



Parte III – Desarrollo Dirigido por Pruebas

Tema 6: Principios del desarrollo dirigido por pruebas




1



Objetivo

- Aprender cómo usan los equipos de desarrollo ágil el desarrollo dirigido por pruebas
- Conocido por su acrónimo en inglés: Test Driven Development
- Los programadores deben probar todo lo que puede llegar a fallar, utilizando pruebas automatizadas que deben ejecutarse perfectamente en todo momento



2


2

174

24/2/20
Todo el código que está en el repositorio ha sido probado.
Cuando nos descarguemos código debe seguir funcionando.

Principios básicos de pruebas en metodologías ágiles

- Como el código se comparte y se puede modificar rápidamente, se debe asegurar que se deja en un estado en el que se asegure que no falle
 - Se han de probar todas las clases
 - Se debe poder ejecutar todas las pruebas en todo momento
- Es mejor escribir las pruebas en un principio. Prueba un poco, codifica un poco
 - Las pruebas no son de usar y tirar.
 - Se almacenan con el código fuente, son un elemento permanente del sistema
 - Información que se debe guardar en el gestor de configuración




3

Las pruebas se hacen en cada paso de la integración, y deben irse arrastrando, no hacer pruebas de usar y tirar. Deben estar almacenadas en el repo.

3

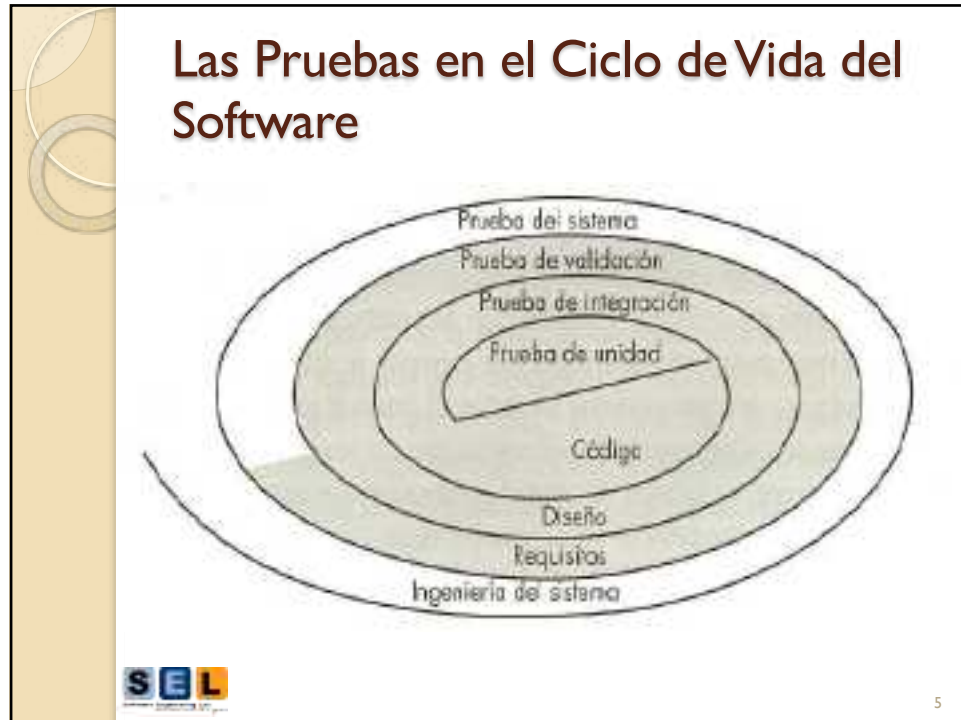
Principios básicos de pruebas en metodologías ágiles

- Solamente se publica una clase cuando se ejecutan correctamente todas las pruebas que se han formulado
 - Por tanto, en el código disponible para el resto de miembros, se ejecutan satisfactoriamente todas las pruebas
 - Esto facilita enormemente la detección de errores en las modificaciones de las clases
- Cada día, una pareja elabora un conjunto de casos de prueba y código fuente
 - Las posibilidades de tener que reescribir mucho código cuando se producen errores disminuyen considerablemente
- Aumenta la percepción de seguridad en el software desarrollado



4

4



5



6

Pruebas Unitarias

- Verifican la unidad más pequeña de software: el método.

Técnicas de Pruebas de unidad

Funcionales o caja negra

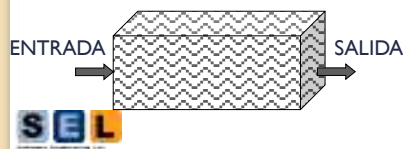
Estructurales o caja blanca

→ No se conoce la estructura que se quiere probar

→ Se conoce la estructura y se puede probar todos los caminos.

