

# Software Verification and Validation

## TTU - CS 5374

An Introduction to Testing

# Software Testing Techniques – A Quick Look

- What is it?
  - Once source code is available, software must be tested to uncover and fix possible errors before delivery to the customer
  - Goal is to design a series of test cases that have high likelihood of finding errors
  - Software testing techniques provide systematic guidance for designing tests s.t.
    1. Exercise the internal logic of software component
    2. Exercise the input and output domains of the program
- Who does it?
  - During early stages of testing, a software engineer performs all tests
  - As the testing process progresses, testing specialists may involved
  - Good software testers
    - Must try to break any program

# Software Testing Techniques – A Quick Look

- Why is it important?
  - Reviews and other SQA activities are necessary, but not sufficient
  - Before delivering products to customers, we must test it
  - To find the highest possible number of errors, tests must be conducted systematically using discipline techniques
- What are the steps?
  - Software requirements are exercised (functional testing)
  - Internal program logic is tested (structural testing)
- What is the work product?
  - A set of test cases designed to exercise both internal logic and external requirements. That includes expected and actual results.
- How do I ensure that I have done it right?
  - Try hard to break the software
  - Review your test cases for thoroughness

# What Do We Mean by Testing?

- Two types of program analysis
  - Static analysis – Inspecting code without actually running it
    - Formal Methods
  - Dynamic analysis – Inspecting code by executing it
    - Software testing
- Software testing involves:
  1. Preparing test plan
  2. Developing effective test cases
  3. Running executable code on the selected test cases in controlled environments
  4. Recording the behavior of software
  5. Possibly adding more test cases
  6. Analyzing whether the observed behavior is acceptable or not
- Major question: When to stop?
  - Stopping criteria with respect to test plan
  - An optimization problem and research direction

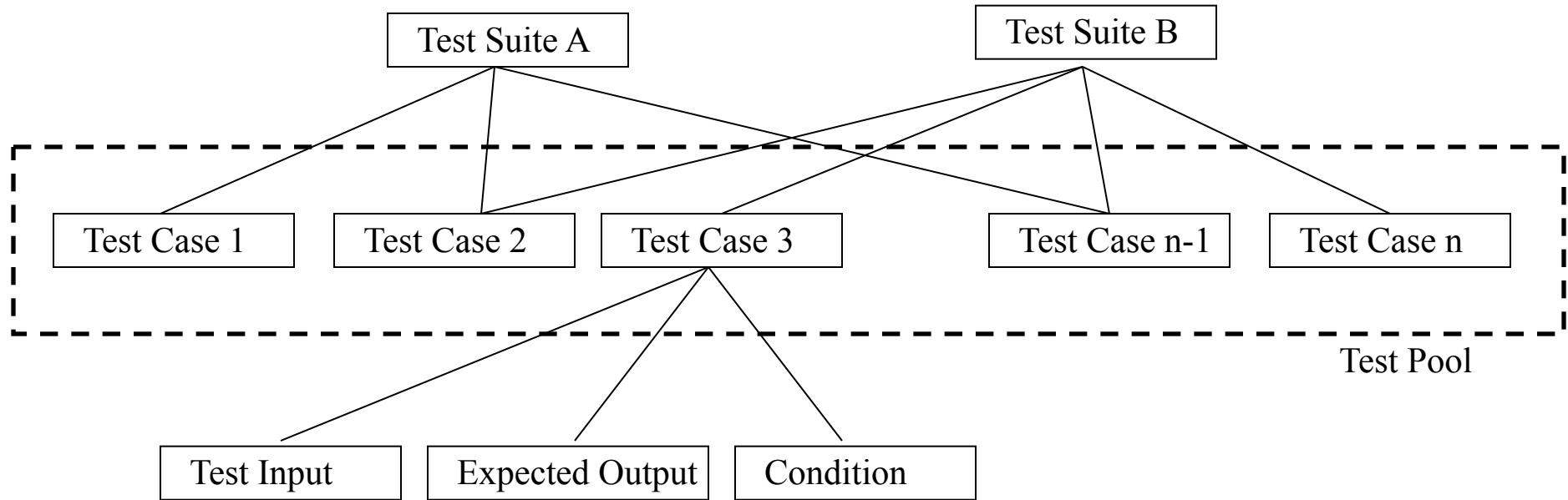
# Testing Terminologies

- Failure
  - Event in which program performs incorrectly
    - E.g. Producing wrong output
- Fault
  - Problem in program causing failure
    - E.g. Using “<” instead of “<=”
- Bug
  - Used informally for both failure and fault
- Defect
  - Used for both failure and fault and also for problems in specs and design, etc.
- Error
  - Mistake programmer made in introducing fault
    - E.g. Thinking that using “<” was right

