Software Testing

IF WE BUILD OUR SOFTWARE WITH NO BUGS, WE CAN MAKE A 10% RETURN ON OUR INVESTMENT.
10%
BUT IF WE DO A POOR JOB, WE CAN MAKE A 40% RETURN BY SELLING UPGRADES AND SERVICE.
BUT DON'T WORRY. WE ONLY HAVE THE BUDGET FOR A POOR JOB.
I CAN'T REMEMBER IF WE'RE CHEAP OR SMART.
PHEW!
Dilbert.com DilbertCartoonist@gmail.com
2·3·11 ©2011 Scott Adams, Inc./Dist. by UFS, Inc.

CAN YOU TEST THE SOFTWARE TODAY?
NO, I'M MAKING MAJOR CHANGES TUESDAY.
YOU COULD TEST THE CURRENT VERSION.
I WISH PEOPLE WOULDN'T SLAP THEIR FOREHEADS AND SAY "AYE-YI-YI-YI" EVERY TIME I TALK.
www.dilbert.com scottadams@aol.com
9/21/00 © 2000 United Feature Syndicate, Inc.

OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTERNOON!
S. Adams E-mail: SCOTTADAMS@AOL.COM
11/13 © 1995 United Feature Syndicate, Inc.(NYC)

Outline

- What is testing?
- Why test?
- Who are testers?
- When to test?
- How to approach testing?
- Where in the lifecycle to test?

What is testing?

PAIR UP AND TAKE 5 MINS

Testing Definitions

- Testing is an organized process of identifying discrepancies (variance between the actual and expected results)
- The objective of testing is to systematically uncover errors with a minimum of time and effort
- Testing is a trade-off between risks and economics

Testing Definitions (Cont.)

- Testing is the process of executing a program with the intent of finding errors
- Establishing confidence that a program does what it is supposed to do
- Any activity aimed at evaluating an attribute or capability of a program or system

Testing Definitions (Cont.)

- Testing is the process of demonstrating that errors are not present (not possible)
- Testing is a destructive Process??!!
  - What does this mean?
- Testing is a sadistic Process??!!
  - What does this mean?

Testing Limitations

- Systems and Programs can not be completely tested
- There is no such thing as a debugged system

False Beliefs About Testing

- Testing is easy, or at least a lot easier than design and programming
- Anyone can do testing
- No prior training or experience is needed
- Improvements in tools and processes eliminates the need for test
- Errors are just bad luck

Why test?

PAIR UP AND TAKE 5 MINS

Testing Fundamentals - 1

- Software Testing is a critical element of producing quality software.
- It represents the ultimate review of the requirements specification, the design, and the code.
- It is the most widely used method to insure software quality.
- Many organizations spend 40-50% of development time in testing.