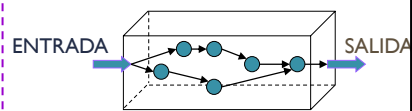
FUNCIONALES

- Se centra en las entradas y las salidas, NO en la estructura interna.
- Es imposible probar todas las entradas y salidas posibles. Se seleccionarán las más relevantes.



ESTRUCTURALES

- Se centra estructura interna.
- Consiste en probar todos los caminos de ejecución.



7

Estrategia de Pruebas Unitarias

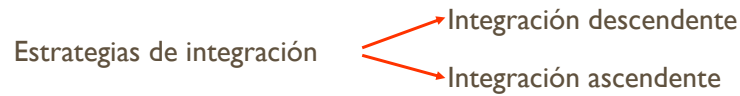
- Crear una clase de prueba – El nombre debe recordar a la clase que se está probando
- Crear un método para crear y configurar (Setup Method) el objeto a probar. En este método se incluirá, si es necesario, un simulador de datos de entrada
- Escribir una prueba
 - Si el método que se prueba cambia el estado del objeto probado, codificar una aserción
- Ejecutar la prueba
 - Si el código falla, corregir el código fuente (o la especificación del caso de prueba) y repetir la prueba.
 - Si el código no falla, pasar a la siguiente prueba.
- Cuando se ejecutan satisfactoriamente todas las pruebas preestablecidas, registrar el código fuente y la prueba en el servidor de configuración



8

8

Pruebas de Integración



- **Descendente:** Los módulos se integran al descender por la jerarquía de control, comenzando con el módulo de control principal (programa principal).
- **Ascendente:** Los módulos se integran desde los componentes de los niveles más bajos de la estructura de la aplicación hasta el módulo de control principal.



9

9

Principio – Paso I

- Los desarrolladores escriben casos de prueba automatizados para verificar que una nueva funcionalidad que se pretende desarrollar o una mejora funciona correctamente antes de escribir cualquier tipo de código que la implemente
- La premisa básica consiste en que el desarrollo comienza cuando se escribe el código de una prueba que falta para la parte más simple de la funcionalidad que se necesita implementar
- La prueba falla porque todavía no hemos implementado código alguno que permita que la prueba funcione correctamente y sin errores
- El valor de este primer paso reside en formalizar mediante código el resultado que se debe proporcionar en esta primera parte de funcionalidad y así tener una idea muy precisa del código funcional que tenemos que generar en este momento



10

10

Principio – Paso II

- El segundo paso en consiste en desarrollar el código más simple posible que permita que ese caso de prueba que hemos codificado anteriormente funcione sin error
- Una vez que hemos conseguido este paso, el código funcional que acabamos de generar es reorganizado o ‘refactorizado’ (actualizado) para asegurar que satisface las reglas y recomendaciones que se incluyen en el estándar de codificación



11

11

Publicación del código

- Pasos para la publicación del código
 1. Las pruebas funcionan correctamente
 2. El código cumple el estándar
 3. Se registra el código funcional, el de pruebas y el log de las pruebas
 4. Se informa al resto del equipo
- Con ello:
 - Se facilita el trabajo de identificar y resolver defecto, porque el espacio sobre el que se tiene que realizar se reduce al código modificado en la última sesión de trabajo
 - No es necesario remontarse al código desarrollado hace días o semanas



12

12

Herramientas de pruebas dirigidas por código

- La definición del caso de prueba se realiza mediante el desarrollo del denominado código de prueba
- Ejemplos: Familia XUnit y similares
- Apropriadas para las pruebas unitarias, de integración o de rendimiento
- Los casos de prueba son código construido que deben configurar los datos de entrada requeridos para las pruebas, ejecutar el código funcional a probar y comprobar que el resultado obtenido de la ejecución es igual al resultado esperado



13

13

Dificultades y recomendaciones

- Sistematización del procedimiento
- Necesidad de un cambio de cultura
- Necesidad de cambio en las rutinas del equipo
- Necesidad de trabajar en pequeños incrementos de código que suelen ser resueltos en pocas horas (menos de una jornada de trabajo) por lo que el coste de calidad se mantiene también bajo



14

14

} Que te acostumbres cuando no te ha
 hecho así antes.
 } Que te trabajes de otra
 manera, se han cogido
 hábitos.

Las pruebas son la documentación más fiable.

