# Testing

## Automatización de Pruebas

# ¿Qué son las pruebas?

La prueba de software es una investigación que permite obtener información acerca de la calidad del producto.

# Principios

- Las pruebas reducen riesgos
- Automatizar las pruebas es costoso
- Asumir que los errores son parte del proceso.
- La etapa de pruebas es una inversión.

# Tipos de Prueba

Pruebas Unitarias:

- Automatizables
- Completas
- Repetibles o Reutilizables
- Independientes

Pruebas E2E

# Tipos de Pruebas

Pruebas de Integración

Probar todos los elementos unitarios que forman parte de un proceso

# Tipos Pruebas

Pruebas de Sistema

- Generalmente son de caja negra.
- Analizan defectos globales o de comportamiento (seguridad-rendimiento)

# Tipos de Prueba

Pruebas de validación

- Comprueban que el software producido cumpla con las especificaciones
- Casos de Prueba

Pruebas exploratorias

# Tipos de Prueba

Pruebas de Rendimiento

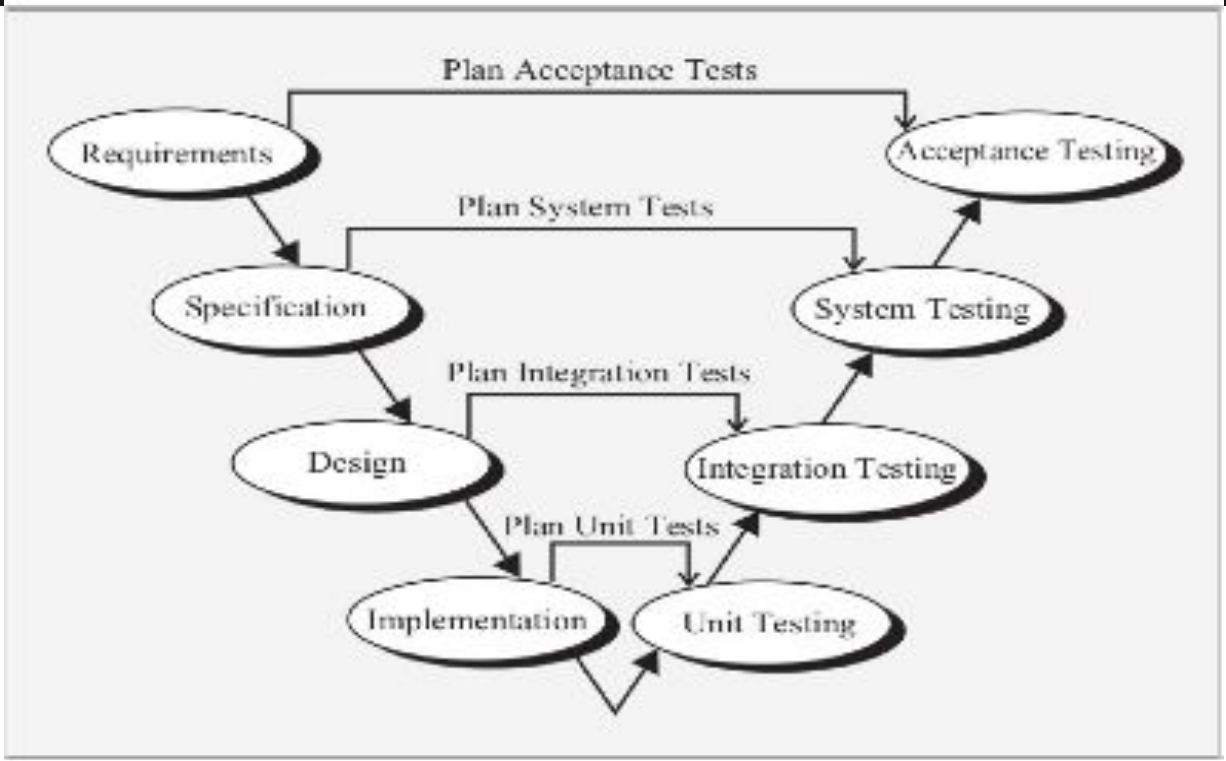
- Pruebas de carga
- Pruebas de estrés
- Pruebas de estabilidad (soak testing)
- Pruebas de picos (spike testing)

