Desarrollo de Aplicaciones Web Empresariales

Exequiel Fuentes Lettura
exequiel.fuentes@ucn.cl

Información de contacto

- Exequiel Fuentes Lettura
  - Email: exequiel.fuentes@ucn.cl
  - Horario de Atención: Jueves y Viernes, bloque C

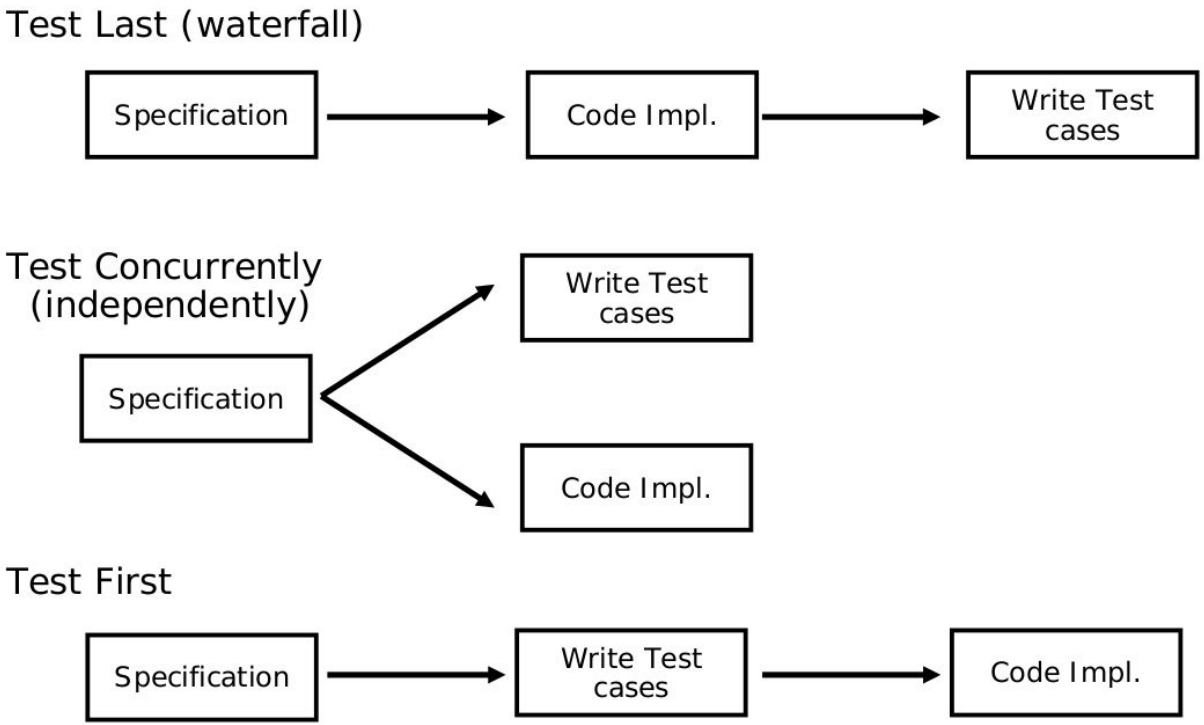
- Departamento de Ingeniería de Sistemas y Computación
  - Oficina: Y1 - 329
  - http://www.disc.ucn.cl

Introducción

- Test-driven Development es una práctica de programación que indica a los desarrolladores escribir nuevo código sólo si una prueba automatizada ha fallado.

- La meta de TDD es: código limpio que funcione.

Posibles procesos de pruebas



Qué es TDD

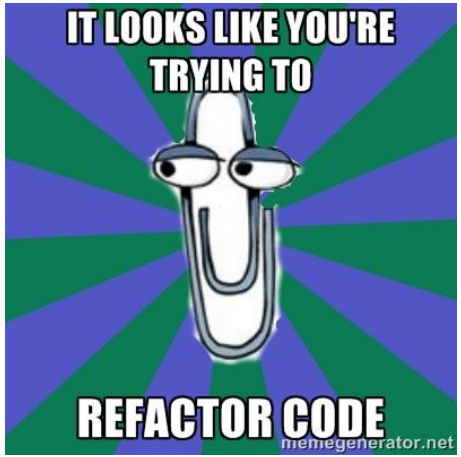
- TDD es una técnica donde se escribe una prueba antes de escribir la implementación.
- Las pruebas dirigen o dictan el código a ser desarrollado:
  - Las pruebas proporcionan una especificación de lo que una pieza de código realmente hace.
  - Pensar acerca de la prueba es analizar lo que el sistema debería hacer.
  - Algunos argumentan que las pruebas son una parte de la documentación.
- Principalmente, son pruebas de unidad.
- Es posible ejecutar pruebas de regresión automatizadas.