# Testing Terminologies

- Test input
  - Input data to run a test
  - E.g. “-l” in the Unix command “ls -l”
- Test case
  - The set of {test input, expected output, condition}
    - Condition: we need to satisfy in order to generate the expected result
- Passing test case
  - The test case in which software behaves as expected
- Failing test case
  - The test case in which software behaves incorrectly
- Test suite
  - A set of test cases
- Test pool
  - Set of all test cases developed

# Testing Terminologies



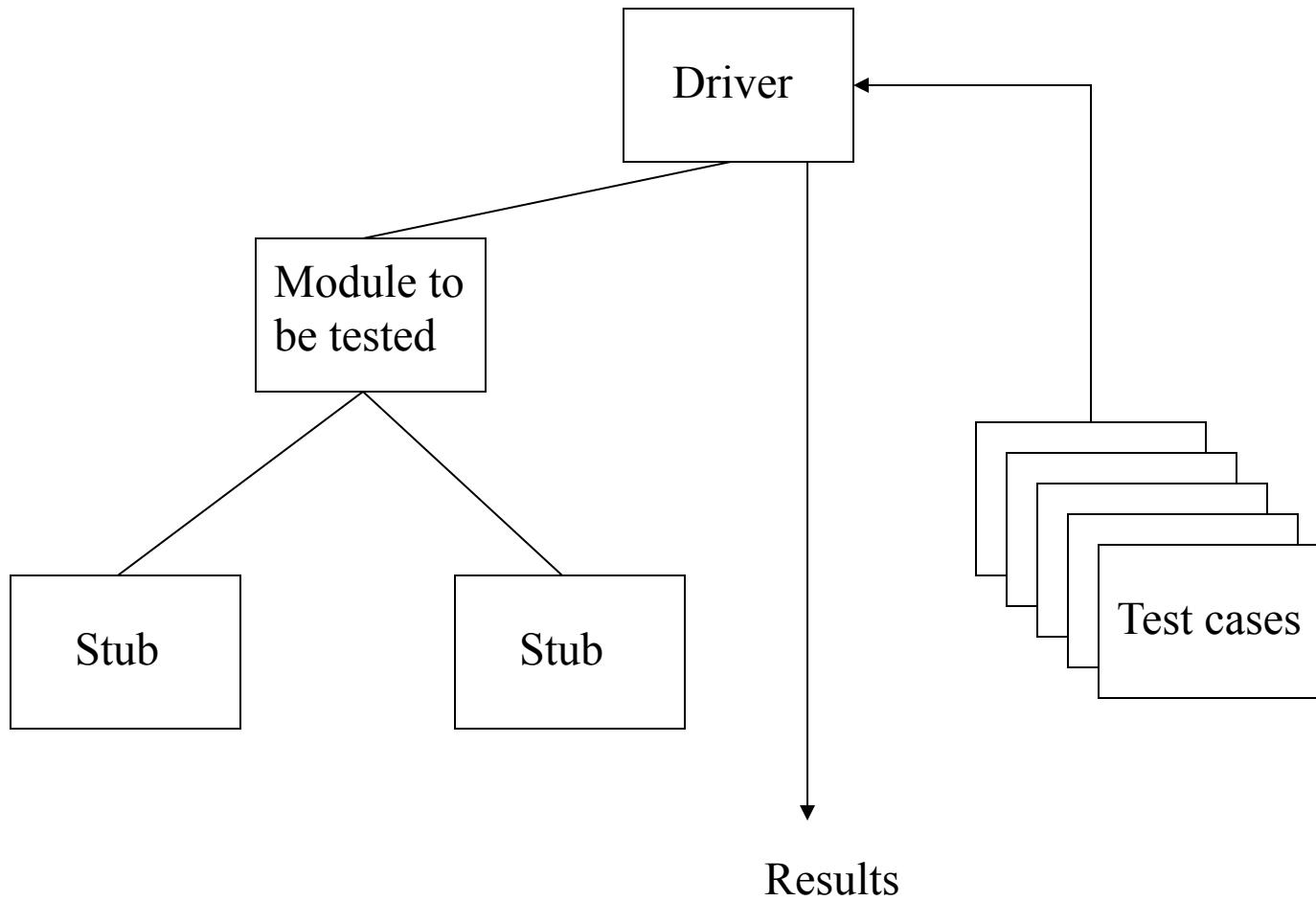
## Test Equivalence

- Two test cases are test equivalent if software fails (or passes) on both test cases
- Test equivalence is an equivalence relationship
  - A binary relation on a set that specifies how to partition the set into subsets such that every element of the larger set is in exactly one of the subsets
    - Reflexivity – For any test case  $t$ , we have  $t R t$
    - Symmetry - For any two test cases  $t_1$  and  $t_2$  at the same partition, if  $t_1$  fails  $t_2$  will fail and visa versa
    - Transitivity – For any three test cases  $t_1, t_2, t_3$  at the same partition, if  $t_1$  fails causes  $t_2$  to fail, and if failing  $t_2$  causes  $t_3$  to fail, then failing  $t_1$  causes  $t_3$  to fail

# Testing Terminologies

- Test oracle
  - A source of expected results for a test case
- Test driver
  - Nothing more than a “main program” that accepts test case data, passes such data to the component (to be tested) and prints relevant results
- Test stub
  - Serves to replace modules that are subordinate (called by) the component to be tested
  - A dummy subprogram that does minimal data manipulation, print verification of entry, and returns control to the module undergoing testing
- Both test driver and stub must be developed, but not delivered with the final product
- Test harness (Automated Test Framework)
  - A collection of software and test data configured to test a program unit by running it under varying conditions and monitoring its behavior
  - Essential for automating test activities

# Test Driver and Stubs



# Testing Terminologies

- Functional testing (Black box testing)
  - Generating test cases based on specs, design documents, user manual ,etc.
  - Answering: What test cases shall I use to exercise my program?
  - No source code is available
- Structural testing (White box testing)
  - Testing the internal structure of code
  - Source code must be available
- Gray box testing
  - Testing using both functional and structural techniques
- Alpha test
  - Testing product using real users at the developer's site
- Beta test
  - Testing product using real users at the user' s site

# Testing Terminologies

- Unit testing
  - Testing modules, routines, classes, etc.
- Integration testing
  - Testing interaction of units
- System testing
  - Testing whole programs as a system
- Acceptance testing
  - Testing conducted to approve the acceptance of system by client
- Regression testing
  - Testing for multiple releases of software
- Adequacy Criterion
  - A test adequacy criterion is a predicate that is true (satisfied) or false (unsatisfied) for a pair of {program, test suite}

