CISC/CMPE 327 Software Quality Assurance

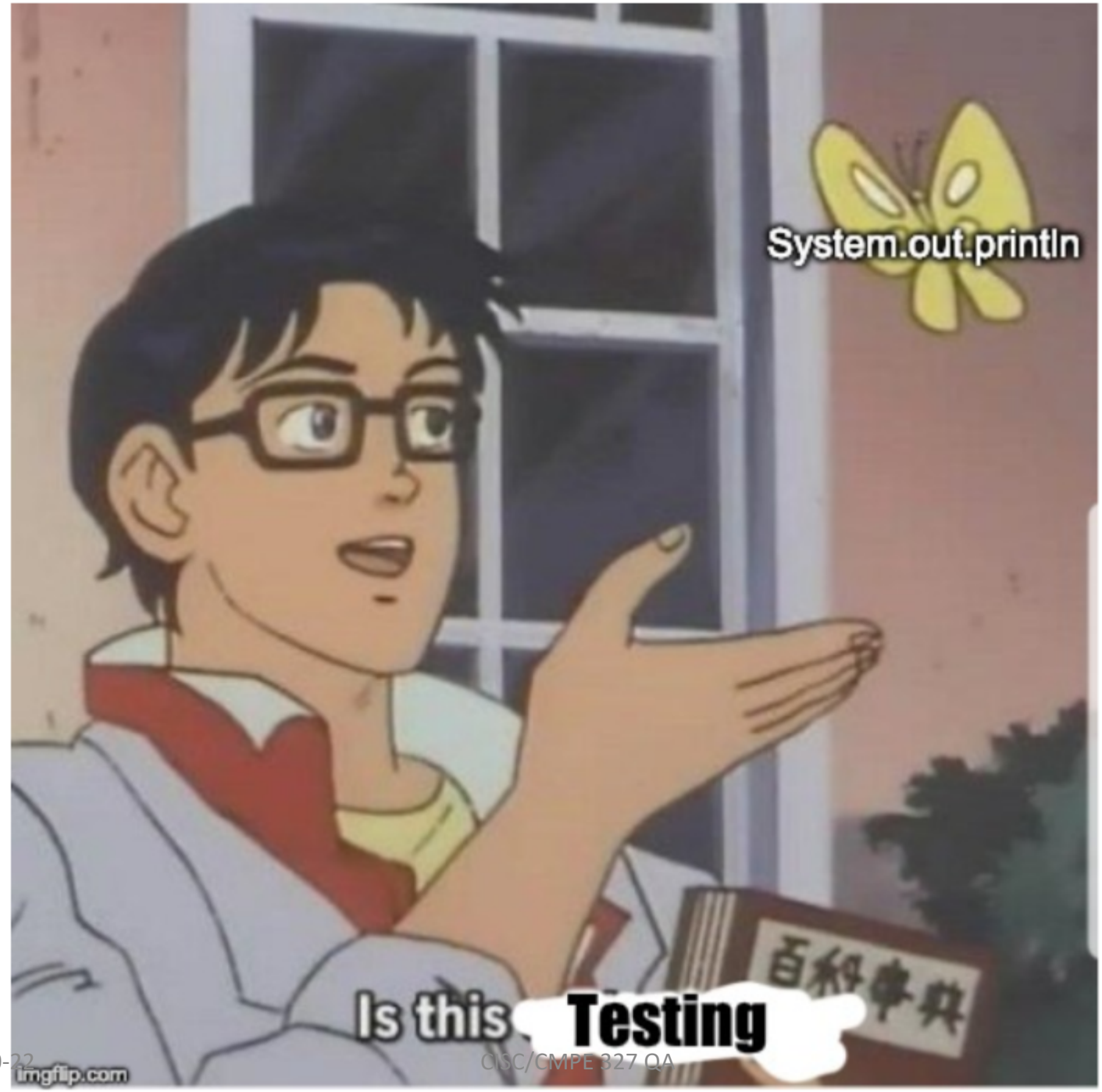
Queen's University, 2019-fall

Part II-1

Systematic Testing

Introduction to Systematic Testing

- Outline
  - Today we begin a thorough look at software testing
  - Definitions: What is software testing?
  - Levels of specifications
  - Levels of testing:
    unit, integration, system, acceptance
  - Types of testings