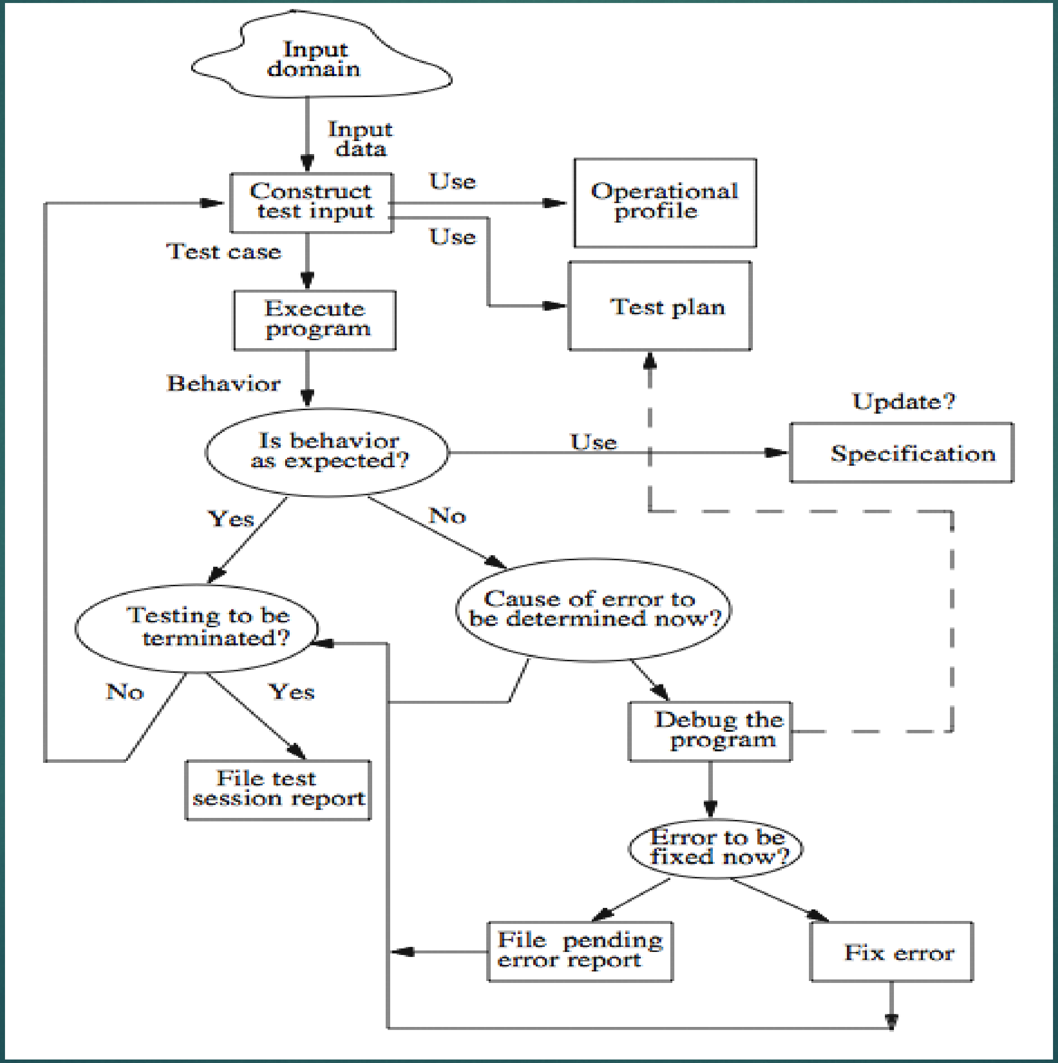
Testing Fundamentals - 2

- Testing is the process of determining if a program has any errors.

- Debugging is concerned with finding where defects occur (in code, design or requirements) and removing them (fault identification and removal).

- Even if the best review methods are used (throughout the entire development of software), testing is <u>necessary</u>.

A test/debug cycle



Testing Fundamentals - 3

- Testing is the one step in software engineering process that could be viewed as destructive rather than constructive.

- A successful test is one that uncovers an as yet undiscovered defect.

- For most software systems, exhaustive testing is impossible.

Who are testers?

PAIR UP AND TAKE 5 MINS

Are Testers Different?

- Look for failures, rather than successes in software
- Imagining ways to go wrong, rather than right
- Searching for weakness, rather than strengths

Are Testers Different? - 2

- Detail-oriented, good communication skills
- Are skeptical, will check everything themselves
- Understand the domain and requirements.

Testers Attitude

- Testers hunt errors
- Testers are destructive in a creative way
- Testers pursue errors and not people
- Testers add value

Testing Principles

- Test cases must be written for invalid and unexpected, as well as valid and expected input conditions
- Thoroughly inspect the results of each test
- The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that testing

Testing Principles - 2

- Testing is an extremely creative and intellectually challenging task
- A good test case is one that has a high probability of detecting an undiscovered error
- Examining a program to see if it does not do what it is supposed to do is only half of the battle

Testing Principles - 3

- The quality of the test process determines the success of the test effort (testing life cycle)
- Prevent defect migration by using early life-cycle testing techniques
  - More than half of the errors are usually introduced during the requirement phase. How do we do this?

Testing Principles - 4

- We must take responsibility for improving the testing process, including use of tools
- Testing is a professional discipline requiring trained skilled people (not an entry position)
- Cultivate a positive team attitude of creative destruction, “test to break”

How Do Testers Detect Errors

- By examining the internal structure and design
- By examining the functional user interface
- By examining the design objectives
- By examining the users’ requirements
- By executing code
- …

When to test?

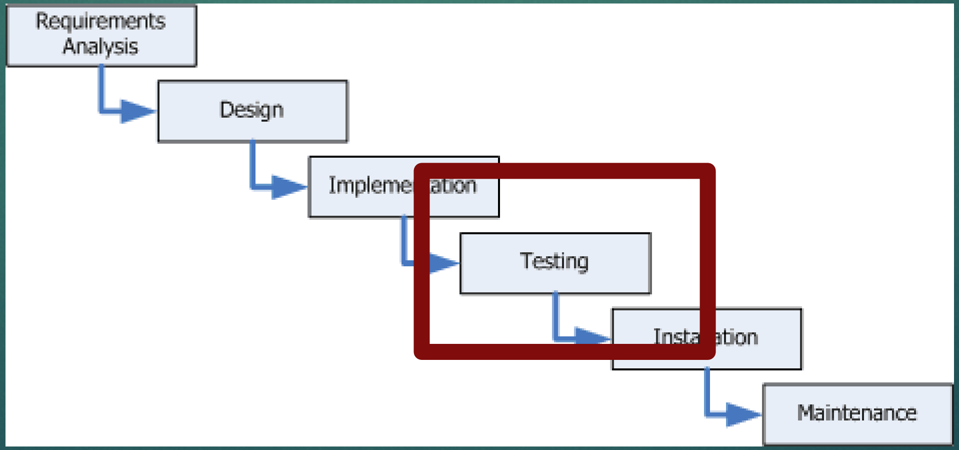
PAIR UP AND TAKE 5 MINS

An evolutionary view of testing!

