Test Driven Development

INTRODUCCIÓN AL TEST DRIVEN DEVELOPMENT

ÍNDICE

DESCRIPCIÓN

- Es una metodología de desarrollo de software que se basa en la creación de pruebas automatizadas antes de escribir el código de producción.
- En este enfoque, el ciclo de desarrollo se inicia con la definición de pruebas que describen el comportamiento esperado del software. Luego, se procede a escribir el código mínimo necesario para que esas pruebas pasen con éxito, y finalmente, se realiza la refactorización del código para mejorar su estructura y calidad sin alterar su comportamiento.

CARACTERISTICAS TDD

1. El TDD es un ciclo de desarrollo que consta de diferentes pasos:

- -Paso Rojo: Se escribe una prueba automatizada que falle por que el código de producción no existe o no cumple los requisitos.
- -Paso Verde: Se escribe un código de producción mínimo necesario para que la prueba automatizada pase satisfactoriamente.
- -Paso Refactor: Se mejora y refactoriza el código para que sea más limpio, eficiente y cumpla mejor con las características.

2. Pruebas automatizadas: Las pruebas escritas en código se ejecutan automáticamente al modificar el código de producción, esto garantiza que las pruebas sean repetibles y confiables.

3. Enfoque de los requisitos: Antes del código de producción se escriben pruebas con los requisitos del software, esto ayuda a comprender lo que se espera del código.

4. Incremental y evolutivo: Se construye el software paso a paso agregando funciones y pruebas a medida que avanza el desarrollo, esto permite detectar problemas antes y garantizar que el código funcione correctamente.

5. Calidad del código: Fomenta la alta calidad de los códigos ya que las pruebas ayudan a identificar problemas y garantizan que el código cumpla los requisitos.
Además mejora tanto estructura como la mantenibilidad del código.

6. Retroalimentación inmediata: Como las pruebas se ejecutan automáticamente es fácil detectar los errores en el código y permite corregirlos antes y más fácil.

7. Documentación viva: Las pruebas escritas son documentación viva del software donde describe el comportamiento del código, esto facilita la comprensión incluso para desarrolladores que no participaron al inicio.

8. Facilita la colaboración: Promueve la colaboración ya que las pruebas claras y los requisitos definidos facilitan la comunicación y trabajo en equipo.

VENTAJAS

- **Calidad del código:** al requerir pruebas automatizadas.
- **Diseño mejorado:** Mejor diseño de software.
- **Retroalimentación inmediata:** Cuando una prueba falla se puede solucionar al momento antes de seguir.
- **Reducción de errores:** se pueden identificar y corregir errores antes de que lleguen a la etapa de producción
- **Confianza en los cambios:** Al cambiar el código, las pruebas permiten verificar que las funcionalidades existentes no se han visto afectadas.

DESVENTAJAS

- **Costo inicial de aprendizaje:** tiempo para aprender y adaptarse
- **Mayor tiempo de desarrollo inicial:** El proceso de escribir pruebas antes de escribir el código de producción puede llevar más tiempo
- **Complejidad en el mantenimiento de las pruebas:** Cuando el código evoluciona, las pruebas unitarias también .
- **Posible exceso de pruebas:** Los equipos pueden consumir tiempo y recursos innecesarios a realizar tantas pruebas.
- **No es adecuado para todos los proyectos:** Especialmente aquellos en los que los requisitos son ambiguos o cambian con frecuencia.

FASES (I)

- **Prueba:** Caso de prueba sin hacer código, que se utiliza para ayudar al equipo de ingeniería a entender los "requisitos" antes de empezar a escribir código para el programa real.
- **Ejecutar las pruebas:** Si encuentra fallos como debería, significa que el entorno y el caso de prueba funciona correctamente.
- **Escribir código:** Empieza a escribir el código y debe de asegurarse el programador de que pase la prueba de forma correcta.

FASES (II)

- **Pruebas automatizadas:** Una vez que hacemos diferentes pruebas y escribimos correctamente el código, empezamos con las pruebas automatizadas. Si el resultado de cada prueba es exitoso, significa que cumple con los requisitos que nos ha dado previamente el cliente.
- **Refactorizar el código:** Proceso basado en pruebas, que ayuda a eliminar posible duplicación de códigos de producción y de pruebas (sin dañar ninguna funcionalidad).
- **Repetir:** Se va repitiendo constantemente el ciclo completo basado en pruebas, con una nueva.

ROLES

- ROL DEL PROGRAMADOR
- ROL DEL TESTER
- ROL DEL CLIENTE O USUARIO

ROL DEL PROGRAMADOR

Es el responsable de seguir todo el proceso de TDD, que consta de los siguientes pasos:

-Test: Escribe una prueba automatizada que define el comportamiento deseado del código que aún no ha sido implementado. Esta prueba debe de fallar, ya que el código de producción correspondiente todavía no existe.

-Escribir código de producción: Aquí el objetivo del programador es intentar pasar la prueba sin necesidad de escribir el código perfecto o completo.

-Refactorizar el código: Una vez pasada la prueba, el programador puede refactorizar el código para mejorarlo, sin cambiar su comportamiento. Este ciclo se repite , escribiendo pruebas adicionales y código de producción para construir gradualmente el software.

ROL DEL TESTER

El rol del tester se fusiona con el del programador.

El programador mismo crea las pruebas automatizadas y verifica que el código de producción cumple con los requisitos definidos por las pruebas.

Esto asegura que el código funcione correctamente y no se introduzcan errores.

ROL DEL CLIENTE O USUARIO

El cliente o usuario final del software desempeña un papel importante al definir los requisitos y las pruebas de aceptación que servirán como base para las pruebas unitarias.

Este rol garantiza que el software cumpla con las necesidades y expectativas del negocio o los usuarios.

Gracias por vuestra atención

¿Alguna pregunta?