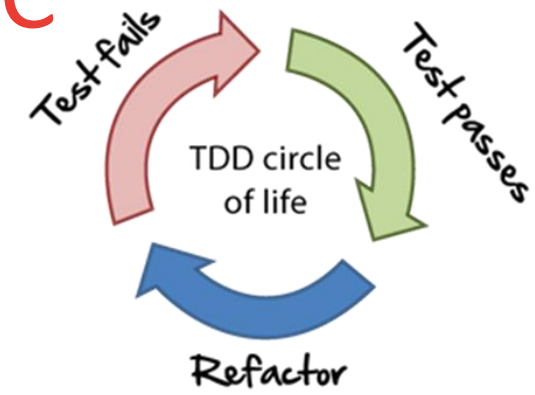
Florida

1º DAM/DAW EDE

U6. Diseño de pruebas de software

AP6 - Pruebas unitarias - TDD - VSC



Pascual Martínez

Ejercicio 1

Descripción:

Sigue los pasos explicados en los enunciados para generar la documentación solicitada.

Objetivo:

Entender TDD y las técnicas que propone, como práctica de Ingeniería del Software.

Bibliografía:

Recursos didácticos de Florida Oberta.

Actividad a realizar:

Se van a plantear una serie de pasos a seguir. Cada uno de los pasos, además de realizarlos, deben quedar correctamente documentados, mediante capturas de pantalla, redacción de texto explicativo …

Pasos a seguir:

1 - Leer y analizar cada paso solicitado.

2 - Ejecutar cada paso hasta completarlos todos.

3 - Generar un **un documento PDF**, debidamente identificado, que incluya cada enunciado con la respuesta correspondiente.

4 - Entregar a través de Florida Oberta en los plazos indicados.

- **Paso 1. Diseña y desarrolla la prueba:**
  - Debemos llevar a cabo un proyecto de ingeniería de software, utilizando TDD como práctica para desarrollar el código.
  - El **proyecto** consta de un **único requerimiento**: “dados 2 números naturales de 0 a 9, el sistema debe poder sumarlos y devolver el resultado”.
  - Dado el requerimiento, diseña una función en JavaScript mediante Visual Studio Code llamada “**pruebaUnitaria()**” que resuelva el caso de prueba para la siguiente **prueba unitaria**: “se entenderá que el programa supera la prueba unitaria y por lo tanto es correcto, si tomando como valores iniciales 3 y 5 devuelve 8 como resultado”.

- **Paso 2. Diseña y desarrolla el código:**
  - Diseña una función llamada "**suma()**" que supere la prueba unitaria definida.
- **Paso 3. Mejora la prueba:**
  - Demuestra y argumenta que la prueba unitaria aplicada no es exhaustiva ni precisa, por lo que, a pesar de que el resultado sea satisfactorio, no garantiza que la funcionalidad sea la correcta. Diseña una nueva prueba unitaria que permita incrementar el nivel de garantía de cumplimiento del requerimiento.

- **Paso 3. Mejora la prueba. Recomendación**:
  - En este caso, se propone inicializar dos arrays con números aleatorios del 0 al 9, que harán las veces de sumandos, y un tercer array de totales, con los valores sumados de los dos arrays anteriores.
  - Se llamará de forma iterativa a la función "**suma()**", pasándole como parámetros los valores de los arrays sumandos. Se debe comprobar que el resultado de la función "**suma()**" debe coincidir con el valor del array de totales sumados.
  - Se dará la prueba como satisfactoria o superada, si todas las llamadas a la función "**suma()**" son satisfactorias.

- **Conclusiones:**
  - Realizando los pasos tal y como se propone en la actividad, se demuestra cómo **se puede diseñar o desarrollar la prueba antes que el código**.
  - Es importante aprender a **acotar la precisión de una prueba**, de ese modo podremos conocer también y de antemano, la fiabilidad del resultado de nuestro código.
  - En nuestro caso, queda demostrado que probar una función de suma, con sólo una llamada y sumando sólo 2 números, no tenía un **nivel de precisión adecuado**. En el momento probemos la función de forma iterativa y con números aleatorios, incrementamos notablemente el nivel de fiabilidad de la prueba.