TEST DRIVEN DEVELOPMENT

Alexandre Currás Rodríguez, Daniel Duque Puga, David Vila Fernández

Introducción

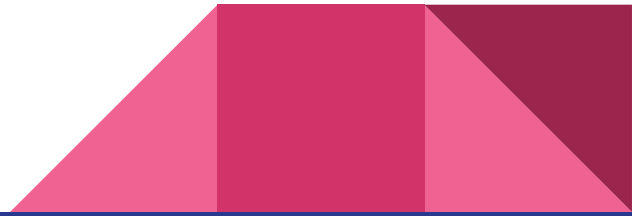
Ciclo de desarrollo

Buenas prácticas

Estilo de desarrollo

Mocks

Conclusion (beneficios y limitaciones)

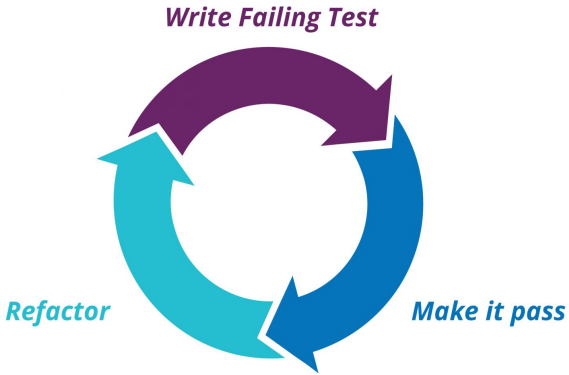


Introducción

- Práctica de IS creada por Kent Beck
- Involucra dos prácticas: escribir pruebas (TFD) y refactorizar.
- Utilización de pruebas unitarias
- Intención: código limpio que funcione
- Traducir requisitos a pruebas

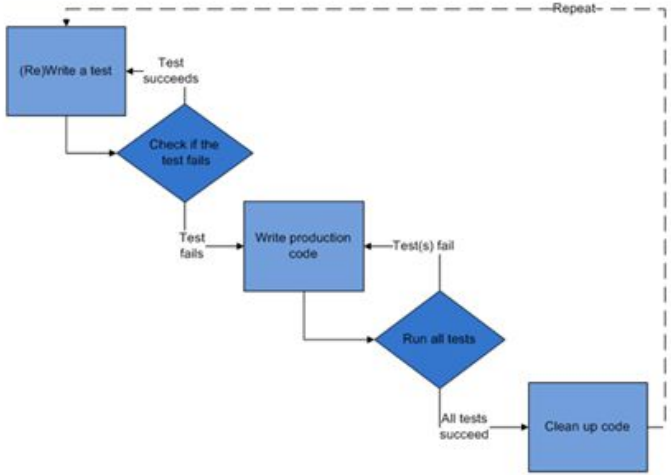
Introducción

- Necesario un sistema flexible
- Es útil si se escriben pruebas relativamente pequeñas



Ciclo de desarrollo

1. Elegir un requisito.
2. Codificar la prueba
3. Verificar que la prueba falla
4. Escribir la implementación
5. Ejecutar las pruebas automatizadas
6. Refactorización
7. Actualización de la lista de requisitos



Fuente de la imagen: Wikipedia

Buenas prácticas

- Repositorio de pruebas para facilitar integración
- Mejor problemas pequeños cada pocas horas
- Centrarse en el resultado del test
- Tratar el código de test igual que el de producción
- Durabilidad, legibilidad y mantenibilidad
- Redudir dependencias entre los test
- Reuniones de equipo sobre los test

Malas prácticas

- Demasiadas dependencias.
- Test con código poco eficiente, ralentiza el proceso
- No refactorizar

Estilo de Desarrollo

Principios:

- SOLID
- “Mantenerlo simple”
- “No lo vas a necesitar”

Se hacen los test antes que las funcionalidades

Se verifica que los test fallan al principio

Mantener las pruebas de unidad pequeñas

Objetos Mock

- Objetos que simulan el comportamiento de objetos reales
- Se usan cuando es imposible o impracticable usar un objeto real
- Implementan la interfaz del objeto real
- Disminuyen la complejidad de los tests
- Reducen el tiempo de preparación de los test

Ventajas

- Menor tiempo de debug
- Menos errores
- Menos redundancia
- Código fiable
- Diseño orientado a necesidad
- Software modular
- Software de mayor calidad en menos tiempo

Desventajas

- Automatización de pruebas
- Interfaces de usuario
- Falsa seguridad
- Pérdida de la visión general
- Pronunciada curva de aprendizaje

TDD

ALL CODE IS GUILTY

UNTIL PROVEN INNOCENT