INTRODUCTION TO SOFTWARE TESTING

QA, QC AND TESTING

QA – QUALITY ASSURANCE

QC – QUALITY CONTROL

TESTING

QA – QUALITY ASSURANCE

- Setting up adequate processes to ensure the quality of the product.
- Prevent possible bugs instead of detect bugs.
- Process oriented activities.

QC- QUALITY CONTROL

- Making sure that the product correspond to the requirements.
- Requirements verification.
- Product oriented activities.

TESTING

- Exploring to system to check how it works.
- Detect possible bugs (different techniques).
- Product oriented.

Quality assurance
Quality Control
Testing

ERROR, DEFECT, FAILURE

ARE THEY THE SAME?

MISTAKE MADE BY A HUMAN ACTION OR TOOL.

INCORRECT BEHAVIOR OF THE SYSTEM DUE TO A DEFECT.

ERROR MANIFESTATION DURING THE EXECUTION.

According ISTQB (International Software Testing Qualifications Board)

SEVEN TESTING PRINCIPLES

1. TESTING SHOWS THE PRESENCE OF DEFECTS, NOT THEIR ABSENCE

2. EXHAUSTIVE TESTING IS IMPOSSIBLE

3. EARLY TESTING SAVES TIME AND MONEY

4. DEFECTS CLUSTER TOGETHER

5. BEWARE OF THE PESTICIDE PARADOX

6. TESTING IS CONTEXT DEPENDENT

7. ABSENCE-OF-ERRORS IS A FALLACY

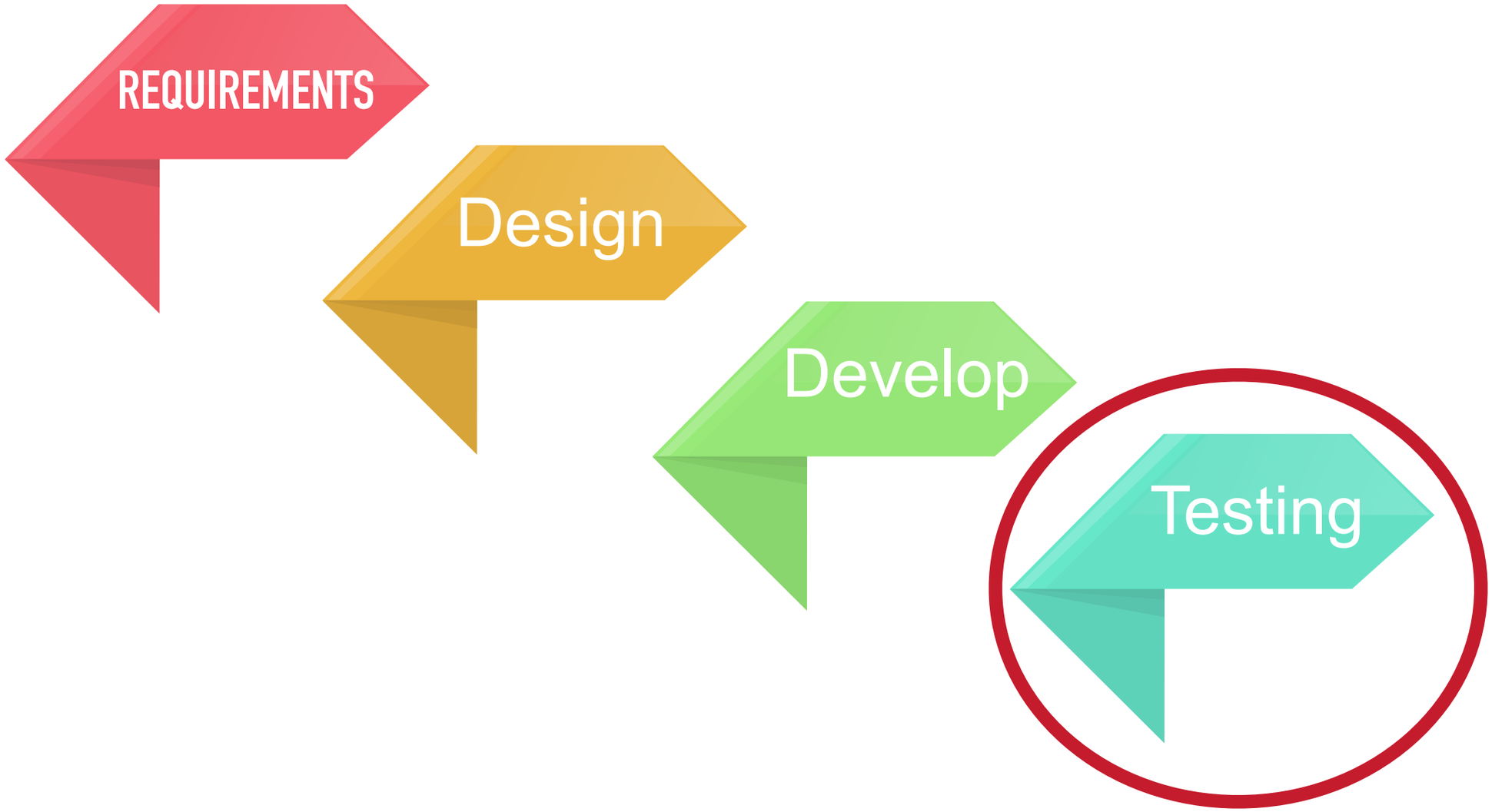
TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFECYCLE

WHAT TYPE OF SDLC DO YOU KNOW?

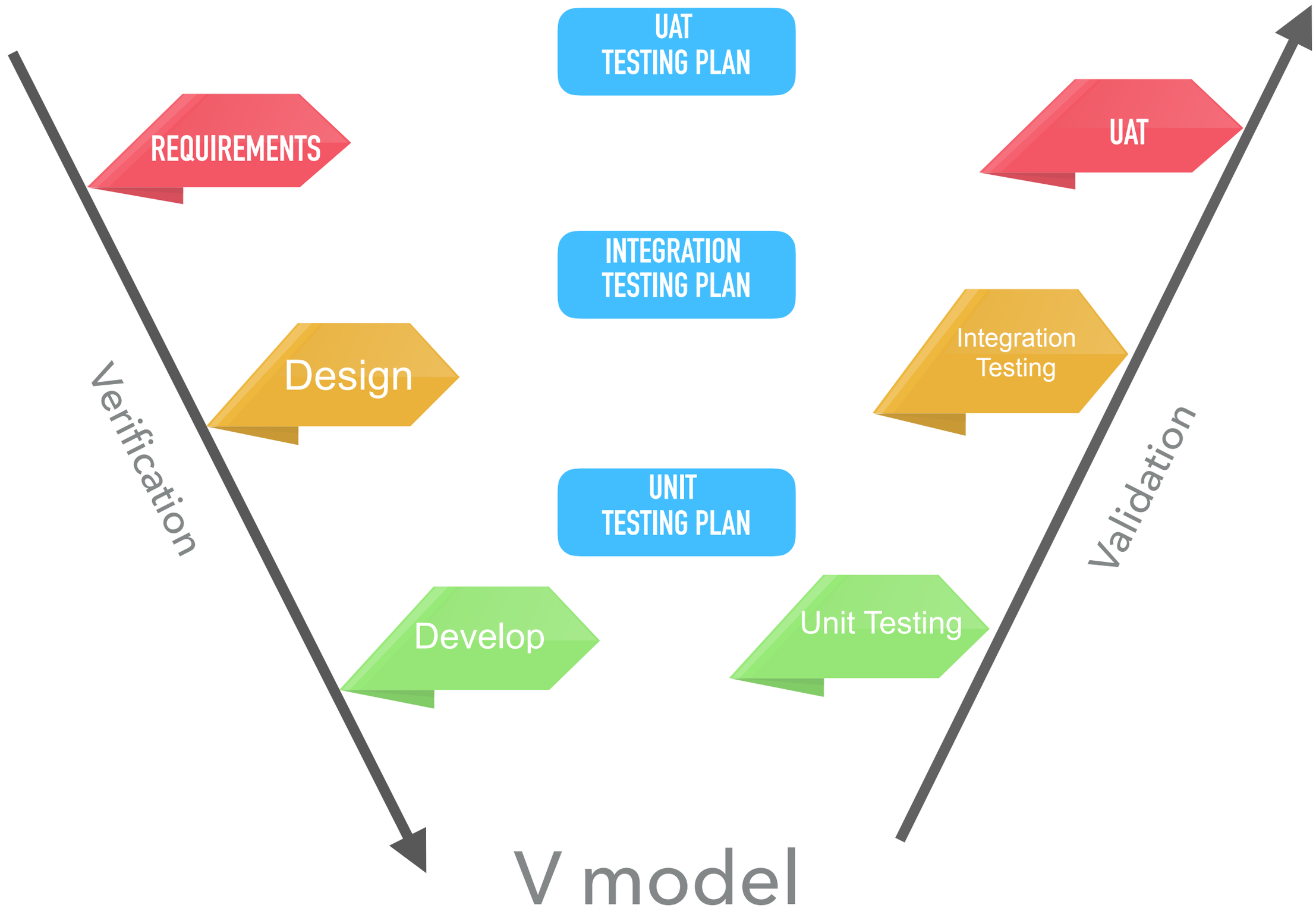
- Sequential development models
- Iterative and incremental development models

- Sequential development models
  - Examples:
    - Waterfall
    - V Model

- WaterFall



TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFECYCLE



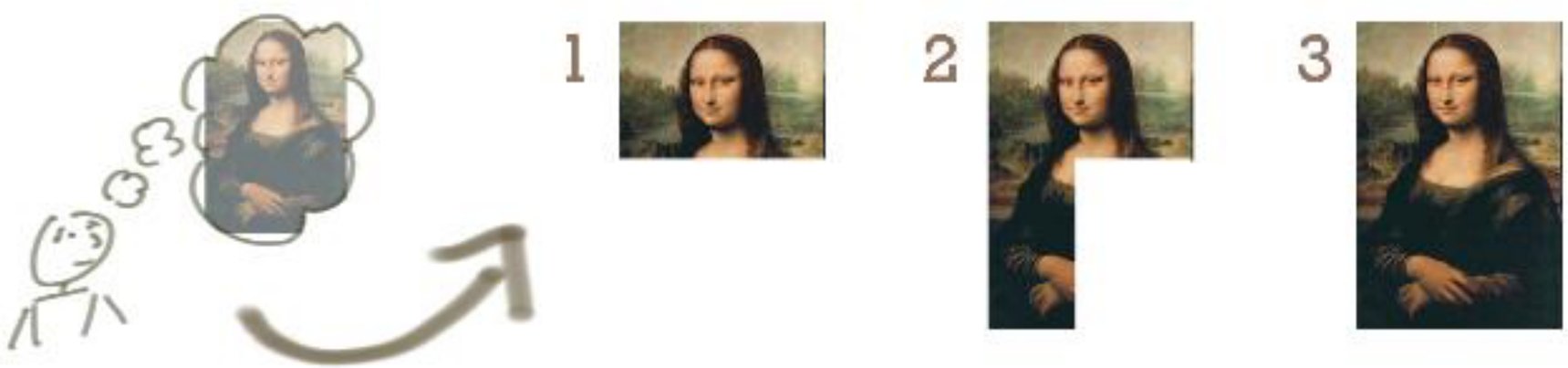
- Iterative and incremental development models

The Mona Lisa Analogy By Jeff Paton

- Incremental

Incrementing calls for a fully formed idea
"incrementing" builds a bit at a time.

Jeff Paton

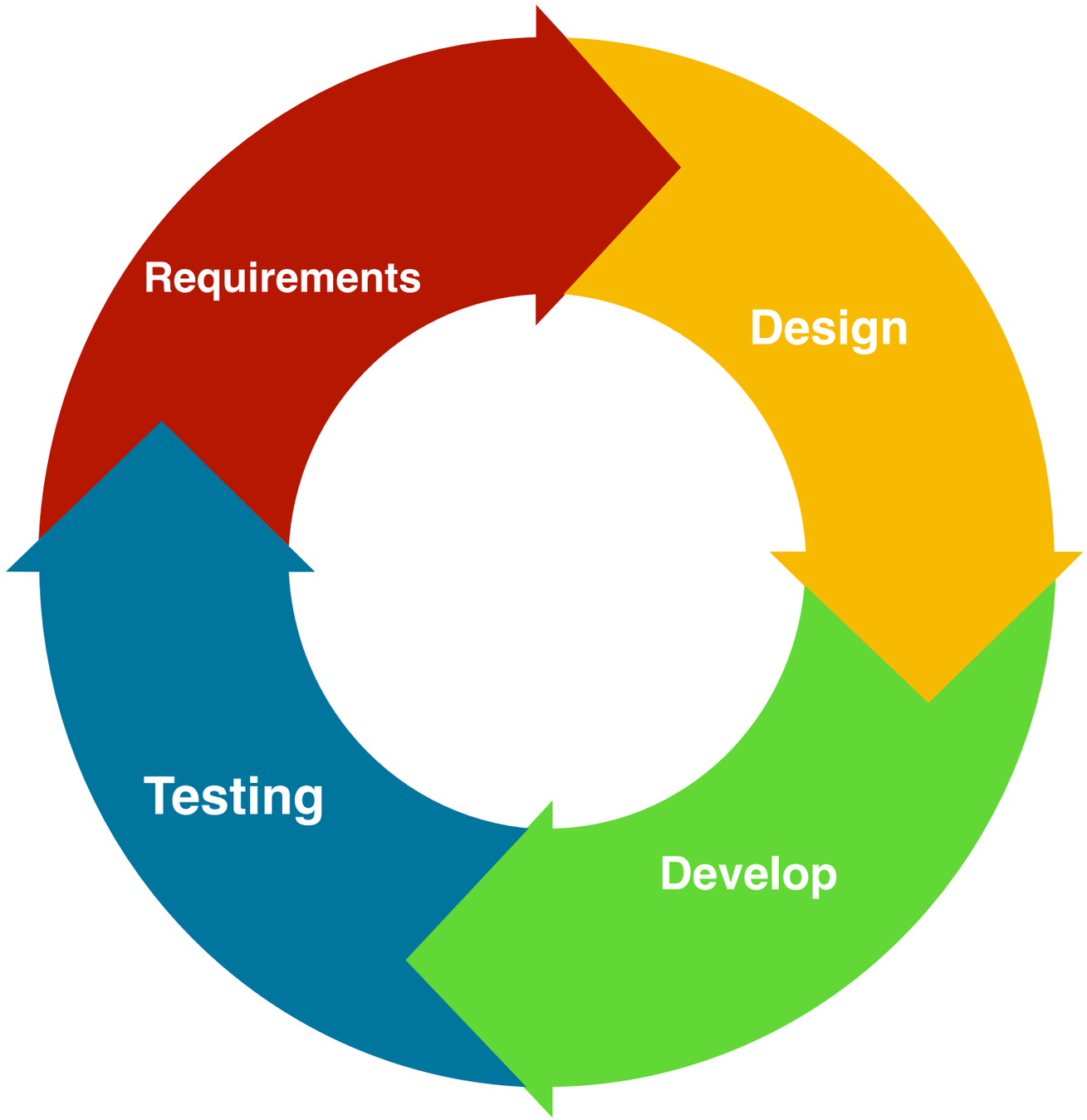


Iterative

Iterating allows you to move from vague idea to realization
"iterating" builds a rough version, validates it, then slowly builds up quality

Jeff Paton

- Agile software development

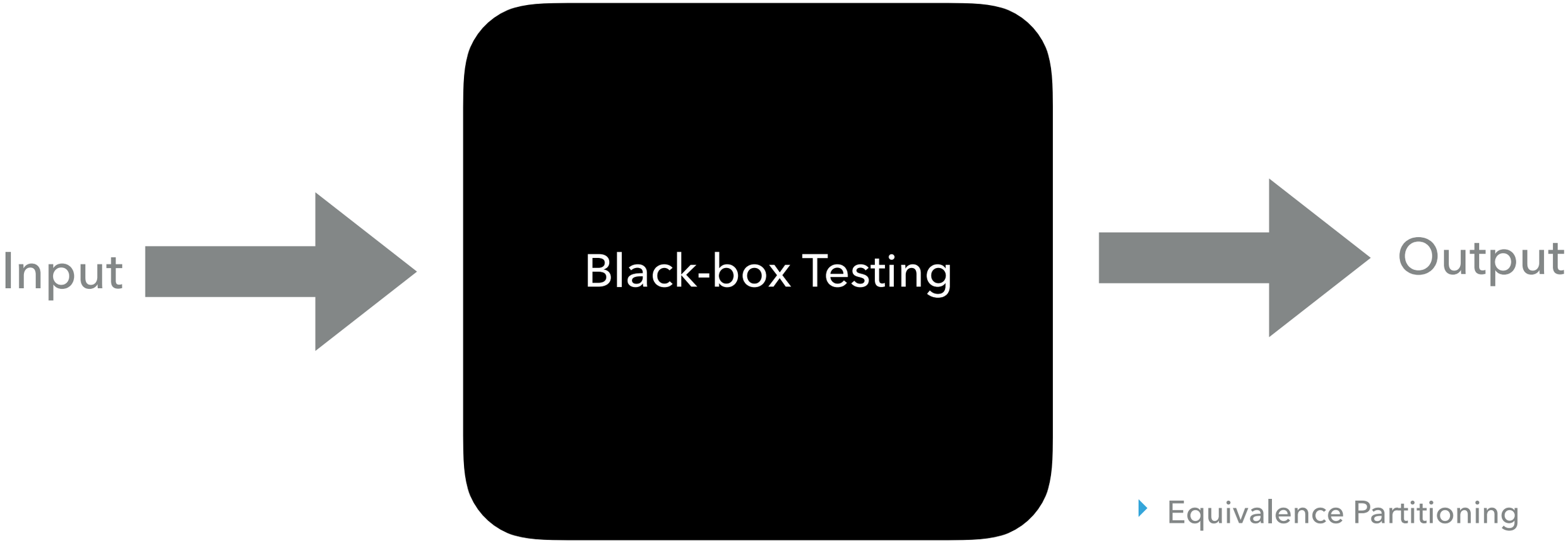


IS THIS A CLASS OF SOFTWARE TESTING OR SDLC?

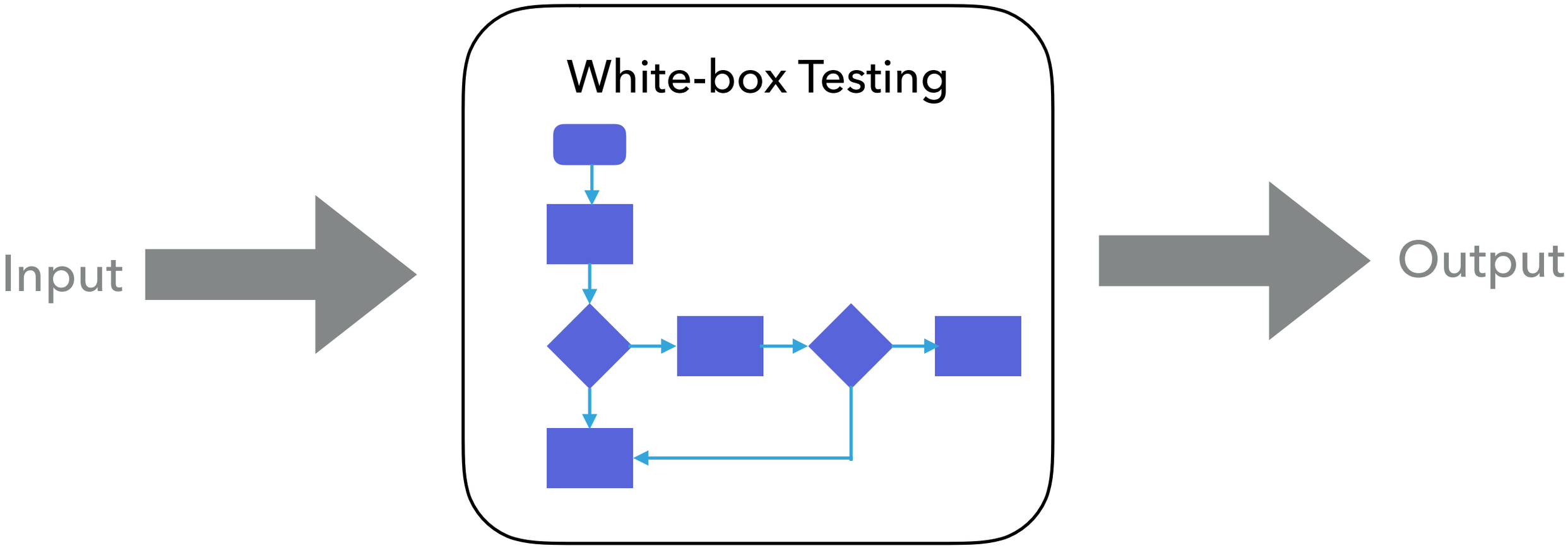
- No matter which software development lifecycle model is chosen.
- Test activities.
  - Early testing principle.

TEST TECHNIQUES

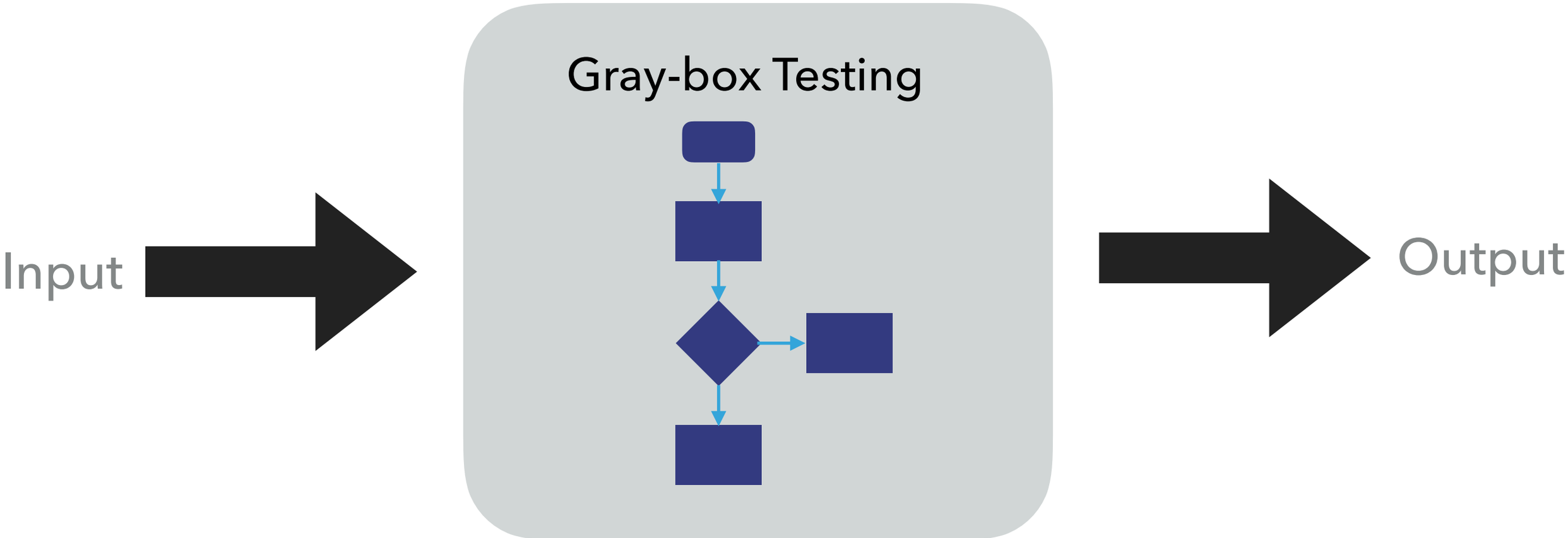
TEST TECHNIQUES



- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Testing
- State Transition Testing
- Use Case Testing



- Statement Testing
- Decision Testing


Gray-box Testing
Input
Output

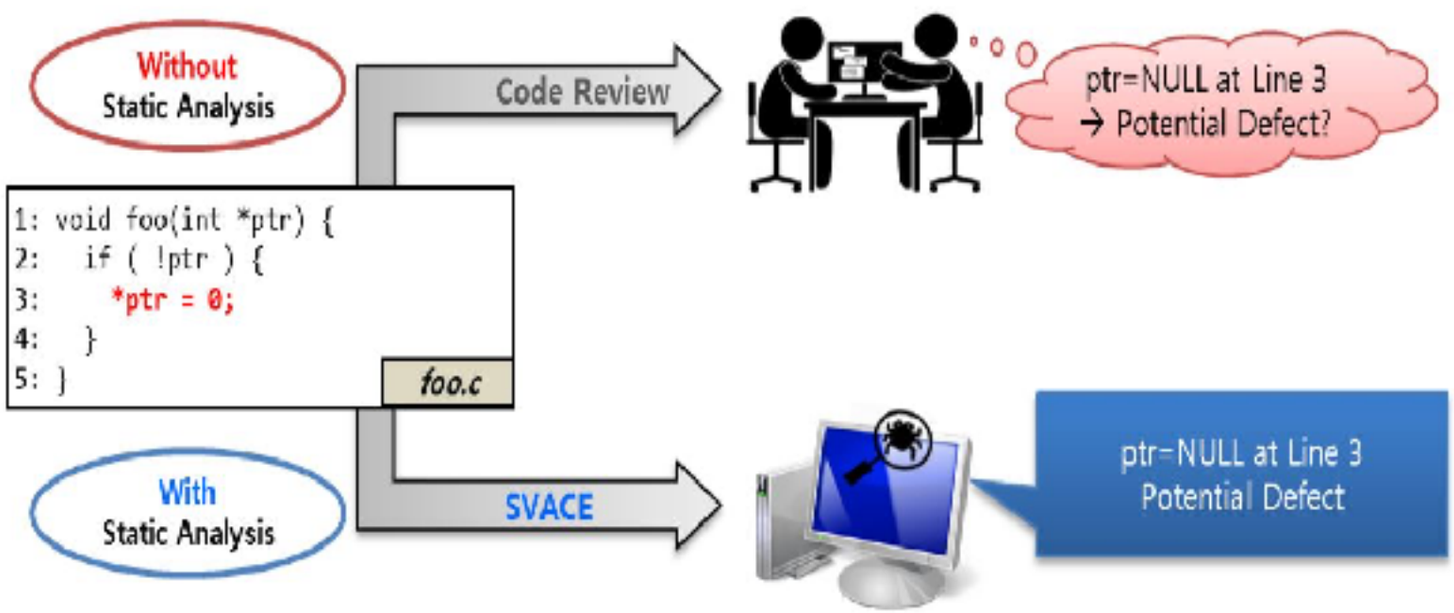
EXPERIENCE-BASED TESTING

- Error Guessing
- Exploratory Testing
- Checklist-based Testing

STATIC & DYNAMIC TESTING

STATIC TESTING

- Software is tested without execution.
- Two Parts:
  - Reviews
  - Static analysis