Show it works	(demonstration)	1950’s
Find defects	(detection)	1970’s
Assess Quality	(evaluation)	1980’s
Control Quality	(prevention)	1990’s
Test Driven Development	(prevention)	2000’s

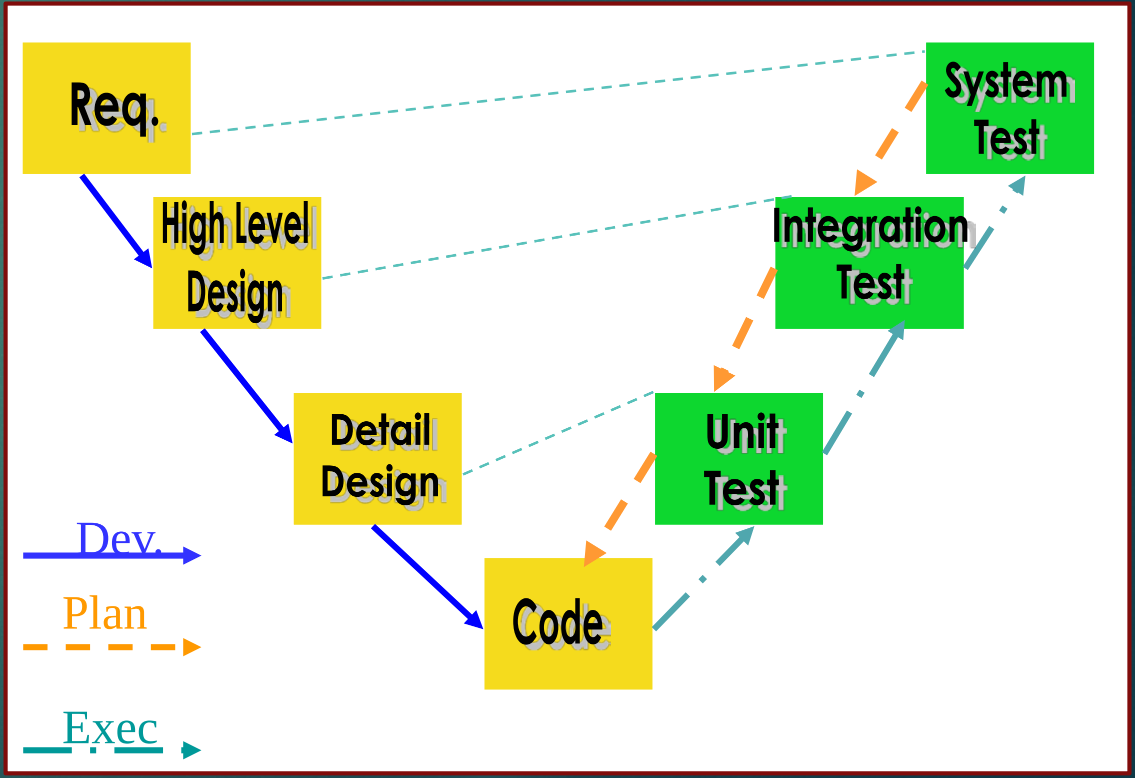
Three Views of Testing

- 1. Traditional (old Waterfall)
  - Late in the development process, after the design and coding
  - Mostly focused on the code