Pruebas automatizadas

- Considerar:
  - Código que no es probado, se asume que no funciona.
  - Código que no es probado por pruebas de regresión puede eventualmente fallar.
  - Si no está automatizado, el trabajo no ha terminado.
- Un framework de pruebas de unidad permite crear pruebas de unidad y de regresión de manera efectiva y eficiente.
  - Java: JUnit, TestNG, Mockito, Arquillian, entre otros.
  - Python: unittest, pyUnit, Nose, py.test, doctest, entre otros.
  - Javascript: Karma, Protractor, Buster.js, Mocha, entre otros.

Refactorización

- Refactorizar es reestructurar o transformar el código sin cambiar su comportamiento.
- Palabras claves: Reestructurar, simplificar y mejorar.
- Peligro: Refactorizar es una excelente forma para ‘quebrar’ código.



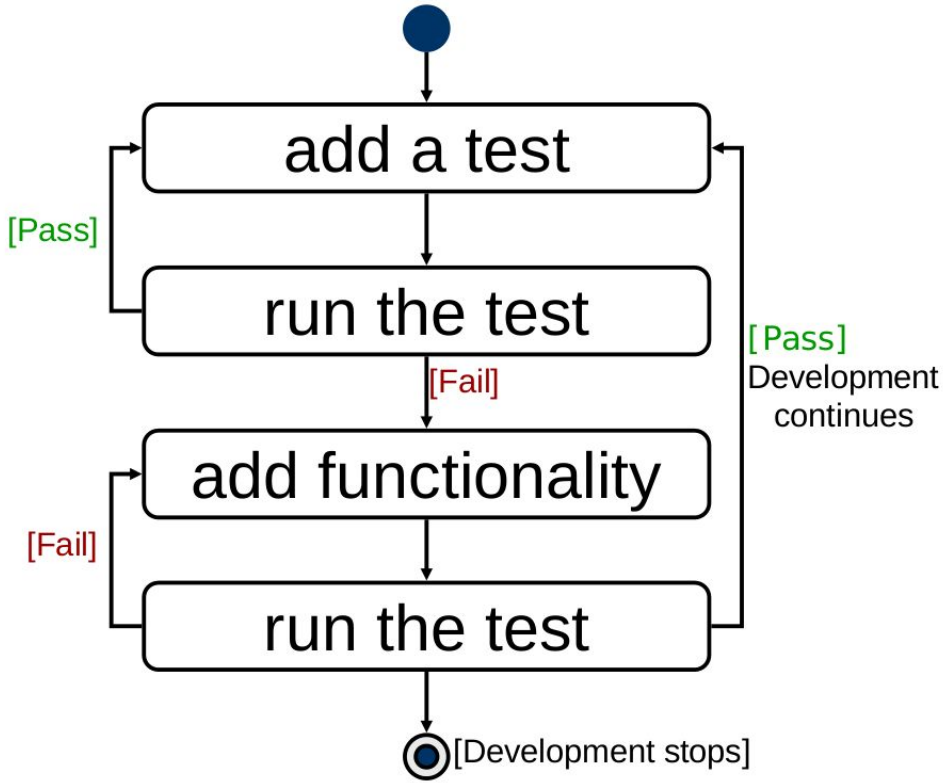
Pruebas de regresión

- Código nuevo y cambios a código viejo puede ‘afectar’ el resto del código base. ‘Afectar’ a veces significa ‘quebrar’.
- Regresión es una recaída a un estado menos perfecto o desarrollado.
- Las pruebas de regresión verifican que el código no tenga una regresión.
- Las pruebas de regresión son requeridas para tener un código estable y mantenible.

Pasos en TDD

1. Escribir una prueba.
2. Compilarla. No debería compilar porque no se ha escrito la implementación del código.
3. Implementar sólo el código para que la prueba compile.
4. Ejecutar la prueba y ver si falla.
5. Implementar sólo el código para que la prueba pase.
6. Ejecutar la prueba y ver si pasa.
7. Refactorizar si es necesario y sola una vez.
8. Repetir.