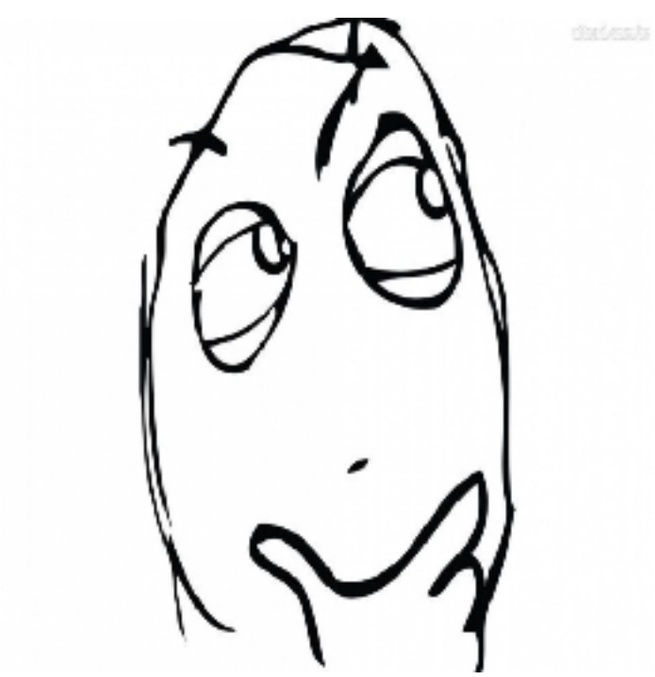


DYNAMIC TESTING

- Software is tested with execution.
- Examples:
  - Unit Testing
  - Integration Testing
  - E2E Testing

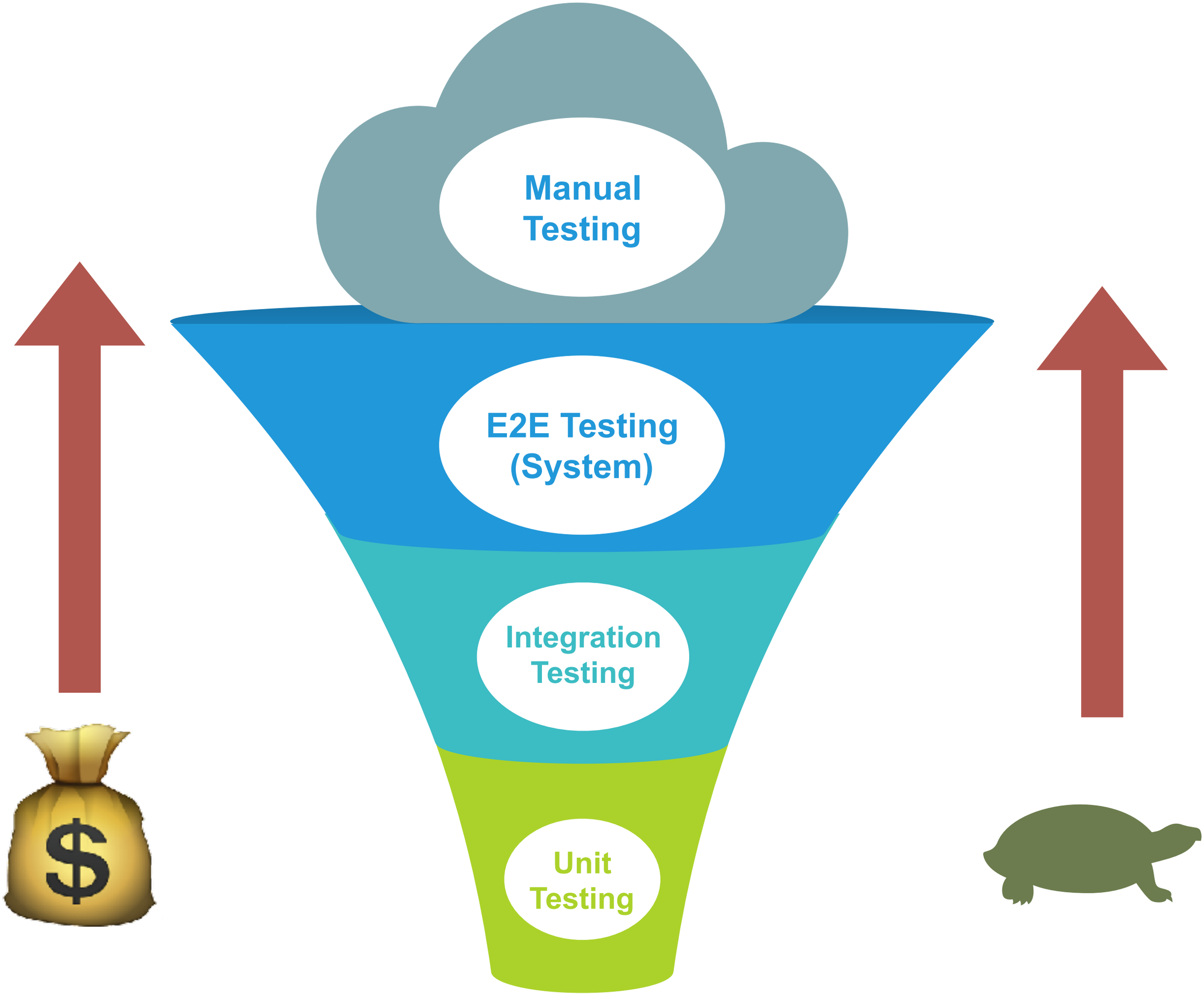
WHAT'S THE MAIN DIFFERENCE BETWEEN BOTH?

- Static Testing
  - Find defect not failures.
- Dynamic Testing
  - Find failures not defects.



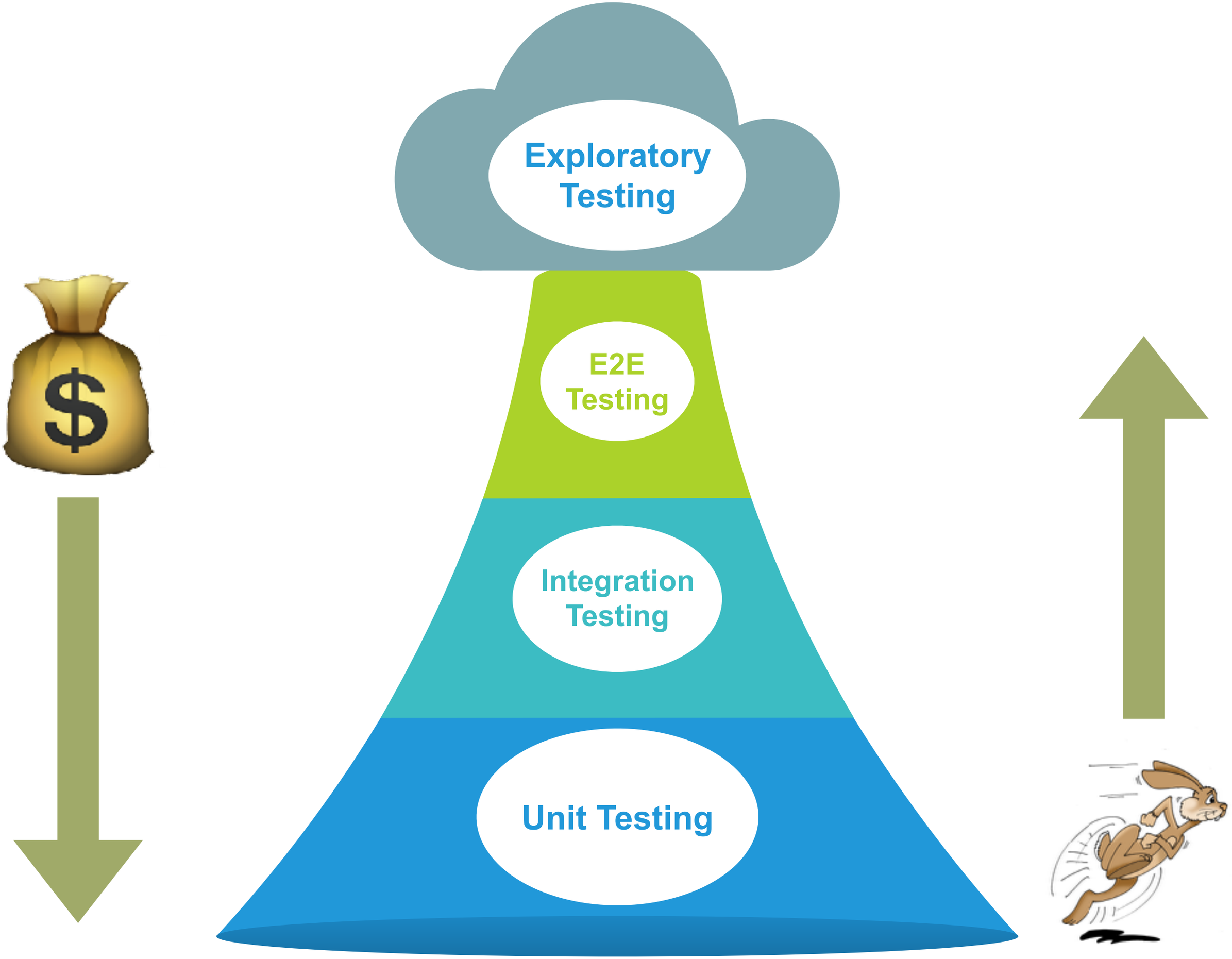
TEST PYRAMID

TRADITIONAL SOFTWARE DEVELOPMENT


Manual Testing
E2E Testing (System)
Integration Testing
Unit Testing

"ICE SCREAM TESTING MAKES EVERYONE WANT TO SCREAM!"

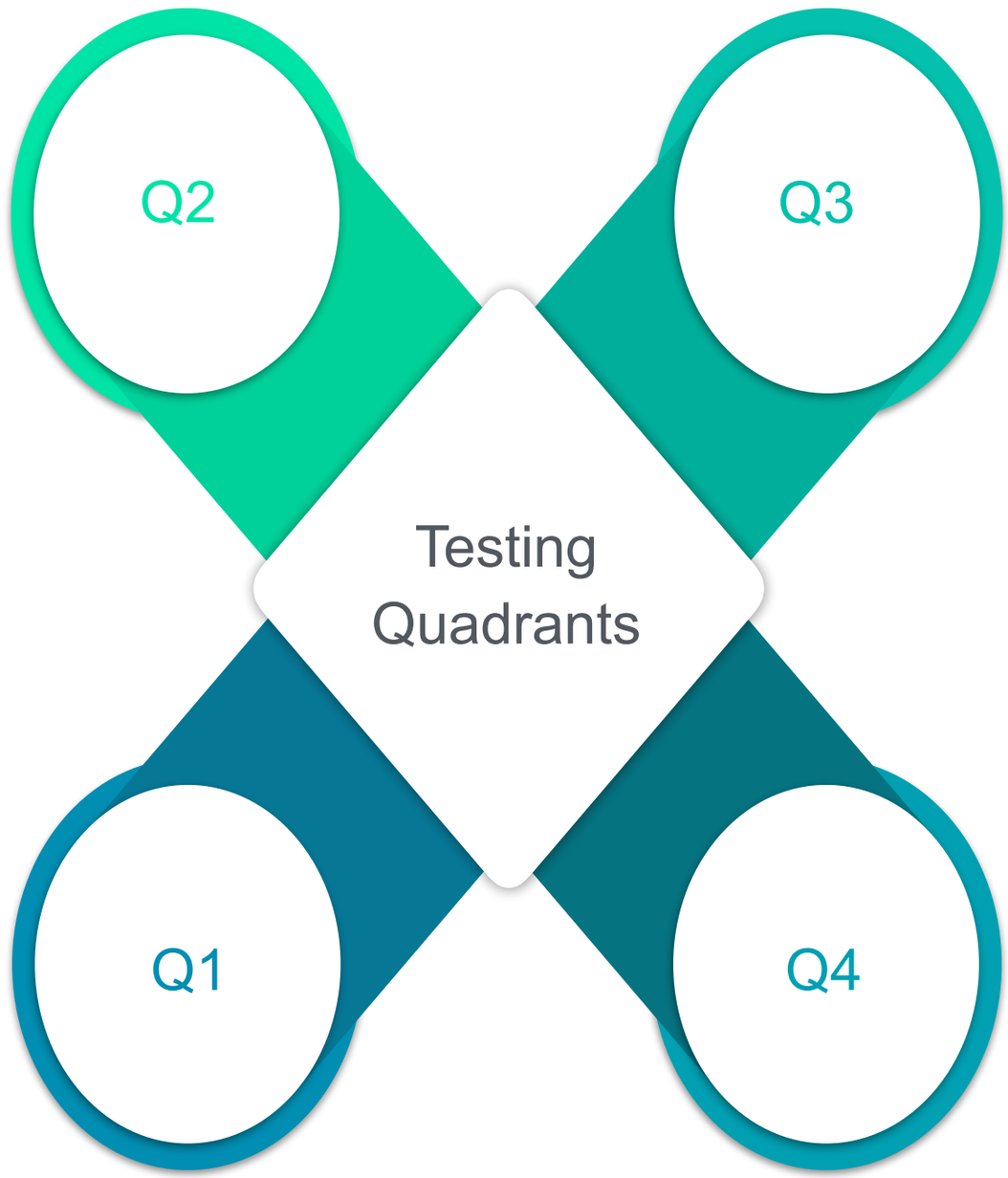
AGILE SOFTWARE DEVELOPMENT


Exploratory Testing
E2E Testing
Integration Testing
Unit Testing

TESTING QUADRANTS

- Defined by Brian Marick.
- Helps to ensure that all test types and test levels are included in the development cicle.
- Can be business or technology facing.
- Can be manually (supported by tools), automated or combination.

- System level.
- Business facing.
- Functional tests.
- Manually or Automated.
- Regression test suites.



- System or user acceptance level.
- Business facing.
- Critique tests for the product.
- Exploratory testing.
- Often Manually.

- Unit level.
- Technology Facing.
- Automated.
- Included in the CI process.

- System or operational acceptance level.
- Technology facing.
- Performance, load, stress, data migration.
- Monitoring.
- Automated.

WHY TESTING IS NECESSARY?

WHY TESTING IS NECESSARY?

- Humans make mistakes ALL the time.
- Reduce risks of failures.
    - Failures could be expensive or dangerous.
- Quality of the system.
- Meet the customer requirements.

THANK YOU.

FUNCTIONAL AND NON-FUNCTIONAL TESTING

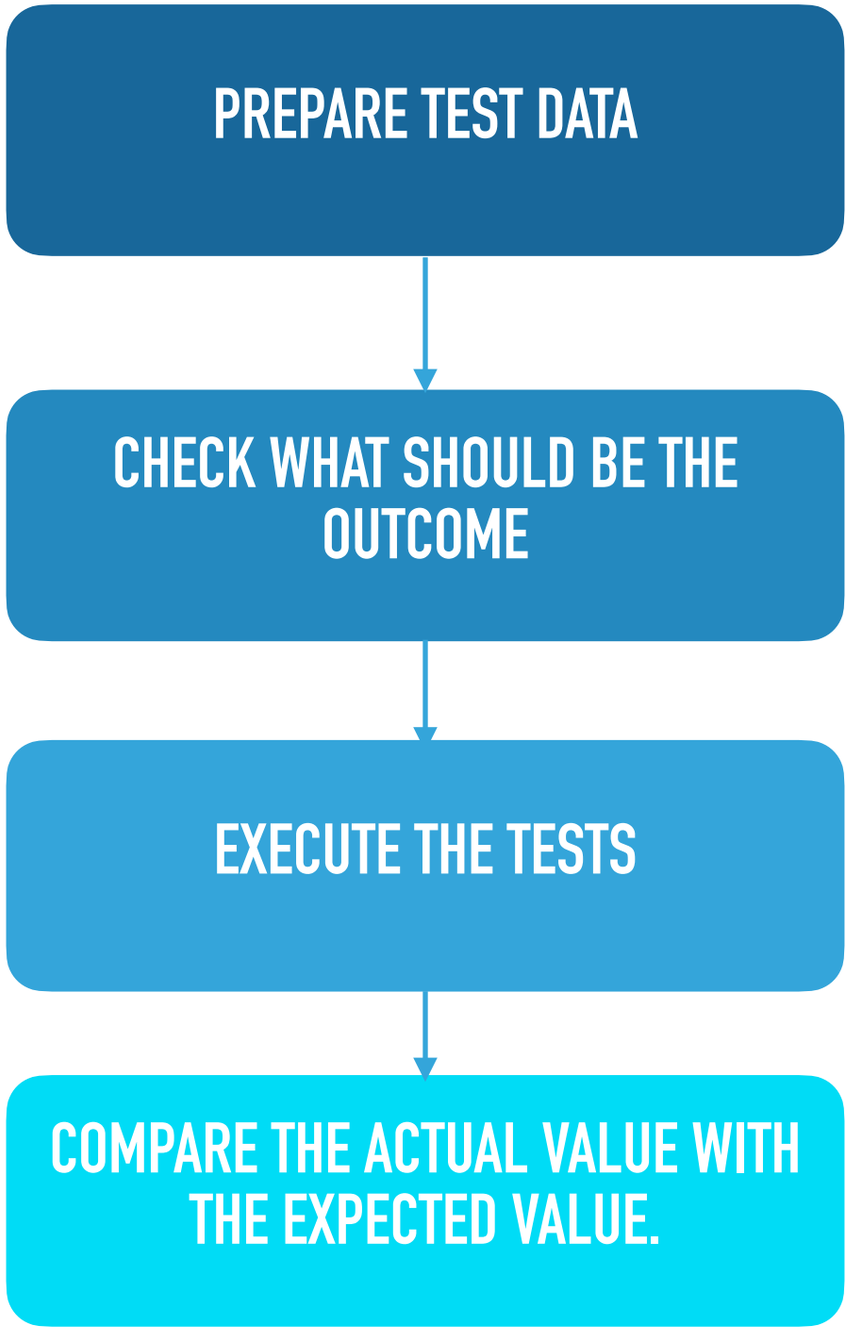
FUNCTIONAL TESTING

FOR YOU, WHAT IS FUNCTIONAL TESTING?

FUNCTIONAL TESTING

- Verifies each function of the software.
- Project Requirements.
- Simulate actual system usage.
- Automated or manual?

Steps to do functional testing:



FUNCTIONAL TESTING

- Types of functional testing:
  - Unit Testing
  - Smoke Testing
  - Sanity Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing
    - UAT
    - Alpha Testing
    - Beta Testing
    - Contract and regulatory testing
  - Negative Testing

Unit Testing

- Basic level of testing.
- Smaller building blocks of a program or system.
- Automated.
- Developers.

EXAMPLE

```java
/**
 * Created by josea.wolff on 1/19/19.
 */
public class ExampleTest {

    public String getTextInUpperCase(String text){
        return text.toUpperCase();
    }

    public String getTextInLowerCase(String text){
        return text.toLowerCase();
    }
}
```

```java
import Example.ExampleTest;
import org.testng.Assert;
import org.testng.annotations.Test;

/**
 * Created by josea.wolff on 1/19/19.
 */
public class Tests {

    @Test
    public void upperCaseTest(){
        System.out.println("\n########################## upperCaseTest ####################################");
        String expectedResult = "INTRO TO DEVOPS";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInUpperCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##########################################################################");
        Assert.assertEquals(actualResult,expectedResult);
    }

    @Test
    public void lowerCaseTest(){
        System.out.println("\n########################## lowerCaseTest ####################################");
        String expectedResult = "intro to devops";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInLowerCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##########################################################################");
        Assert.assertEquals(actualResult,expectedResult);
    }
}
```

EXAMPLE

```
Example                                                    9ms
  <no name>                                                9ms
    Tests                                                  9ms
      lowerCaseTest                                        9ms
      upperCaseTest                                        0ms
```

```
All 2 tests passed – 9ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
objc[32713]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMach
  .jdk/Contents/Home/bin/java (0x106cc24c0) and /Library/Java/JavaVirtualMachines/jdk1.8
  .jdk/Contents/Home/jre/lib/libinstrument.dylib (0x106d4e4e0). One of the two will be u

########################### lowerCaseTest ####################################
Expected Result: intro to devops
ActualResult: intro to devops
##############################################################################

########################### upperCaseTest ####################################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
##############################################################################

===============================================
Example
Total tests run: 2, Failures: 0, Skips: 0
===============================================

Process finished with exit code 0
```

EXAMPLE

```
/**
 * Created by josea.wolff on 1/19/19.
 */
public class ExampleTest {

    public String getTextInUpperCase(String text){
        return text.toUpperCase();
    }

    public String getTextInLowerCase(String text){
        return text.toUpperCase();
    }
}
```

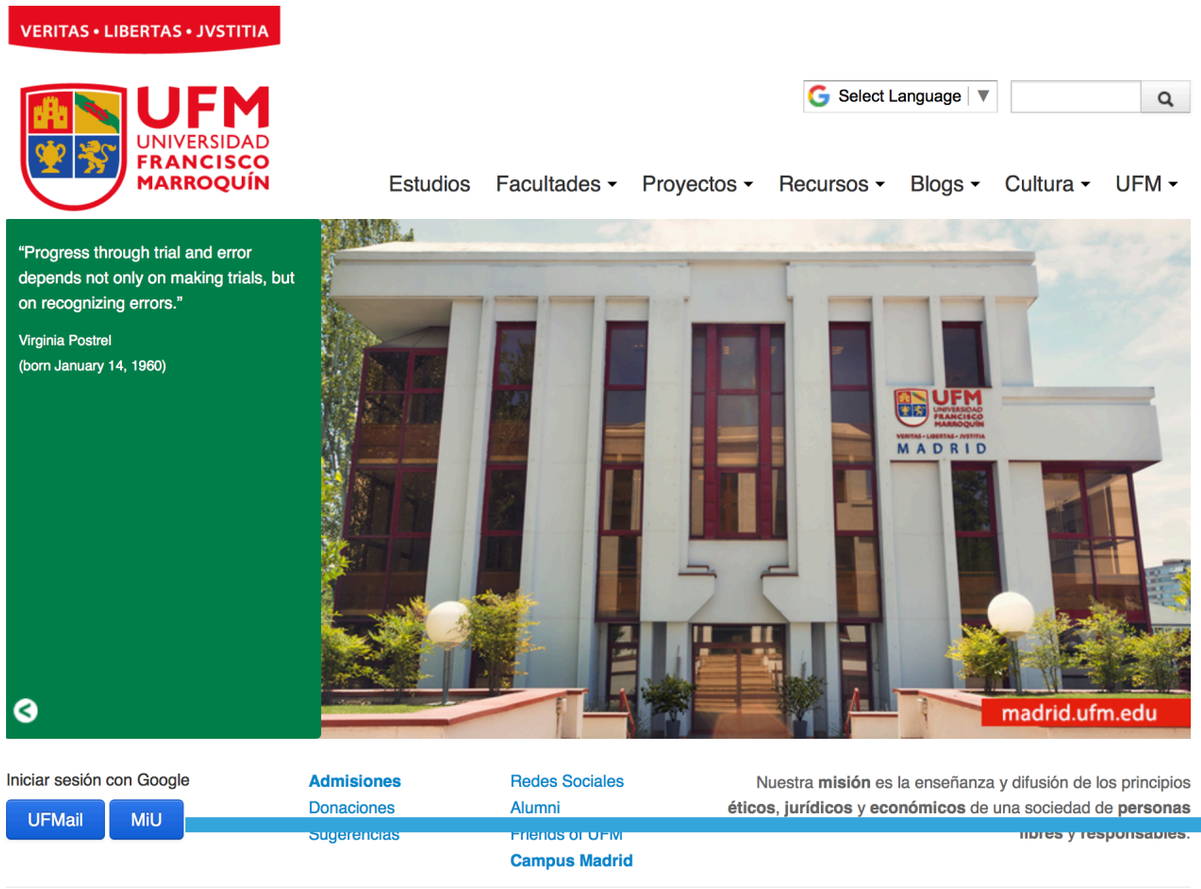
```
Example                                  10ms
  <no name>                              10ms
    Tests                                10ms
      lowerCaseTest                      10ms
      upperCaseTest                       0ms
```

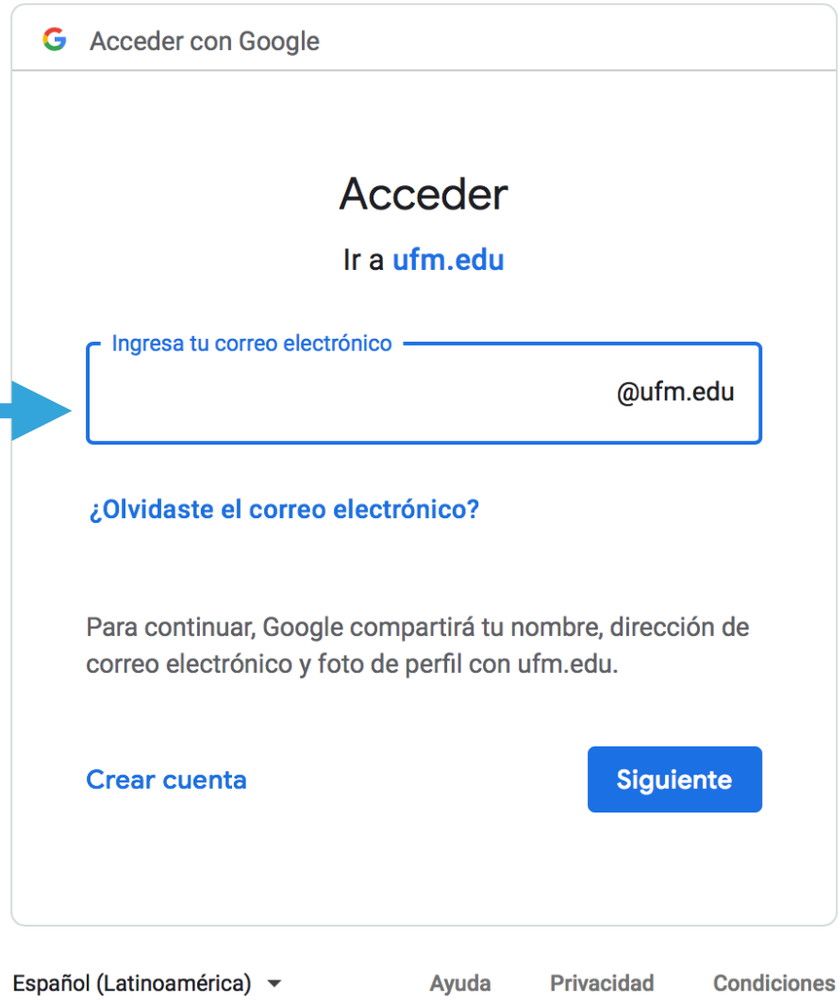
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/ja
objc[32816]: Class JavaLaunchHelper is implemented in both /Library/Jav
  and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/
  one is undefined.