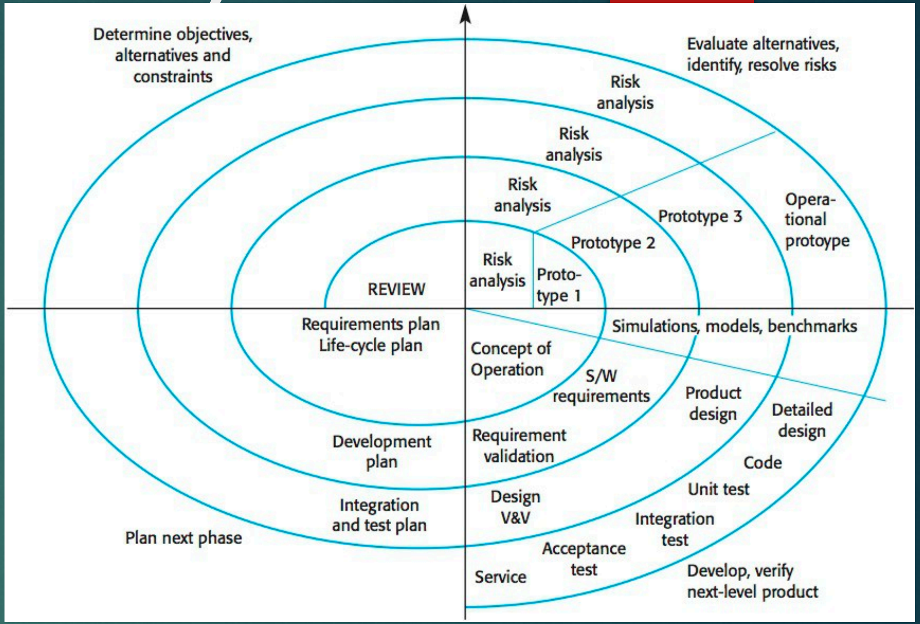
Three Views of Testing (Cont.)

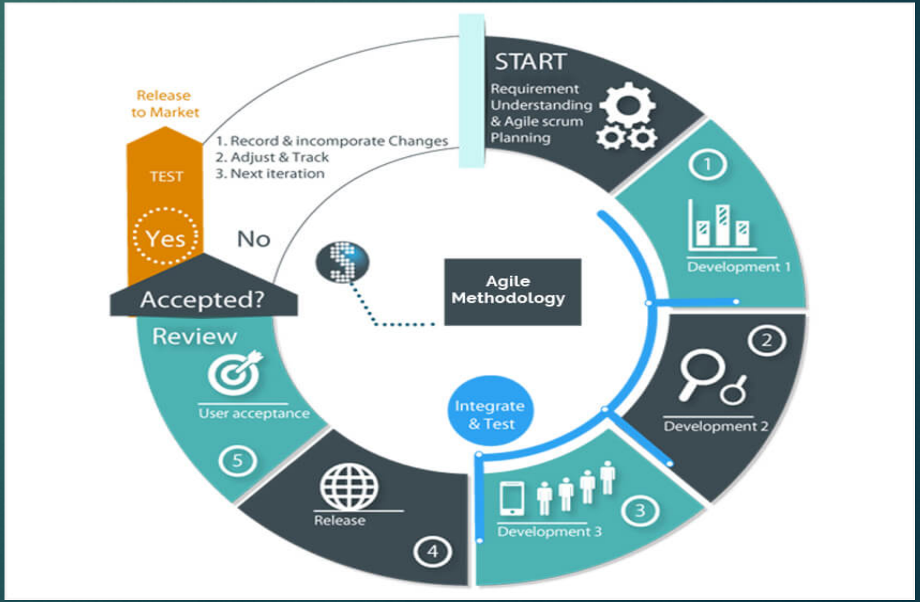
- 2. Comprehensive Testing (Waterfall, V model)
  - Test development starts very early in development process
  - Test planning parallels analysis, and test design parallels design
  - Substantial user involvement
  - Advantages
    - Earlier, better test plan
    - Quality of spec. & design increases due to feedback



Three Views of Testing (Cont.)

- 3. Latest View (waterfall, V, spiral, agile)
  - Test and quality assurance starts early
  - Each step and component is validated before moving ahead
  - Walkthroughs and inspections are utilized routinely
  - Total quality management (TQM)
  - On going process improvement





Basic Forms of Testing

- Full Testing (Preventive Testing)
  - Starts no later than Requirements phase and continues through acceptance testing
- Partial Testing
  - Begins after functional design
  - Less effect on Requirements and functional design
- Endgame Testing (Acceptance Testing)
  - A validation oriented approach
  - No influence on requirements and design

Preventive Testing

Preventive Testing

- A new way of thinking
- Test driven development
- Using testing to influence and control software designs and developments.
- Implicit concurrent engineering of testware and software

Preventive Testing (Cont.)

- “The preventive testing is built upon the observation that one of the most effective ways of specifying something is to describe (in detail) how you would recognize (test) the something if someone ever gave it to you.”

  -Bill Hetzel

Preventive Testing (Cont.)

- “The defect that is prevented doesn’t need repair, examination, or explanation. The first step is to examine and adopt the attitude of defect prevention. This attitude is called, symbolically: Zero Defects.”

