to specify refinements of the test approach and to identify the features to be covered by the design and its associated tests.

- Test cases specification
  - to define a test case identified by a test design specification. The test case spec documents the actual values used for the input along with the anticipated outputs.

  - Test cases are separated from test designs to allow for use in more than one design and to allow foe reuse in other situations.

SW Test Documentation

- Test procedure specification
  - to identify all steps required to operate the system and execute the specified test cases in order to implement the associated test design.
  - The procedures are separated from test design specifications as they are indented to be followed step by step and should not have extraneous detail.

SW Test Documentation

- Test Log
  - to provide a chronological record of relevant details about the execution of tests.
- Test incident report
  - to document any test execution event which requires further investigation
- Test summary report
  - to summarize the results of the testing activities associated with one or more test design specs and to provide evaluations based on these results

SW Testing Tools

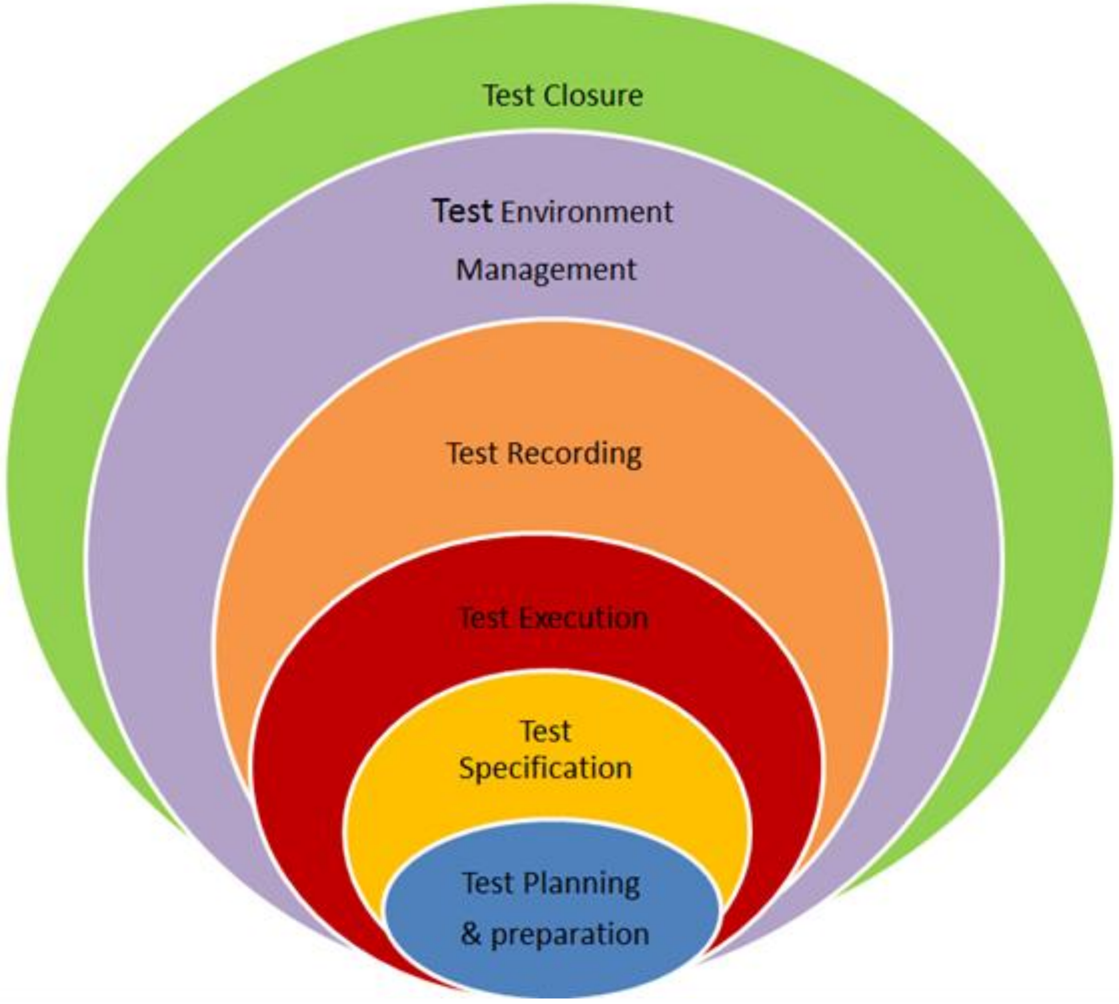
- Capture/playback tools
  - capture user operations including keystrokes, mouse activity, and display output
  - these captured tests form a baseline for future testing of product changes
  - this makes regression testing easier
- Coverage analyzers
  - tell us which parts of the product under test have been executed (covered) by the current tests
  - identifies parts not covered
  - varieties of coverage - statement, decision, … etc.

SW Testing Tools

- Memory testing (bounds-checkers)
  - detect memory problems, exceeding array bounds, memory allocated but not freed, reading and using uninitialized memory
- Test case management
  - provide a user interface for managing tests
  - organize tests for ease of use and maintenance
  - start and manage test execution sessions that run user-selected tests
  - provide seamless integration with capture/playback and coverage analysis tools
  - provide automated test reporting and documentation
- Tools for performance testing of client/server applications

SW Testing Support Tools

- Defect tracking tools
  - used to record, track, and generally assist with the management of defects
  - submit and update defect reports
  - generate pre-defined or user-defined management reports
  - selectively notify users automatically of changes in defect status
  - provide secured access to all data via user-defined queries


Test Closure
Test Environment Management
Test Recording
Test Execution
Test Specification
Test Planning & preparation