############################ lowerCaseTest ############################
Expected Result: intro to devops
ActualResult: INTRO TO DEVOPS
###################################################################################

java.lang.AssertionError: expected [intro to devops] but found [INTRO TO DEVOPS]
Expected :intro to devops
Actual   :INTRO TO DEVOPS
 <Click to see difference>


    at org.testng.Assert.fail(Assert.java:96)
    at org.testng.Assert.failNotEquals(Assert.java:776)
    at org.testng.Assert.assertEqualsImpl(Assert.java:137)
    at org.testng.Assert.assertEquals(Assert.java:118)
    at org.testng.Assert.assertEquals(Assert.java:453)
    at org.testng.Assert.assertEquals(Assert.java:463)
    at Tests.lowerCaseTest(Tests.java:29) <16 internal calls>
    at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)
    at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:84) <4 internal calls>
    at org.testng.IDEARemoteTestNG.run(IDEARemoteTestNG.java:72)
    at org.testng.RemoteTestNGStarter.main(RemoteTestNGStarter.java:127)


############################ upperCaseTest ############################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
###################################################################################

===============================================
Example
Total tests run: 2, Failures: 1, Skips: 0
===============================================


Process finished with exit code 0
```

Smoke Testing

- Any idea?
- Help to decide if is feasible to continue with the rest of the tests.
- Use the Pareto principle.
- Manual or Automated.
- Normally used on:
    - Integration and System Level.
- Pesticide paradox principle - Important

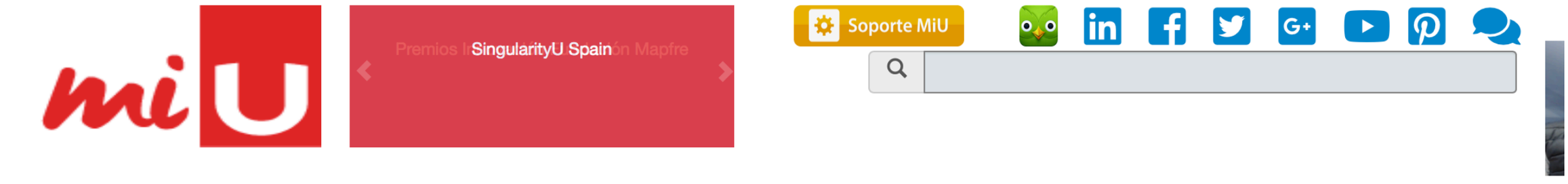
EXAMPLE

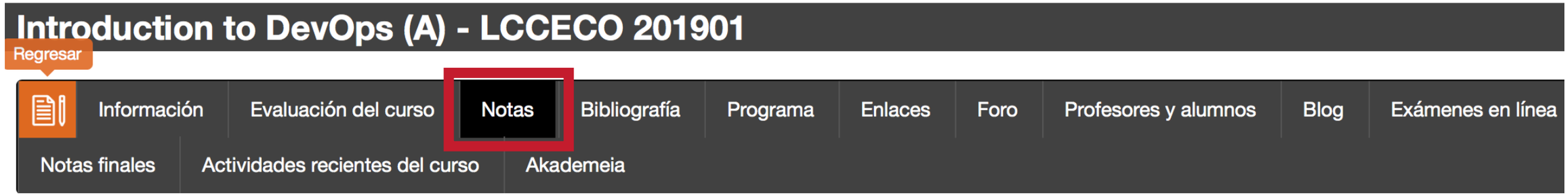




Integration Level

EXAMPLE



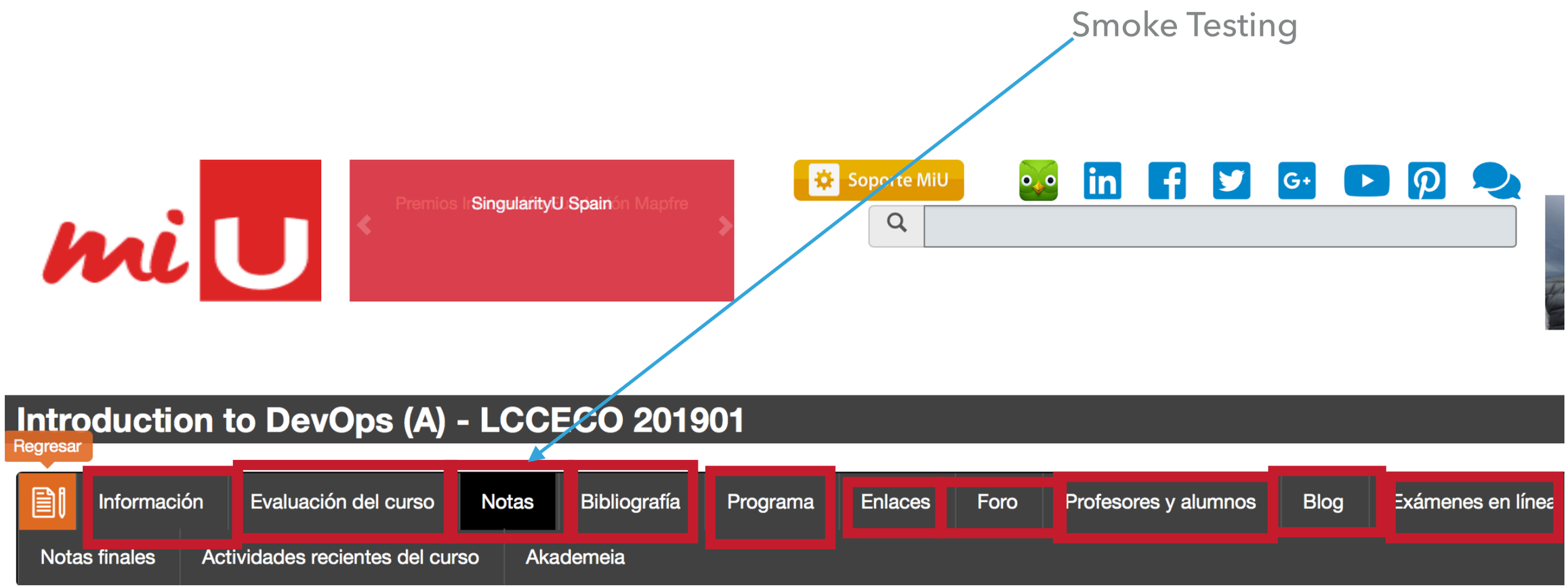


System Level

Sanity Testing

- Checks bugs reported in the previous build.
- Checks if breaks another functionalities.
- Checks that the expected functionality works fine.
- Avoid wasting time and money.
- Subset of Acceptance Criteria.
- Checks the main functionalities without going deeper.

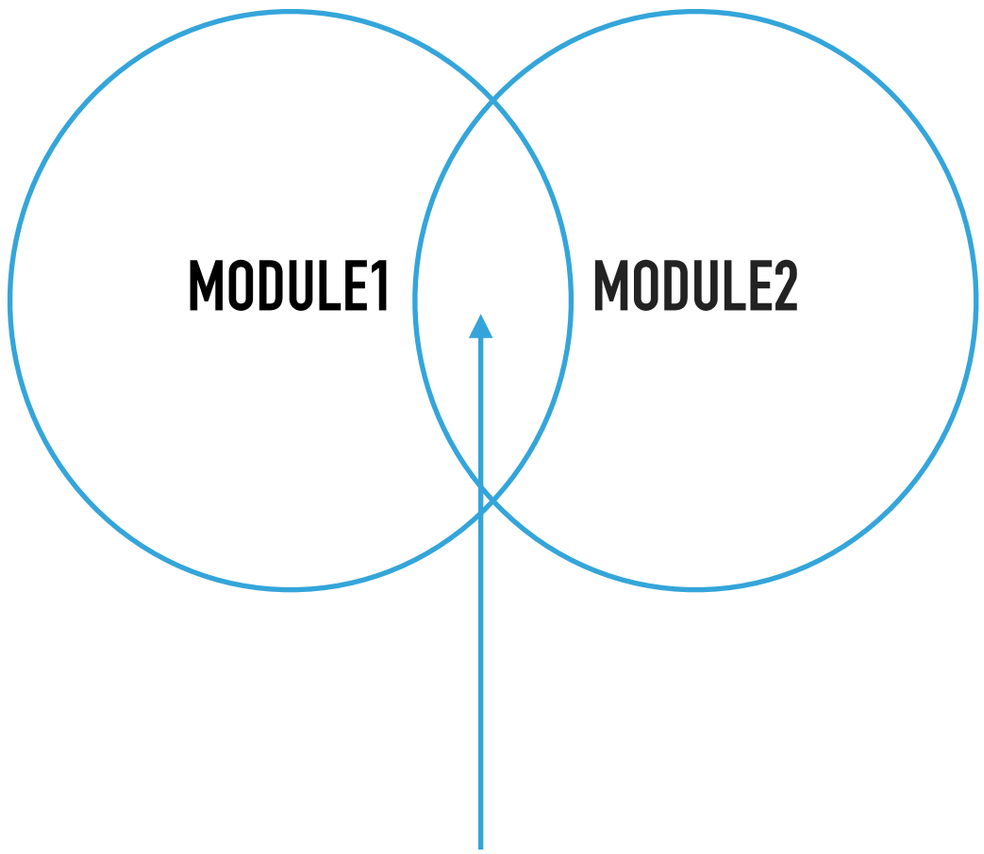
EXAMPLE


Smoke Testing
SingularityU Spain
Soporte MiU
Introduction to DevOps (A) - LCCECO 201901
Regresar
Información
Evaluación del curso
Notas
Bibliografía
Programa
Enlaces
Foro
Profesores y alumnos
Blog
Exámenes en línea
Notas finales
Actividades recientes del curso
Akademeia

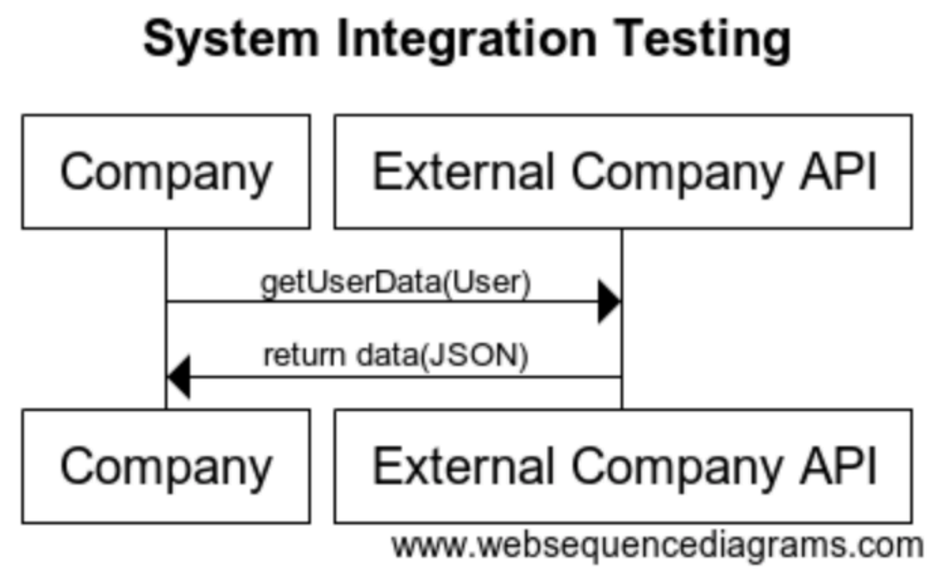
Integration Testing

- After Unit Testing occurs.
- Small unit are combined.
- Two levels:
  - Component Integration Testing.
  - System Integration Testing.
- Testers/Developers.

EXAMPLE



Component Integration Testing



System Testing

- After Integration Testing occurs.
- Behavior and capabilities of a whole system.
- End-to-End Testing.
- Produces information that is used by stakeholders to make release decisions.
- Test Environment.
- Automated or manual.
- Testers.

Analogy of System Testing

Acceptance Testing

- Acceptance criteria is needed.
- Focuses on the behavior of the whole system.
- System complete and will work as expected.
- Significant number of defects in some cases is considered a major project risk.
- Common forms to do acceptance testing:
  - UAT.
  - Alpha testing.
  - Beta testing.
  - Contractual and regulatory acceptance testing

UAT

- Environment ≈ Production.
- Main Objective:
  - Building confidence that the user can use the system to meet their needs.
- Techniques:
  - Error Guessing.
  - Exploratory Testing.
  - Check List based Testing
- Who does these tests?
  - Real users or a separate team in the company.

Alpha Testing

- Environment ≈ Production.
- Dedicated test team in the company.
- Not open to the market.
- Black box testing and white box testing.
- Reduce risk.

Beta Testing

- Environment = Production.
- Real Users.
- Open to the market.
- Black box testing.
- Detection of defects related to the conditions and environment(s)
- In some cases happens after alpha testing occurs.

Contractual and regulatory acceptance testing

- Contractual Testing
  - Contract's acceptance criteria -
  - Real Users or independent testers.
- Regulatory acceptance testing:
  - Example: Legal or security.
  - Real users, independent testers, regulatory agencies.

Negative Testing

- Error path testing or failure testing.
- Validate the behavior of the application with invalid data.
- Real and more scenarios.
- How do you can define the valid and invalid data?
    - Equivalence partitioning.
    - Boundaries Values.

EXAMPLE

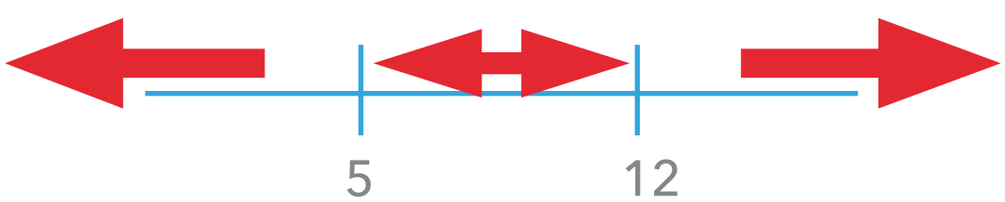
Specifications:

No symbols and numbers should be accepted.

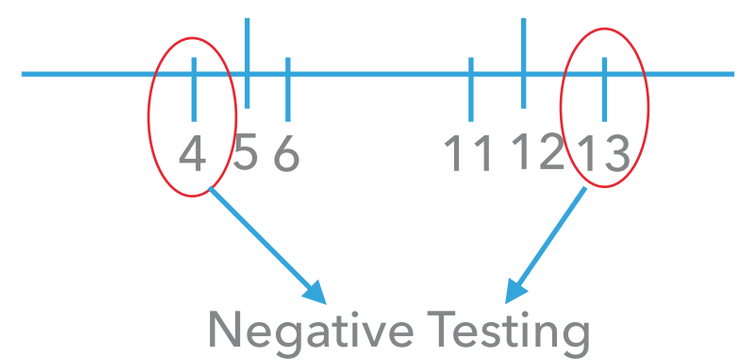
5 to 12 characters.

Name:

Equivalence Partitioning.



Boundaries values.



NON-FUNCTIONAL TESTING

FOR YOU, WHAT COULD BE A NON-FUNCTIONAL ASPECT?

NON-FUNCTIONAL TESTING

Some non-functional aspects:

- Performance.
- Usability.
- Security.
- Visual.
- Resilience

NON-FUNCTIONAL TESTING

- How well the system behaves.
- Common is performed at all levels.
- Late discovery = dangerous to the success of the project.
- Black-box testing.
    - Example: Boundary value analysis can be used to define the stress conditions for performance testing.
- Involve special skills.

PERFORMANCE TESTING

- The performance of an application is measured from different perspectives.
- Different types of performance testing.
  - Load testing.
  - Stress testing.
- Continuously monitored.

LOAD TESTING

- Modeling the expected usage of the application.
- Simulate multiple users concurrently.
- Checks how the application behave during normal or hight usage.
- Helps to determine how many users can be handle.
- Find bugs that are not able to detect using another testing technique.

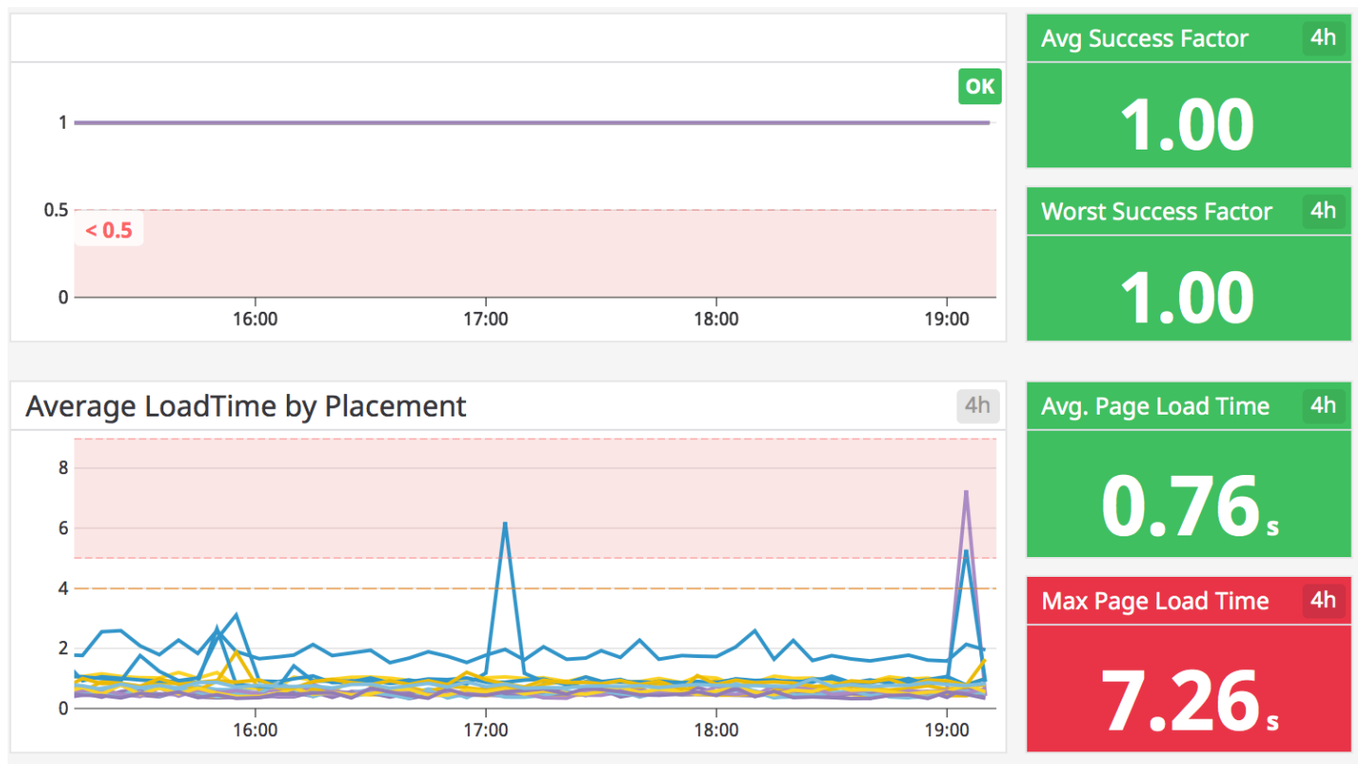
STRESS TESTING

- Load placed on the system is increased beyond the normal usage.
- How the application response.
- Long period of time. (Hours or days).
- Test the robustness of the hardware system.

WHAT'S THE MAIN DIFFERENCE BETWEEN LOAD AND STRESS TESTING.

PERFORMANCE MONITORING

- System Owners.
- Performance degradations. (Example?).
- Tools.



USABILITY TESTING

- Evaluate the ease of use which the end users of the system access the application.
- ISO 9214–11
  - Effectiveness.
  - Efficiency.
  - Satisfaction.
- Key principles of usability testing:
  - Communication.
  - Provide information users wants.
  - Offer simple.

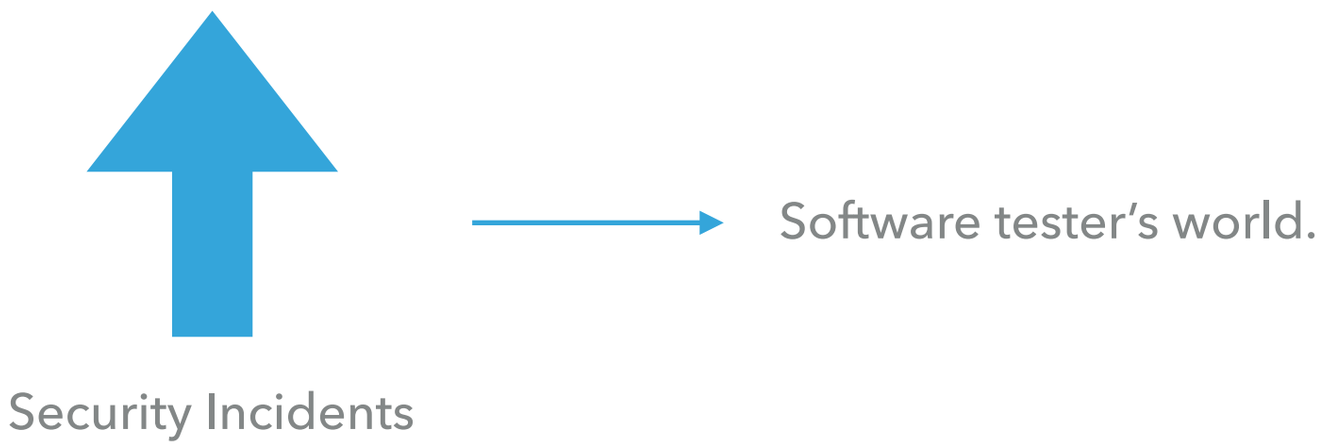
USABILITY TESTING

- Discover needs and expectations.
- Critical for usability testing:
  - Visibility.
  - Match between system and the real world.
  - Ease learning.
- Feedback direct from target audience.

USABILITY TESTING – EXAMPLE

SECURITY TESTING

- Was performed by network administrators or system developers.



- 3 of 4 web sites are vulnerable to an attack and 75% of the hacks occur at the application level (Gartner's reports).

SECURITY TESTING

- Main Objectives:
  - Security requirements.
  - Identify and solve vulnerabilities of the application.
- Should be developed in each phase of the SDLC.
- Manual or automated.
- Examples ?
  - IP Restriction.
  - Ports Enable.
  - Pen testing (Penetration testing).

VISUAL TESTING

- Main Objectives:
  - Visual requirements.
  - Rendering in different browsers/OS
- Manual or automated.
- Examples ?
  - Responsive
  - Element positions.

RESILIENCE TESTING

- Ensure the application perform well in real conditions.
- Acceptance service level to business.
- Recovery.
- Example:

EXERCISE

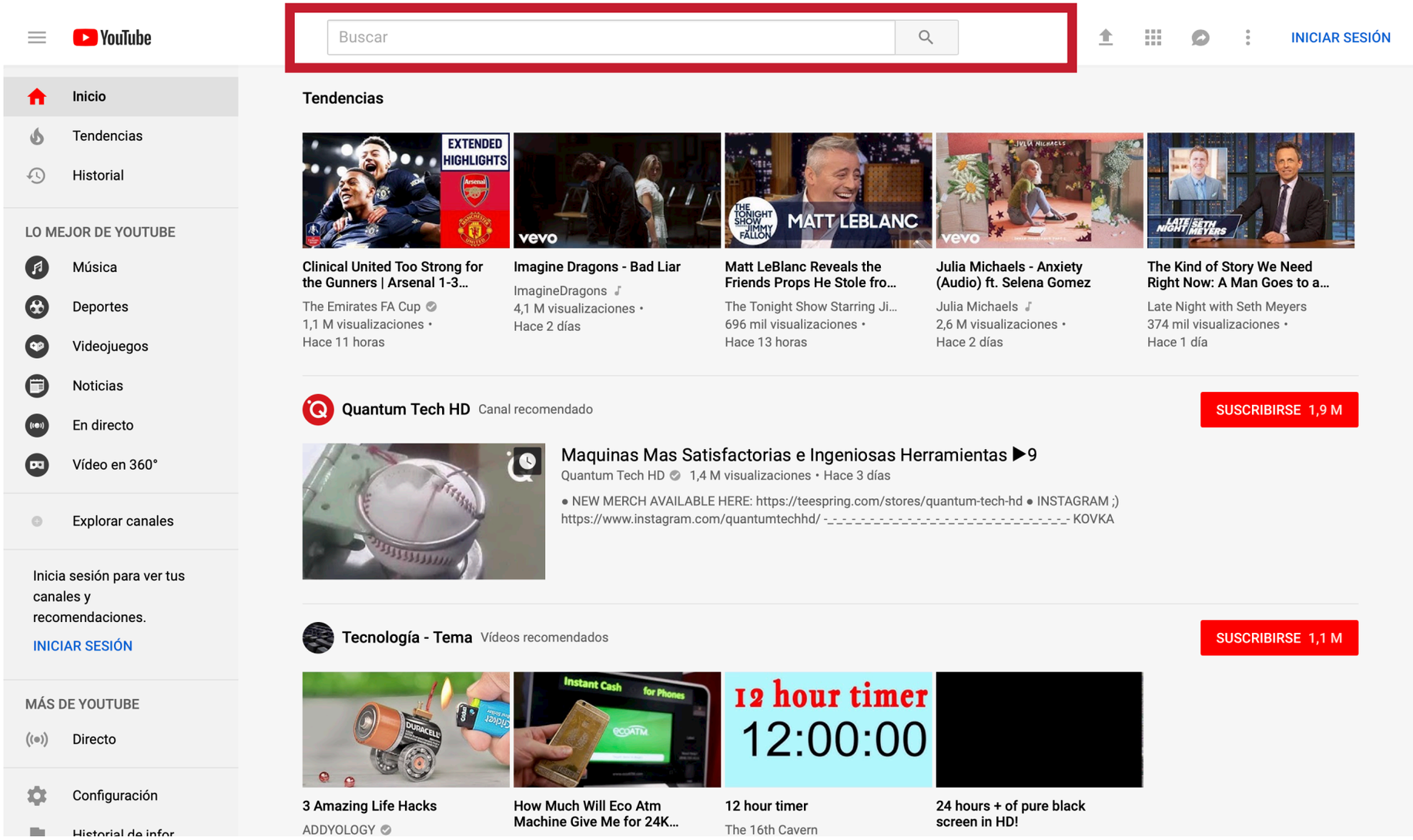
- Select one of the following websites.

- Identify 6 functional and non-functional aspects.
- Describe what type of functional and non-functional testing you would used and why.

EXAMPLE – YOUTUBE

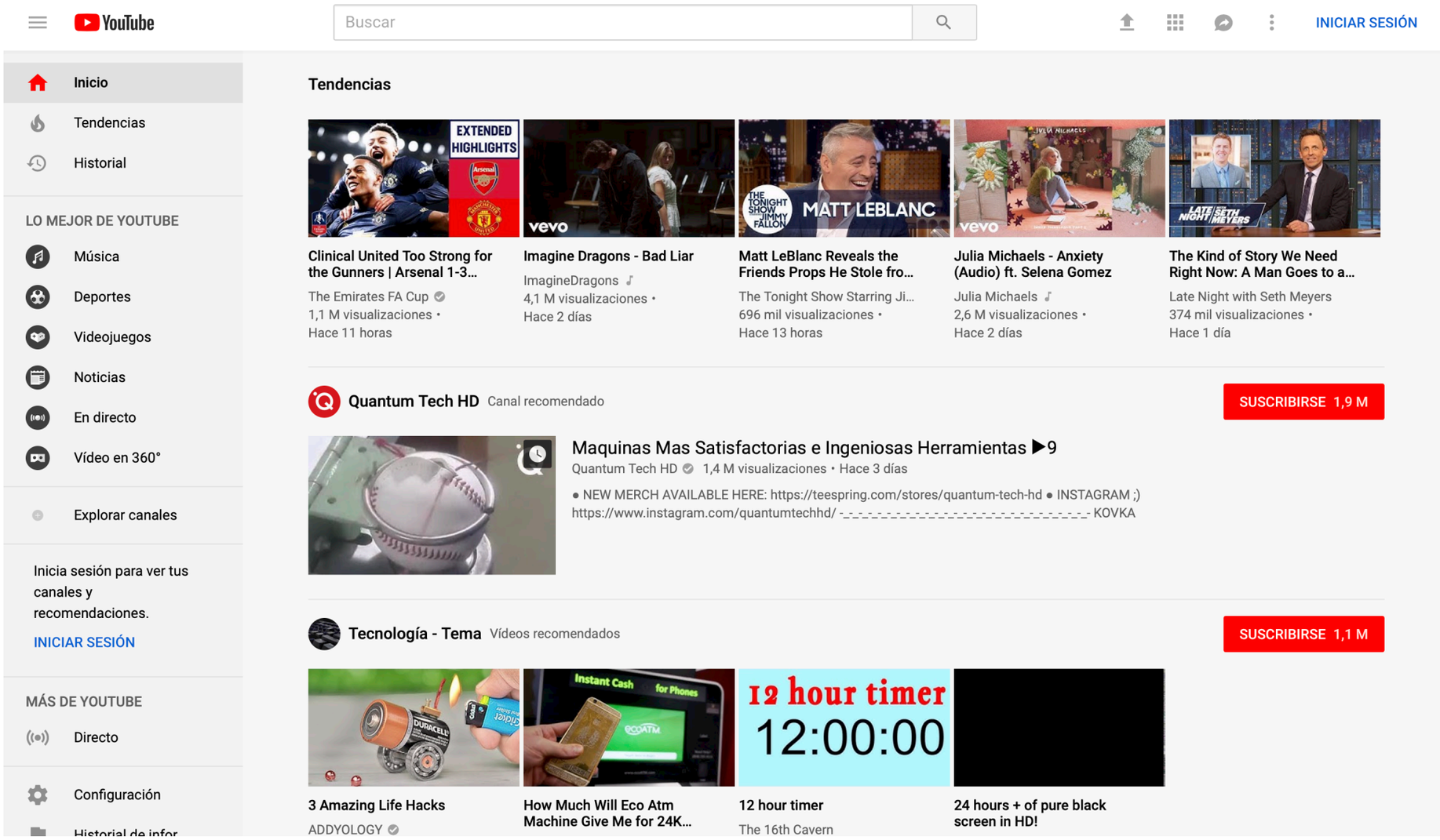
YouTube

YOUTUBE - FUNCTIONAL TEST



- Search Option:
    - Functional Type: Smoke Test
    - Why: Because for me the search option is the most important of youtube. If the user is not able to search video youtube doesn't accomplish their main use and smoke testing help us to check if is feasible continue with the rest of the tests (making sure that the most critical functionalities are working).

YOUTUBE - NON-FUNCTIONAL TEST



- Search Option:
  - Non-Functional Type: Load Test
  - Why: We have to be sure if youtube can make different searches simultaneous with a quick response. Load test simulate multiple users concurrently

THANK YOU.

TESTING TOOLS

TESTING TOOLS

- A lot of testing tools with different purposes.
- Different types:
  - Test Management tools.
  - Static Analysis tools.
  - Automated testing tools.
  - Load and stress Testing tools.
  - Defect tracking tools.
  - Mobile testing tools.
  - CrossBrowser testing tools.
  - API Testing tools.
  - Visual Testing tools.
  - Etc….

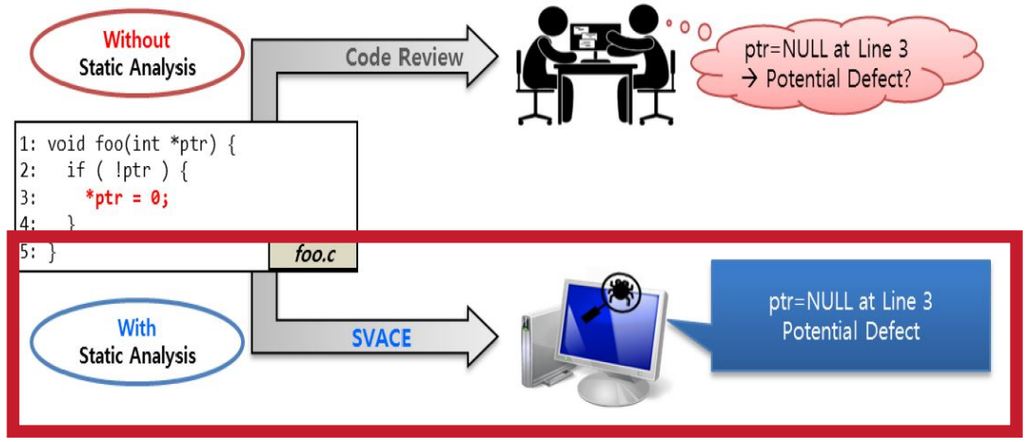
- Benefits:
  - Order.
  - Quick Feedback.
  - Visibility.
  - Automation.

TEST MANAGEMENT TOOLS

- Everything that a tester does day a day.
- Integration with different tools (Example: JIRA)
- Control of test activities.
- Metric collector.
- Used by testers.

STATIC ANALYSIS TOOLS

- What is static analysis?
- Analyse the code.
- Understand the structure of the code.
- Coding standars.
- Different languages.
- Mostly used by the developers.



STATIC ANALYSIS TOOLS

