Software Testing Strategies

A Generic Software Testing

- Testing begins at the module level and works "outward" toward the integration of the entire computer-based system
- Different testing techniques are appropriate at different points in time
- Testing is conducted by the developer of the software and an independent test group
- Testing and debugging are different activities, but debugging must be accommodated in any testing strategy

Verification and Validation (V & V)

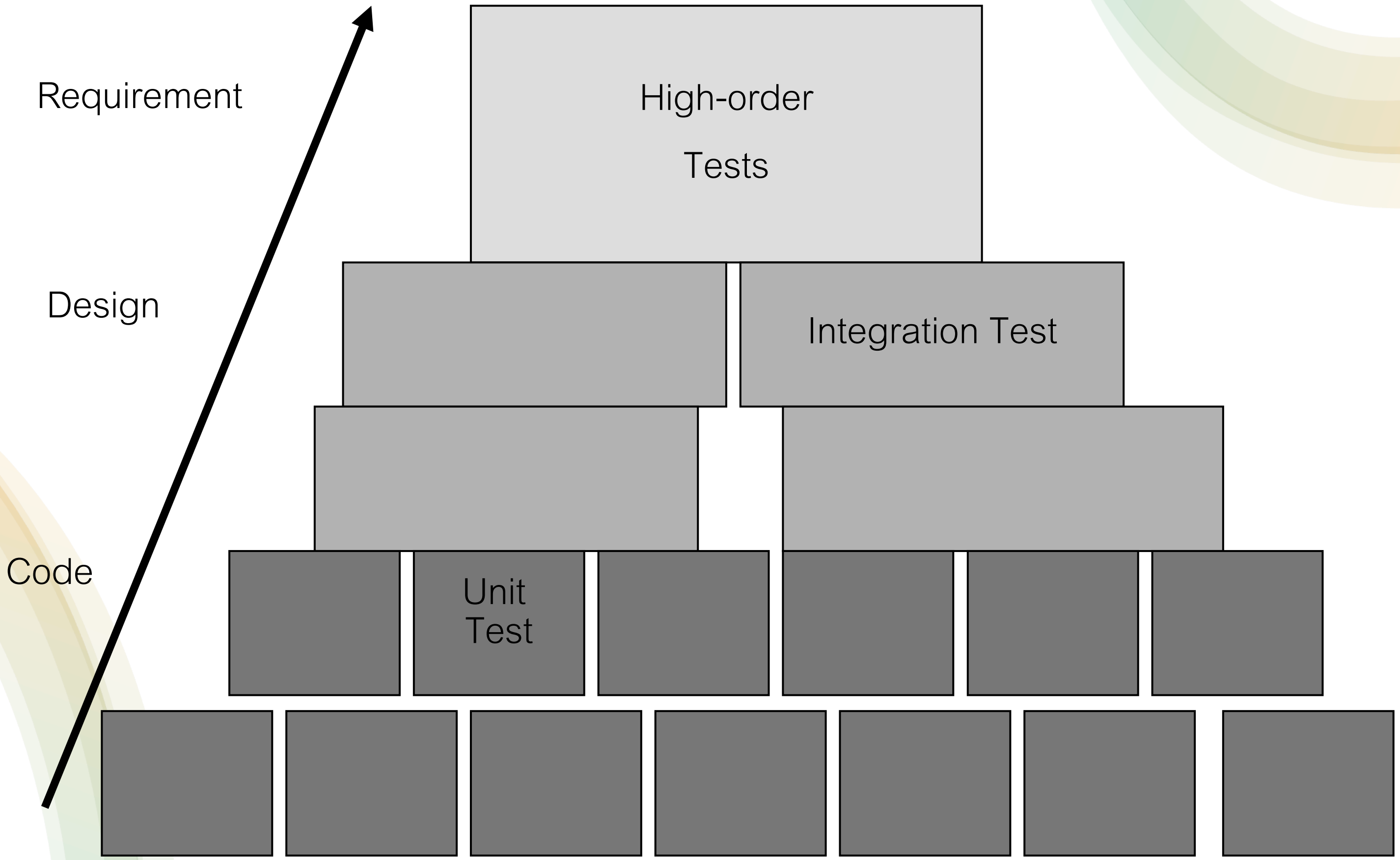
Verification refers to the set of activities that ensure that software correctly implements a specific function

Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements

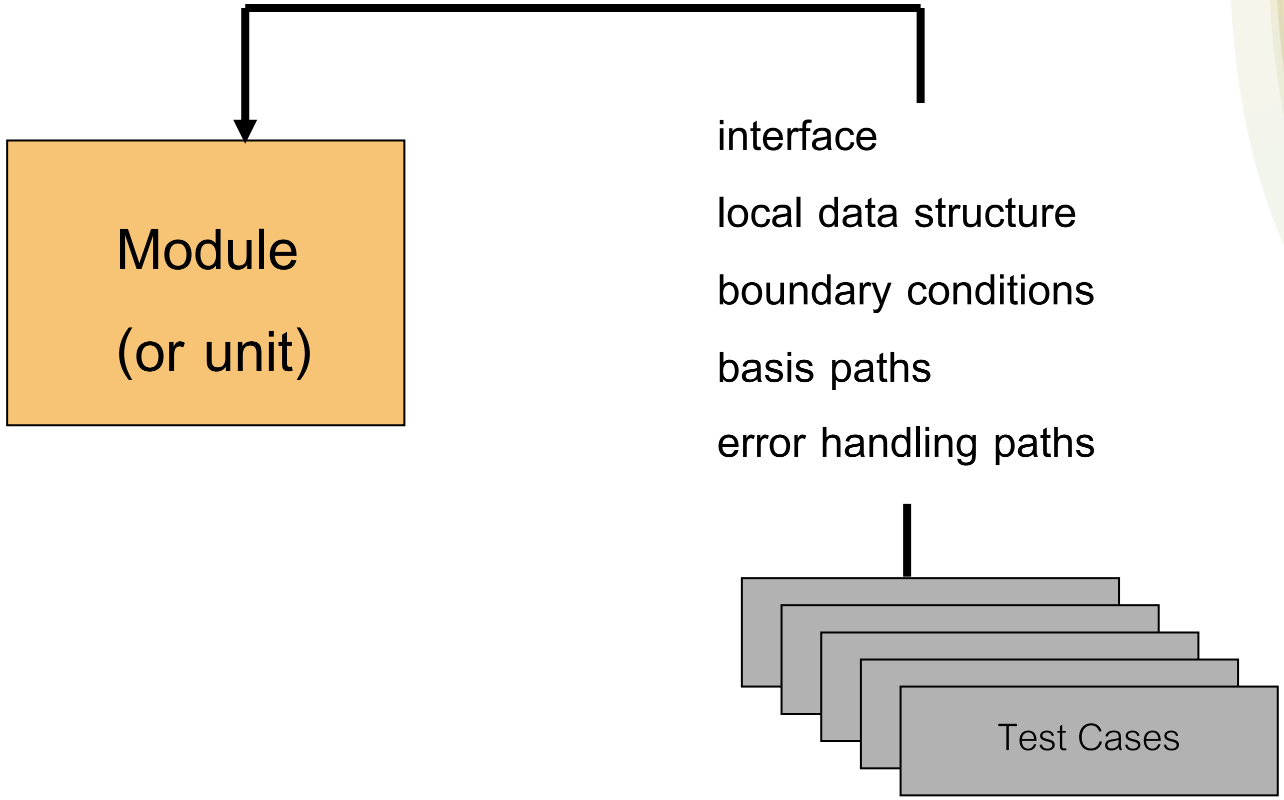
Boehm states that

- **Verification**: "**Are we building the product right**?"
- **Validation**: "**Are we building the right product**?"