System.out.println
Is this
Testing
imgflip.com

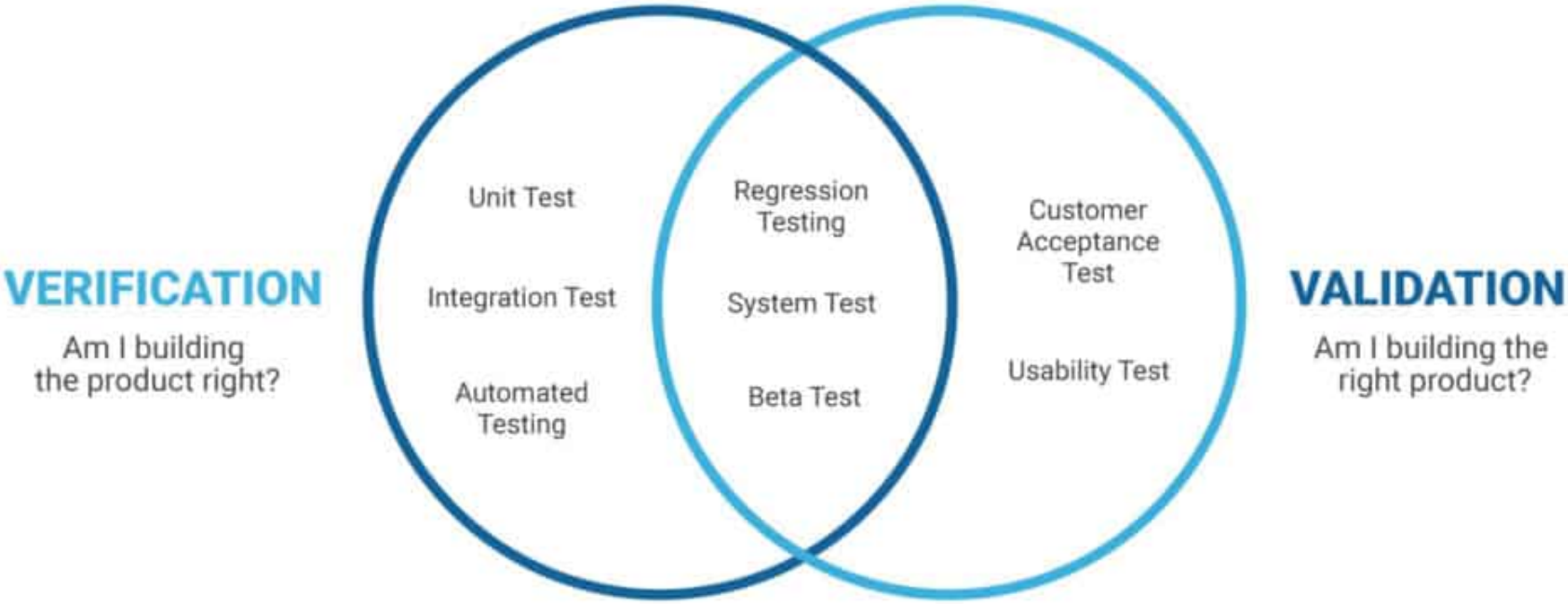
What is Testing?

- Testing is the process of executing software in a controlled manner to answer a question:
  - "Does the software behave as specified?"
- Specification
- Properties
- Testing is often associated with the words validation and verification

Verification vs. Validation

- Verification
  - Given a specification
    - Answers the question "are we doing the job right?"

- Validation
  - Answers the question "are we doing the right job?"


VERIFICATION
Am I building the product right?
Unit Test
Integration Test
Automated Testing
Regression Testing
System Test
Beta Test
Customer Acceptance Test
Usability Test
VALIDATION
Am I building the right product?

Testing vs. Debugging

- Debugging is not Testing
  - Debugging -> analyzing and locating bugs [when something wrong]
  - Testing -> methodically searching for and exposing bugs

Debugging -> supports testing but cannot replace it

What is Systematic Testing?

- An explicit discipline or procedure (a **system**) for
  - choosing and creating test cases
  - executing the tests and documenting the results
  - evaluating the results, ~~possibly~~ **automatically**
  - deciding when we are done (enough testing)

What is Systematic Testing?