```
▼ <div id="make-model-zopcode-container">
  ▼ <p>
      <label class="body-boxes-step2-col2-label" id="makeSelect"> MAKE:</label> = $0
    ▶ <select id="makeSelect" name="makeSelect" class="body-boxes-step2-select" style="width: 200px;">…</select>
    </p>
  ▶ <p>…</p>
  ▶ <p>…</p>
  ▶ <p>…</p>
    <div class="clearboth"> </div>
```

Same Id

- id: Specifies a unique id for the element.

AUTOMATED TESTING TOOLS

- Annotations.
- WebApplications/Desktop Application.
- Integration with other tools.
- Different programming languages.
- Developers and Testers.

TestNG

LOAD AND STRESS TESTING TOOLS

- Web applications.
- Helps to ensure the performance of your application.
- Robustness of the application.
- Free and paid.
- Testers, developers or system administrators.

DEFECT TRACKING TOOLS

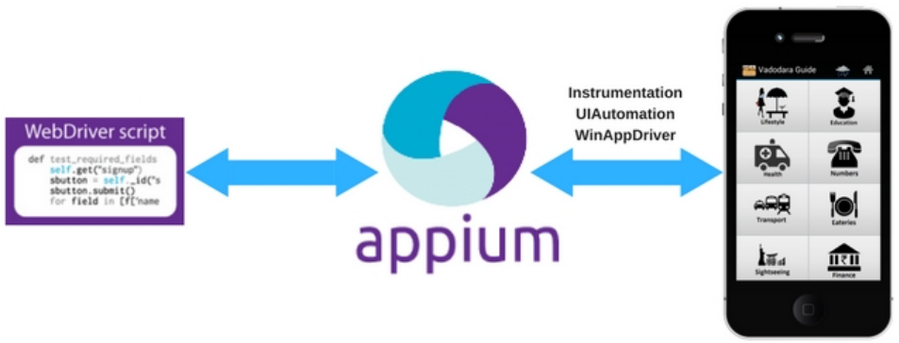
- Information centralized.
- Report bugs.
- Assigning.
- Project status.
- History, comments.
- Visibility.

REDMINE
flexible project management

Bugzilla

MOBILE TESTING TOOLS

- Native or web application.
- Mobile is being more used now.
- Functionality and usability testing.
- Compatibility.
- Performance.
- Automated or manual.
- Equivalence partitioning?



CROSSBROWSING TESTING TOOLS

- Features supported.
- Behaviour in different browsers.
- Web Standars.
- Functional tests.
- Manual or automated.
- Testers.
- Why you should use this type of tools?

SAUCELABS
Testing at the speed of awesome.

TestObject
A PART OF SAUCE LABS

API TESTING TOOLS

- What is API?
    - Application program interface.
- WebApplication/Mobile Applications.
- Integration testing.
- Functionality, Performance and security.
- No GUI (Graphic user interface)
- **Automated** or manual.
- Testers.

VISUAL TESTING TOOLS

- Element position.
- Rendering
- Integration with cross browser testing tools.
- Automated.
- Real users.

applitools

EXERCISE

- Pairs.
- Research one tool
  - Presentation.
  - What's the purpose of the tool.
  - How it works.
  - Tool: pro and cons.

THANK YOU.

SOURCE CODE MANAGEMENT

WHAT IS A VERSION CONTROL SYSTEM?

- Collaboration with the team.
- History track.
- Easy rollbacks.
- Non-Distributed.
  - Subversion, CVS.
- Distributed.
  - Git, Mercurial.

NON-DISTRIBUTED VS DISTRIBUTED

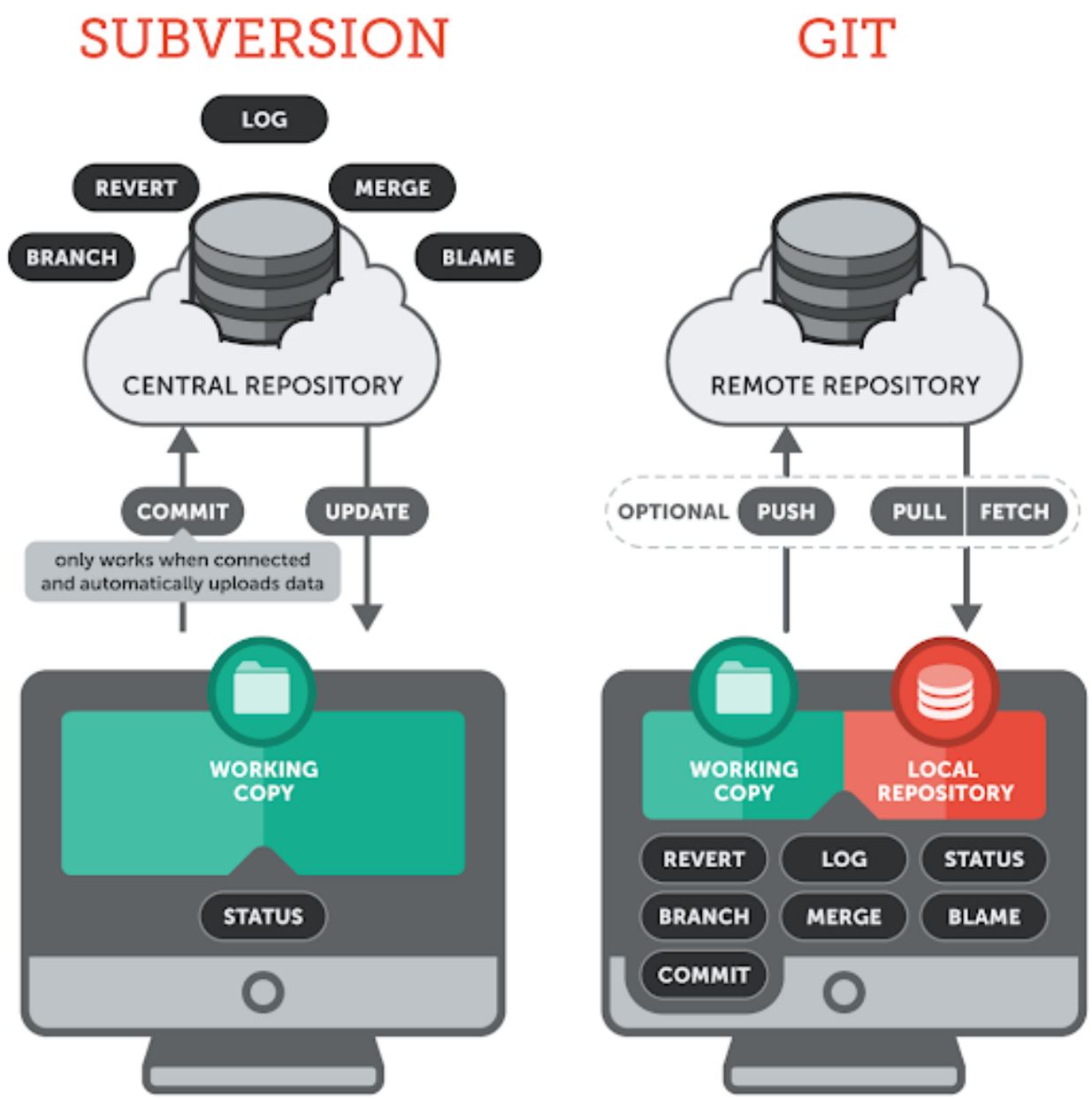


Photo: https://scriptcrunch.com/295/

INDIVIDUALS AND INTERACTIONS

OVER

PROCESSES AND TOOLS

GIT

- Created by Linus Torvalds - 2005.
- Created mainly for programmers but can track any files.
- Free & open source tool.
- Distributed.

Download: ***https://git-scm.com***

Optional: ***https://www.sourcetreeapp.com***

GIT – BASIC COMMANDS

- Git init
- Git clone -b <branch> <remote>
- Git add
- Git commit -m “commit message”
- Git push -u origin master
- Git pull
- Git merge

GIT – BRANCHES

Git branch <branchName>

Git checkout <branchName>

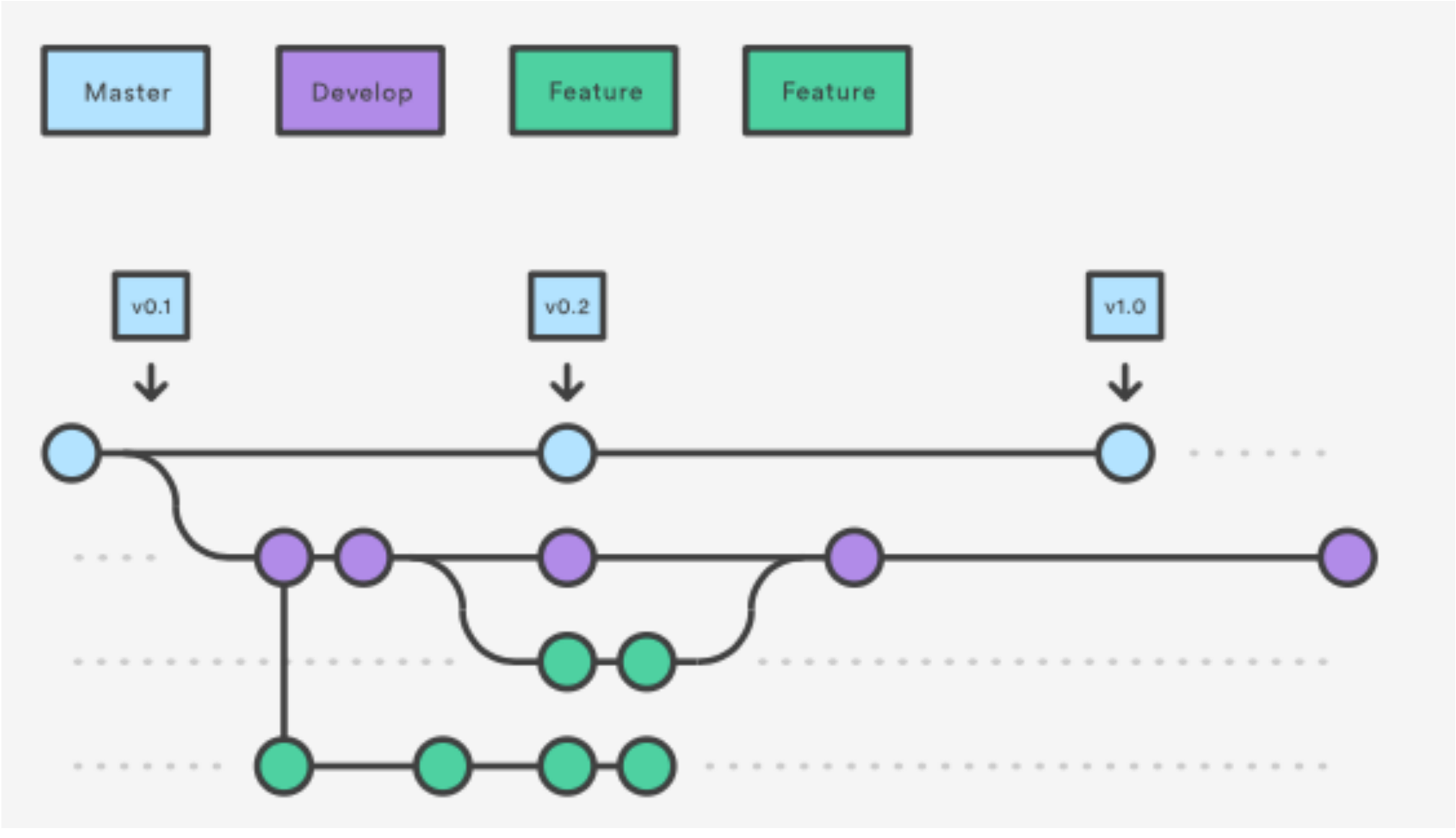
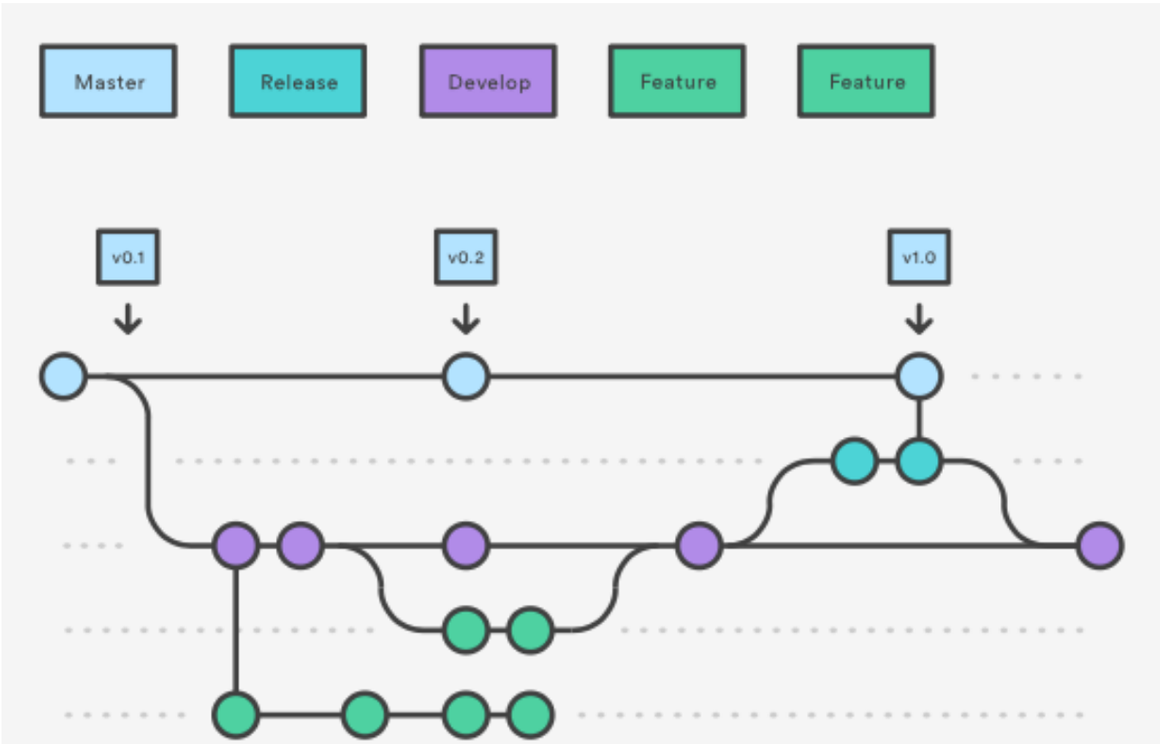


Photo: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

GIT – BRANCHES: GOOD PRACTICE

- Master - Protected.
  - Production or release branch.
- CI/develop/integration
  - CI branch (used to execute AT and UAT)
- Feature brach
  - Development of each feature or ticket.

GIT – MERGE & MERGE CONFLICTS



index.html styles.css

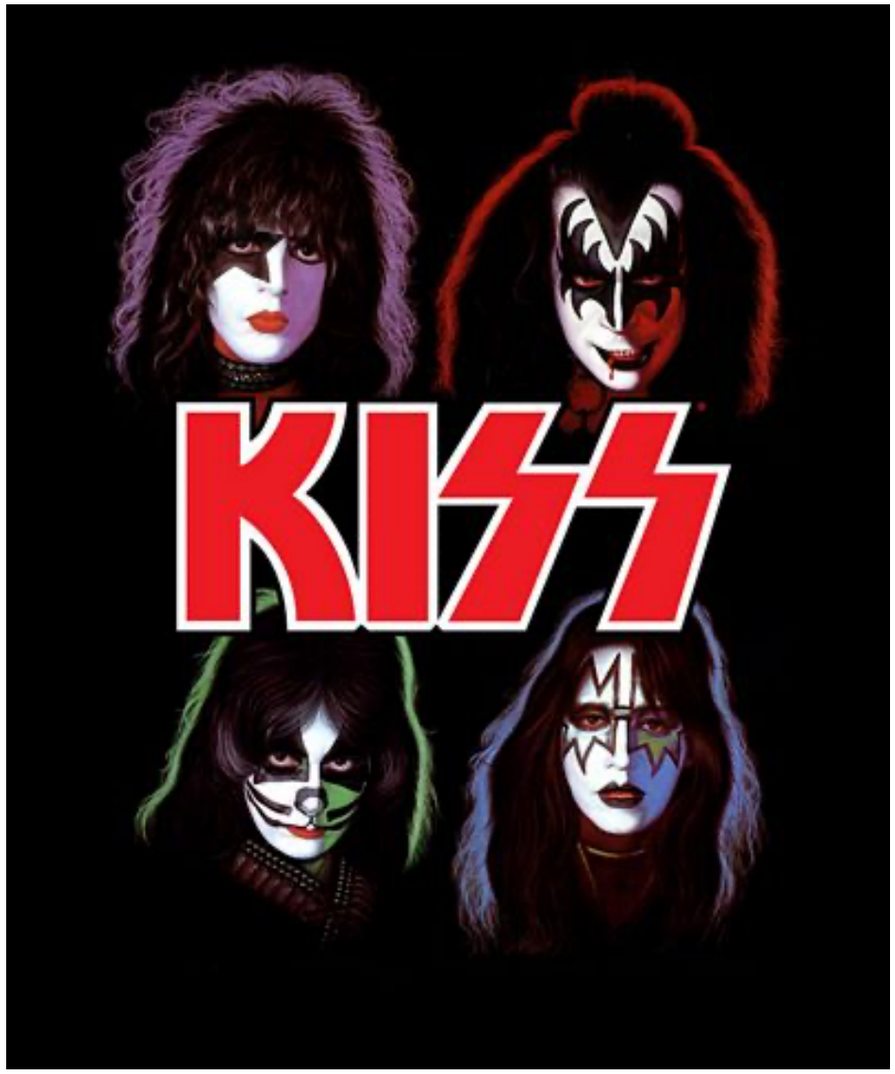
```
<!DOCTYPE html>
<html>
<head>
  <title>GC Merge Demo</title>
  <link href="styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
<<<<<<< HEAD
  <h1>Grand Circus Merge Demo</h1>
=======
  <h1 class="header">Merge Demo</h1>
>>>>>>> 1d46372af5a97f8ef05b9eecb82712382cc5f31c
  <p>
    Demo the merge.
  </p>
  <p class="footer">
    Grand Circus Detroit
  </p>
</body>
</html>
```

EXAMPLE

- Work in couples.
- Create a new project.
- Create different branches.
- Upload some files (in each branch).
- Update same file.
- Merge to develop or integration branch.
- Fix merge conflicts
- Merge to master.

10 Minutes.

SOME BEST PRACTICES IN SOFTWARE DEVELOPMENT



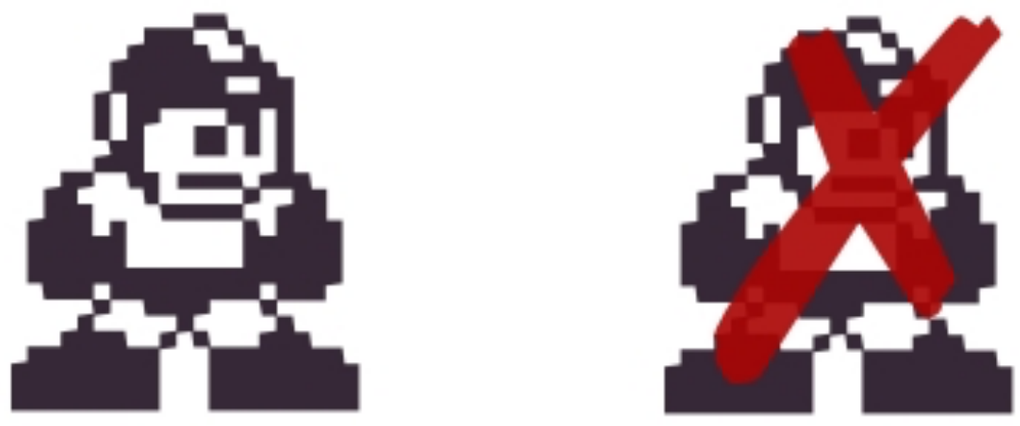
Keep it simple, s....

Y.A.G.N.I

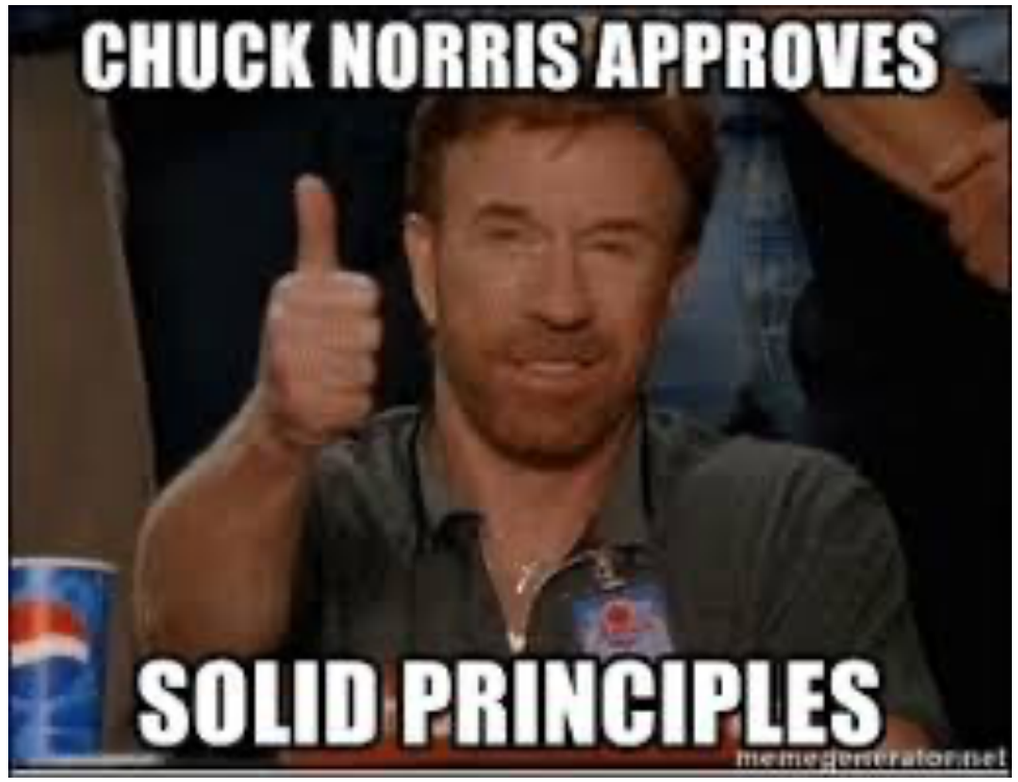
Don't do unnecessary code!

D.R.Y

DRY: Don`t Repeat Yourself

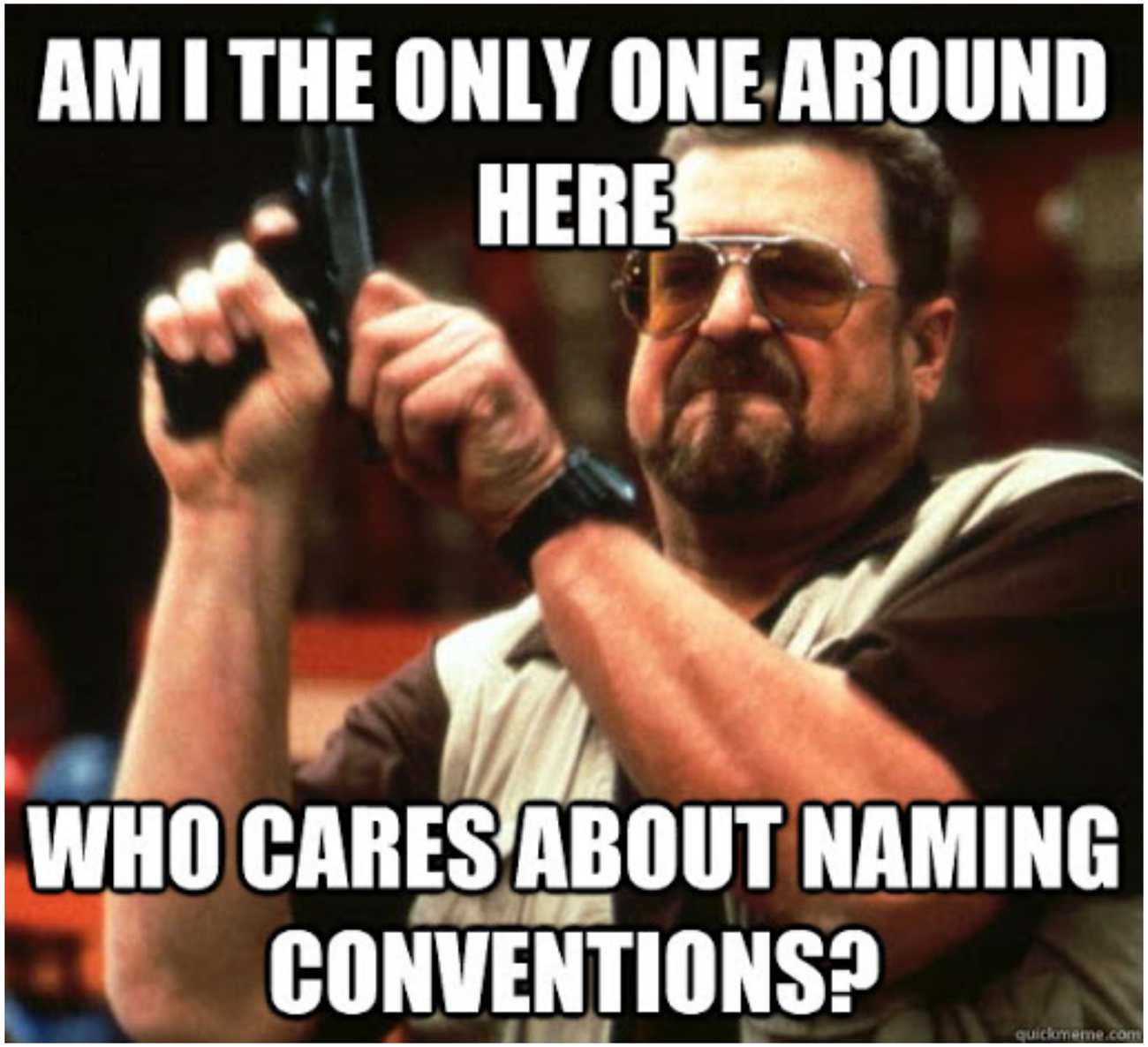


Avoid Duplication!



Use Solid principles.

Meaningful name in variable, methods, classes, etc



Follow a naming convention.

Popular naming conventions

- **Camel Case:**
  - **Eg:** camelCaseName
- **Snake Case:**
  - **Eg**: snake_case_name
- **Kebab Case:**
  - **Eg:** kebab-case-name

LONGEST PLACE NAME IN THE WORLD
Taumatawhakatangihangakoauauotamateaturipukakapikimaungahoronukupokaiwhenuakitanatahu

Name length

Use verbs in methods: Save, close, get, load, etc

Boolean methods: is, has, can.

Use nouns and nouns combinations for classes.

SOLID PRINCIPLES

WHAT IS SOLID?

- Robert C. Martin (Uncle Bob).
- Acronym for 5 principles using OOP.
- Purpose:
    - Understandable.
    - Easier to maintain
    - Easier to extend.

SOLID ACRONYM

S Ingle Responsibility Principle

O pen/Closed Principle

L iskov Substitution Principle

I interface Segregation Principle

D ependency Inversion Principle

S Ingle Responsibility Principle

- Only one single responsibility.
- “Do one thing and do it well”.
- “Reason to change”- Robert C. Martin.

Example

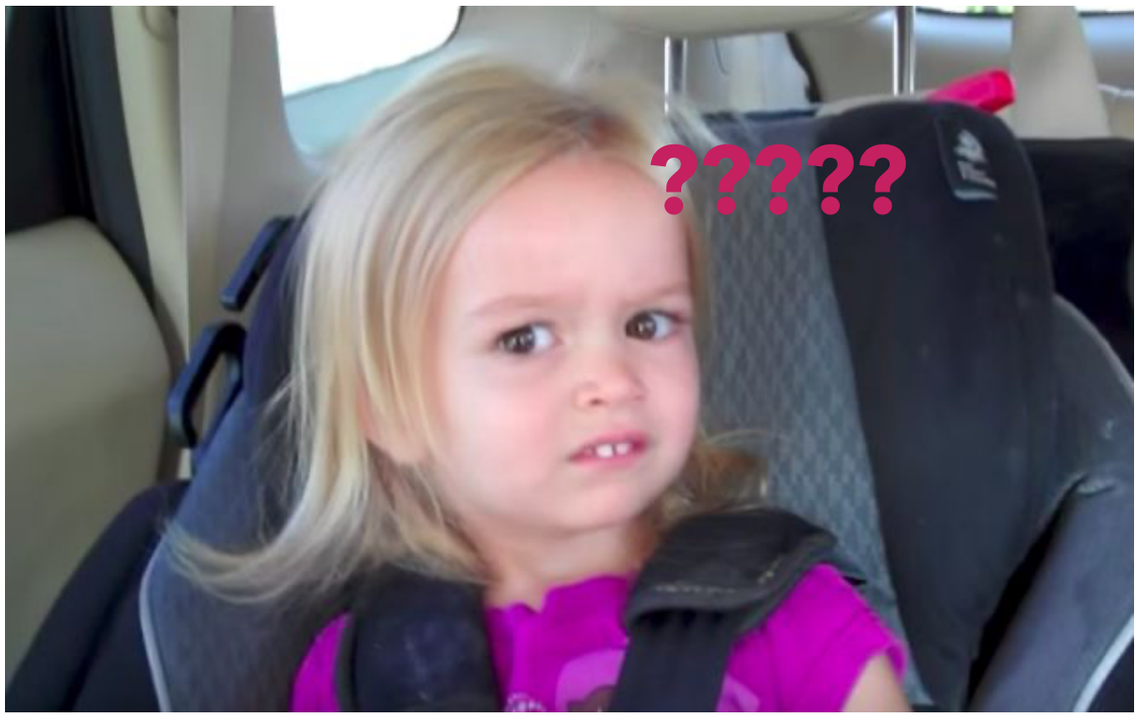
O pen/Closed Principle

- Open for extension BUT close to modification.
- Use inheritance and/or implementing.

Example

L iskov Substitution Principle

- If **"x"** is a subtype of **"y"** that means that the objects of type **"y"** may be replaced with objects of type **"x"**.



L iskov Substitution Principle

"A subclass should behave in such a way that it will not cause problems when used instead of the superclass."

Example

interface Segregation Principle

- Interfaces.
- “No one should be forced to depende on methods it does not use”
- Y.A.G.N.I.
- Create multiple interfaces.

Example

D ependency Inversion Principle

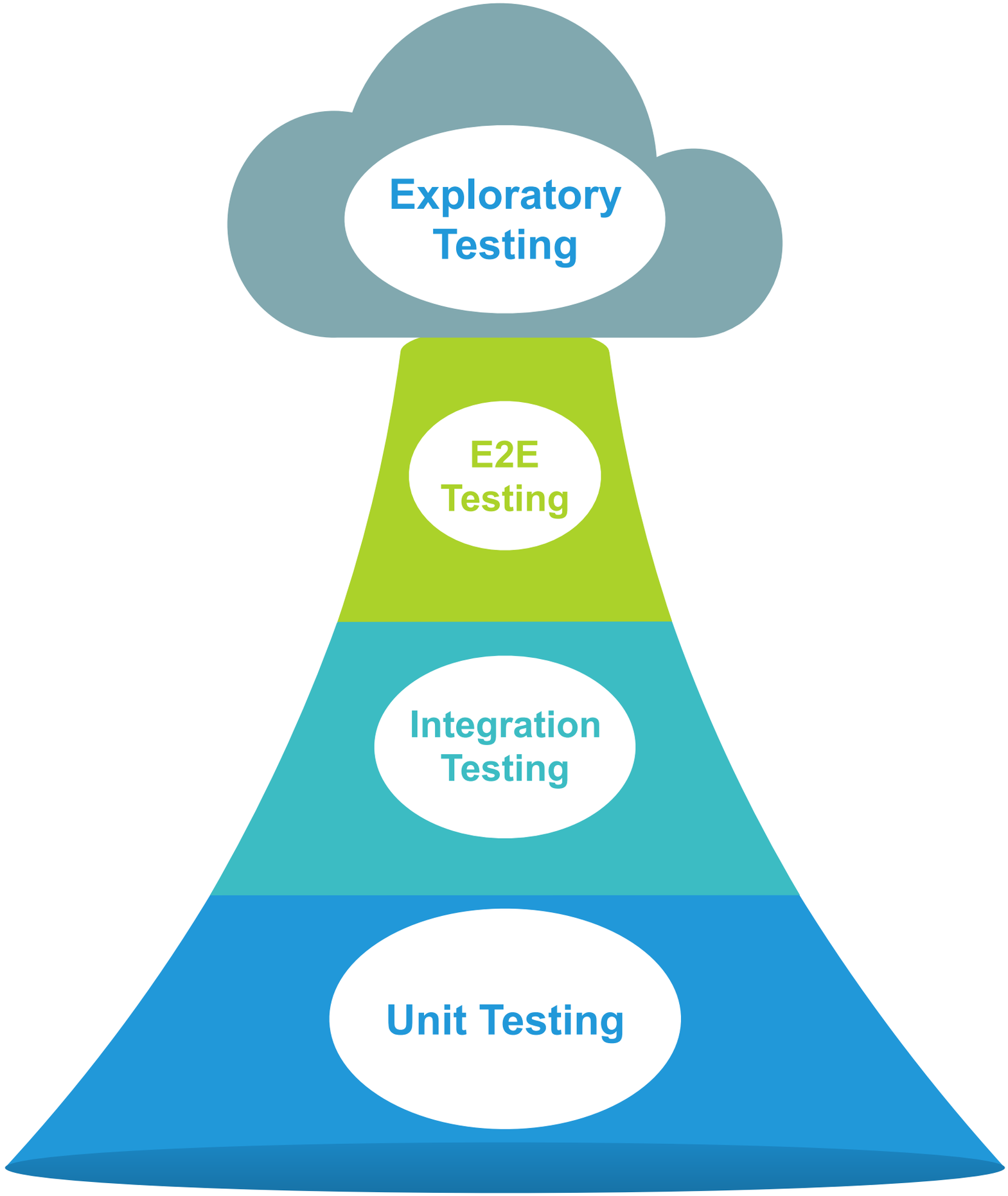
- High level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depende on details. Details should depend on abstractions.

Example

SOLID
Software Development is not a Jenga game

THANK YOU.

TESTING GROUPS USING TESTING TOOLS


Exploratory Testing
E2E Testing
Integration Testing
Unit Testing

- Build automation tool.
- Used primary for Java.
- Two aspects:
  - How the software is built.
  - Dependencies.
- XML file.

- Groupid - Project identifier across all projects.
- Artifactid - Jar name without version.
- Version - Jar version (if you distribute it)

POM File

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.ufm.introdevops</groupId>
    <artifactId>testing</artifactId>
    <version>1.0</version>

    <dependencies>
    <!--  ********************** AW DEPENDENCIES ************************-->
        <!-- https://mvnrepository.com/artifact/org.testng/testng -->
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>6.14.3</version>
            <scope>test</scope>
        </dependency>


    </dependencies>

</project>
```