# Tipos de Pruebas

Pruebas de aceptación

- Pruebas con las que el cliente aceptará el proyecto

# Modelo desarrollo en V

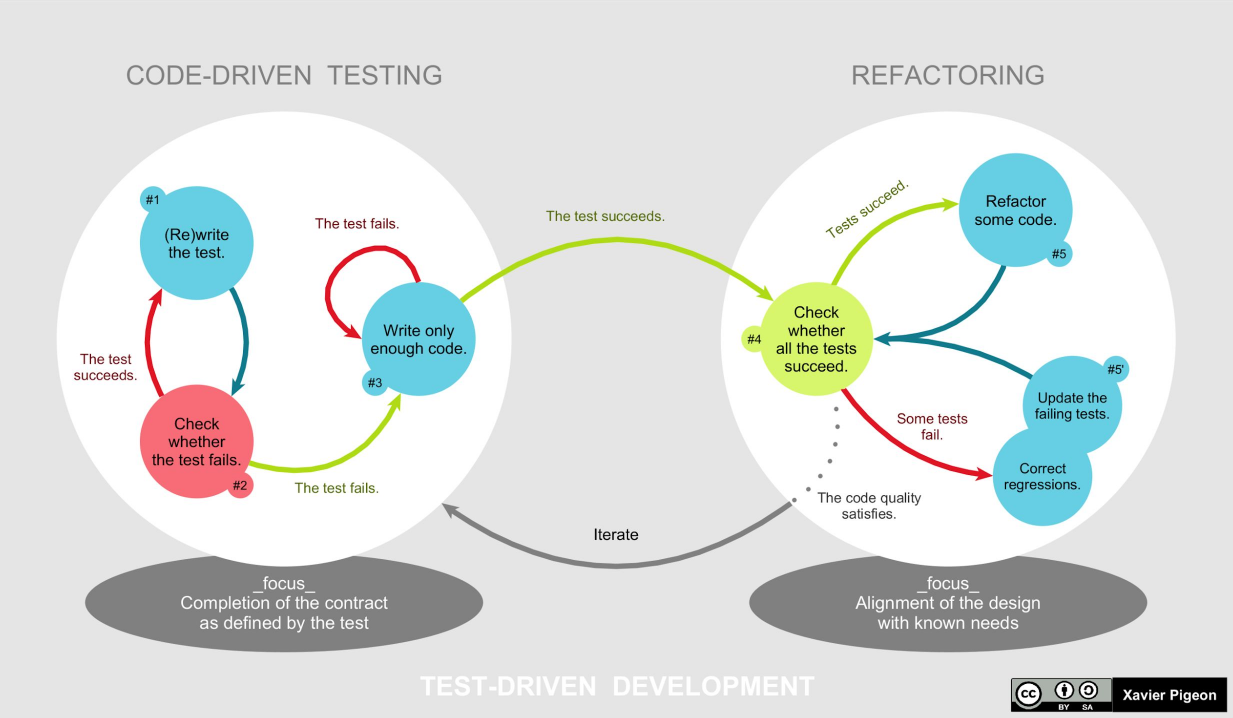


# TDD o Test-Driven Development

## Desarrollo dirigido por tests

Es una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito

# TDD o Test-Driven Development



# Automatización de Pruebas

Beneficios:

- Ejecutar pruebas de regresión
- Ejecutar más casos y con mayor frecuencia
- Ejecutar pruebas difíciles de realizar en forma manual
- Lanzar más rápidamente nuevas versiones

# Automatización Pruebas

Problemas:

- Expectativas poco realistas
- Falsa seguridad
- Costo Mantenimiento
- Depender de las herramientas de prueba

# TestNG

TestNG es un marco de prueba inspirado en JUnit y NUnit, pero presenta algunas funcionalidades nuevas que lo hacen más potente y fácil de usar.

