Introducción a Unit testing / Prueba de componentes

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 Generalidades

“

El objetivo principal es aislar cada **unidad** del sistema para identificar, analizar y corregir los defectos.

”

Generalidades

La prueba de componente, a menudo, se realiza de forma **aislada** del resto del sistema, dependiendo del modelo de ciclo de vida de desarrollo de software y del sistema, lo que puede requerir objetos simulados, virtualización de servicios, arneses, stubs y controladores.

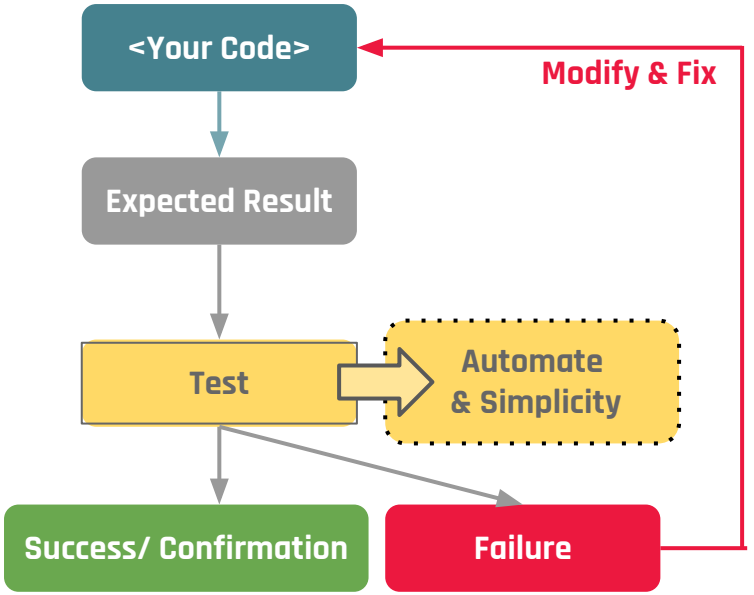
Este tipo de pruebas puede cubrir:

1. La **funcionalidad**: por ejemplo, la exactitud de los cálculos.
2. Las características **no funcionales**: por ejemplo, la búsqueda de fugas de memoria
3. Las propiedades **estructurales**: por ejemplo, pruebas de decisión.

Proceso

En general, cuando no se sigue un enfoque TDD, el proceso es el siguiente:

1. Se crea el código del software.
2. Se definen los resultados esperados.
3. Se ejecuta el test.
    - **a)** Si el test pasa, se confirma el resultado esperado.
    - **b)** Si el test falla, se modifica el código para solucionar el defecto encontrado

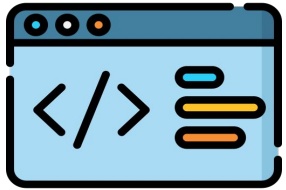


Lo ideal es automatizar los test para poder simplificar el proceso de prueba.

Ventajas

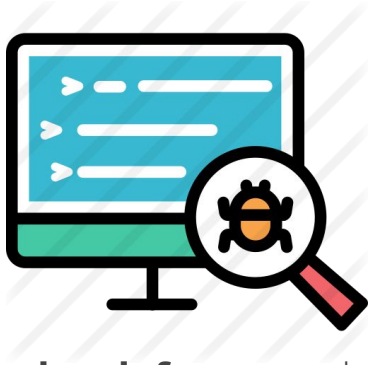
Estas son algunas de las ventajas de realizar pruebas de componente o unit test dentro de un proyecto:

Reduce el costo de las pruebas, ya que los defectos se capturan en una fase temprana.



Mejora el diseño y código del software debido a que permite una mejor refactorización del mismo.

Ventajas



Reduce los defectos en las funciones recientemente desarrolladas o reduce los errores al cambiar la funcionalidad existente.

En modelos de desarrollo incrementales e iterativos donde los cambios de código son continuos, la **prueba de regresión de componente automatizada** juega un papel clave en la construcción de la **confianza** en que los cambios no han dañado a los componentes existentes.

Las pruebas unitarias **inapropiadas** harán que los defectos se propaguen hacia pruebas de nivel superior y esto conducirá a un **alto costo de reparación de defectos** durante las pruebas del sistema, las pruebas de integración e incluso las pruebas de aceptación de usuario. Si se realizan las pruebas unitarias **adecuadas** en el desarrollo inicial, al final se **ahorra esfuerzo, tiempo y dinero**.

2 | Frameworks

Frameworks

Las pruebas unitarias pueden ser de dos tipos:

Manuales

Se puede emplear un documento instructivo paso a paso.

Automatizadas

Se necesita de un framework automatizado para escribir los scripts de prueba.

¿Qué se necesita para automatizar los unit test?

TEST RUNNER

Es una herramienta que ejecuta los test y muestra los resultados en forma de reporte.

Por eJemplo: Mocha (https://mochajs.org/)

ASSERTION LIBRARY

Es una herramienta que se utiliza para validar la lógica de prueba, las condiciones y resultados esperados.

Por ejemplo: Chai (https://www.chaijs.com/guide/)

JEST es un framework que incluye tanto el test runner como la assertion library

Frameworks más utilizados

Junit: herramienta de prueba de uso gratuito que se utiliza para el lenguaje de programación Java. Proporciona afirmaciones para identificar el método de prueba. Esta herramienta prueba los datos primero y luego los inserta en el fragmento de código.

NUnit: es un marco de trabajo de pruebas unitarias ampliamente utilizado para todos los lenguajes .net. Es una herramienta de código abierto y admite pruebas basadas en datos que pueden ejecutarse en paralelo.

JMockit: es una herramienta de prueba unitaria de código abierto. Es una herramienta de cobertura de código con métricas de sentencia y decisión. Permite hacer mocks de API con sintaxis de grabación y verificación. Esta herramienta ofrece cobertura de sentencia, cobertura de decisión y cobertura de datos.

Frameworks más utilizados

EMMA : es un conjunto de herramientas de código abierto para analizar y reportar código escrito en lenguaje Java. Emma admite tipos de cobertura como método, sentencia, bloque básico. Está basado en Java, por lo que no tiene dependencias de bibliotecas externas y puede acceder al código fuente.

PHPUnit: es una herramienta de prueba unitaria para programadores PHP. Toma pequeñas porciones de código que se denominan unidades y prueba cada una de ellas por separado. La herramienta también permite a los desarrolladores usar métodos de confirmación predefinidos para afirmar que un sistema se comporta de cierta manera.

Framework para JavaScript

Para JavaScript vamos utilizar el framework JEST. Si queremos configurarlo, debemos instalar:

1. NodeJS (https://nodejs.org/)
2. Un IDE (el recomendado es Visual Studio Code - https://code.visualstudio.com/)
3. JEST (https://jestjs.io/)


DigitalHouse>
Coding School