TestNG

- Open source automation testing framework.
- Inspired from JUnit and NUnit but introducing new functionalities.
- Key Features:
  - Groups
  - Reports (Basic)
  - Integrated with other tools (Jenkins, Maven)
  - Annotations.

TestNG

- Testng.xml
  - Configuration file in testng.
  - Define test suites.
  - Include or exclude.
  - Parallelism.

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Example" preserve-order="true" allow-return-values="true" verbose="1">
    <test name="Example">
        <groups>
            <run>
                <exclude name="brokenTests" />
                <include name="regressionTests" />
            </run>
        </groups>
        <classes>
            <class name="Tests"/>
        </classes>
    </test>
</suite>
```

TestNG

- Common annotations:

@BeforeSuite:	The annotated method will be run before all tests in this suite have run.
@BeforeTest:	The annotated method will be run before any test method belonging to the classes inside the tag is run.
@BeforeClass:	The annotated method will be run before the first test method in the current class is invoked.
@BeforeMethod:	The annotated method will be run before each test method.
@Test:	The test method or class
@AfterMethod:	The annotated method will be run after each test method.
@AfterClass:	The annotated method will be run after all the test methods in the current class have been run.
@AfterTest:	The annotated method will be run after all the test methods belonging to the classes inside the tag have run.
@AfterSuite:	The annotated method will be run after all tests in this suite have run.

TestNG

- @Test Annotations attributes:
  - **description:** A string that gives information about the test it is attached to.
  - **TimeOut:** This is the maximum number of milliseconds for a test run.
  - **priority:** When this attribute is not declared in your tests, the order in which the test methods are run is random.
  - **dependsOnMethods:** When you want to run a test only after another test has run successfully.
  - **enabled:** This attribute has boolean values. This attribute is only worth specifying explicitly when you do not want a certain test method to be run, by setting the attribute to false
  - **groups:** This attribute is useful for grouping together tests that relate to the same functionality, are of the same importance or are of the same type.

TestNG

```
@Test(description = "Upper Case testt",
        timeOut = 20000,
        priority = 1,
        dependsOnMethods = "lowerCaseTest",
        groups = {"example", "test"})
public void upperCaseTest(){
    System.out.println("\n########################### upperCaseTest ############################
    String expectedResult = "INTRO TO DEVOPS";
    ExampleTest ex =  new ExampleTest();
    String actualResult = ex.getTextInUpperCase("Intro to devops");
    System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
    System.out.println("#########################################################################
    Assert.assertEquals(actualResult,expectedResult);
}
```

More annotations that we will cover in the next classes…

HOW CAN I DECIDE IF THE TEST PASS OR FAIL?

TestNG

- Assertions:
  - Hard Assert.
  - Soft Assert.

WHAT COULD BE THE DIFFERENCE?

TestNG

- Hard Assert:
  - Throw an AssertException after the test fails and the test is marked as failed.
  - Next test is executed.

TESTING GROUPS USING TESTING TOOLS

```java
import ...

/**
 * Created by josea.wolff on 1/19/19.
 */
public class Tests {

    @Test(description = "Upper Case testt",
            timeOut = 20000,
            priority = 1,
            //dependsOnMethods = "lowerCaseTest",
            groups = {"example", "test"})
    public void upperCaseTest(){
        System.out.println("\n########################### upperCaseTest ####################################");
        String expectedResult = "INTRO TO DEVOPS";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInUpperCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##############################################################################");
        Assert.assertEquals(actualResult,expectedResult);
        System.out.println("--> Hard assertion");
    }
```

Tests › upperCaseTest()

Tests passed: 1 of 1 test – 17 ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
objc[1087]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java
  (0x10f76d4c0) and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/jre/lib/libinstrument.dylib (0x10f7e74e0). One of the two
  will be used. Which one is undefined.

############################ upperCaseTest ####################################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
##############################################################################
--> Hard assertion

===============================================
Example
Total tests run: 1, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

TESTING GROUPS USING TESTING TOOLS

```
*/
public class Tests {

    @Test(description = "Upper Case testt",
            timeOut = 20000,
            priority = 1,
            //dependsOnMethods = "lowerCaseTest",
            groups = {"example", "test"})
    public void upperCaseTest(){
        System.out.println("\n########################### upperCaseTest ####################################");
        String expectedResult = "INTrRO TO DEVOPS";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInUpperCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##############################################################################");
        Assert.assertEquals(actualResult,expectedResult);
        System.out.println("--> Hard assertion");
    }
```

Tests › upperCaseTest()

ment... ×

Tests failed: 1 of 1 test – 16 ms

```
16 ms  ########################### upperCaseTest ####################################
16 ms  Expected Result: INTrRO TO DEVOPS
16 ms  ActualResult: INTRO TO DEVOPS
16 ms  ##############################################################################
16 ms

       java.lang.AssertionError: expected [INTrRO TO DEVOPS] but found [INTRO TO DEVOPS]
       Expected :INTrRO TO DEVOPS
       Actual   :INTRO TO DEVOPS
        <Click to see difference>


           at org.testng.Assert.fail(Assert.java:96)
           at org.testng.Assert.failNotEquals(Assert.java:776)
           at org.testng.Assert.assertEqualsImpl(Assert.java:137)
           at org.testng.Assert.assertEquals(Assert.java:118)
           at org.testng.Assert.assertEquals(Assert.java:453)
           at org.testng.Assert.assertEquals(Assert.java:463)
           at Tests.upperCaseTest(Tests.java:22) <7 internal calls>
           at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
           at java.util.concurrent.FutureTask.run(FutureTask.java:266)
           at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
           at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
           at java.lang.Thread.run(Thread.java:745)

       ===============================================
       Example
       Total tests run: 1, Failures: 1, Skips: 0
       ===============================================
```

?????

TestNG

- Soft Assert:
  - Collect errors during @Test.
  - Does not throw an AssertException.
  - Continue with the next step after an assertion happens.

TESTING GROUPS USING TESTING TOOLS

Project

- Java Projects [Examples] ~/Library/Mob
  - .idea
  - src
    - main
      - java
        - Example
      - resources
    - test
      - java
        - Tests
  - target
  - test-output
  - Examples.iml
  - pom.xml
  - testng.xml
- External Libraries
- Scratches and Consoles

testing | testng.xml | Tests.java | ExampleTest.java

```
8   */
9   public class Tests {
10
11      @Test(description = "Upper Case testt",
12              timeOut = 20000,
13              priority = 1,
14              //dependsOnMethods = "lowerCaseTest",
15              groups = {"example", "test"})
16      public void upperCaseTest(){
17          SoftAssert softAssert = new SoftAssert();
18          System.out.println("\n########################## upperCaseTest ##################################");
19          String expectedResult = "INTRO TO DEVOPS";
20          ExampleTest ex =  new ExampleTest();
21          String actualResult = ex.getTextInUpperCase("Intro to devops");
22          System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
23          System.out.println("##########################################################################");
24          softAssert.assertEquals(actualResult,expectedResult);
25          softAssert.assertEquals( s: "Fail test",expectedResult);
26          System.out.println("--> Continue");
27          System.out.println("--> Hard assertion");
28      }
29
30      @Test
31      public void lowerCaseTest(){
32          System.out.println("\n########################## lowerCaseTest ##################################");
```

Tests › upperCaseTest()

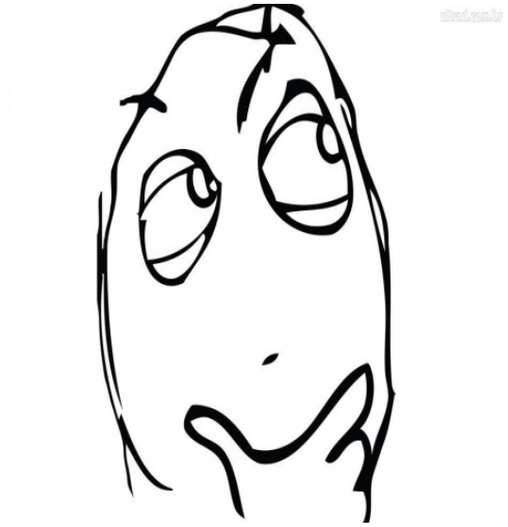
Run: /Users/josea.wolff/Library/Mobile Document...

Tests passed: 1 of 1 test – 20 ms

- Example 20 ms
  - <no name> 20 ms
    - Tests 20 ms
      - upperCaseTest 20 ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
objc[1122]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Hom
 (0x1050cb4c0) and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/jre/lib/libinstrument.dylib (0x10514
 will be used. Which one is undefined.

########################## upperCaseTest ##################################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
##########################################################################
--> Continue
--> Hard assertion

===============================================
Example
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

WHY THE TEST NOT FAILED?



- Assert.assertAll() - Last Statement in the @Test.

TESTING GROUPS USING TESTING TOOLS

Java Projects › src › test › java › Tests

/Users/josea.wolff/Library/Mobile Docum...o Devops/Clases/Java Projects/

Project

- Java Projects [Examples] ~/Library/Mob
  - .idea
  - src
    - main
      - java
        - Example
      - resources
    - test
      - java
        - Tests
  - target
  - test-output
  - Examples.iml
  - pom.xml
  - testng.xml
- External Libraries
- Scratches and Consoles

testing | testng.xml | Tests.java | ExampleTest.java

```
import ...