  -Philip Crosby

Key Elements of Preventive Testing

- Tests are used as requirements models
- Testware design leads software design
- Defects are detected earlier or prevented all together
- Defects are systematically analyzed
- Testers and developers work together

Preventative Testing Steps

- Plan Strategy
  - Identify objectives
  - Develop detail test plan
    - (Level/Phase/Sprint Plan)
- Perform Test
  - Design tests
    - (Cases, Procedures and supporting Testwares)
  - Execute Test
- Measure
  - Collect and analyze the result

How to approach testing?

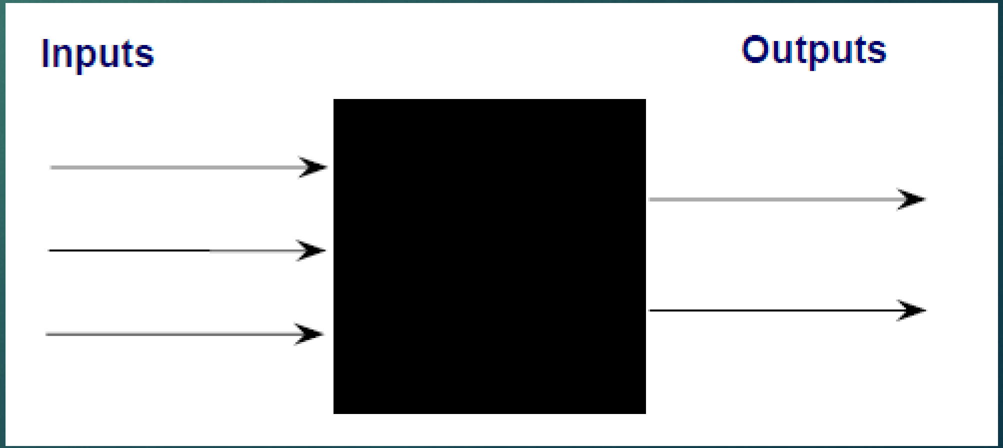
PAIR UP AND TAKE 5 MINS

Approaches to Software Testing

- Testing **strategies** defines in what context to perform tests.
- Testing **methods** defines how to design test cases.
- Testing **techniques** defines how to find the defects.
- Testing **levels** defines at what level of development the testing is conducted.

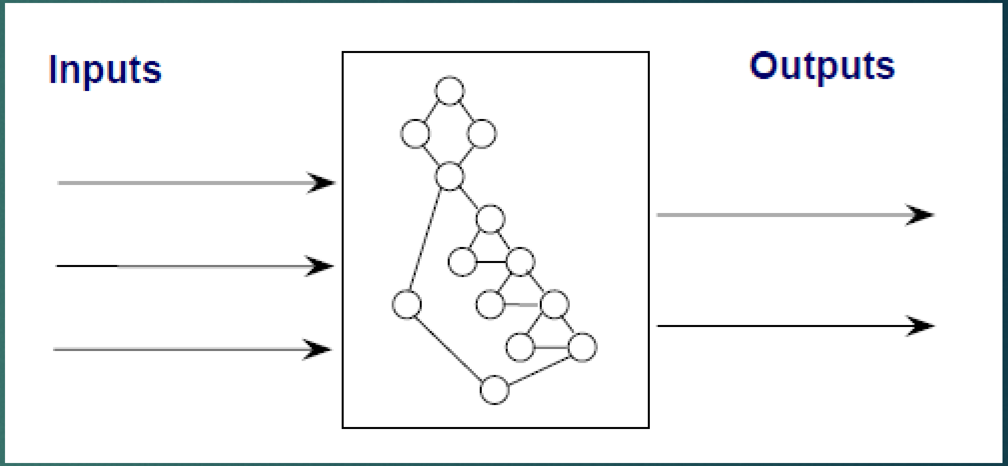
Testing Methods

- Black Box (Functional) Testing
  - Testing is based on the external behavior of the product.
  - Requires no insight to how the internal structure of software behaves
  - Based on specification and inputs



Testing Methods - 2

- White Box (Structural) Testing
  - Test cases are based on internal structure of the program and a specific level of coverage.
  - Examines the internal workings of a program, in order to assure that the internal operation performs according to specification, and “all” internal components have been exercised.
- Gray Box Testing
  - Some combination of both



Pair up and take 8 minutes: Compare both methods

Black-box

- Advantages:

- Disadvantages

White-box

- Advantages

- Disadvantages

Comparison of both methods

Black-box

- Advantages:
  - Test cases can be developed earlier
  - Faster & cheaper
  - No need for technical understanding of the code
  - No need for test case modification if code change
- Disadvantages
  - Failed test cases show presence of defects but no guidance where or why
  - No assurance of coverage?