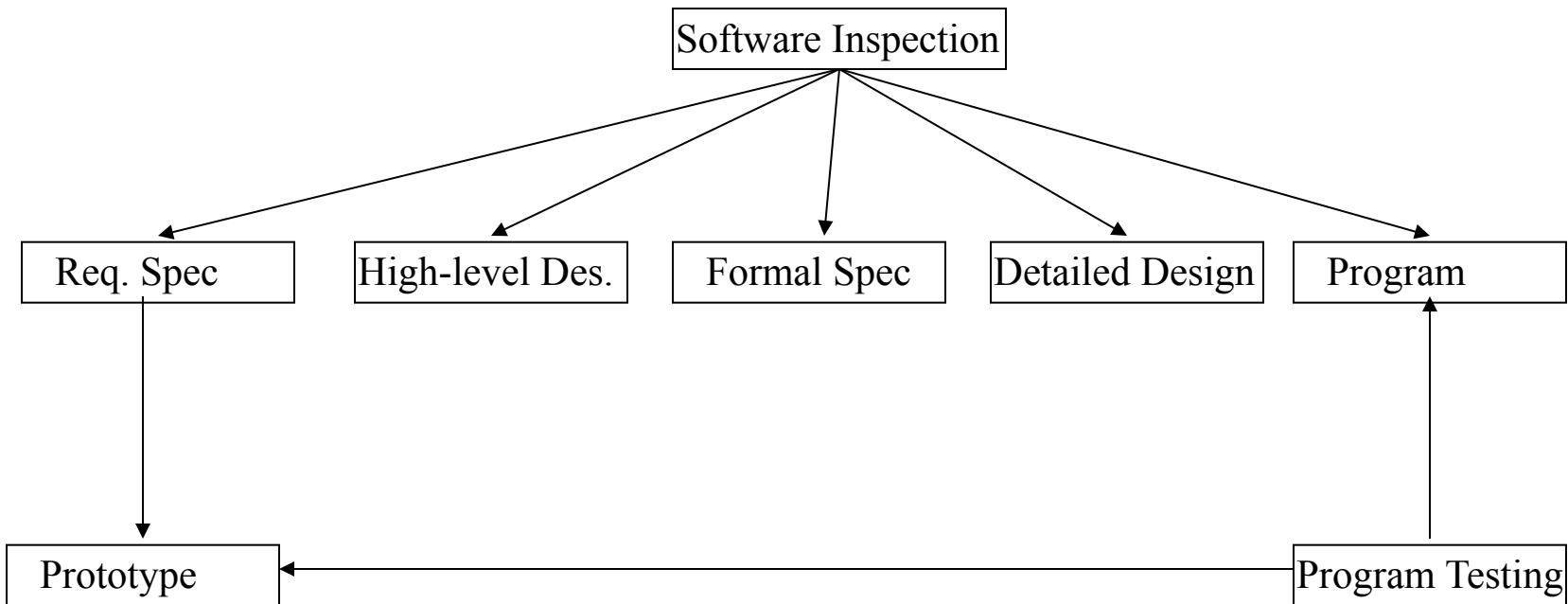
# Verification vs. Validation

# Quality Assurance vs. Quality Control

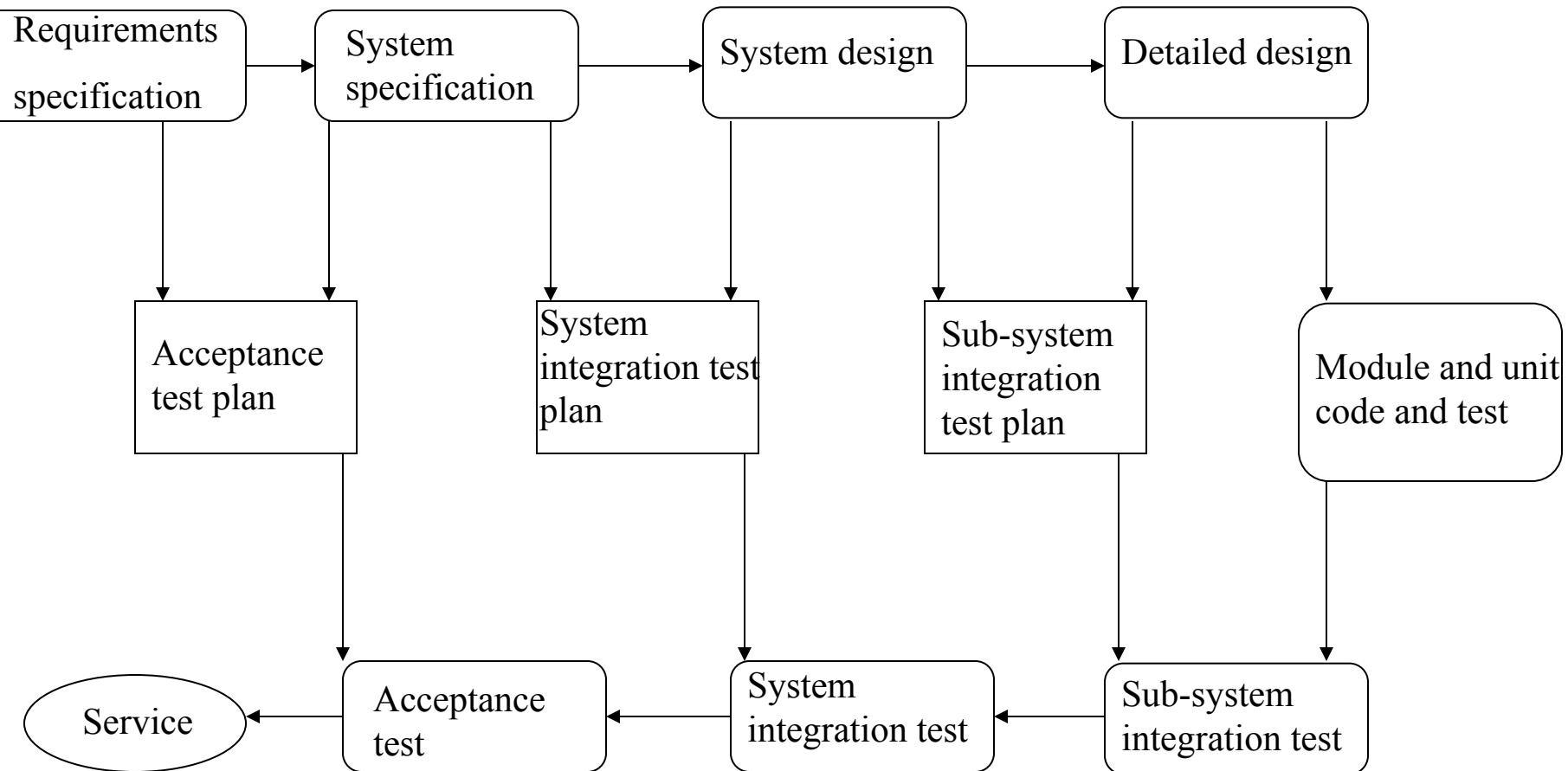
- Verification
- Quality assurance
  - A quality process to evaluate whether or not a product, service, document, or system complies with a regulation, standard, specification, or conditions imposed at the start of a development phase
  - Often an internal process
  - Are we building the thing right?
    - “Building it right” checks that the documented development process was followed
  - Takes place at the end of each phase
- Validation
- Quality control
  - A process of establishing documented evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements.
  - Often involves acceptance and suitability with external customers
  - Are we building the right thing?
    - “Building the right thing’ refers back to the user's needs
  - Takes place before delivery