# TestNG - Ventajas

- Puede producir informes HTML de ejecución
- Soporta anotaciones
- Los casos de prueba se pueden agrupar y priorizar más fácilmente
- Permite pruebas en paralelo
- Genera registros
- Permite parametrizar los datos

# TestNG - Pasos

**Paso 1**: Escribir la lógica de negocio de la prueba

**Paso 2**: Insertar las anotaciones de TestNG en el código

**Paso 3**: Agregar la información sobre su prueba (por ejemplo, los nombres de clase, nombres de métodos, nombres de grupos, etc.) en un archivo testng.xml

**Paso 4**: Ejecutar TestNG

# TestNG - Notaciones

**@BeforeSuite**

**@AfterSuite**

**@BeforeTest**

**@AfterTest**

**@BeforeGroups**

**@AfterGroups**

**@BeforeClass**

**@AfterClass**

**@BeforeMethod**

**@AfterMethod**

**@Test**

# Ejemplo de test

```
public class Anotaciones {

   @Test
   public void testCase1() {
      System.out.println("This is the Test Case 1");
   }
   @Test
   public void testCase2() {
      System.out.println("This is the Test Case 2");
   }
   @BeforeMethod
   public void beforeMethod() {
      System.out.println("This will execute before every Method");
   }
   @AfterMethod
   public void afterMethod() {
      System.out.println("This will execute after every Method");
   }
}
```

# Grupos de test

```java
public class Grupos {
   @Test (groups = { "Car" })
   public void Car1() {
      System.out.println("Batch Car - Test car 1");
   }
   @Test (groups = { "Scooter" })
   public void Scooter1() {
      System.out.println("Batch Scooter - Test scooter 1");
   }
   @Test (groups = { "Scooter" })
   public void Scooter2() {
      System.out.println("Batch Scooter - Test scooter 2");
   }
   @Test (groups = { "Car", "Sedan Car" })
   public void Sedan1() {
      System.out.println("Batch Sedan Car - Test sedan 1");
   }
}
```

# Grupos de test  config xml

```
<suite name="Suite">
  <test name="Practice Grouping">
    <groups>
      <run>
        <include name="Car" />
        <!--include name="Scooter" /-->
      </run>
    </groups>
    <classes>
      <class name="testNG.Grupos" />
    </classes>
  </test>
</suite>
```

# Resultados del test

| Método assertxxx() | Qué comprueba |
| --- | --- |
| assertTrue(expresión) | comprueba que expresión evalúe a true |
| assertFalse(expresión) | comprueba que expresión evalúe a false |
| assertEquals(esperado,real) | comprueba que esperado sea igual a real |
| assertNull(objeto) | comprueba que objeto sea null |
| assertNotNull(objeto) | comprueba que objeto no sea null |
| assertSame(objeto_esperado,objeto_real) | comprueba que objeto_esperado y objeto_real sean el mismo objeto |
| assertNotSame(objeto_esperado,objeto_real) | comprueba que objeto_esperado no sea el mismo objeto que objeto_real |
| fail() | hace que el test termine con fallo |

# Otro ejemplo

```
public void testIndiceNoValido()
{
  String s = "mensaje";
  try {

    char c = s.charAt(-5);
    fail("Deberia haber lanzado una excepcion");

  } catch (IndexOutOfBoundsException e) {
    // si sale por aqui es que la prueba ha ido bien
  }
}
```

# Proyecto

pom.xml
Agregar dependencia

```
<dependencies>
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>6.14.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Ejecutar el test

mvn clean test

# Ejercicio

Utilizando TDD crear un método que devuelva los n caracteres del principio y del final de una frase.