Pasos en TDD

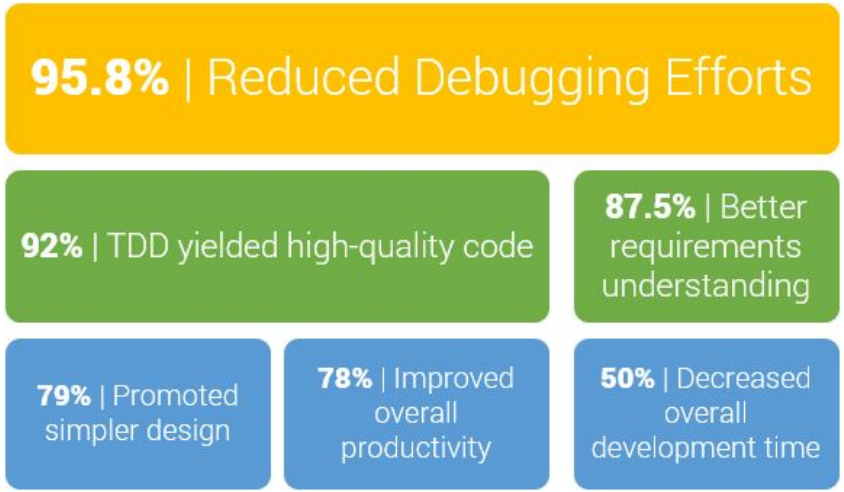

add a test
[Pass]
run the test
[Pass]
Development
continues
[Fail]
add functionality
[Fail]
run the test
[Development stops]

Beneficios

- Eficiencia:
  - Identifica efectos tempranamente.
  - Identifica causas más fácilmente.
- Alto valor para el esfuerzo en las pruebas:
  - Produce sistemas más confiables.
  - Mejora la calidad de las pruebas.
  - Reduce los imprevistos.
  - Código base estable.
- Reduce los defectos de inyección:
  - Pequeños arreglos tienen 40 veces más errores que código nuevo.
  - Entonces, incrementar pruebas de unidad y verificar continuamente.

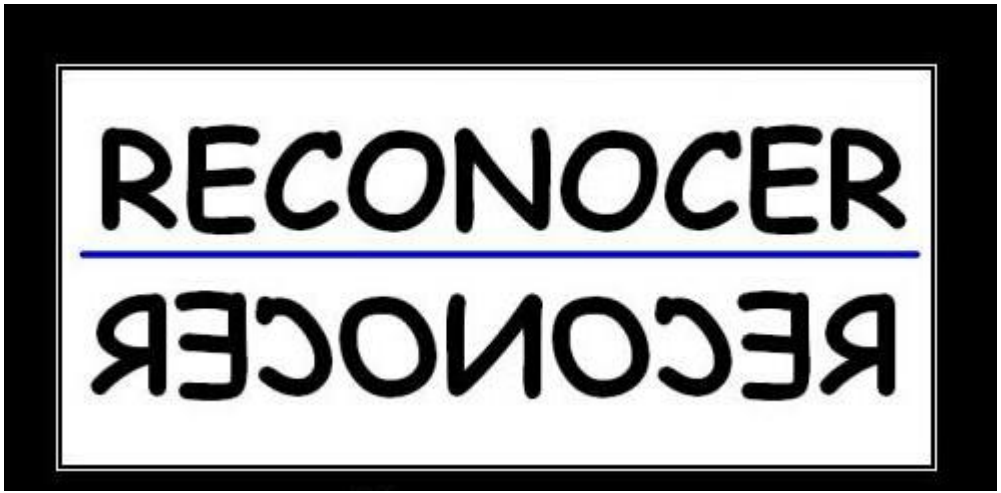
Beneficios

- Mejor vida para el programador:
  - Trabajar en el código sin miedos.
  - Nadie quiere dar soporte a un sistema frágil.
  - Con las pruebas, si el código falla se sabe que falló y quién lo provocó.
- Es fácil agregar cambios:
  - Agregar una funcionalidad.
  - Nuevos requerimientos.
  - Refactorizar.



Ejemplo: Palíndromo

- Palíndromo es una palabra, número o frase que se lee igual hacia adelante que hacia atrás.
- Implementar un algoritmo que verifique que una palabra es palindromo utilizando TDD.
- Ver video adjunto.



Ejercicio: Ingresar clave de usuario

- Una clave de usuario válida debe contener:
  - Debe contener al menos 8 caracteres ASCII.
  - Debe contener al menos un carácter numérico.
  - Debe contener al menos una letra mayúscula.
  - Debe contener al menos una letra minúscula.
  - Debe contener al menos un carácter no-alfanumérico.
  - No puede contener espacios en blanco.
- Debe retornar un valor booleano. True si la clave es correcta y False de lo contrario.
- Implementar la solución en Java utilizando TDD que verifique que la clave ingresada cumple con las exigencias anteriores.
- Subir el código a un repositorio Git. La idea es ver a través de los commits que se entendió el concepto de trabajar con TDD.

¿Preguntas?