White-box

- Advantages
  - Failed test cases will show where the code failed (why, where)
  - Can show coverage of code
  - Higher confidence on quality of code
- Disadvantages
  - Require fully implemented code to develop test cases
  - Much difficult to develop test cases since it requires an analysis of the structure of code
  - Need for test case modification if code change
  - Cost more

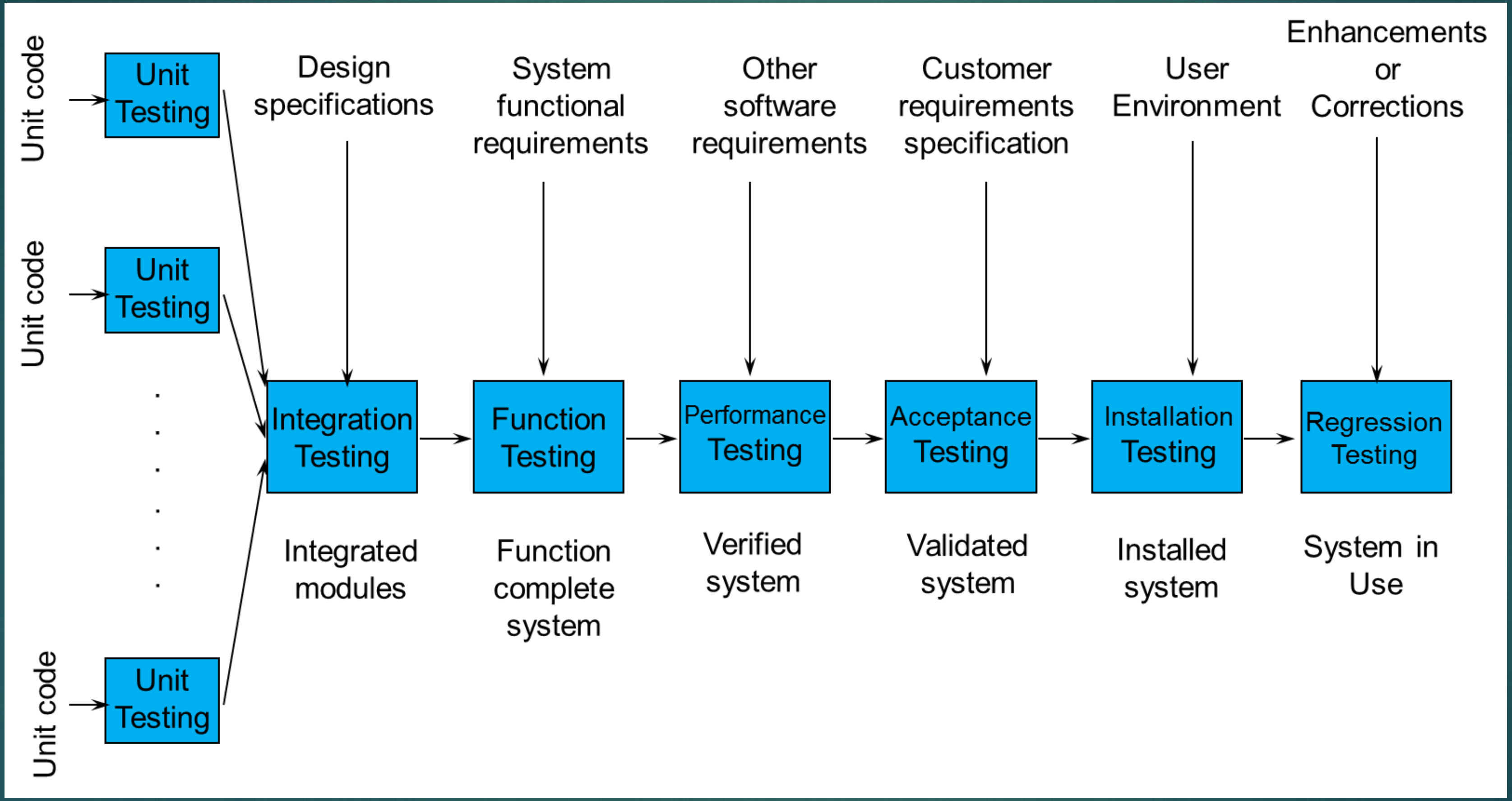
Black-box and White Box Testing

- We’ll cover specific techniques next week

Where in the lifecycle to test?

