Prueba unitaria

umh2818-TADS

En programación, una **prueba unitaria** es una forma de comprobar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto.

1 Características

Para que una prueba unitaria tenga la *calidad suficiente* se deben cumplir los siguientes requisitos:

Automatizable No debería requerirse una intervención manual. Esto es especialmente útil para *integración continua*.

Completas Deben cubrir la mayor cantidad de código.

Repetibles o Reutilizables No se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para *integración continua*.

Independientes La ejecución de una prueba no debe afectar a la ejecución de otra.

Profesionales Las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

Aunque estos requisitos no tienen que ser cumplidos al pie de la letra, se recomienda seguirlos o de lo contrario las pruebas pierden parte de su función.

2 Ventajas

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

Fomentan el cambio Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar

refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Simplifica la integración Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las *pruebas de integración*.

Documenta el código Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.

Separación de la interfaz y la implementación Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando *objetos mock* (*mock object*) para simular el comportamiento de objetos complejos.

Los errores están más acotados y son más fáciles de localizar Dado que tenemos pruebas unitarias que pueden desenmascararlos.

3 Limitaciones

Es importante darse cuenta de que las pruebas unitarias no descubrirán todos los errores del código. Algunos enfoques se basan en la generación aleatoria de objetos para amplificar el alcance de las pruebas de unidad.^[1] Esta técnica se conoce como *testing aleatorio* (RT, por *random testing*). Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto. Además, puede no ser trivial anticipar todos los casos especiales de entradas que puede recibir en realidad la unidad de programa bajo estudio. Las pruebas unitarias sólo son efectivas si se usan en conjunto con otras pruebas de software.

4 Herramientas

- **JUnit**: Entorno de pruebas para Java creado por Erich Gamma y Kent Beck. Se encuentra basado

en SUnit creado originalmente para realizar pruebas unitarias para el lenguaje Smalltalk.

- **TestNG**: Creado para suplir algunas *deficiencias* en JUnit.
- **JTiger**: Basado en anotaciones, como TestNG.
- **SimpleTest**: Entorno de pruebas para aplicaciones realizadas en PHP.
- **PHPUnit**: Sistema para la realización pruebas unitarias en PHP.
- **PHPUnit**: Versión del framework para lenguajes C/C++.
- **NUnit**: Versión del framework para la plataforma.NET.
- **FoxUnit**: Entorno OpenSource de pruebas unitarias para Microsoft Visual FoxPro
- **MOQ**: Entorno para la creación dinámica de objetos simuladores (mocks). «MOQ».
- **QUnit**: Librería para pruebas unitarias en Javascript. Creada por la fundación jQuery, ha sido reescrita para ser independiente de la librería jQuery.
- **libunittest**: Librería portable para pruebas unitarias en C++ que usa el nuevo estándar C++11.
- **CUnit**: Entorno para escribir, administrar y correr test unitarios en lenguaje C.
- **PyUnit**: Framework para la elaboración de pruebas unitarias en python.

5 Referencias

- [1] Barrientos, Pablo Andrés (25 de abril de 2014). *Enfoque para pruebas de unidad basado en la generación aleatoria de objetos*. p. 101. Consultado el 28 de abril de 2014.

6 Origen del texto y las imágenes, colaboradores y licencias

6.1 Texto

- **Prueba unitaria** *Fuente:* https://es.wikipedia.org/wiki/Prueba_unitaria?oldid=87430428 *Colaboradores:* Vanbasten 23, Jonik, Echner, Rodrigouf, Guille.hoardings, Alhen, Yrbot, YurikBot, KnightRider, CEM-bot, Thijs!bot, JAnDbot, TXiKiBoT, Biasoli, Cinevoro, Volkov-Bot, AlleborgoBot, Zerial, Botito777, MastiBot, Luckas-bot, Nallimbot, Gepser, Waeswaes, ChuispastonBot, Fdbozzo, Sphera5, KLBot2, Invadibot, Bibliofilotransornado, MahdiBot, Nutibara, Gonzalo.villarreal, Leammau y Anónimos: 33

6.2 Imágenes

6.3 Licencia del contenido

- Creative Commons Attribution-Share Alike 3.0