

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

PG-C01

Monitorización del proceso de pruebas con Git-Travis-Codecov

Alumnos Iván Arjona Alonso
 Marta Monje Blanco

Tutores Carlos López Nozal
 Jesús Alonso Abad
 DEPARTAMENTO DE INGENIERÍA CIVIL
 Área de Lenguajes y Sistemas Informáticos

Burgos, 25 de febrero de 2018



Este documento está licenciado bajo [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/)

Índice de contenido

1	Introducción	3
2	Pruebas realizadas	3
3	Cobertura	5
4	Cuestiones	6
5	Conclusiones	7

Índice de ilustraciones

Ilustración 1: Captura de los teses pasados.....5

Ilustración 2: Gráfica de cobertura y complejidad.....5

Ilustración 3: Tabla de commits y cobertura.....6

Índice de tablas

1 INTRODUCCIÓN

En esta práctica vamos a realizar una serie de tests sobre un código e iremos monitorizando la cobertura que nos ofrecen mediante el uso de Travis y Codecov.

Vamos a trabajar con Eclipse IDE for Java Developers, con Github, desde el repositorio donde tengamos subida la práctica, con Travis CI y por último, con Codecov.io.

Los enlaces al [repositorio de Github](#), de [Codecov.io](#) y de [Travis CI](#).

2 PRUEBAS REALIZADAS

Para poder comprobar si con las pruebas cubrimos debidamente el código que se nos ha dado, obviamente tenemos que programar las pruebas sobre los diferentes métodos para comprobar que estas son suficientemente cubrientes.

Primera prueba

Comprueba que se obtiene correctamente la instancia:

```
@Test
    public void testGetInstance() {
        ReusablePool pool = ReusablePool.getInstance();
        // No es nulo
        assertNotNull(pool);
        // El objeto devuelto es una instancia de ReusablePool
        assertTrue(pool instanceof ReusablePool);
    }
```

Segunda prueba

Comprueba el método AcquireReusable:

```
@Test(expected = NotFreeInstanceException.class)
    public void testAcquireReusable() throws NotFreeInstanceException {
        Reusable r1 = pool.acquireReusable();
        Reusable r2 = pool.acquireReusable();

        //comprueba que se devuelve una instancia de Reusable
        assertTrue(r1 instanceof Reusable);
    }
```

```
        assertTrue(r2 instanceof Reusable);

        pool.acquireReusable();
    }
}
```

Tercera prueba

Comprueba que se libere el objeto reusable:

```
@Test(expected = DuplicatedInstanceException.class)
public void testReleaseReusable() throws
DuplicatedInstanceException, NotFreeInstanceException {
    Reusable r = pool.acquireReusable();
    pool.releaseReusable(r);
    // Lanza excepción DuplicatedInstanceException
    pool.releaseReusable(r);
}
}
```

Cuarta prueba

Comprueba que la cadena de texto que devuelve el método acquireReusable() acabe de forma prevista:

```
@Test
public void testReusable() throws NotFreeInstanceException {
    Reusable r = pool.acquireReusable();
    String rs = r.util();
    //Comprobamos que el final de la cadena devuelta es esta
    assertTrue(rs.endsWith("Uso del objeto Reutilizable"));
}
}
```

Teses pasados

Los tesis se superan y pasan sin ningún problema:

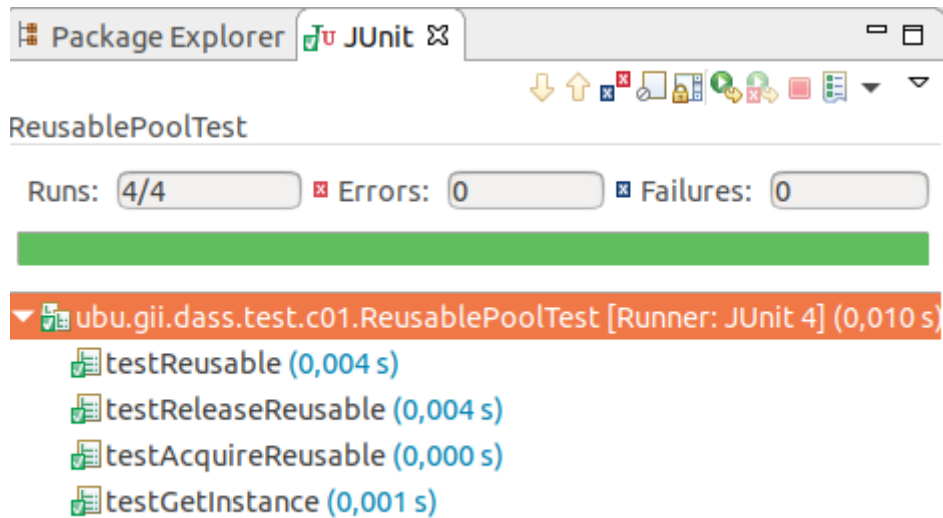


Ilustración 1: Captura de los teses pasados

3 COBERTURA

La cobertura en nuestro caso es del 100% y dicho dato lo podemos comprobar desde codecov.io, el cual para nuestro repositorio nos muestra lo siguiente el siguiente gráfico, donde podemos ver la cobertura y complejidad a través del flujo de commits:

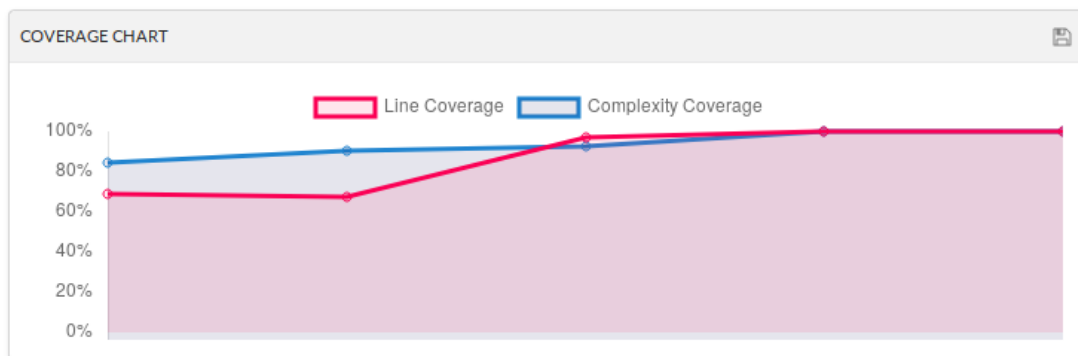


Ilustración 2: Gráfica de cobertura y complejidad

Además de poder seguir la cobertura a través del gráfico, también podemos saber de forma más específica, como cada commit ha intervenido en dicha cobertura.

Commits







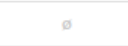









	documentación		100.000%		100.000%	
mmb0093	2 hours ago	master	3246750	✓ CI Passed		
	Prueba instanciar un cliente		100.000%		(+2.858%)	100.000% (+7.143%)
IvanArjona	7 hours ago	master	31b9d57	✓ CI Passed		
	Ignora los test en codecov		97.143%		(+29.902%)	92.858% (+2.381%)
IvanArjona	2 days ago	master	62bcb94	✓ CI Passed		
	test del método Reusable		67.242%		< 80.000% >	(-1.277%)
mmb0093	3 days ago	master	3d0afde	✓ CI Passed		
	Test para el método ReleaseReusable		68.519%		< 66.667% >	85.000%
IvanArjona	3 days ago	master	b74d373			

Ilustración 3: Tabla de commits y cobertura

4 CUESTIONES

¿Se ha realizado el trabajo en equipo?

Sí, hemos trabajado los dos, cada uno desde nuestra cuenta, en el mismo repositorio de github y hay commits de ambos.

¿Tiene calidad el conjunto de pruebas disponibles?

Dado que hacen una cobertura del 100%, nosotros consideramos que sí.

¿Cuál es el esfuerzo invertido en realizar la actividad?

Consideramos que el esfuerzo invertido se puede extraer del tiempo invertido en el trabajo en sí. En total habremos invertido unas 3 horas, de las cuales 2, eran en la clase de prácticas y dado que el número de commits que hemos hecho es menos de 20, consideramos que el esfuerzo ha estado más en familiarizarnos con las herramientas que con la realización persé.

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

Ninguno. Lo único que se ha modificado es que decidimos formatearlo, pero no es un fallo.

5 CONCLUSIONES

Las herramientas de automatización de tesis son muy útiles a la hora de gestionar y controlar una pila de tesis grandes. Quizá en proyectos de mayor tamaño esta utilidad sea más valiosa.

Por lo general agilizan el proceso de pruebas y lo más importante, nos permiten dotar a las pruebas del concepto de calidad.