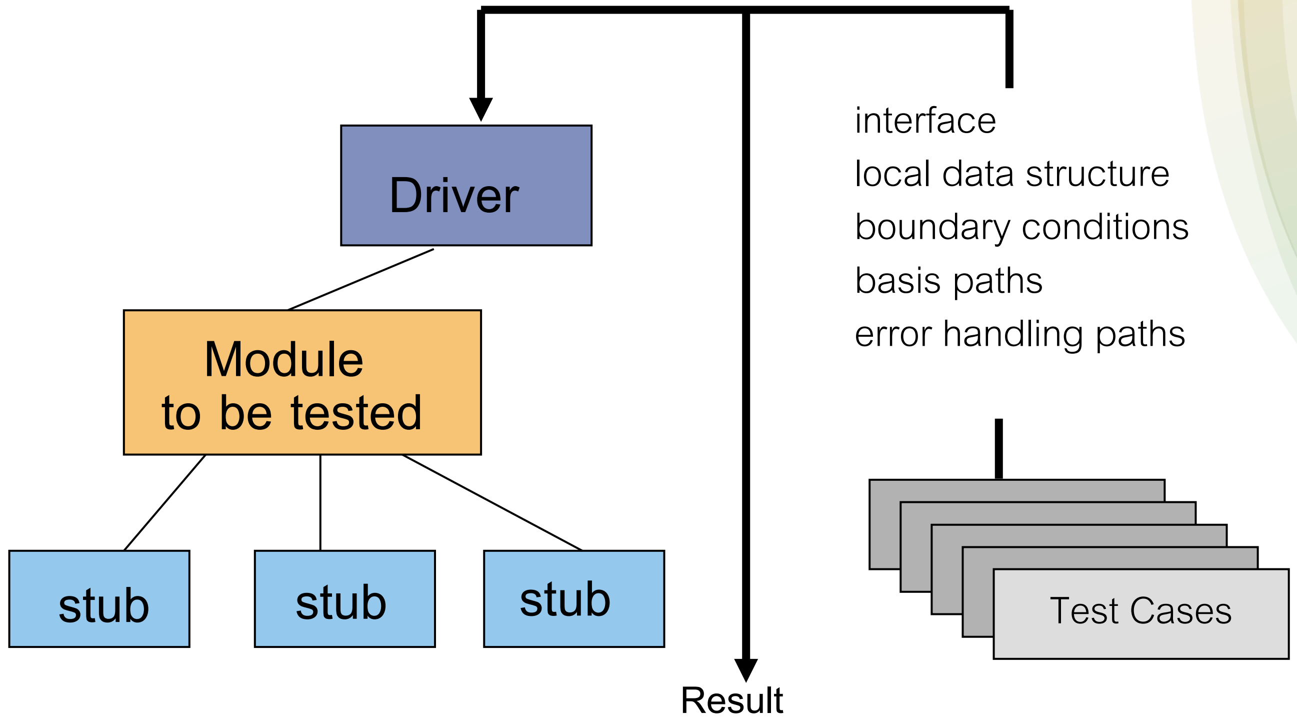
Software Testing Steps



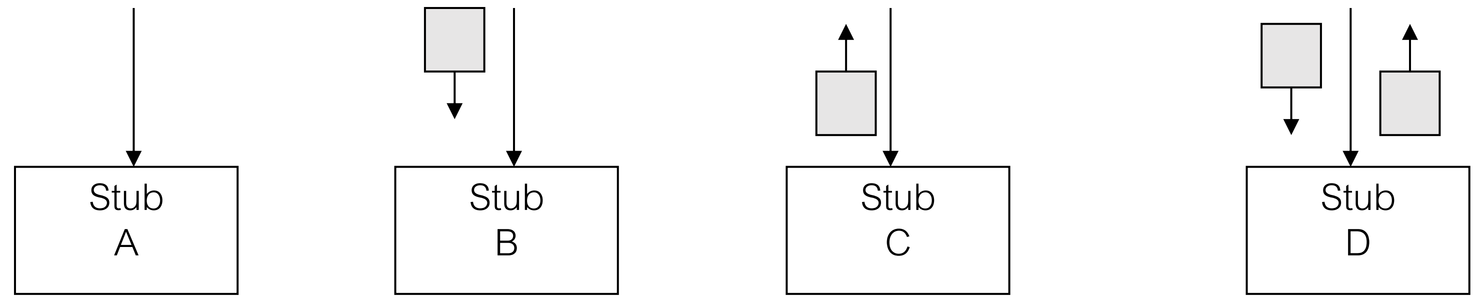
Unit Testing



Unit Testing Environment



Stubs' Complexity



Display
a trace message

Display passed
parameter

Return a value
from a table
(or external file)