/**
 * Created by josea.wolff on 1/19/19.
 */
public class Tests {

    @Test(description = "Upper Case testt",
            timeOut = 20000,
            priority = 1,
            //dependsOnMethods = "lowerCaseTest",
            groups = {"example", "test"})
    public void upperCaseTest(){
        SoftAssert softAssert = new SoftAssert();
        System.out.println("\n########################### upperCaseTest ####################################");
        String expectedResult = "INTRO TO DEVOPS";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInUpperCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##########################################################################");
        softAssert.assertEquals(actualResult,expectedResult);
        softAssert.assertEquals( s: "Fail test",expectedResult);
        System.out.println("--> Continue");
        System.out.println("--> Hard assertion");
        softAssert.assertAll();
```

Tests › upperCaseTest()

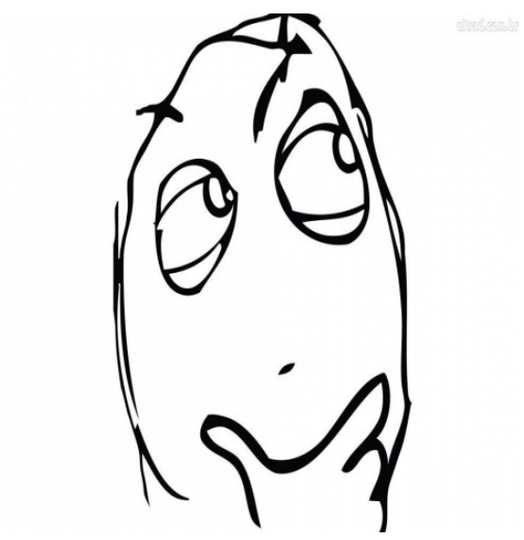
Run: /Users/josea.wolff/Library/Mobile Document...

Tests failed: 1 of 1 test – 16 ms

- Example 16 ms
  - <no name> 16 ms
    - Tests 16 ms
      - upperCaseTest 16 ms

```
objc[1143]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents
  (0x10534d4c0) and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/jre/lib/libinstrument.dylib (0x1
  will be used. Which one is undefined.

########################### upperCaseTest ####################################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
##########################################################################
--> Continue
--> Hard assertion

java.lang.AssertionError: The following asserts failed:
    expected [INTRO TO DEVOPS] but found [Fail test]
Expected :INTRO TO DEVOPS
Actual   :Fail test
 <Click to see difference>


    at org.testng.asserts.SoftAssert.assertAll(SoftAssert.java:43)
    at Tests.upperCaseTest(Tests.java:28) <7 internal calls>
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
```

WHEN YOU SHOULD USE A SOFT ASSERT AND HARD ASSERT?



LET'S
CODE
.IO

THANK YOU.

APPROACHES FOR TEST AUTOMATION SOLUTIONS (TAS)

WHAT IS A TEST CASE?

TEST CASE

- Sequence of actions that are executed against an SUT.
- Can be documented in test procedures.
- Test data.
- Include verification steps to verify the expected result.

APPROACHES TO CREATE A SEQUENCE OF ACTIONS(TEST CASE)

- TAE implements test case directly into automated test scripts.
- TAE design test procedures and transform them into automated scripts.
- TAE uses tools to generate automated test scrips/test procedures.

Reference:
TAE : Test Automation Engineer - ISTQB Advanced Level Test Automation Engineer

APPROACHES FOR AUTOMATION TEST CASES.

- Capture/Playback approach.
- Linear scripting approach.
- Structured scripting approach.
- Data-driven approach.
- Keyword-driven approach.
- Process-driven approach.

CAPTURE/PLAYBACK APPROACH

- Capture interaction with the SUT while performing the sequence of actions.
- **Pros**:
  - Easy to setup and use.
- **Cons**:
  - Scripts = hard to maintain.
  - Depends strongly on the SUT version.
  - Can start only when the SUT is available.

LINEAR SCRIPTING APPROACH

- Start with manual test procedures.
- Run manually (by a test analyst) while the test tool records a sequence of actions.
- Recorded scripts may be edited (e.g: adding comments).
- Replay using the tool.
- **Pros**:
  - No preparation to start automating features.
  - Programming skills are not require (but are usually helpful).
- **Cons**:
  - Effort to automate each test procedures (number of actions).
  - Duplicated code.
  - Non-modular and hard to maintain if the SUT changed.

STRUCTURED SCRIPTING APPROACH

WHAT'S THE DIFFERENCE WITH THE LINEAR SCRIPTING APPROACH?

STRUCTURED SCRIPTING APPROACH

- **Main difference:**
  - Introduction of script library.
- Reusable scripts (same sequence of actions).
- **Pros**:
  - Code maintenance.
  - Reduce the effort to add more test cases.
- **Cons**:
  - Initial effort to create the shared scripts is hard.
  - Programming skills will be required.

DATA-DRIVEN APPROACH

- Builds on the structured scripting approach.
- Inputs are extracted from the scripts
    - Data files
- Main test script can be reused multiple times.
    - Main test scripts is called control script.
- **Pros**:
    - Easy to add more scenarios (Ej: boundary values analysis)
        - Adding scenarios without changing the sourceCode.
- **Cons**:
    - Data files needs to be maintained.
    - Often negative testing may be missed.

KEYWORD-DRIVEN APPROACH

- Adaptation of data-driven approach.
- Actions words.
- High level instructions.
  - eg: SendEmail();
- Special Keywords can be used for verification steps.
- **Pros**:
  - Each keyword represent a number of actions.
  - Easy to maintain, read and write because the complexity is hidden in the keywords.
- **Cons**:
  - Implementation of each keyword.

PROCESS-DRIVEN APPROACH

- Adaptation of keyword-driven approach.
- Scenarios
  - Parametrized using the test data.
- High level instructions. (business language).
- **Pros**:
  - Test procedures are defined from a workflow perspective.
- **Cons**:
  - Complex to maintain if the SUT changes constantly
  - Test data is required.

IMPORTANT: NOT EVERYTHING SHOULD BE AUTOMATED

- Consider:
  - Frequency of use.
  - Complexity to automate.
  - Maturity of test process.
  - Stage of the software lifecycle.
  - Test Environment.
  - Parallel Effort.
  - Design for **testability**.

TESTABILITY

- Parts:
  - Observability
    - Interfaces that give insights into the system.
  - Control(ability)
    - Interfaces to perform actions on the SUT.
  - Clear defined architecture.
    - Clear and understandable interfaces giving control and visibility on all test levels.

METRICS

METRICS

- What do you want to measure?
- Internal.
- External.

INTERNAL METRICS – TAS

- **Tool scripting metrics.**
  - SonarQ.
  - Scripting standars.
- **Speed and efficiency of TAS components.**
  - SUT not performing the same functionality in the same time is a problem.
  - Could increase the load time of the SUT.
- **Automation code defect density.**
  - Good practices.
  - Not less importante than SUT.

EXTERNAL METRICS

- **Time to execute automated test.**
- **Ratio of failures to defects.**
  - Number of automated test that fails for a given defect.
- **Effort to maintain automated tests**
  - Keep tests in sync with the SUT.
  - % of tests that needs maintenance.
- **Effort to build automated tests**
  - Good practices.
  - Scripting approach.
  - Effort required to run the test manually.

EXTERNAL METRICS

- **Automation benefits**
  - Number of defect founded earlier.
  - Cost and time.
  - Number of hours of manual test effort saved.
  - Increase in coverage.
- **Effort to analyze automated test incidents.**
  - Analyze failures.
  - Logging.
    - Expected and actual behavior
    - Actions performed.
- **Number of automated tests.**
  - Shows the progression made.
  - NOTE: Not indicated test coverage (Excluding unit tests)

EXTERNAL METRICS

- **Pass and Fail test cases.**
  - Number of test passed and failed
  - Failures should be analyzed.
    - Problem with the SUT, TAS or Environment.
- **False fail and false pass.**
    - False alarms (problem with the TAS)
    - Reduce confidence in the TAS.
    - Could be Dangerous.
- **Unit test code coverage.**
  - Coverage.
  - Be aligned with a goal.

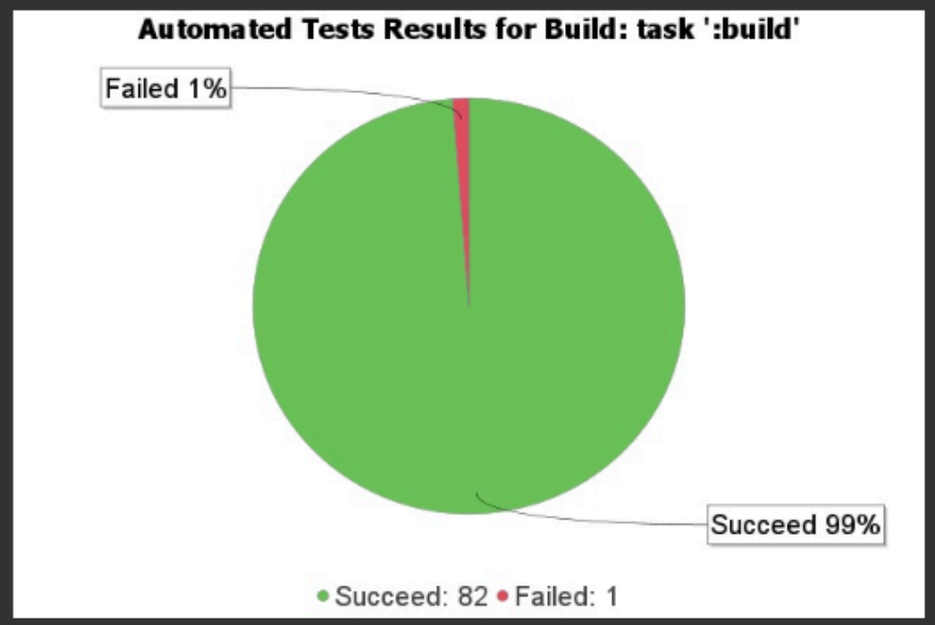
REPORTING

WHY IS IMPORTANT?

IN YOUR OPINION, WHAT ELEMENTS SHOULD BE INCLUDED IN A GOOD REPORT?

- Consider:
  - Testing type.
  - Test details.
  - Understandable.
  - Logging.
  - Environment.
  - **Test Status.**
  - Publish the report.
  - Feedback from the team.

REPORTING



Site	URL	Status	Error Description
Auto Dealer Online - Make Model Page	https://www.auto-dealer-online.com/acura/ilx/	Failed	Status Code: 500 - Status Message: Internal Server Error

REPORTING

FAILED - checkDirect

Class Name: testCases.CheckListings

Test Name: checkDirect

Description: This LoadTimeMonitis check if the url that receive as a parameter display listings.

Groups: all, CheckListings

Error Message: The Placement: 740 - Autobytel with Url: https://www.autobytel.com/car-price-quotes/aweb-fas/?make=ford&model=f-150&series=&zipcode=01010&submit=1&awpop=2&aw-track=false doesn't display Listings

Execution Time: 00 h., 00 min., 30 sec. At 09/22/2020 - 12:13:19

Parameters:

(class java.lang.String) 740 - Autobytel

(class java.lang.String) https://www.autobytel.com/car-price-quotes/aweb-fas/?make=ford&model=f-150&series=&zipcode=01010&submit=1&awpop=2&aw-track=false

Current URL: https://www.autobytel.com/car-price-quotes/aweb-fas/?make=ford&model=f-150&series=&zipcode=01010&submit=1&awpop=2&aw-track=false

This site can't be reached

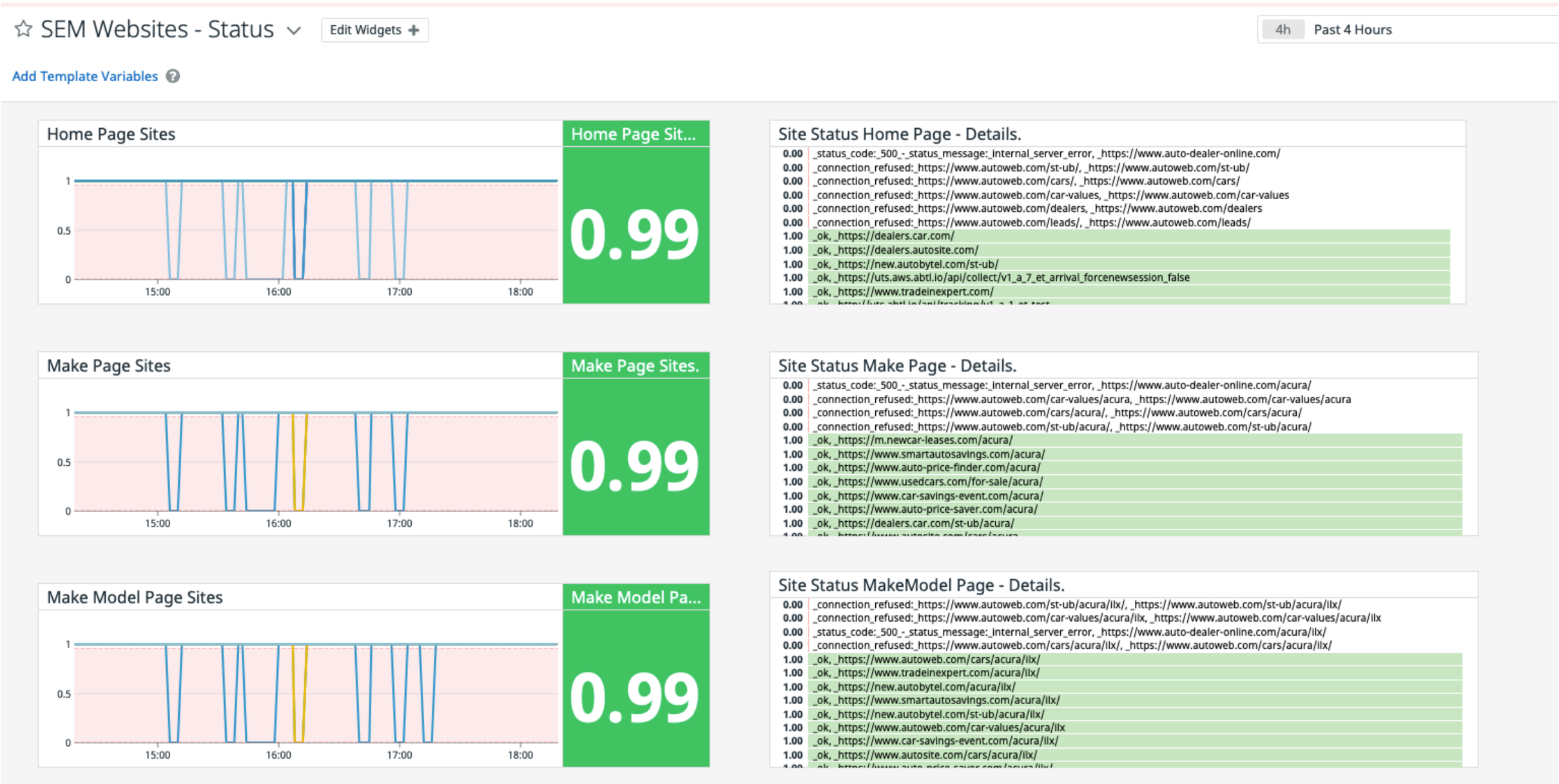
www.autobytel.com refused to connect.

Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED

REPORTING

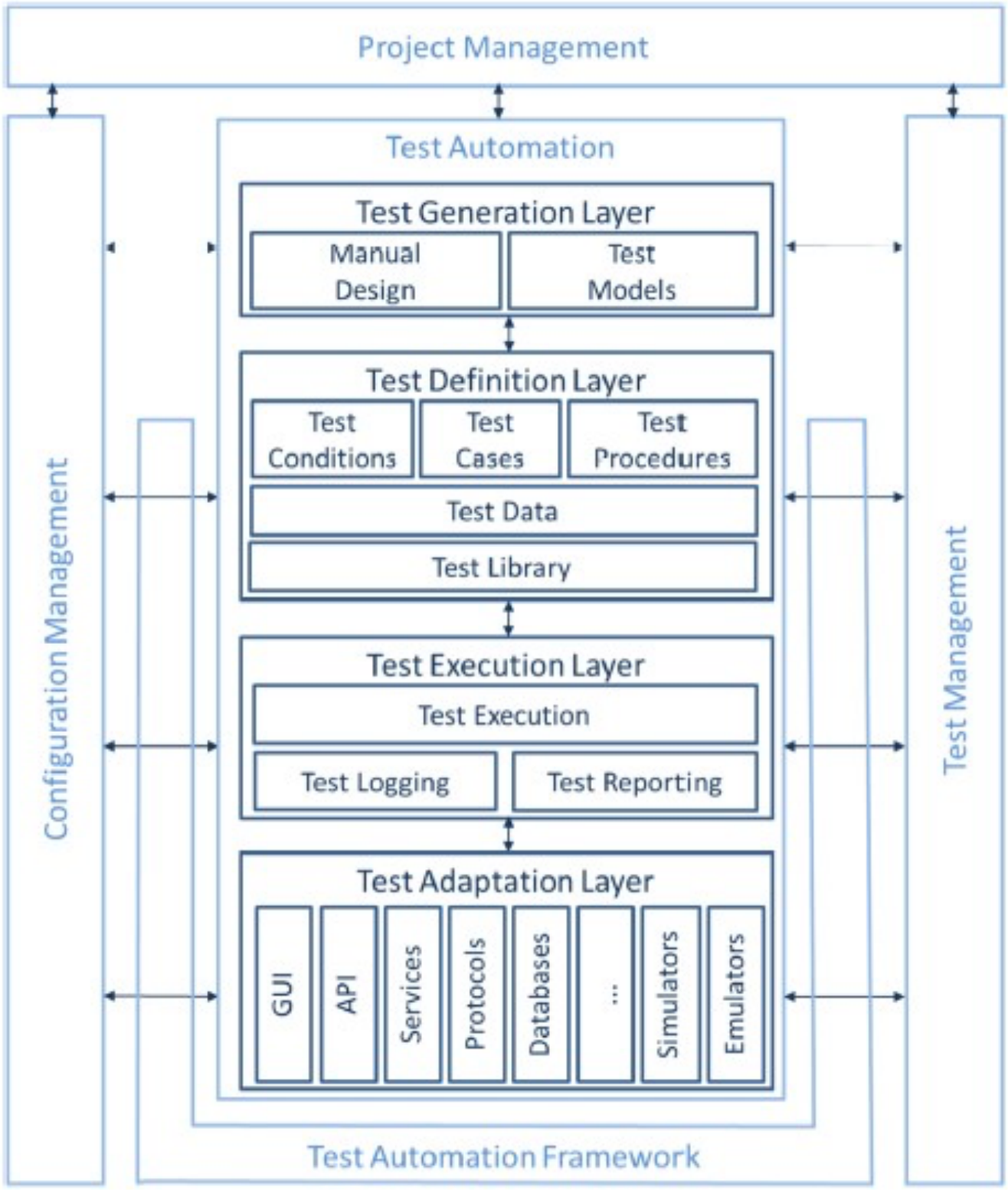

SEM Websites - Status
Edit Widgets
4h
Past 4 Hours
Add Template Variables
Home Page Sites
Home Page Sit...
0.99
Site Status Home Page - Details.
0.00 _status_code:_500_-_status_message:_internal_server_error, _https://www.auto-dealer-online.com/
0.00 _connection_refused:_https://www.autoweb.com/st-ub/, _https://www.autoweb.com/st-ub/
0.00 _connection_refused:_https://www.autoweb.com/cars/, _https://www.autoweb.com/cars/
0.00 _connection_refused:_https://www.autoweb.com/car-values, _https://www.autoweb.com/car-values
0.00 _connection_refused:_https://www.autoweb.com/dealers, _https://www.autoweb.com/dealers
0.00 _connection_refused:_https://www.autoweb.com/leads/, _https://www.autoweb.com/leads/
1.00 _ok, _https://dealers.car.com/
1.00 _ok, _https://dealers.autosite.com/
1.00 _ok, _https://new.autobytel.com/st-ub/
1.00 _ok, _https://uts.aws.abtl.io/api/collect/v1_a_7_et_arrival_forcenewsession_false
1.00 _ok, _https://www.tradeinexpert.com/
Make Page Sites
Make Page Sites.
0.99
Site Status Make Page - Details.
0.00 _status_code:_500_-_status_message:_internal_server_error, _https://www.auto-dealer-online.com/acura/
0.00 _connection_refused:_https://www.autoweb.com/car-values/acura, _https://www.autoweb.com/car-values/acura
0.00 _connection_refused:_https://www.autoweb.com/cars/acura/, _https://www.autoweb.com/cars/acura/
0.00 _connection_refused:_https://www.autoweb.com/st-ub/acura/, _https://www.autoweb.com/st-ub/acura/
1.00 _ok, _https://m.newcar-leases.com/acura/
1.00 _ok, _https://www.smartautosavings.com/acura/
1.00 _ok, _https://www.auto-price-finder.com/acura/
1.00 _ok, _https://www.usedcars.com/for-sale/acura/
1.00 _ok, _https://www.car-savings-event.com/acura/
1.00 _ok, _https://www.auto-price-saver.com/acura/
1.00 _ok, _https://dealers.car.com/st-ub/acura/
Make Model Page Sites
Make Model Pa...
0.99
Site Status MakeModel Page - Details.
0.00 _connection_refused:_https://www.autoweb.com/st-ub/acura/ilx/, _https://www.autoweb.com/st-ub/acura/ilx/
0.00 _connection_refused:_https://www.autoweb.com/car-values/acura/ilx, _https://www.autoweb.com/car-values/acura/ilx
0.00 _status_code:_500_-_status_message:_internal_server_error, _https://www.auto-dealer-online.com/acura/ilx/
0.00 _connection_refused:_https://www.autoweb.com/cars/acura/ilx/, _https://www.autoweb.com/cars/acura/ilx/
1.00 _ok, _https://www.autoweb.com/cars/acura/ilx/
1.00 _ok, _https://www.tradeinexpert.com/acura/ilx/
1.00 _ok, _https://new.autobytel.com/acura/ilx/
1.00 _ok, _https://www.smartautosavings.com/acura/ilx/
1.00 _ok, _https://new.autobytel.com/st-ub/acura/ilx/
1.00 _ok, _https://www.autoweb.com/car-values/acura/ilx
1.00 _ok, _https://www.car-savings-event.com/acura/ilx/
1.00 _ok, _https://www.autosite.com/cars/acura/ilx/

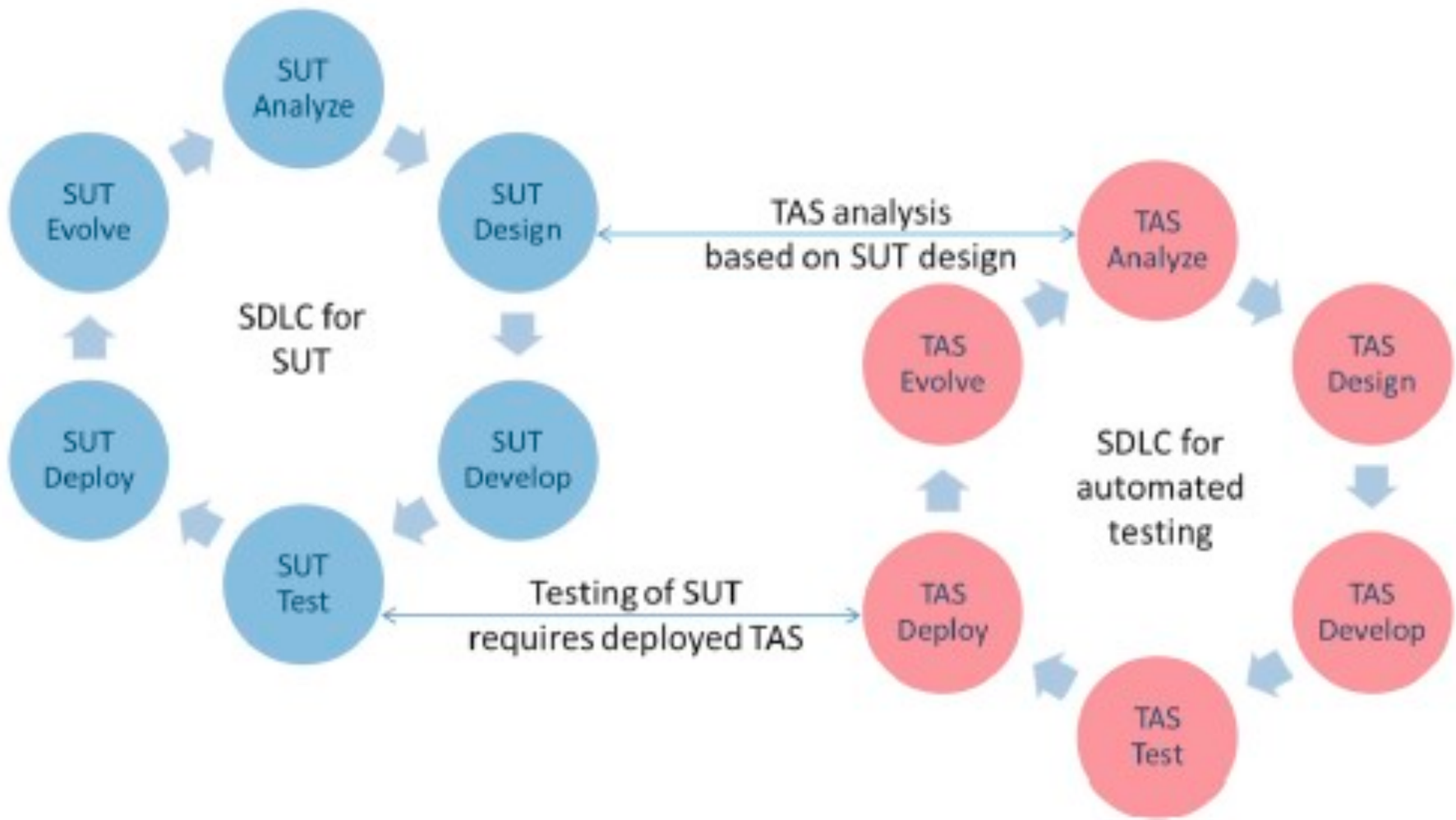
GENERIC TEST AUTOMATION ARCHITECTURE

GENERIC TEST AUTOMATION ARCHITECTURE

- Structure:
  - Test generation
  - Test definition
  - Test Execution
  - Test Adaptation

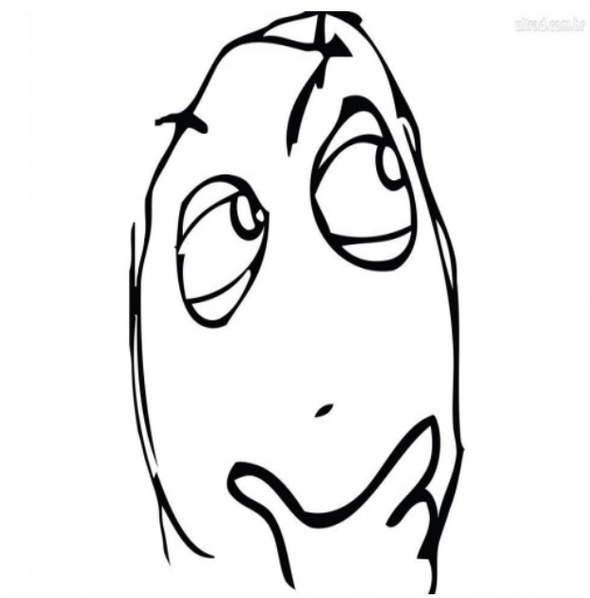
GENERIC TEST AUTOMATION ARCHITECTURE



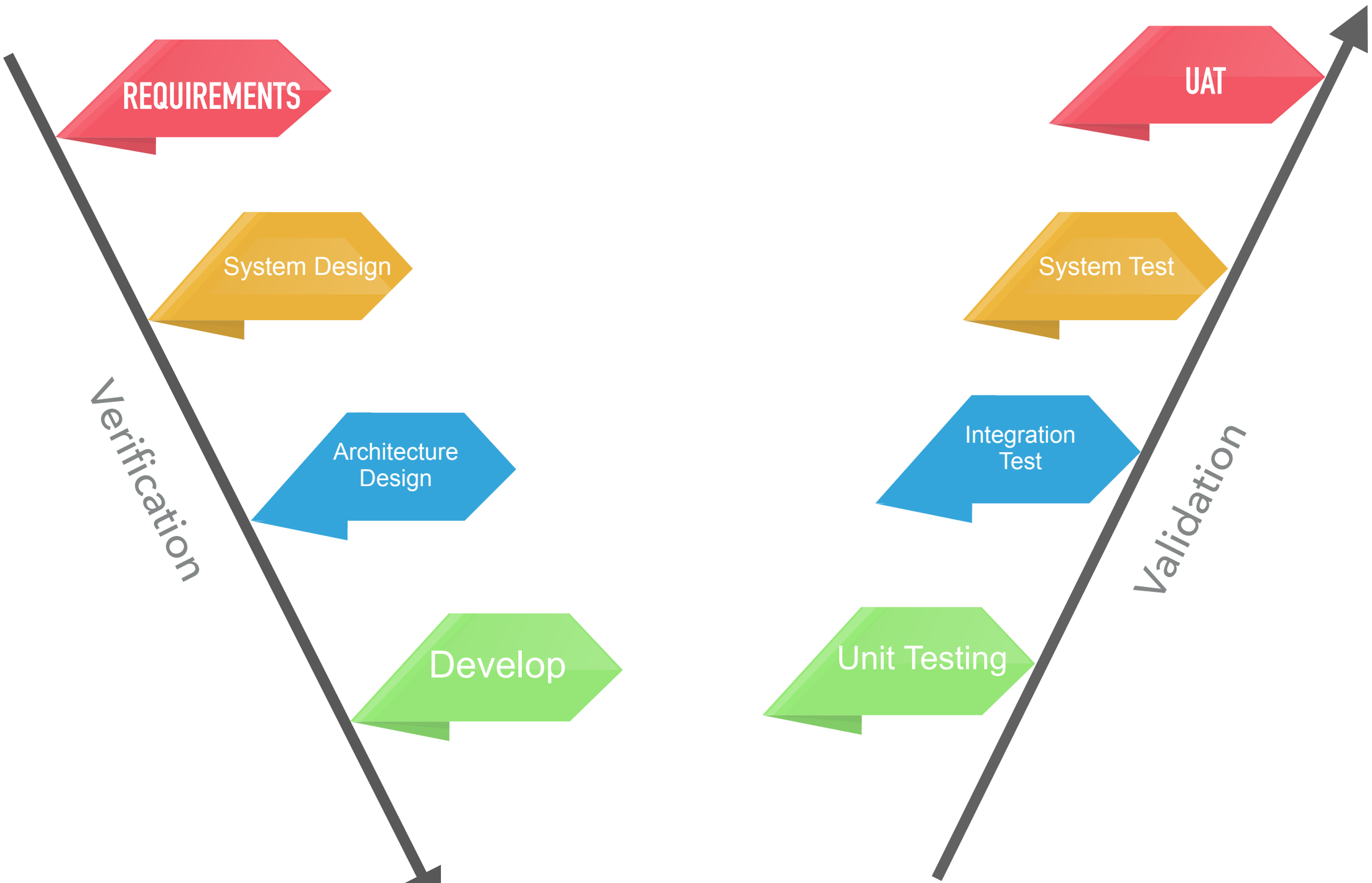

SUT Analyze
SUT Design
SUT Develop
SUT Test
SUT Deploy
SUT Evolve
SDLC for SUT
TAS analysis based on SUT design
Testing of SUT requires deployed TAS
TAS Analyze
TAS Design
TAS Develop
TAS Test
TAS Deploy
TAS Evolve
SDLC for automated testing

V MODEL

WHAT IS THE V MODEL?

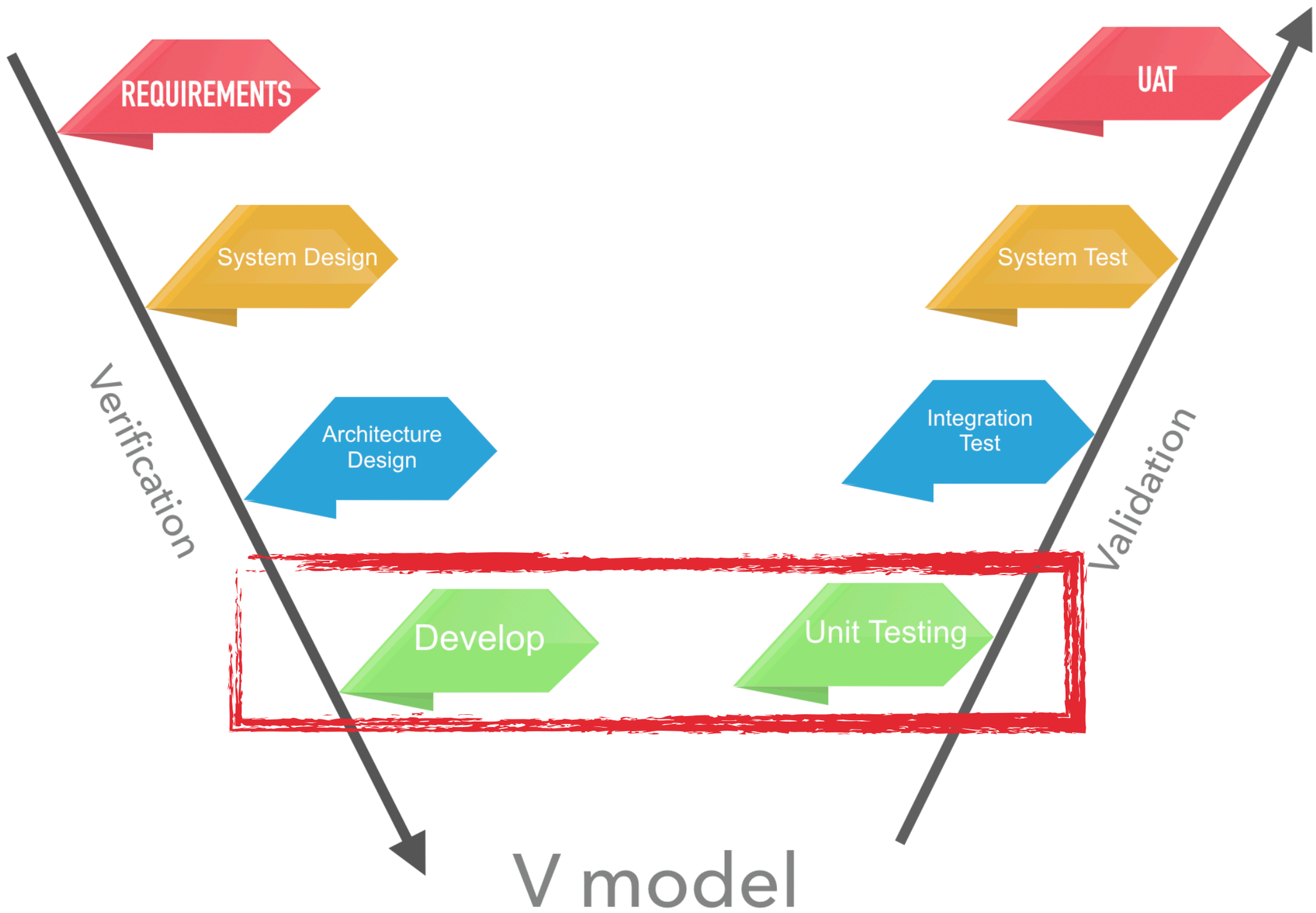


V MODEL



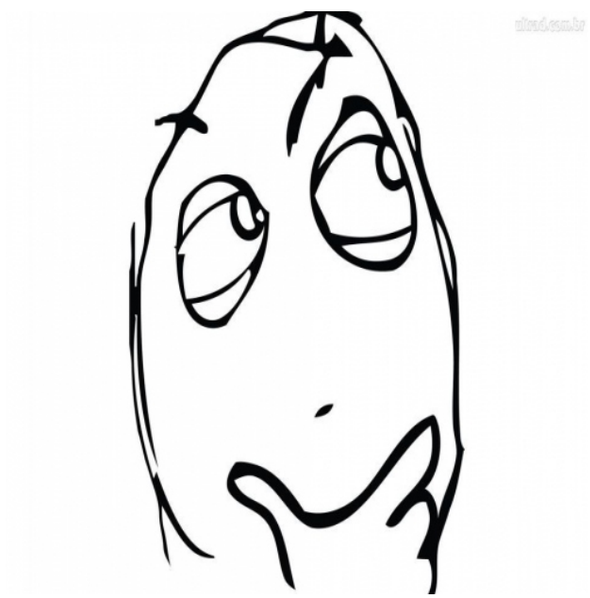
V model

V MODEL

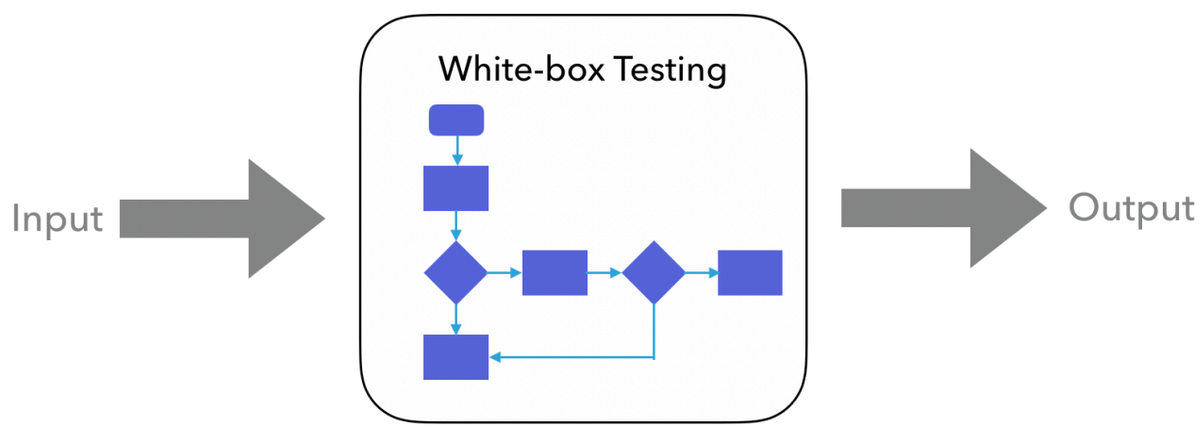

REQUIREMENTS
System Design
Architecture Design
Develop
Unit Testing
Integration Test
System Test
UAT
Verification
Validation
V model

UNIT TESTING

WHAT IS UNIT TESTING?



- First level of test pyramid.
- Testing Quadrants - Q1.
- Confidence in the component's quality.
- Find defects in the components.
- Developers.
- White-box Testing.



- Statement Testing
- Decision Testing

EXAMPLE

