UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

Monitorización del proceso de pruebas con Git

Medición y Calidad

Alumnos Plamen Petyov
Napoleón Cortés
Maria Viyuela
Rodrigo Martinez

Tutor Carlos López Nozal
DEPARTAMENTO DE INGENIERÍA CIVIL
Área de Lenguajes y Sistemas Informáticos

Burgos, 12 de marzo de 2016

Índice de contenido

Índice de ilustraciones

Índice de tablas

1 ENUNCIADO

En la práctica se va simular un pequeño desarrollo de un producto software para realizar mediciones sobre él. El objetivo es establecer un caso de estudio que sirva para caracterizar y evaluar tanto el producto desarrollado como el proceso seguido. (a)

1.1 Descripción del producto

Dado un código de ejemplo del patrón diseño creacional Object Pool, se debe crear una batería de pruebas tal que las coberturas de sus clases sean del 100%. El código de las clases se puede obtener en el repositorio https://github.com/clopezno/poolobject . La batería de pruebas Junit debe estar contenida en la clase ubu.gii.dass.test.c01.ReuseblePoolTest.java.

1.2 Descripción del proceso

El proceso de desarrollo de las batería de pruebas se va a gestionar utilizando el control de versiones del sistema Git proporcionado por el repositorio de proyectos GitHub (https://github.com).

Los pasos para gestionar el procesos son los siguientes:

1. Cada miembro del equipo tiene que estar registrado en GitHub.

2. Uno de los miembros tiene que realizar un fork del repositorio donde se encuentra el código que se quiere probar https://github.com/clopezno/poolobject El nuevo repositorio tiene que ser público.

3. Invitar al resto de miembros del equipo para que puedan participar en el desarrollo del conjunto de pruebas.

4. Cada nuevo test realizado ejecutar un commit/push al repositorio del grupo. El texto del commit tiene que describir el caso de prueba añadido.

2 OBJETIVOS

- Comprender los objetivos de medición relacionados con la caracterización y la evaluación de productos, procesos y recursos software.
- Comprender, aplicar y analizar técnicas de medición sobre entidades de productos software relacionados con conjuntos de pruebas de software.
- Comprender, aplicar y analizar medidas relacionadas sobre entidades de proceso y recursos de prueba software.

3 DESARROLLO

Para proceder a desallorar los metodos testGetInstance, testAcquireReusable y testReleaseReusable hemos utilizado la siguiente tabla para cubrir la cobertura.

Método A Probar	Método De Prueba	Pruebas	Entrada Prueba	Resultado Esperado
getInstance()	testGetInstance()	Que una llamada no devuelva null.	-	TRUE
		Que dos llamadas devuelvan la misma instancia.	-	TRUE
acquireReusable()	testAcquireReusable()	Que una llamada no devuelva null.	Instancia ReusablePool	TRUE
		Que una llamada devuelva una instancia de Reusable.	Instancia ReusablePool	TRUE
		Que dos llamadas devuelvan instancias diferentes.	Instancia ReusablePool	TRUE
		Que una tercera llamada lance excepción	Instancia ReusablePool	NotFreeInstanceException
releaseReusable()	testReleaseReusable()	Que se puedan soltar dos instancias consecutivas de Reusable	Instancia Reusable, Instancia ReusablePool	TRUE
		Que la ultima instancia soltada es la primera en ser adquirida (Funcionamiento de pila)	Instancia Reusable, Instancia ReusablePool	TRUE
		Que al soltar dos veces la misma instancia de	Instancia Reusable, Instancia	DuplicatedInstanceExcepti

		Reusable lance excepción	ReusablePool	on
	testReleaseNewReusable()	Que no se pueda soltar una instancia de Reusable que NO haya sido obtenida a través de la instancia de ReusablePool	Instancia Reusable, Instancia ReusablePool	Fallo (AssertionError)

Tabla 3.1: Pruebas

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
poolobject	74,0 %	171	60	231
src	74,0 %	171	60	231
ubu.gii.dass.c01	62,7 %	79	47	126
Client.java	0,0 %	0	37	37
Reusable.java	23,1 %	3	10	13
DuplicatedInstanceException.java	100,0 %	4	0	4
NotFreeInstanceException.java	100,0 %	4	0	4
ReusablePool.java	100,0 %	68	0	68
ubu.gii.dass.test.c01	87,6 %	92	13	105
ReusablePoolTest.java	87,6 %	92	13	105

Ilustración 1: Cobertura obtenida

4 CUESTIONES RESUELTAS

4.1 ¿Se ha desarrollado el trabajo en equipo?

Adjuntamos tabla con datos cuantitativos con la participación en commits en el proyecto con los miembros del equipo.

