Especialización en sistemas embebidos (CESE)

Testing de software embebido

Clase 4

Laboratorio de Sistemas Embebidos

¿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

¿Para que sirve?

- No más depuración
- Posibilidad de demostrar que el código cumple con su especificación
- Posibilidad de refactor sin miedo a romper algo

¿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

¿Que cuidados hay que tener?

- Evitar que los test corran despacio
- No testear detalles de implementación
- Evitar la dependencia entre los test

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

Beneficios

- Produce mejores diseños
- Da tranquilidad al programador
- Da auditabilidad a los administradores
- Eliminar el miedo al refactor
- Se puede desarrollar el firmware sin el hardware

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: Autorizar o denegar el acceso a una tarjeta
  Como responsable de la seguridad del ambiente controlado
  Quiero poder agregar o quitar tarjetas autorizadas
  Para poder controlar quienes ingresan

Escenario: Estado inicial del equipo
  Dado un equipo con la configuración de fabrica
  Cuando se activa se acerca la tarjeta A al lector de rfid
  Entonces no se autoriza el acceso a la tarjeta A
  Y se emite la secuencia de notas descendentes

Escenario: Autorización de una tarjeta
  Dado un equipo con una cerradura electromagnética
  Y con el tiempo de apertura configurado en 3 segundos
  Y con la lista de tarjetas autorizadas vacía
  Cuando se agrega la tarjeta A a la lista de autorizadas