- Testing is at best complete
  - impossible to ever test completely
- Chooses a particular point of view and tests only from that point of view (the test criterion)
  - e.g., test only that every decision (if statement) can be executed either way

Levels of Specification

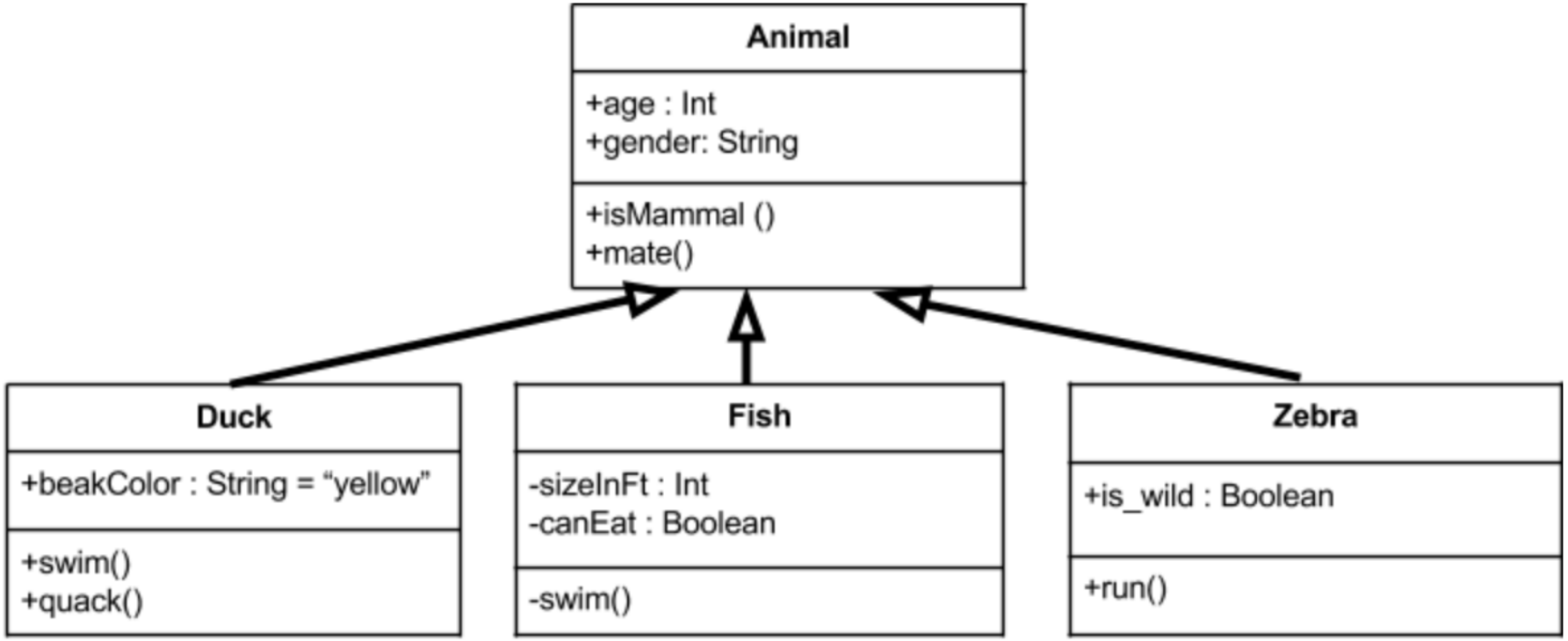
- Three Levels
  1. Functional specifications (or requirements)
  2. Design specifications
  3. Detailed design specifications

Levels of Specification

- 1) Functional specifications (requirements)
  - Precise description of the required behaviour (functionality) of the system
  - What the software should do, not how it should do it
  - Click "Exit" -> "Save" dialog if "has not been saved" -> otherwise "exit"

Levels of Specification

- 2) Design specifications
  - Describe the **architecture** of the design to implement the functional specification
  - Describe the components of the software and how they are to relate to one another


Animal
+age : Int
+gender: String
+isMammal ()
+mate()
Duck
+beakColor : String = “yellow”
+swim()
+quack()
Fish
-sizeInFt : Int
-canEat : Boolean
-swim()
Zebra
+is_wild : Boolean
+run()

Levels of Specification

- 3) Detailed design specifications
  - How to do (code)?
    - component of the architecture
    - individual code units,
  - Data Structure?
  - Data Storage?
  - Algorithms?
  - Input and expected outputs?
  - Invalid inputs?