Although in both V & V we conduct testing, software testing at the code level is what referred as validation

# Verification vs. Validation



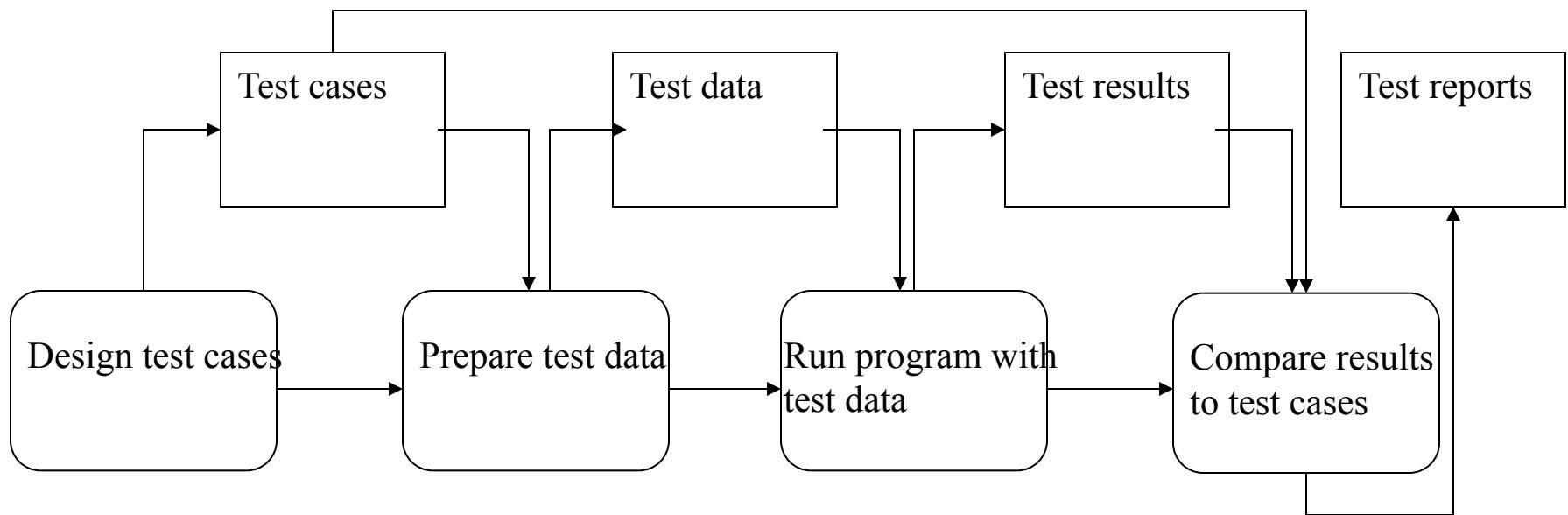
# Verification vs. Validation Planning



## Structure of Software Test Plan

- The testing process
  - A description of the major phases of the testing process
- Requirements traceability
  - Testing should be planned so that all requirements are individually tested
- Tested items
  - The products of the software process which are to be tested should be specified
- Testing schedule
  - An overall testing schedule and resource allocation for this schedule
- Test recording procedures
  - The results of the tests must be systematically recorded
- Hardware and software requirements
  - Required software tools and estimated hardware utilization
- Constraints
  - Constraints affecting the testing process such as staff

# A Simple Testing Process



## Types of Software Testing

- Dynamic analysis
  - Running the code with collected test cases
  - The only way to run actually the end product
- Static analysis
  - Checking correctness of the code with formal methods and theorem proving techniques
  - Usually conducted before dynamic analysis
  - Essential for safety critical system

## Test Case Generation

- Functional testing
  - Tests are derived from the program specification
- Structural testing
  - Tests are derived from the internal logic of the program