Do a table
search foinput
parameter and
associated output
parametr

Integration Testing

Incremental approaches:

- top-down
- bottom-up
- sandwich

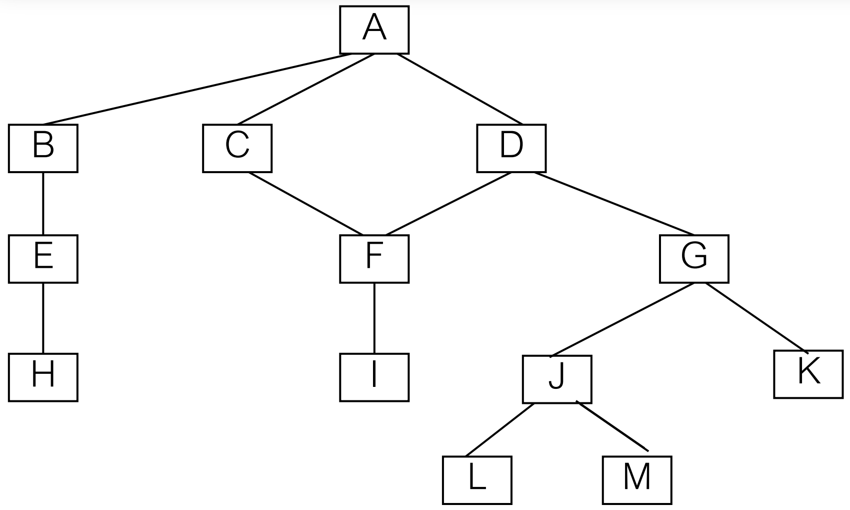
Non-Incremental

- big-bang

Integration Testing: Top Down Approach

- test main first - build stubs
- replace each **stub** with a module
- conduct the new tests and run the old tests
- a new module which has been added might be fail the successful test cases;
  - ➢the fault might be in either the new module or the interfaces between new the new module and the rest of the product

Integration Testing: Top Down Approach



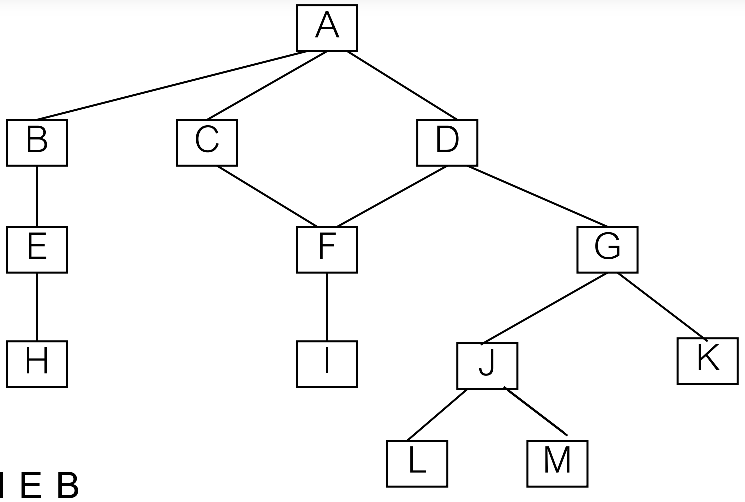
Breadth First: A B C D E F G H I J K L M

Depth First: A B E H C F I D G J L M K

Integration Testing: Bottom Up Approach

- lower level modules are combined into builds or clusters
- develop a **driver** for a cluster
- test the cluster
- replace drive with module higher in hierarchy

Integration Testing: Bottom Up Approach



- Build 1: H E B
- Build 2: I F C D
- Build 3: L M J K G
- after that Integrate Build 1, 2, and 3 with module A