Levels of Testing

- Corresponding Test Levels
  - 3) **Unit** testing addresses the verification that individual components of the architecture meet their detailed design specification

```java
import org.junit.*;
public class WriteAUnitTest {
    // JUnit calls this method one time before all tests
    @BeforeClass
    public static void setUp() {
        // Place code here for any set up required prior to tests
    }
    @AfterClass
    public static void finished() {
        // Place code here for any clean up that should be done after tests are finished
    }
    @Test
    public void testFirstName() {
        Person p=new Person();
        p.setFirstName("Stephen");
        Assert.assertEquals("Stephen", p.getFirstName());
    }
    @Test
    public void testLastName() {
        Person p=new Person();
        Assert.assertNotNull(p.getLastName());
    }
}
```

Levels of Testing

- Corresponding Test Levels
  - 2) **Integration** testing (a.k.a. component testing) verifies that the groups of units corresponding to architectural elements of the design specification can be integrated to work as a whole


Unit test vs. Integration test
freegifmaker.me

Levels of Testing

- Corresponding Test Levels
  - 1) **System** testing
    - verifies that the complete product meets the functional specification
  - 0) **Acceptance** testing
    - **validate** that
      - the software meets their real intentions
      - meet whatever functionally specified
      - accept the result

When she says those four special words

Using Tests

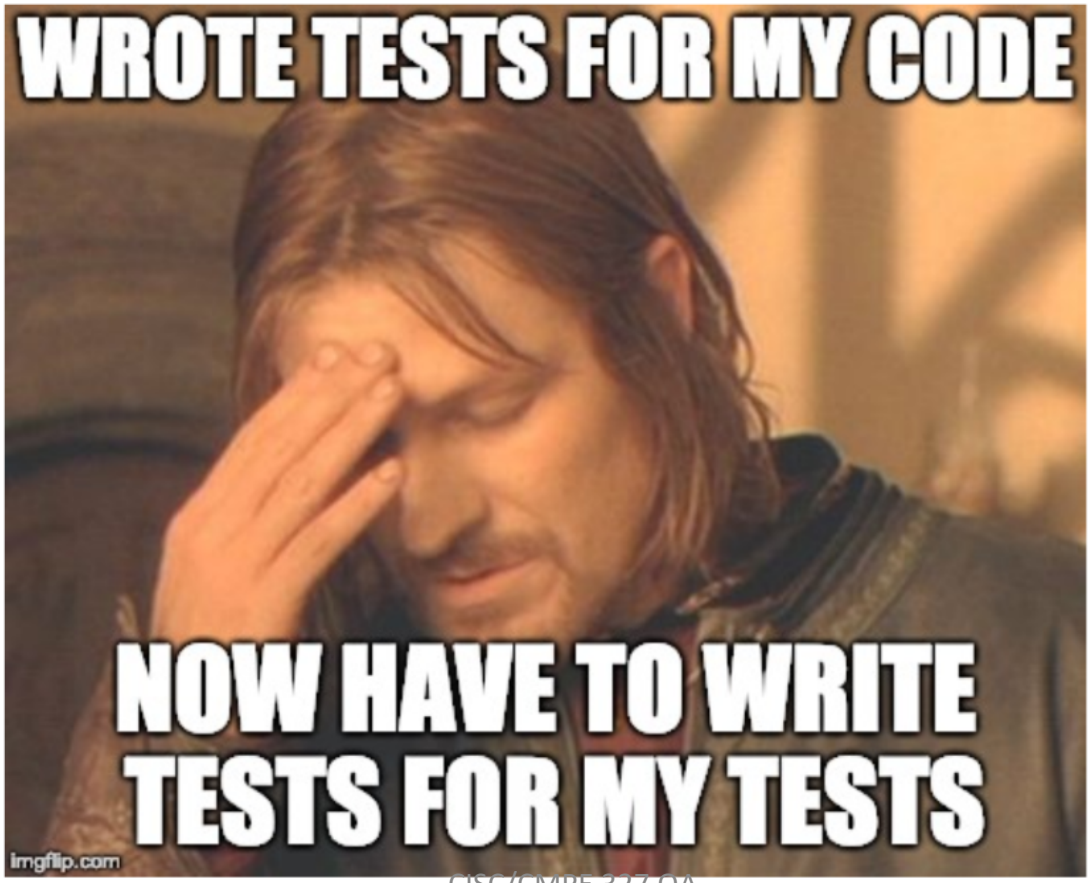
- Evaluating Tests
  - Apply test
  - Evaluate test results
  - FAILED!
    a) the **tests are wrong**:
       UPDATE tests
    b) the **software is wrong**:
       FIX bugs
  - Back to Step 1 until 

Test Evolution

- Tests Don't Die!
  - Testing does not end when the software is accepted by the customer
  - Repeated, modified and extended
    - Continuously monitoring the failed parts when adding new features.
  - NEED Maintenance & Automation

BUT!

