La técnica de programación TDD

Test Driven Development (TDD) es una práctica de ingeniería de software que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización de código (Refactoring). Con esta práctica se consigue entre otras cosas: un código más robusto, más seguro, más mantenible y una mayor rapidez en el desarrollo.

Carácteristicas de TDD

Como TDD es una técnica de desarrollo de software y vimos que el desarrollo de software es un proceso de aprendizaje, podemos concluir que TDD también se puede definir como una técnica de aprendizaje.

Algunas caracteristicas de TDD son:

- Iterativa e incremental.
- Constructivista.
- Está basada en feedback inmediato.
- Recuerda lo aprendido
- Permite asegurarnos de no haber desaprendido.
- Incluye análisis, diseño, programación y testing.

¿Cómo se hace TDD?

TDD consta de 3 simples pasos que se van repitiendo hasta finalizar el desarrollo:

1. Escribir un test
    a. Debe ser el más sencillo que se nos ocurra
    b. Debe fallar al correrlo.
2. Correr todos los tests

    a. Si hay errores, arreglar el modelo e ir al paso 2.
3. Reflexiono. ¿Se puede mejorar el código?
    a. Sí, refactorizar e ir al paso 2.
    b. No. Ir al paso 1.

Al escribir el test estamos definiendo el problema que buscamos resolver. Al arreglar el modelo para que pase el nuevo tests estamos resolviendo el problema (hasta el punto que está definido) y al refactorizar el código estamos haciendo un mejor diseño.

¿Cuando no hago TDD?

- Cuando no tengo feedback inmediato.
- Escribiendo/modificando código antes de escribir un test.
- Escribiendo muchos tests antes de escribir el código.
- Cuando no desarrollo de manera iterativa e incremental.
- Escribiendo la "solución completa" de entrada.
- Haciendo up-front design.
- Escribiendo los tests luego de tener el código, es decir, haciendo testing.

Desarrollo iterativo e incremental

Las reglas de oro del desarrollo iterativo e incremental:

- Escribir el test primero
- El test debe ser el más sencillo posible y debe fallar.
- Hacer la mínima implementación necesaria para que el test pase.
- Refactorizar!

Bad Smells

En esta sección veremos algunas cosas que huelen mal cuando hacemos TDD, y se explicará lo que significa cada situación y algunas sugerencias para resolver el problema.

Situación	Significa que
Un nuevo test funciona de entrada la primera vez que lo corro.	● El caso ya esta testeado, con lo cual hay que borrar el test. ● No estoy haciendo un desarrollo iterativo e incremental. Controlar la ansiedad. Hacer "baby steps"
Tardo mucho para hacer funcionar un test.	● El test era muy complejo, no estoy haciendo desarrollo incremental. Borrar test y hacerlo más simple. ● El modelo es muy complejo. Debo refactorizar.
Tardo mucho en escribir un test porque el setup es muy largo.	● No estoy "reflexionando". Debo pensar cómo simplificar la creación de objetos de prueba.
Tardo mucho en escribir un test porque el act es muy largo.	● El test es muy complejo, no estoy haciendo el test más sencillo. Borrar el test y hacerlo más simple. Particionar el test en varios casos. ● El modelo es muy complejo. Debo refactorizar.
Tardo mucho en escribir un test porque son muchos assertions.	● El test es muy complejo, estoy testeando varios casos juntos. Particionar el test en varios casos. ● No estoy "reflexionando" sobre qué significa cada assertion. Debo refactorizar.
Me lleva mucho tiempo decidir qué test escribir.	● Le estoy dando mucha importancia pensando cual es el más simple. Actuar!!!

	Hay que tener feedback para aprender y mejorar!!! ● El dominio de problema es muy complejo. Aceptarlo. ● Terminé!!!
Los tests tardan mucho en correr.	● El test depende de algo externo que lo hace demorar. Desacoplar esa dependencia.

Conclusiones

Veamos algunas conclusiones sobre la práctica de programación TDD:

- Lo más importante: Feedback inmediato para poder reflexionar.
- Recordar que es iterativo e incremental.
- No preocuparme por los nombres de los tests al principio.
- No preocuparme por la organización de los tests al principio.
- Hacer un test por caso. Y Organizar bien los test.
- El test no es el fin, es un medio.
- Usar la computadora como soporte o medio de aprendizaje (no correr el sistema en nuestra cabeza).
- Un programador no es buen programador si no es un buen tester.
- TDD ayuda a generar diseños menos acoplados.
- TDD no implica buen diseño!
    - Los buenos diseños los hacen los buenos diseñadores.
    - Pensar en objetos para tener buenos diseños.
- TDD nos da confianza para realizar cambios cuando el modelo así lo requiera.