Desarrollo Guiado por Tests

CC51H

Ignacio Ortega C.

Es conocido como TDD, por sus siglas en inglés, Test Driven Development.

Agenda

- ¿Que és?
- Ejemplo Práctico
- Conclusiones

¿Qué es TDD?

- Escribir código limpio que funciona.
- Combinación de *test-first programming* y **refactoring**.
- Objetivo: Especificación. Pensar el diseño antes del código funcional.

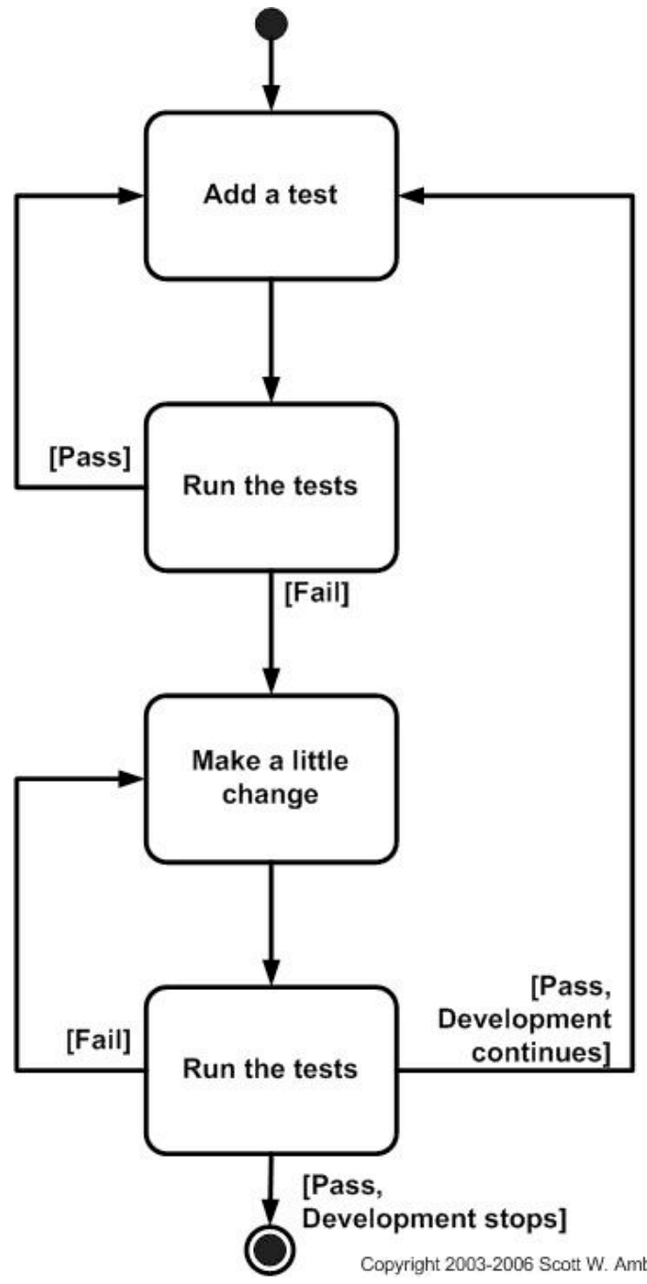
John Reffries lo define como “Clean code that works”.
Pero primero debiesemos entender que es un test significa evaluar código, evaluar que el código funcione de la forma esperada. Se puede decir además que el test es software que hace mejor software.

Test-First programming nos indica que primero debemos escribir un test antes de escribir cualquier código funcional.

Refactoring significa alterar la estructura del interna del software sin modificar su comportamiento externo.

TDD: El Proceso

- Averiguar que hacer. Diseño.
- Escribimos un test para la nueva funcionalidad y la dejamos fallar.
- Código pecaminoso para pasar (¡progreso!).
- Implementación completa, pasando la prueba.



¿Como empezar? Se recomienda escribir un test simple, que se tenga confiaza en que se pueda pasar rápidamente. Poner la pelota en juego. Si es un problema complejo, comezar por el caso más sencillo.

Ejemplo Práctico

- Tarea 1: Clasificador de figuras geométricas.

Figuras:
- 2D: Triángulos y Cuadriláteros
- 3D: tetraedros y pirámides

Criterios:
- 2D :
  - ángulo mínimo
  - razón de aspecto
- 3D:
  - angulo diedro (ad) mínimo
  - la razón entre volumen v de la figura y su lado más largo, elevado a 3: vl = v/l 3.

Formato de Archivos:

- 2D:

```
2D
Triangulo
0 0 1 0 0 1
Cuadrilatero
0 0 2 0 2 2 0 3
```

- 3D

```
3D
Tetraedro
0 0 0 1 0 0 0 1 0 0 0 1
Tetraedro
0 0 0 2 0 0 0 2 0 0 0 3
Piramide
0 0 0 1 0 0 0 1 0 0 0 1 1 1 0
```

TDD: Ventajas

- Pequeños pasos al escribir software
- “The act of writing a unit test is more an act of design than of verification. It is also more an act of documentation than of verification. The act of writing a unit test closes a remarkable number of feedback loops, the least of which is the one pertaining to verification of function”

Es importante destacar, que una de las ventajas es que el software que se construye se hace de forma incremental, y el ciclo expuesto se realiza continuamente, es decir continuamente hacemos diseño y agregamos test de forma de especificar el comportamiento que se desea para luego implementar sólo el código que pase las pruebas. Una vez pasadas debemos pensar en refactorizar, si tenemos código duplicado, factorizarlo, o si el diseño que tenemos se adapta a los nuevos requerimientos.

Conclusiones

- No hay código sin pruebas asociadas
- El código se origina y permanece sólido
- Las pruebas perduran
- Las pruebas son documentación
- Efecto psicológico

Con TDD el código que escribo es funcional pues hay una prueba que lo respalda. El efecto psicológico se consigue cuando escribimos código que no tiene un test que lo respalda y no podemos asegurar que sea funcional.