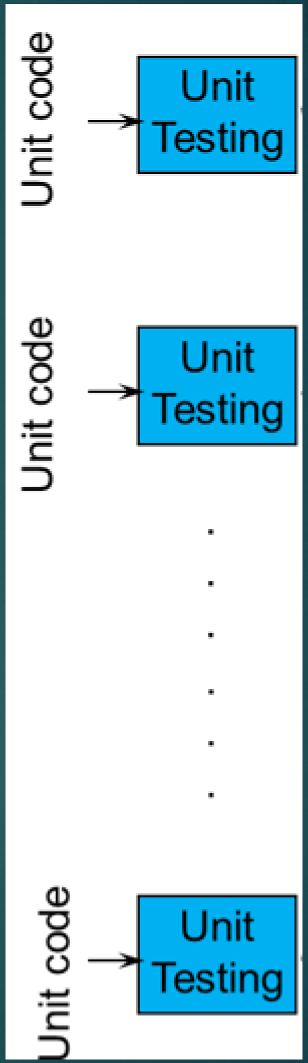
PAIR UP AND TAKE 5 MINS

Different levels of testing



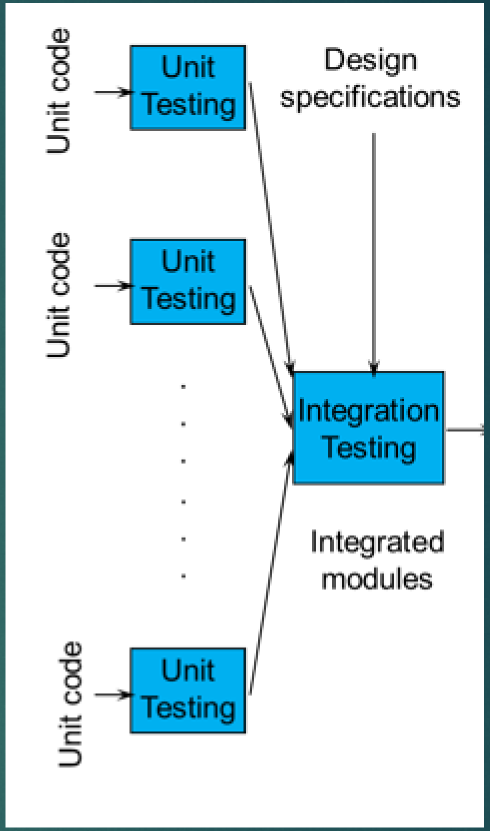
Testing Level

- Unit Testing
  - Remove defect at the earliest stage of implementation
  - Smallest testable element (module, object, etc.)
  - Usually written by the developer
  - Both static and dynamic
    - Static (Code Review, Data Flow, etc.)
    - Dynamic (Code Coverage, Condition Coverage, etc.)
  - Usually white box testing, some black box



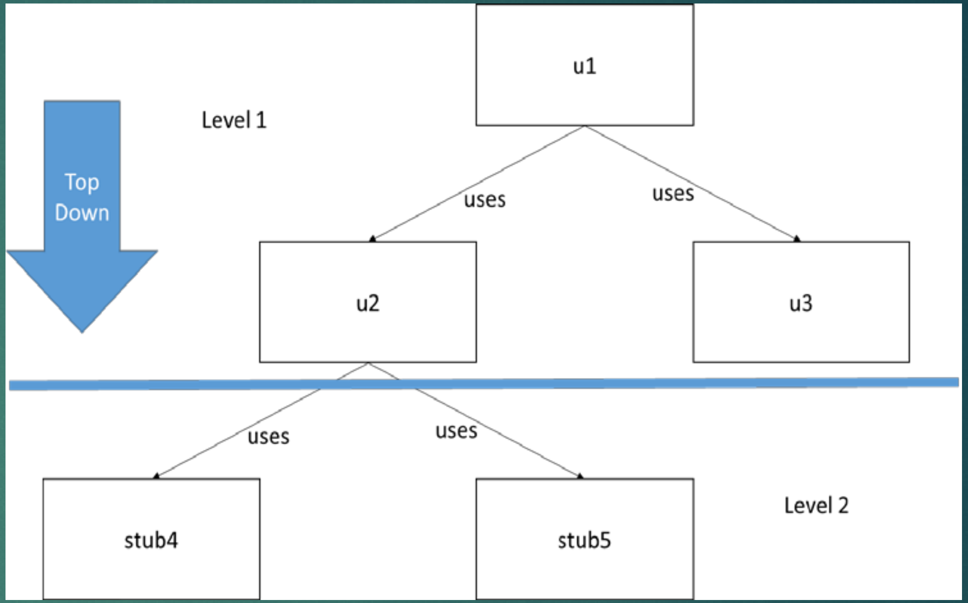
Testing Level - 2

- Integration Testing
  - Testing the interfaces between the units
  - Integration can be:
    - Top-down
    - Bottom-up
    - Big-bang