Steps for Integration Testing

- All modules should be unit tested
- Choose integration testing strategy
- Do WB/BB, test input/output parameters
- Exercise all modules and all calls
- Keep records (test results, test activities, faults)

System Testing

- starts after integration testing
- ends when
  - we have successfully determined system capabilities
  - we have identified and corrected known problems
  - we confidence that system is ready for acceptance

Components of System Testing

- Requirement-based functional tests
- Performance Capabilities
- Stress or Volume tests
- Security Testing
- Recovery Testing
- Quality attribute - reliability, maintainability, integrity

Requirement-Based System Test

- to demonstrate that all functions are available
- test cases derived from requirements
- exercise all functions, classes of output, and system status
- all valid input data is accepted
- all invalid input data is rejected without system failure
- test interfaces to other systems

Requirement-Based System Test (cont.)

- look for systematic coverage: use a functional coverage matrix
- the coverage matrix is different from the unit-testing
- we are now planning the testing for a group of programs instead of a single one

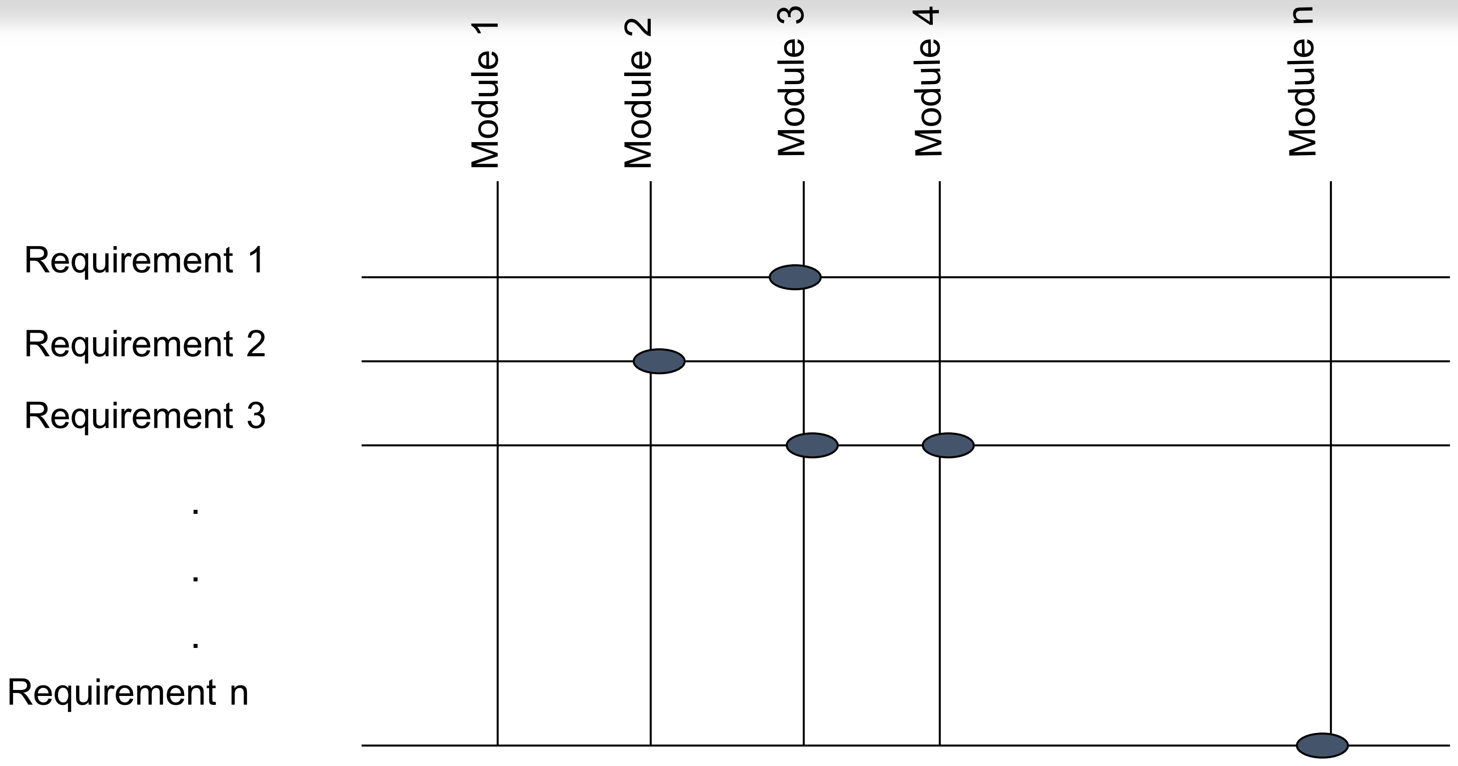
The Requirements Validation Matrix

- is matrix of requirements versus test cases
- to organize all requirements and ensure that tests are specified for each of them
- The matrix lists each requirement and refers to the test cases or situations that have been created to test it

The Requirements Validation Matrix

Requirement No.	Requirements Description	Test Cases ID	Status
1		87,88,89	
2		81-88,102	
3			
4		103-106	

A Conceptual Traceability Matrix



Performance capability tests

- examine performance limits and show that performance objectives are met
- evaluate parameters such as: respond time, memory requirements, run-time requirements, and file sizes
- need source of transaction, lab, and instrumentation - hardware or software to monitor
- look for hardware or software units that limit performance

Stress or Volume Testing

- is designed to confront the system with abnormal monitors
- drive the system to its limit and determine whether it breaks down
- first test to specification, then break and analyze
- determine how many transactions or records can be operationally supported
- demand resources in abnormal quality, frequency, and volume

Recovery and Security Testing

- Recovery Testing:
  - ➢ to confirm that the system with switchover capabilities resumes processing without transaction compromise
  - ➢ look for transaction fidelity
  - ➢ can the system recover from any situations that make the system crash