  Cuando se acerca la tarjeta A al lector de rfid
  Entonces se autoriza el acceso a la tarjeta A
  Y se emite la secuencia de notas ascendentes
```

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

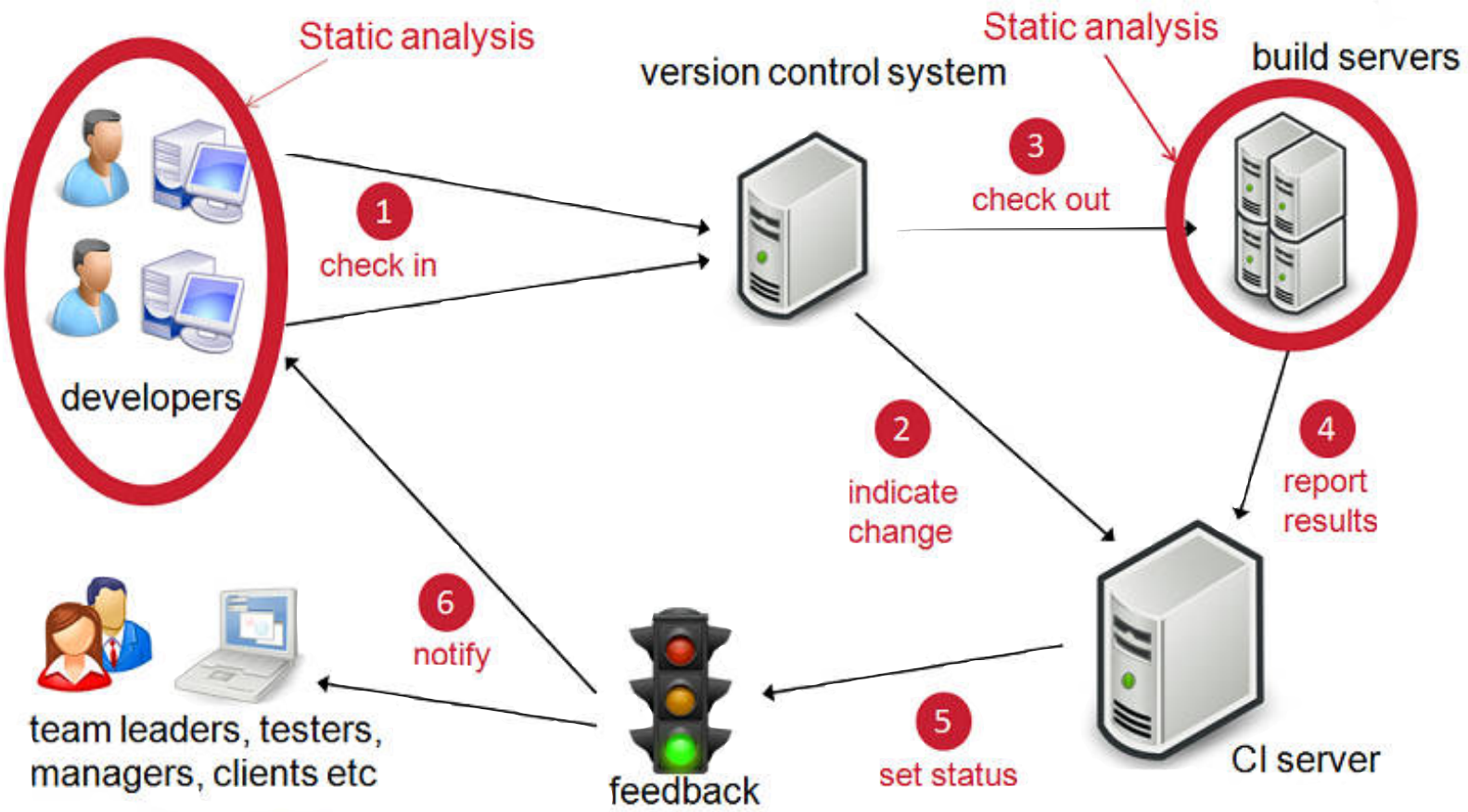
Desventajas

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

Un paso mas: Integración continua

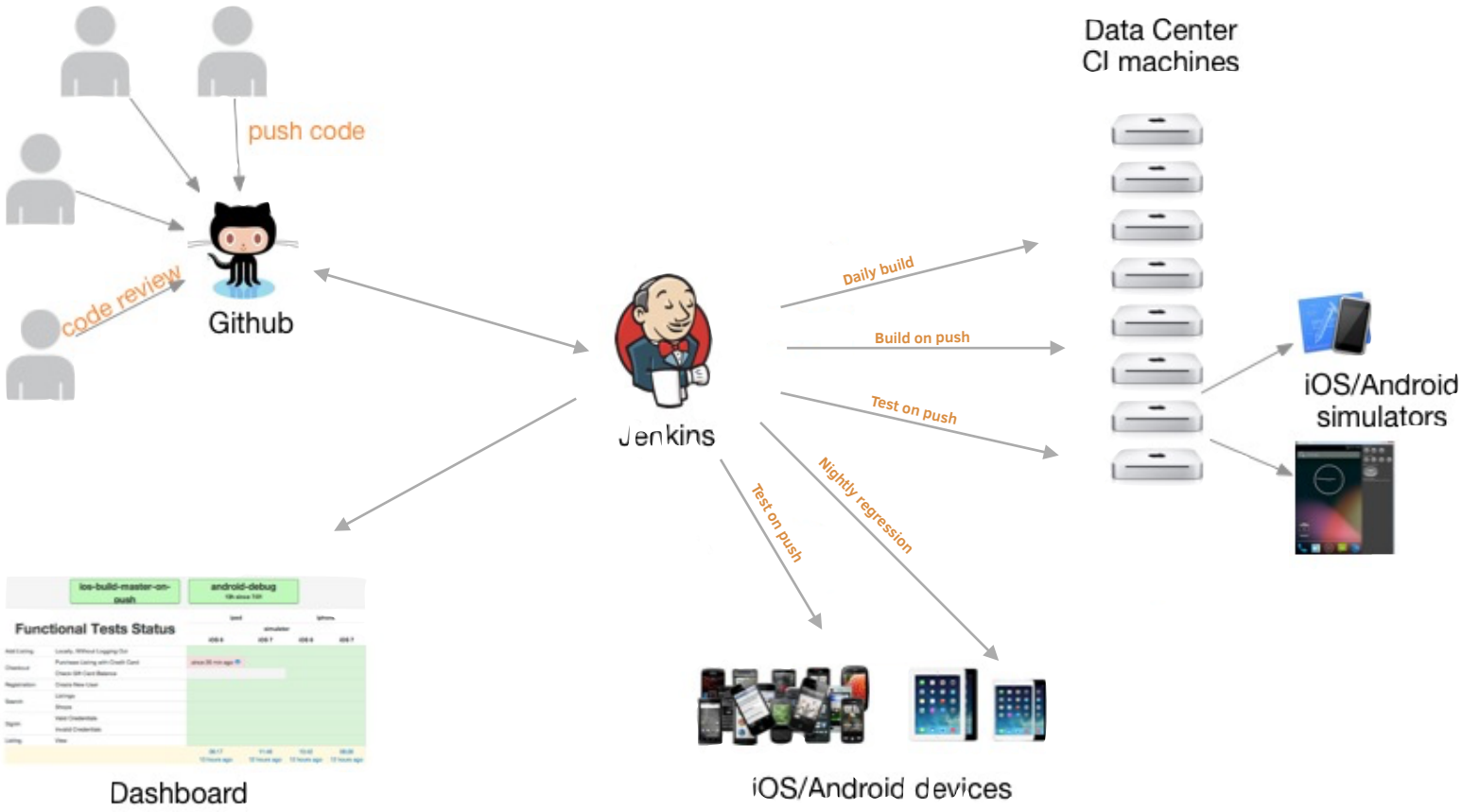
- La idea de integración continua es compilar y probar el proyecto en forma automática
- Para poder implementarlo es imprescindible:
  - Tener el código almacenado en un repositorio
  - Tener test automatizados para ejecutarlos desde una linea de comandos

Esquema de trabajo



https://www.qa-systems.com/tools/qa-c/continuous-integration/

Aplicación en embebidos y móviles



https://codeascraft.com/2014/02/28/etsys-journey-to-continuous-integration-for-mobile-apps/

Servidor virtuales de CI

- Existen servidores gratuitos para proyectos open source
  - http://drone.io
  - http://travis-ci.org
- Drone se configura en el sitio y por lo tanto monitores todas las ramas.
- Travis se configura en un archivo, si no existe en una rama entonces no se prueba.
- Ambos corren en servidores virtuales, solo sirven para probar software.

Bibliografía

- BDD in Action
  John Ferguson Smart
- The OLD Object Mentor Blog Site
  http://butunclebob.com
- Ejemplo simple de temporizadores con BDD
  https://bitbucket.org/labmicro/temporizadores
- Refactoring: Improving the Design of Existing Code
  Martin Flower