Behaviour Driven Development

Electrónica IV

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

https://facetvirtual.facet.unt.edu.ar/course/view.php?id=165

¿Qué es TDD?

- Es una técnica de programación pensada para poder construir software en forma incremental
- No es una tecnica de Testing
- El concepto básico es no escribir absolutamente nada de código sin escribir antes una prueba de aceptación
- Estas pruebas de aceptación son los Test
- Es la sistematización de la postergación: se demora la escritura del código hasta que las pruebas nos obligan.
- El objetivo de completar la implementación se alcanza naturalmente cuando se completan todas las pruebas

¿Como se usa?

1) Se captura la funcionalidad esperada para el software

2) Se escriben las pruebas de aceptación del mismo

3) Se escribe el código para aprobar las pruebas

Beneficios

- Produce mejores diseños
- Eliminar el miedo al refactor
- Posibilidad de demostrar que el código cumple con su especificación
- Se puede desarrollar el firmware sin el hardware

FIRST

- Fast: Los test deben correr rápido
- Isolated: No debe haber dependencia entre los test
- Repeteable: Se debe poder repetir el test n veces y obtener el mismo resultado
- Self-Checking: El propio test debe determinar si el resultado es correcto o no
- Timely: Se deben escribir al mismo tiempo que el código de producción

Como se aplica en embebidos

- Dual targetting
- Mock del hardware
- Inyección de dependencias

BDD: Behaviour Driven development

- Surge como una mejora para facilitar la adopción de TDD.
- Es un conjunto de buenas practicas pensadas para mejorar el desarrollo de software.
- Define un lenguaje ubicuo simple, con sentencias en el lenguaje nativo para facilitar la comunicación entre las parte.
- Busca generar Test desacoplados de la implementación y enfocados en los resultados esperados

Gherkin

- Es una plantilla para escribir las historias de usuario
- Permite capturar los requisitos y las pruebas de aceptación en un formato simple
- Permite automatizar la generación de los Test directamente desde las especificaciones
- Utiliza una estructura y palabras claves para restringir el formato de las mismas

Ejemplo

Característica: Permitir el ingreso y egreso de zonas controladas
Como usuario del sistema de alarma
Quiero poder desactivar el sistema antes de que suene la sirena
Para poder instalar el teclado de la alarma dentro de la casa

Escenario: Salida normal de la casa
Dado un sistema de alarma instalado, funcional y desarmado
Cuando armo el sistema en modo ausente
Entonces un sonido me indica que inicia el tiempo de salida
Cuando transcurren 5 segundos
Y abro la puerta principal
Y cierro la puerta principal
Entonces no se registra una situación de alarma
Cuando transcurren 40 segundos
Entonces un sonido me indica que se agotó el tiempo de espera
Y el sistema queda armado

Ventajas

- Enlaza la pruebas de aceptación con los requisitos en forma directa y automática.
- Proporciona una herramienta efectiva para analizar los requisitos
- Permite involucrar a los clientes en la definición de las pruebas de aceptación

Desventajas

- Por el momento no existen herramientas para lenguaje C
- Los mayores beneficios se obtienen cuando se trabaja con lenguajes dinámicos como Python o Ruby