- Security Testing
  - ➢ test security issue of the system

Acceptance Testing

- to provide clients/users with confidence and insure that the software is ready to use
- begins when system test is complete
- test cases are subset of the system test
- acceptance tests are based on functionality and performance requirements
- take typical day’s transactions, month or year of operation

Acceptance Testing (cont.)

- is usually complete when the client is satisfied
- is formal and held to a predefined schedule and duration
- we need to test for a replacement system
- acceptance test run for customized software product but if we develop software for many users, alpha and beta test are required

Constraints on Acceptance Testing

- tests must be run on operational hardware and software
- tests must stress the system significantly
- all interfaced systems must be in operation throughout the test
- the test duration should run a full cycle
- tests should exercise the system over a full range inputs
- all major functions and interfaces should be exercised
- if running the entire test cannot be completed, a new run should include a complete start

Alpha Test

- conduct at developer's site
- invite users
- developers interact with users
- record errors, and usage problems

Beta Test

conduct at customer site by users

developer usually not present

users record and report problems, developers fix them and then release

Regression Test

involves a retesting of software after changes have been made to insure that its basic functionality has not been affected by the changes

insure that no new errors have been introduced

involves rerunning old test cases

automation is necessary since this process can be very time consuming

Debugging
(Fault localization)

- when tests found faults or defects
- is programmer's responsibility
- purpose of debugging
  - ➢locate fault (find causes and prevention)
  - ➢correct it
  - ➢retest to ensure that you have removed bug and not introduced other

Bug Consequences

- **Mild** - misspell output, lack of white space
- **Moderate** - output may be misleading or redundant
- **Annoying** - users need tricks to get system to work
- **Disturbing** - refuses to handle legitimate transaction
- **Serious** - looses track of transaction and its occurrence
- **Very serious** - bug causes system to incorrect transaction

Bug Consequences (cont.)

- **Extreme** - problem limited to a few user or a few transaction type, frequent, and arbitrary
- **Intolerable** -long term unrecoverable corruption of database, system shutdown may need to occur
- **Catastrophic** - system fails
- **Infectious** - corrupts other systems, system that causes loss of life