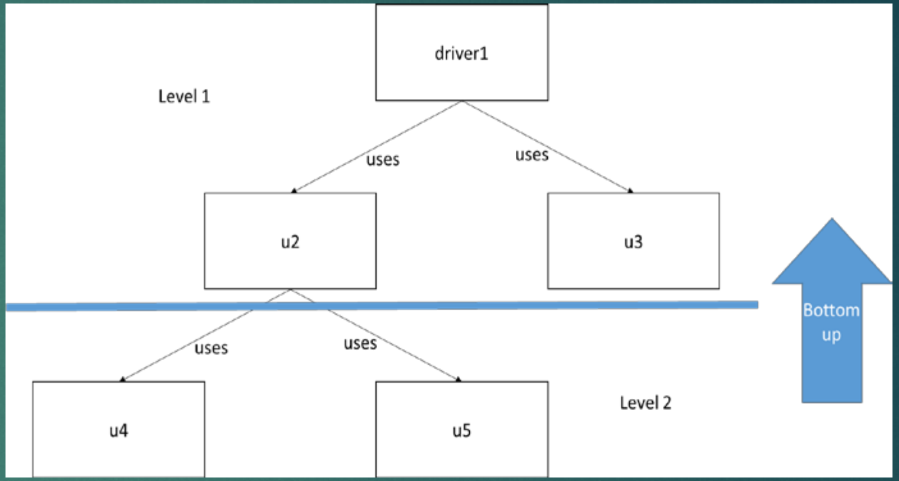
Top-Down

- Units are integrated from highest level to lowest
- “Stubs” are used to provide the necessary data to the higher level components



Bottom-up

- Lowest level units are integrated and tested then joined with higher level components
- Drivers are used to pass communication between low level components.

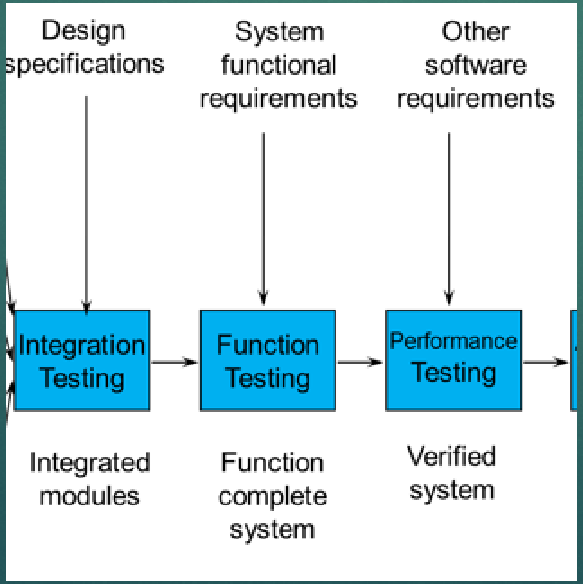


Big-bang

- All units are integrated at once
- Then tested.
- Only practically applicable for small systems
  - Why is that?

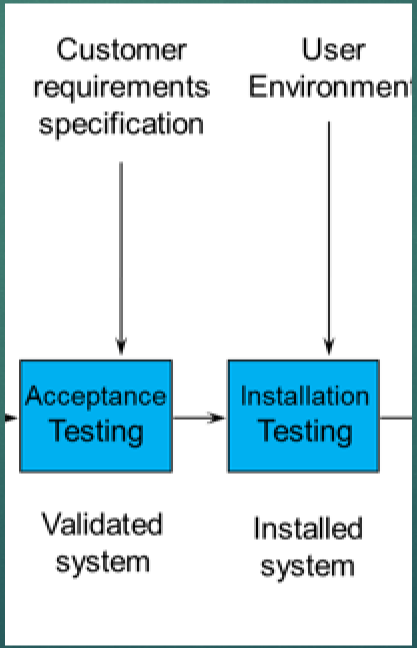
Testing level - 3

- System Testing
  - Testing the fully integrated system
  - To show the system meets the system requirements
  - Mostly black box testing, some white box



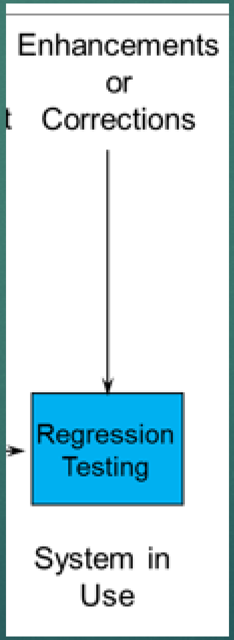
Testing level - 4

- Acceptance Testing
  - Testing the system for operational readiness
  - Mainly concentrating on non-functional requirements
  - Black box testing



Testing level - 5

- Regression Testing
  - Testing the system when a change has been applied
    - Assuring the change did not introduce new defects.
  - White-box testing.


Enhancements or Corrections
Regression Testing
System in Use