Diseño y Pruebas II

WIS TESTING REPORT

Grupo: C1.02.07
Repositorio: https://github.com/DP2-C1-02-07/Acme-L3
Autor/es:

- Javier Nunes Ruiz - **javnunrui@alum.us.es**
- Manuel Palacios Pineda - **manpalpin@alum.us.es**
- Manuel Carnero Vergel - **mancarver1@alum.us.es**
- Pablo Martínez Valladares - **pabmarval@alum.us.es**
- Julio Navarro Rodríguez - **julnavrod@alum.us.es**

Fecha: 15/02/2023

1. Índice

2. Tabla de versiones

Versión	Fecha	Descripción
1.0	12/02/2023	Apartados 3, 4, 5.1 y 7
1.1	13/02/2023	Apartados 5.2, 6 y 7

3. Resumen del documento

Somos el grupo C1.02.07 de la asignatura DP2 de la Universidad de Sevilla, y en el siguiente informe expondremos el conocimiento del equipo sobre *testing* de sistemas de información.

4. Introducción

Durante la formación académica de los integrantes del equipo, hemos cursado una asignatura acerca de uno de los aspectos más importantes durante el desarrollo de software, el *testing*. Esta asignatura es Diseño y Pruebas I, la cual a su vez, se centra en los tests unitarios.

5. Contenidos

5.1. Testing

Entendemos como *testing*, el proceso de evaluar y verificar que un producto software se comporta como era de esperar. Los beneficios de un buen *testing* incluyen la reducción del coste del desarrollo software, así como mejorar el rendimiento del código. El *testing* se basa en 2 conceptos principales:

- Verificación: el software diseñado y desarrollado debe cumplir los requisitos.
- Validación: el software debe satisfacer las necesidades del cliente.

El conocimiento principal del grupo se centra en las pruebas unitarias (*unit testing*).

5.2. Pruebas Unitarias

Una prueba unitaria tiene como sujeto bajo prueba *(SUT)* una unidad (métodos y funciones individuales).

Hay dos tipos de pruebas unitarias: positivas (aquellas que comprueban el correcto funcionamiento del código) y negativas (aquellas que comprueban casos anormales, como por ejemplo, las excepciones).

Para que una prueba unitaria se considere como tal debe seguir las siguientes características *FIRST*:

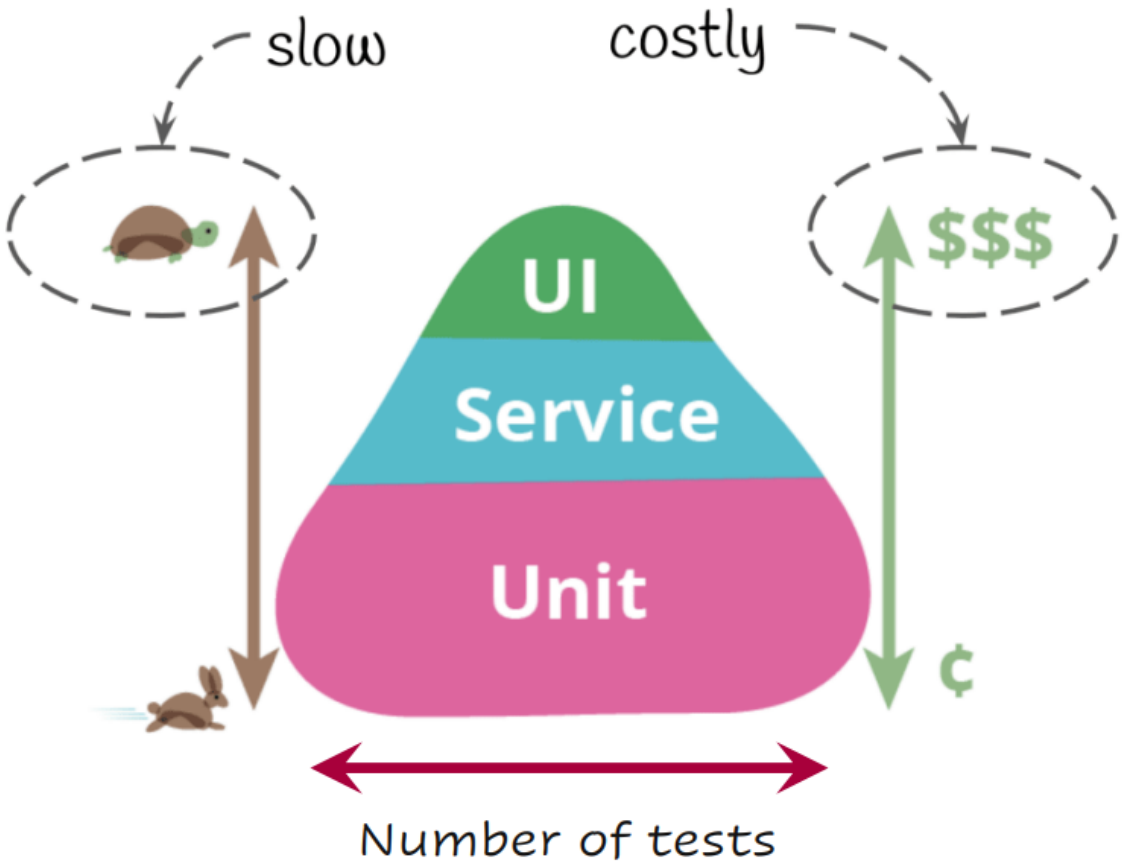
- Fast: fáciles y rápidas de ejecutar.
- Independent: no deben depender de otras condiciones provocadas por otros tests.
- Repeatable: no deben depender de factores externos, como la fecha actual por ejemplo.
- Self-checking: debe de poder determinar si es un éxito o no sin depender de ningún humano que supervise el resultado.

- Timely: deben ser creados o actualizados al mismo tiempo que el código bajo prueba.

La estructura interna de la prueba unitaria debe estar formada por 3 partes:

- Arrange/Fixture: se configuran los datos a usar en la prueba.
- Act: se ejecuta el sujeto de la prueba.
- Assert: se comprueba que se devuelve el resultado esperado.

Mediante la siguiente imagen, podemos ver fácilmente la relación entre la cantidad, el precio y la rapidez de los tests en función del tipo de prueba.



Para el siguiente apartado, debemos introducir el concepto de *test double*. Un *test double* es un objeto que pretende ser un objeto real en un test. Les damos uso para evitar efectos secundarios y configuraciones complejas. También es conveniente conocer el concepto de *seams*, que son lugares donde se puede alterar el comportamiento del programa sin editarlo.

Al fin y al cabo, un *test double* es un *seam* que aíslan el comportamiento del *SUT* de sus colaboradores.

Hay varios tipos de *test double*, pero nos centraremos en los dos siguientes:

- Stubs: no tienen lógica, pero devuelven respuestas enlatadas. Se usan cuando se necesita que algún colaborador devuelva un valor específico para conseguir un estado concreto del *SUT*.
- Mocks: se requieren unas expectativas de cómo debería funcionar, para luego compararlas con el resultado real. Se usan para analizar la interacción entre objetos.

Para finalizar, anotamos que los frameworks conocidos para el *testing* son:

- JUnit: para pruebas unitarias
- Mockito: para *tests doubles*.

6. Conclusiones

Para finalizar, podemos concluir que, al cursar la asignatura DP1, hemos obtenido un conocimiento básico sobre las pruebas unitarias. Con la asignatura DP2 buscamos ampliar más este conocimiento y obtener más experiencia aplicándolo, así como estudiar nuevas formas de *testing*.

7. Bibliografía

- BP Logix: definición de verificación y validación en el contexto de software *testing*.
- IBM: definición de software *testing*.
- Objc: definición de *mocks*.
- Presentaciones del Tema 1, 8 y 9 de DP1 Curso 2022-23, Universidad de Sevilla.