```java
/**
 * Created by josea.wolff on 1/19/19.
 */
public class ExampleTest {

    public String getTextInUpperCase(String text){
        return text.toUpperCase();
    }

    public String getTextInLowerCase(String text){
        return text.toLowerCase();
    }
}
```

```java
import Example.ExampleTest;
import org.testng.Assert;
import org.testng.annotations.Test;

/**
 * Created by josea.wolff on 1/19/19.
 */
public class Tests {

    @Test
    public void upperCaseTest(){
        System.out.println("\n########################## upperCaseTest ####################################");
        String expectedResult = "INTRO TO DEVOPS";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInUpperCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##########################################################################");
        Assert.assertEquals(actualResult,expectedResult);
    }

    @Test
    public void lowerCaseTest(){
        System.out.println("\n########################## lowerCaseTest ####################################");
        String expectedResult = "intro to devops";
        ExampleTest ex =  new ExampleTest();
        String actualResult = ex.getTextInLowerCase("Intro to devops");
        System.out.println("Expected Result: " + expectedResult + "\nActualResult: " + actualResult);
        System.out.println("##########################################################################");
        Assert.assertEquals(actualResult,expectedResult);
    }
}
```

EXAMPLE

```
Example                                                    9ms
  <no name>                                                9ms
    Tests                                                  9ms
      lowerCaseTest                                        9ms
      upperCaseTest                                        0ms
```

```
All 2 tests passed – 9ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
objc[32713]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMach
  .jdk/Contents/Home/bin/java (0x106cc24c0) and /Library/Java/JavaVirtualMachines/jdk1.8
  .jdk/Contents/Home/jre/lib/libinstrument.dylib (0x106d4e4e0). One of the two will be u

########################### lowerCaseTest ####################################
Expected Result: intro to devops
ActualResult: intro to devops
##############################################################################

########################### upperCaseTest ####################################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
##############################################################################

===============================================
Example
Total tests run: 2, Failures: 0, Skips: 0
===============================================

Process finished with exit code 0
```

EXAMPLE

```
/**
 * Created by josea.wolff on 1/19/19.
 */
public class ExampleTest {

    public String getTextInUpperCase(String text){
        return text.toUpperCase();
    }

    public String getTextInLowerCase(String text){
        return text.toUpperCase();
    }
}
```

```
Example                                  10ms
  <no name>                              10ms
    Tests                                10ms
      lowerCaseTest                      10ms
      upperCaseTest                       0ms
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/ja
objc[32816]: Class JavaLaunchHelper is implemented in both /Library/Jav
  and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/
  one is undefined.

############################ lowerCaseTest ############################
Expected Result: intro to devops
ActualResult: INTRO TO DEVOPS
###################################################################################

java.lang.AssertionError: expected [intro to devops] but found [INTRO TO DEVOPS]
Expected :intro to devops
Actual   :INTRO TO DEVOPS
 <Click to see difference>


    at org.testng.Assert.fail(Assert.java:96)
    at org.testng.Assert.failNotEquals(Assert.java:776)
    at org.testng.Assert.assertEqualsImpl(Assert.java:137)
    at org.testng.Assert.assertEquals(Assert.java:118)
    at org.testng.Assert.assertEquals(Assert.java:453)
    at org.testng.Assert.assertEquals(Assert.java:463)
    at Tests.lowerCaseTest(Tests.java:29) <16 internal calls>
    at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)
    at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:84) <4 internal calls>
    at org.testng.IDEARemoteTestNG.run(IDEARemoteTestNG.java:72)
    at org.testng.RemoteTestNGStarter.main(RemoteTestNGStarter.java:127)


############################ upperCaseTest ############################
Expected Result: INTRO TO DEVOPS
ActualResult: INTRO TO DEVOPS
###################################################################################

===============================================
Example
Total tests run: 2, Failures: 1, Skips: 0
===============================================


Process finished with exit code 0
```

STATEMENT AND DECISION COVERAGE

- What are they?
- Statement Coverage
    - Executable statements in the code.
- Decision Coverage
    - Exercises the decisions in the code and tests the code that is executed based on the decision outcomes.

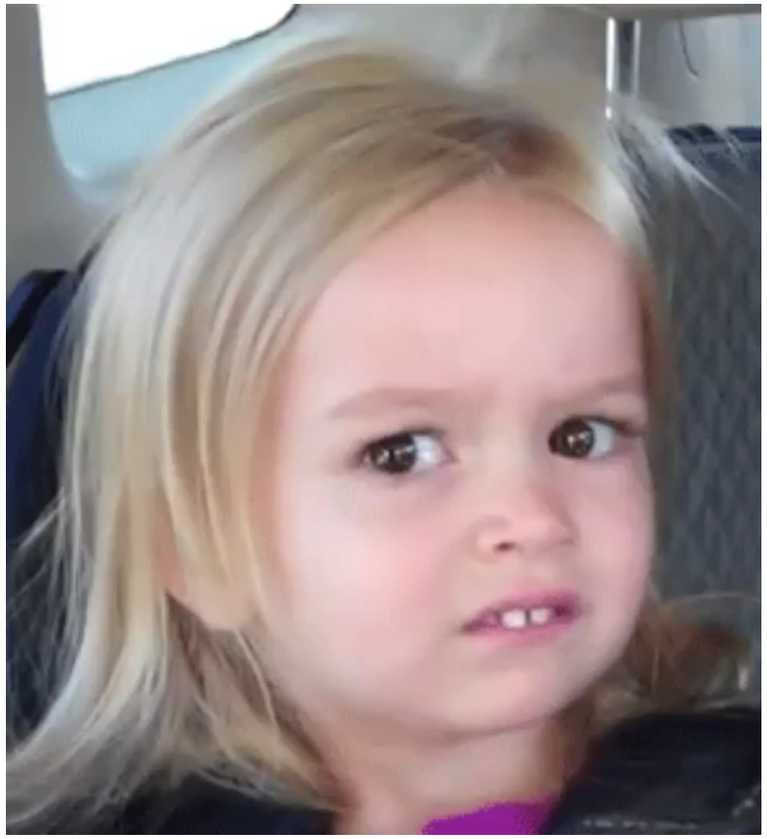
$$Statement\ Coverage = \frac{Number\ of\ executed\ statements}{Total\ number\ of\ statements} \times 100$$

$$Decision\ Coverage = \frac{Number\ of\ executed\ Decisions/Branches}{Total\ number\ of\ Decisions/Branches} \times 100$$

100% decision coverage

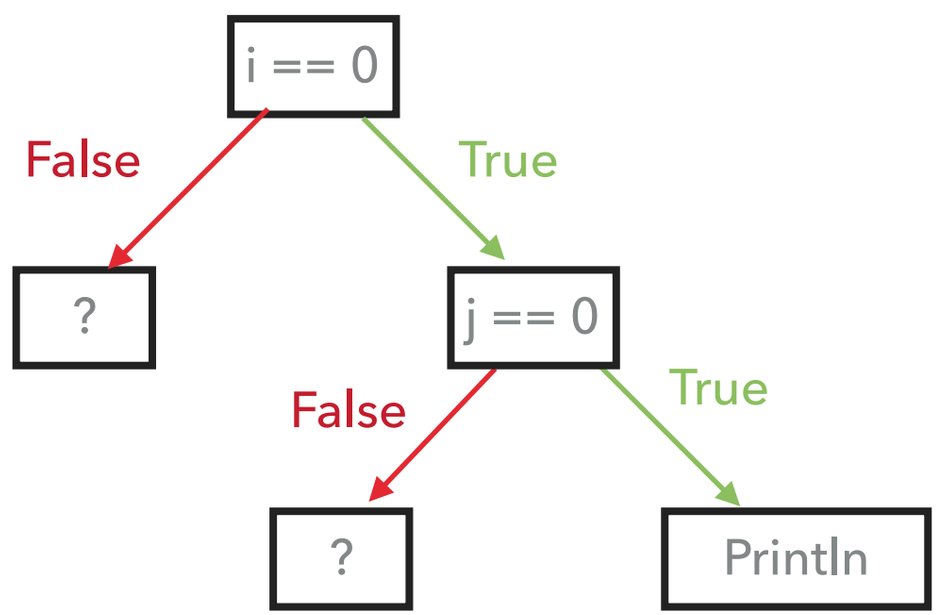
means

100% of statement coverage **not viceversa**.



EXAMPLE

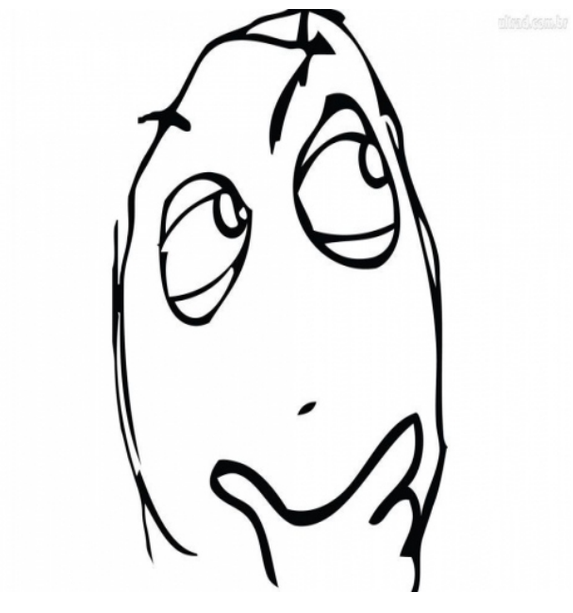
```
@Test(groups = {"Example"}, description = "Example")
public void example(){
    int i = 0;
    int j = 0;
    if(i == 0){
        if( j > 1){
            System.out.println("j is > than 1");
        }
    }
}
```



EXAMPLE