Miembros del equipo	# Número de commits	# Porcentaje de participación (commits miembro/totales)
Plamen	2	40.00%
María	1	20.00%
Napoleón	1	20.00%
Rodrigo	1	20.00%

Tabla 4.1: Commits realizados

Estos datos se pueden verificar a continuación:

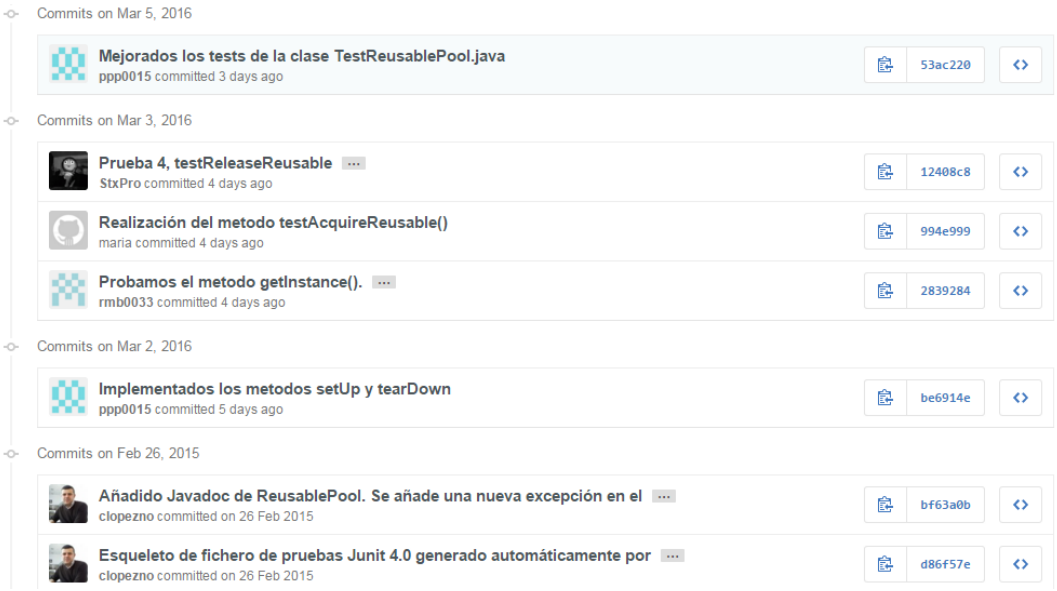

Ilustración 2: Commits en el repositorio

4.2 ¿Tiene calidad el conjunto de pruebas disponibles?

Si, tenemos una cobertura del #100% de la clase ReusablePool #(68 instrucciones cubiertas de las 68 disponibles), para comprobar esto ver *ilustracion1*.

4.3 ¿Cuál es el esfuerzo invertido en realizar la actividad?

Para calcular el tiempo empleado en esta actividad (esfuerzo invertido) lo mediremos a partir de **dias** que hemos hecho commits, ya que no tenemos la medida exacta en horas o minutos. Medida cuantitativa: #3 (Dias)

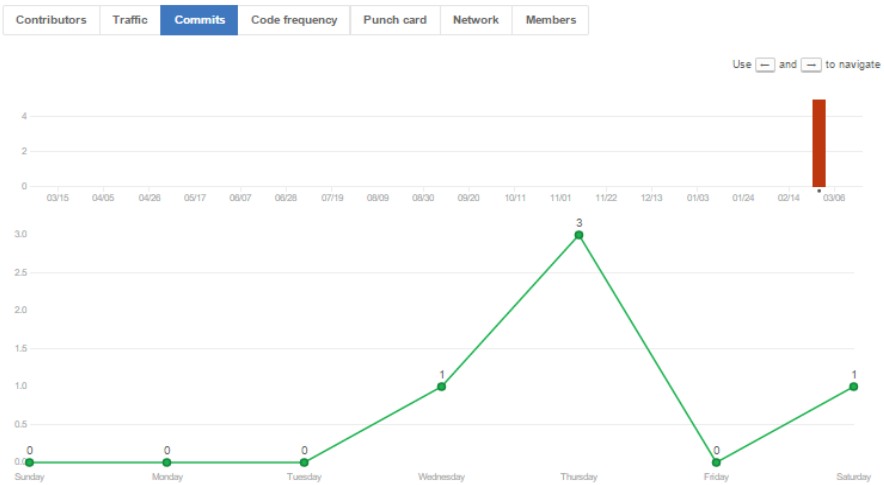

Ilustración 3: Actividad de los commits

4.4 ¿Cuál es el número de fallos encontrados en el código?

Se ha podido localizar un fallo en el código. El método releaseReusable(), usado para devolver una instancia de la clase Reusable, está mal implementado, de modo que permite devolver instancias de Reusable que no hayan sido obtenidas mediante una instancia de ReusablePool. Ésto conlleva a un aumento de las instancias en el pool y por lo tanto a la violación de la condición en el método acquireReusable() (ver código del método)