Beneficios - I

- Ayuda a asegurar la calidad del código que se desarrolla porque permite centrarse en los requisitos que se debe satisfacer antes de empezar a escribir el código
- Es útil para mantener el código simple fácil de probar entender y modificar porque está dividido en pequeños pasos cuya consecución se corresponde con cada una de las pruebas que el desarrollador va superando en las etapas establecida por la estrategia de desarrollo



15

15

Beneficios - II

- Proporciona documentación acerca de cómo funciona el sistema que estamos intentando desarrollar y que se encuentra registrada en el código fuente facilitando la transmisión de este conocimiento a todas las personas que se incorporan a nuestro equipo de trabajo
- Permite construir un conjunto de pruebas de regresión fácilmente repetibles cuando se actualice cada una de las partes del código
- Actúa como un facilitador de las prácticas de adaptación al cambio porque permite que estos se introduzcan sobre versiones del código de las que se tiene certeza acerca de su correcto funcionamiento



16

16

Preguntas frecuentes acerca de las pruebas (I)

- ¿Cómo realizar las pruebas cuando se necesitan o registran datos de una base de datos?
 - Si los datos se utilizan para realizar otras operaciones, utilizar datos procedentes de ficheros
 - Pruebas de apertura de la conexión de la base de datos
 - Los datos recuperados se pueden contrastar con otros existentes en un fichero o incluidos a mano en el probador



17

17

Preguntas frecuentes acerca de las pruebas (II)

- ¿Qué hago si las pruebas se ejecutan muy lentamente?
 - Esto producirá que haga menos casos de prueba y los realice menos frecuentemente
 - Razones:
 - Las pruebas hacen más de lo necesario
 - Por ejemplo, para probar la verificación de la contraseña, se inserta un usuario nuevo – Cambiar la configuración de las pruebas
 - Una parte del sistema funciona muy lentamente
 - Identificar la zona, de esta manera se podrá optimizar el código, logrando un software más rápido u unas pruebas también más rápidas.



18

18

Preguntas frecuentes acerca de las pruebas (III)

- ¿Qué pasa si es complicado saber cómo probar una clase?
 - Crear la clase es complicado
 - Recodificar estableciendo un constructor que permita crear la clase fácilmente
 - La clase es dependiente de muchos objetos complicados de crear y cargar
 - Reorganización de código



19

19

Preguntas frecuentes acerca de las pruebas (IV)

- ¿Es correcto probar una clase a través de otras (pertenecientes a la aplicación) que ya lo utilizan?
 - Posiblemente encontraremos todos los defectos, pero:
 - Las limitaciones de la clase a probar pueden que no se manifiesten hasta que el software está en producción (pruebas de caja negra) – Es mejor ejercitar las pruebas de caja blanca
 - Cuando las clases pertenecientes a la aplicación fallan es más difícil depurar el error, no sabes si es de esa clase o de la que usa



20

20

Preguntas frecuentes acerca de las pruebas (V)

- ¿Cómo se sabe que se ha probado todo lo que pudiera fallar?
 - Mezcla de conocimiento y experiencia
 - Cuando se produce un error en una prueba de aceptación es que faltan una o varias pruebas unitarias
 - Escribe la clase de prueba y pensar que otras cosas se podrían probar.



21

21

Preguntas frecuentes acerca de las pruebas (VI)

- ¿Qué pasa con los errores que solo surgen cuando se integran dos clases?
 - No debería producirse, pero de hacerlo hay problemas de comunicación y coordinación
 - Nombres de clases inconsistentes
 - Inconsistencia en la definición y utilización de parámetros



22

22

Preguntas frecuentes acerca de las pruebas (VII)

- ¿... y la Interfaz Gráfica de Usuario?
 - No introducir lógica de negocio en las clases de interfaz de usuario
 - Transmisión de eventos y recepción de datos de las clases intermedias
 - ... y de vez en cuando, perder tiempo en ejercitar la interfaz de usuario o utilizar herramientas avanzadas de grabación de diálogos



23

23

Herramientas de grabación y reproducción (I/2)

- Útiles para la realización de pruebas de integración y del sistema
- Ejemplos: Selenium, TestingWhiz y Sagi
- Las pruebas se definen grabando una secuencia de pasos en la interfaz de usuario y determinando los resultados que se deben conseguir después de cada paso
- Para ejecutar la prueba, la máquina repite automáticamente los pasos verificando las condiciones establecidas para el final de cada uno de ellos



24

24

Herramientas de grabación y reproducción (2/2)

- Cuando un paso falla la prueba finaliza con error
- Es importante establecer un guion para la prueba que incluya todos los pasos a dar y la comprobación de las condiciones de éxito en cada uno de ellos y tener en cuenta los posibles problemas debidos a cuestiones de sincronización y tiempo de respuesta de las comunicaciones asociadas a las funcionalidades que se quiere probar



25

25

Cómo ejecutar las pruebas de sistema

- Si el programa es batch, editar un fichero de entrada con todos los datos posibles, hacer un programa que lea el fichero de entrada y ejecute el programa para cada dato de entrada
- Utilizar el mismo truco para las funcionalidades de informe (batch o no)
- Escribir las pruebas en un entorno automatizado de pruebas (JUnit)
- Permitir a los usuarios escribir las pruebas en una hoja de cálculo, haciendo un programa que las lea y las ejecute automáticamente.
- Hacer un simulador de entrada de datos y hacer que los usuarios lo utilicen para registrar los datos de entrada



26

26