```
@Test(groups = {"Example"}, description = "Example")
public void example(){
    int i = 0;
    int j = 0;
    if(i == 0){
        if( j > 1){
            System.out.println("j is > than 1");
        }
    }
}
```

WHAT'S WRONG WITH THAT CODE?

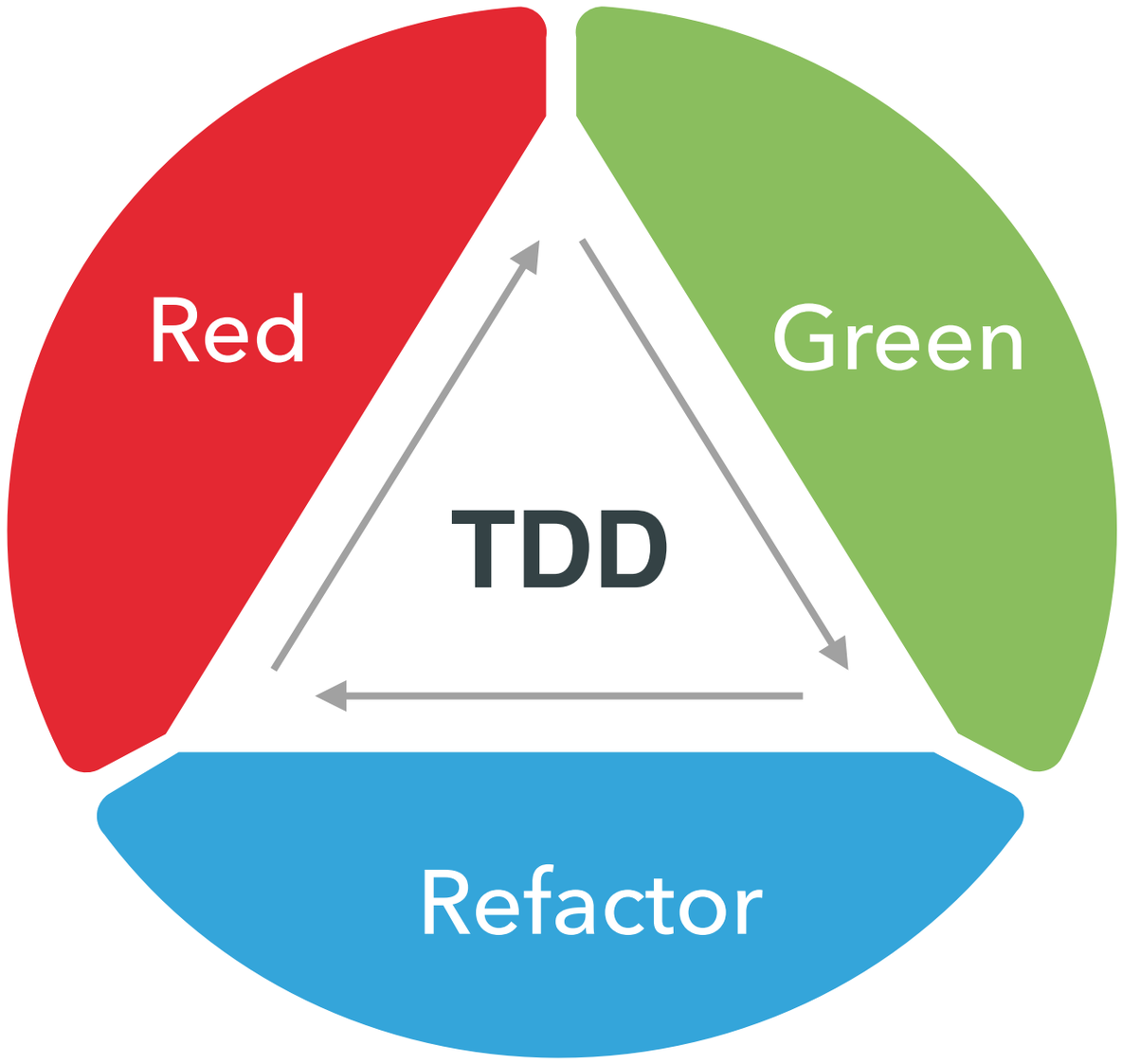


TDD

TEST DRIVEN DEVELOPMENT

- Also called Red-Green-Refactor.
- Test first programming concept.
- Pros:
    - Modular code.
    - Good architecture.
    - Documentation, etc.
- Cons:
    - Hard to write.
    - At the beginning the development could be slow.
    - Legacy code.

Test-Driven Development (TDD)



TESTNG – PARAMETERS

PARAMETERS

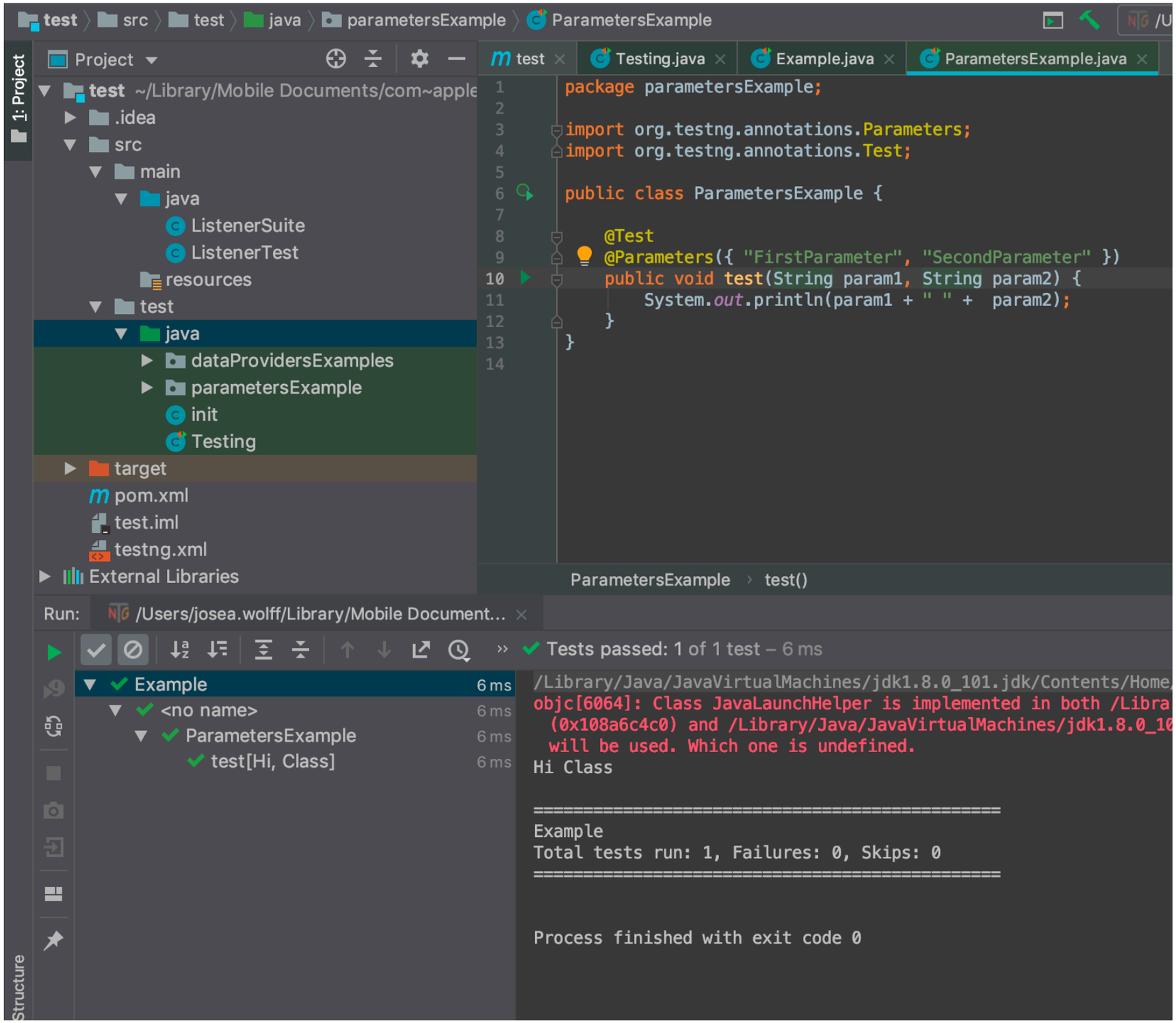
- Run multiple times the same test with different parameters.
- Specify in the testng.xml file.
- @Before/ @After and @Factory annotations.
- Can be optionals (@Optional).

PARAMETERS – EXAMPLE

```java
package parametersExample;

import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class ParametersExample {

    @Test
    @Parameters({ "FirstParameter", "SecondParameter" })
    public void test(String param1, String param2) {
        System.out.println(param1 + " " +  param2);
    }
}
```

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Example" preserve-order="true" allow-return-values="true" verbose="1">
    <parameter name="FirstParameter" value="Hi"/>
    <parameter name="SecondParameter" value="Class"/>
    <test name="Example">
        <classes>
            <class name="parametersExample.ParametersExample"/>
        </classes>
    </test>

    <listeners>
        <listener class-name="ListenerTest"/>
        <listener class-name="ListenerSuite"/>
    </listeners>
</suite>
```

PARAMETERS – EXAMPLE



PARAMETERS – EXAMPLE2 (@OPTIONAL)

```java
package parametersExample;

import org.testng.annotations.Optional;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class ParametersExample {

    @Test
    @Parameters({ "FirstParameter", "SecondParameter" })
    public void test(@Optional String param1, String param2) {
        System.out.println(param1 + " " +  param2);
    }
}
```

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Example" preserve-order="true" allow-return-values="true" verbose="1">
    <!--<parameter name="FirstParameter" value="Hi"/>-->
    <parameter name="SecondParameter" value="Class"/>
    <test name="Example">
        <classes>
            <class name="parametersExample.ParametersExample"/>
        </classes>
    </test>

    <listeners>
        <listener class-name="ListenerTest"/>
        <listener class-name="ListenerSuite"/>
    </listeners>
</suite>
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Co
objc[6105]: Class JavaLaunchHelper is implemented in
  (0x10d56e4c0) and /Library/Java/JavaVirtualMachines/
  will be used. Which one is undefined.
null Class

===============================================
Example
Total tests run: 1, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

PARAMETERS – EXAMPLE2 (@OPTIONAL)

```java
public class ParametersExample {

    @Test
    @Parameters({ "FirstParameter", "SecondParameter" })
    public void test(@Optional("HI") String param1, String param2) {
        System.out.println(param1 + " " +  param2);
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.j
objc[6135]: Class JavaLaunchHelper is implemente
  (0x10f21c4c0) and /Library/Java/JavaVirtualMach
  will be used. Which one is undefined.
HI Class

===============================================
Example
Total tests run: 1, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Example" preserve-order="true" allow-return-values="true" verbose="1">
    <!--<parameter name="FirstParameter" value="Hi"/>-->
    <parameter name="SecondParameter" value="Class"/>
    <test name="Example">
        <classes>
            <class name="parametersExample.ParametersExample"/>
        </classes>
    </test>

    <listeners>
        <listener class-name="ListenerTest"/>
        <listener class-name="ListenerSuite"/>
    </listeners>
</suite>
```

TESTNG – DATAPROVIDERS

DATAPROVIDERS

- What inconvenient can you see with the parameters?
- testng.xml is not sufficient.
- Complex parameters (Example: API).
- Method that returns an array of array of objects.
- @DataProvider.
- Specify in the @Test.

DATAPROVIDER – EXAMPLE (STATIC STRUCTURE)

test 〉 src 〉 test 〉 java 〉 dataProvidersExamples 〉 Example

Project

- test ~/Library/Mobile Documents/com~apple
  - .idea
  - src
    - main
      - java
        - ListenerSuite
        - ListenerTest
      - resources
    - test
      - java
        - dataProvidersExamples
        - parametersExample
        - init
        - Testing
  - target
  - pom.xml
  - test.iml
  - testng.xml
- External Libraries

test | Testing.java | Example.java | testng.xml

```java
package dataProvidersExamples;

import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class Example {

    @DataProvider(name = "ExampleDP")
    public static Object[][] ExampleDP() {
        return new Object[][] { { "1", "1" }, { "2", "2" }};
    }

    @Test(dataProvider = "ExampleDP")
    public void exampleDP(String firstParameter, String secondParameters){
        System.out.println(firstParameter + " " + secondParameters);
    }
}
```

Run: Example.exampleDP

Tests passed: 2 of 2 tests – 6 ms

Default Suite	6 ms
test	6 ms
Example	6 ms
exampleDP[1, 1]	6 ms
exampleDP[2, 2] (1)	0 ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
objc[6157]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtu
  (0x1023144c0) and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/H
  will be used. Which one is undefined.
1 1
2 2

===============================================
Default Suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

DATAPROVIDER – EXAMPLE (DYNAMIC STRUCTURE)

test 〉 src 〉 test 〉 java 〉 dataProvidersExamples 〉 ExampleDynamicStructure

Project

- test ~/Library/Mobile Documents/com~apple
  - .idea
  - src
    - main
      - java
        - ListenerSuite
        - ListenerTest
      - resources
    - test
      - java
        - dataProvidersExamples
          - ExampleDynamicStructure
          - ExampleStaticStructure
        - parametersExample
        - init
        - Testing
  - target
  - pom.xml
  - test.iml

test | Testing.java | ExampleStaticStructure.java | ExampleDynamicStructure.java

```
1   package dataProvidersExamples;
2
3   import ...
10
11  public class ExampleDynamicStructure {
12
13      @DataProvider(name = "ExampleDP")
14      public Iterator<Object[]> locales() {
15          List<Object[]> data = new ArrayList<Object[]>();
16          for (int i = 0; i < 15; i++){
17              data.add(new Integer[]{i});
18          }
19          return data.iterator();
20      }
21
22      @Test(dataProvider = "ExampleDP")
23      public void exampleDP(int number){
24          if(number <= 10){
25              System.out.println(number );
26          }else{
27              Assert.fail( s: number + " is greater than 10");
28          }
29
```

ExampleDynamicStructure 〉 exampleDP()

Run: ExampleDynamicStructure.exampleDP

Tests failed: 4, passed: 11 of 15 tests – 9 ms

Default Suite	9 ms
test	9 ms
ExampleDynamicStructure	9 ms
exampleDP[0]	4 ms
exampleDP[1] (1)	0 ms
exampleDP[2] (2)	0 ms
exampleDP[3] (3)	1 ms
exampleDP[4] (4)	0 ms
exampleDP[5] (5)	1 ms
exampleDP[6] (6)	0 ms
exampleDP[7] (7)	0 ms
exampleDP[8] (8)	0 ms
exampleDP[9] (9)	0 ms
exampleDP[10] (10)	1 ms
exampleDP[11] (11)	2 ms
exampleDP[12] (12)	0 ms
exampleDP[13] (13)	0 ms
exampleDP[14] (14)	0 ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
objc[6189]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVir
  (0x10bd194c0) and /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents
  will be used. Which one is undefined.
0
1
2
3
4
5
6
7
8
9
10

java.lang.AssertionError: 11 is greater than 10

    at org.testng.Assert.fail(Assert.java:96)
    at dataProvidersExamples.ExampleDynamicStructure.exampleDP(ExampleDynamicSt
    at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)
    at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:84) <4 internal
    at org.testng.IDEARemoteTestNG.run(IDEARemoteTestNG.java:73)
    at org.testng.RemoteTestNGStarter.main(RemoteTestNGStarter.java:123)
```

EXAMPLE

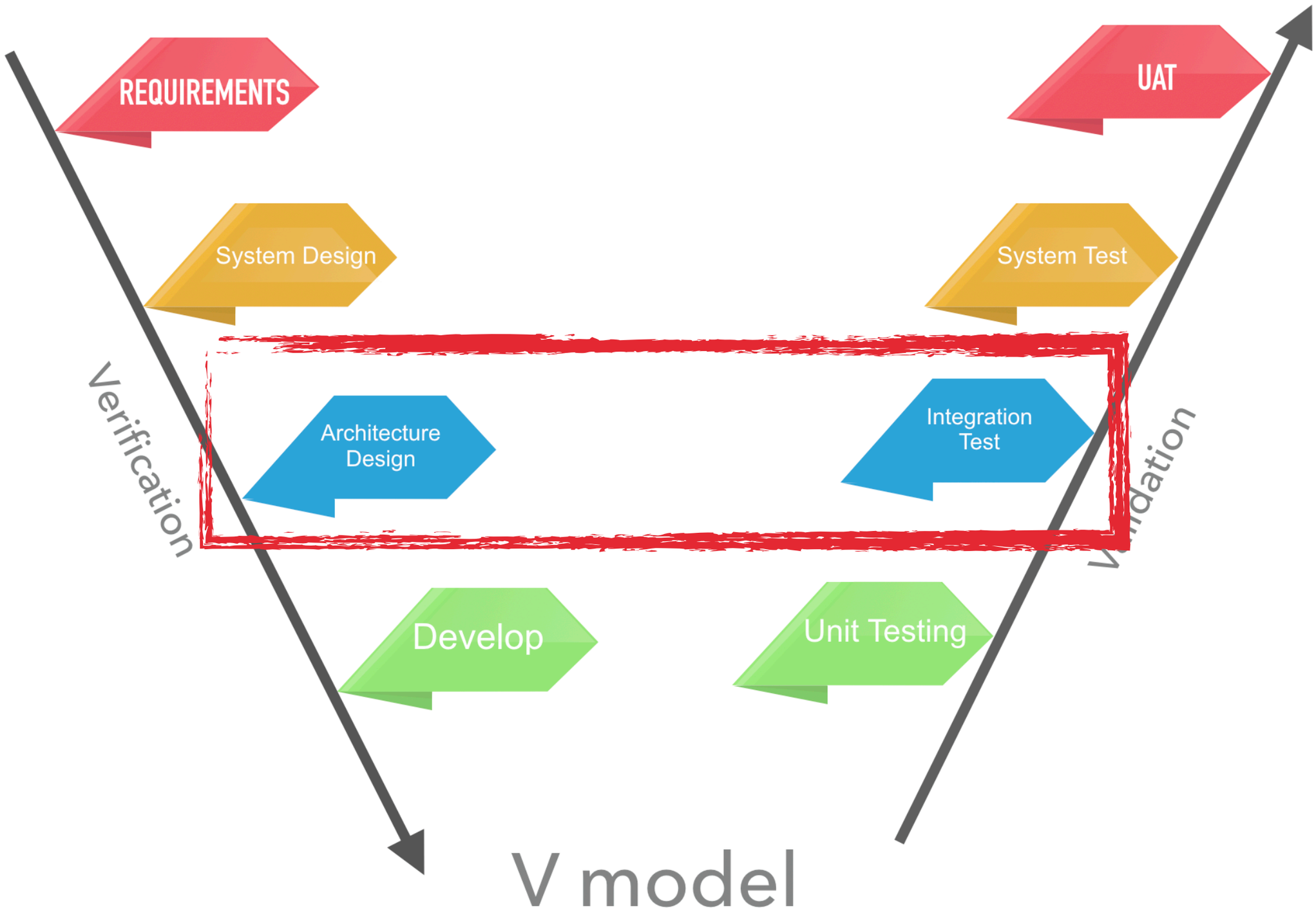
- Calculator
  - +
  - -
  - *
  - /

LET'S
CODE
.IO

THANK YOU.

V MODEL – INTEGRATION TESTING

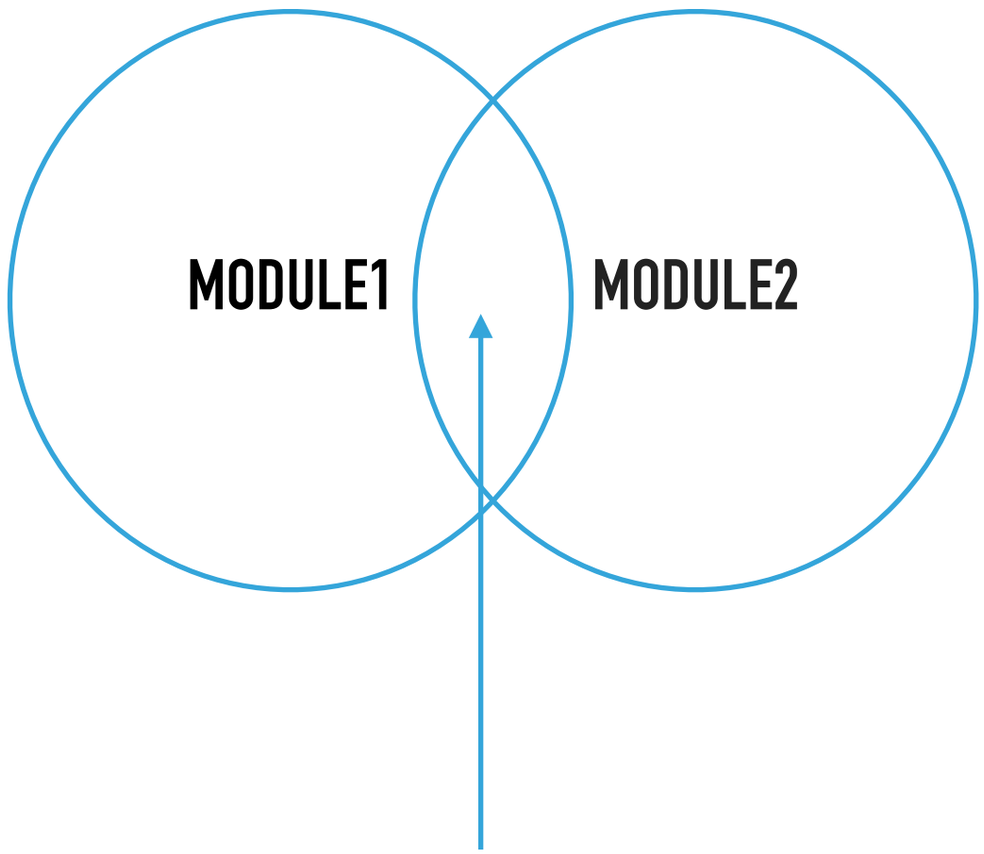
V MODEL


REQUIREMENTS
UAT
System Design
System Test
Verification
Architecture Design
Integration Test
Validation
Develop
Unit Testing
V model

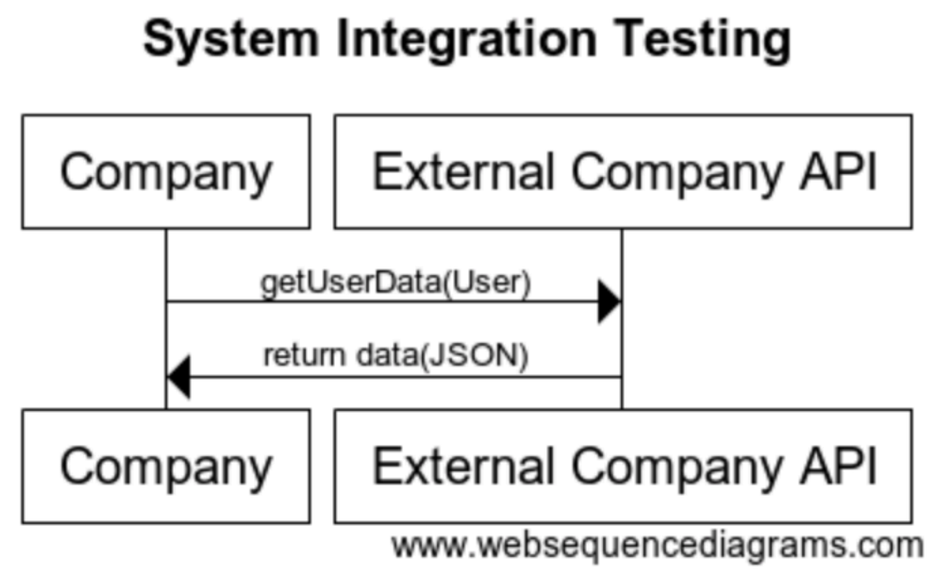
INTEGRATION TESTING

- After Unit Testing occurs.
- Small unit are combined.
- Two levels:
  - Component Integration Testing.
  - System Integration Testing.
- Testers/Developers.

EXAMPLE



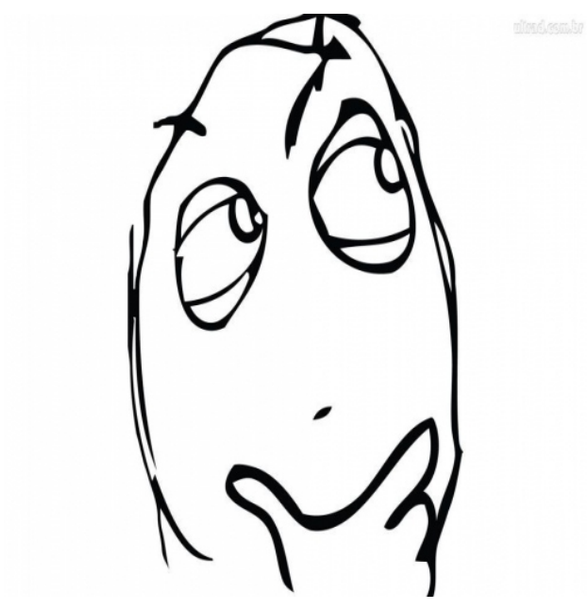
Component Integration Testing



STRATEGIES– INTEGRATION TESTING

STRATEGIES- INTEGRATION TESTING

- Big Bang
- Incremental
  - Top Down
  - Botton Up



BIG BANG APPROACH

- All modules finished.
- Time.
- Difficult to find the cause of the problems.

INCREMENTAL APPROACH – TOP DOWN

- Not wait until all modules are finished.
- Driver
  - Simulate the data that should be passed to the other module.

INCREMENTAL APPROACH – BOTTOM UP

- Not wait until all modules are finished.
- Stub
  - Simulate the data that should be returned to the other module.

INTEGRATION TESTING – TOOLS

EXAMPLES

Let try it…

THANK YOU.