- Test Adequacy

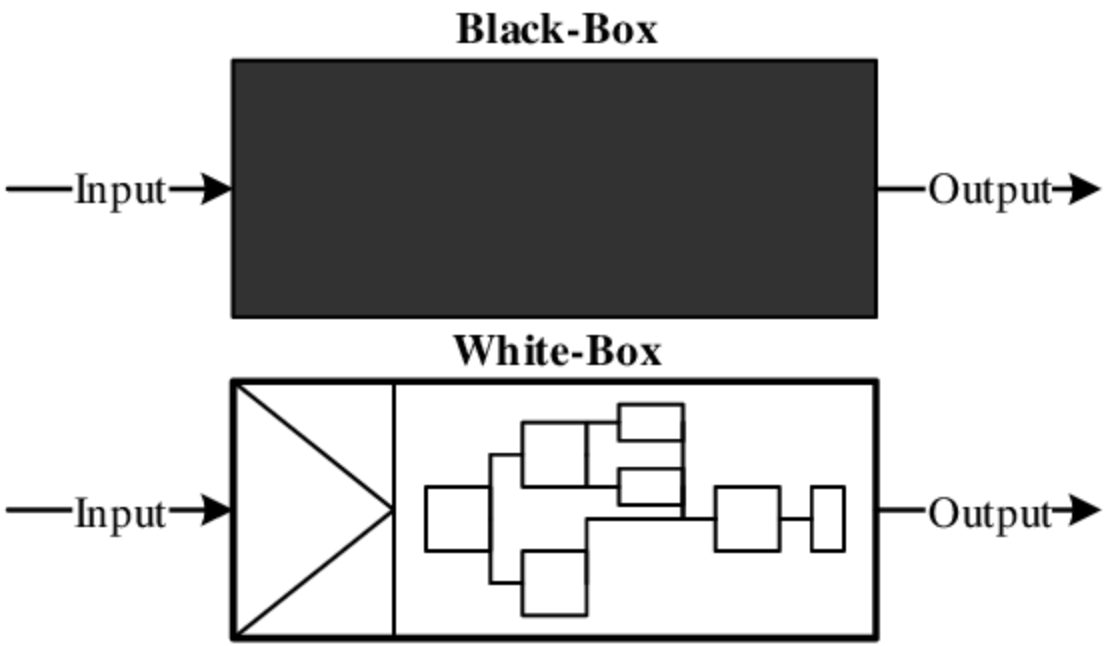


Testing in the Life Cycle

- Kinds of Tests
  - Back box vs. White Box
  - Regression Test
  - Failure Test

Testing in the Life Cycle

- **Black box** testing methods are based on the software's **specifications**
- **White box** (or glass box) testing methods are based on the software's **code**

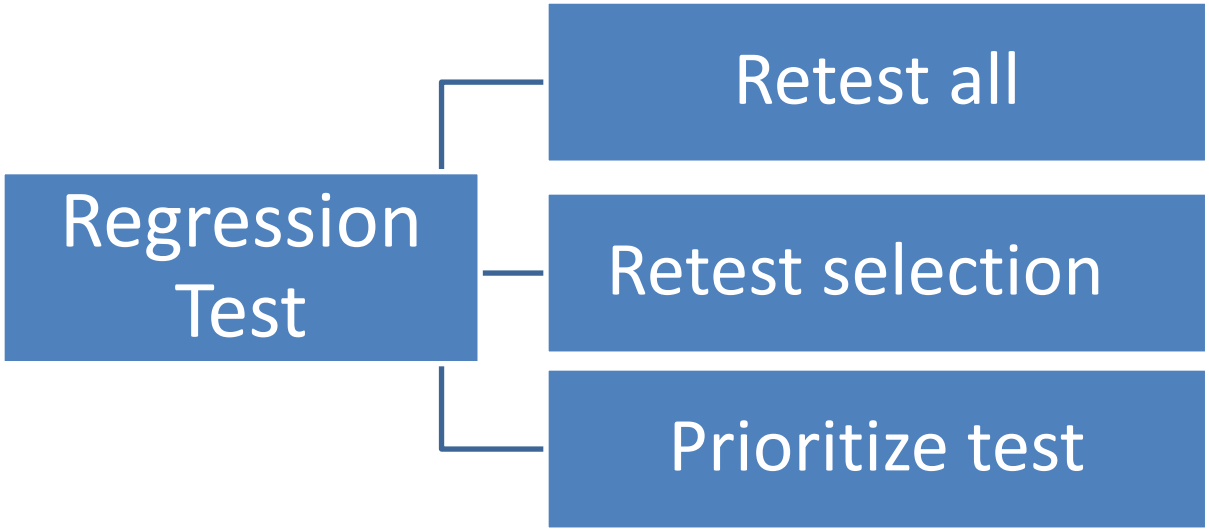


-ility Testing

- System characteristics for quality or testing
  - Capability:
    - The required functions?
  - Reliability:
    - Resist failure in all required situations?
  - Usability:
    - Easy to use?
  - Performance:
    - Fast? Responsive? Scalable?
  - Security:
    - secure?

Regression Test

- Codebase changed?
  - re-running functional and non-functional tests



Regression Test

- Millions of test case…
- Frequent update
- Cost? $$$
- Maintenance?

Testing in the Life Cycle

- Failure Tests
  - Test known/discovered/fixed failures
  - Known observed inputs -> caused the past failures

Test Design (A2)

- Design of Tests
  - A difficult and tricky engineering problem
  - A set of stages
    - High level test -> detailed test procedures
  - Typical test design stages are:
    - test strategy
    - test planning
    - test case design
    - test procedure

Test Strategy (A2)

- Test Strategy
  - the overall approach to testing
  - Levels of testing?
  - Methods?
  - Techniques?
  - Tools?
  - Standards?
  - …
- overall quality plan, by PM, driven by business
- static

Test Plans (A2)

- Test Plans
  - how the test strategy will be carried out
    - the items to be tested
    - the level they will be tested at
    - the order they will be tested in
    - the test **environment**
    - Responsibility?
    - Coverage?
  - project-wide, or procedure-wise
  - By Test Lead or Test Manager


I heard you want to be a web developer
Here are a few devices to test your site

Test Case Design (A2)

- Test Case Design
  - a set of test cases for each item to be tested at each level
  - Each test case specifies
  - how the implementation is to be tested
  - how we will know if the test is successful
  - Input -> action[s]/event[s] -> expected response

Test Case Design (A2)

- Test Case Design (continued)
  - positive testing (should do)
  - negative testing (shouldn't do)
  - separately by level: unit, integration, system, and acceptance

Test Procedures (A2)

- Test Procedures
  - the **process** for conducting test cases
  - For each level
    - the process for running and evaluating the test cases
      - test harnesses (run part of the system)
    - test scripts
    - testing tools (frameworks)
      - GitHub Actions

GitHub Actions (A2)

GitHub Actions / **Build** successful 5 days ago in 19s

- ✓ Set up job
- ✓ Run actions/checkout@v1
- ✓ Set up Python 3.7
- ✓ Install dependencies
- ✓ Lint with flake8
- ✓ Test with pytest
- ✓ Complete job

Test Reports (A2)

- Documenting Test Results
  - Output of test execution results file,
  - Summarized in a readable report
  - Concise, easy to read, and to clearly point out failures
  - A **standardized** form
  - With tools/framework
    - pytest xxxx --junitxml="result.xml"
    - There is an HTML option

Summary

- Introduction to Testing
  - Testing addresses primarily the verification that software meets its specifications
  - Without some kind of specification, we cannot test
  - Testing is done at several levels, corresponding to the levels of functional, design, and detailed specifications in reverse order
  - Testing is not finished at acceptance, it remains for the life of the software system

Summary

- Introduction to Testing
  - Testing is not just a one time task, it is a continuous process that lasts throughout the software life cycle
  - Effective testing requires careful engineering, similar and parallel to the process for design and implementation of the software itself
  - An overall test strategy drives test plans, test case design, and test procedures for a project