



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 12

Pruebas Unitarias

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

5 de octubre de 2020

1. Anuncios curso
2. Pruebas unitarias

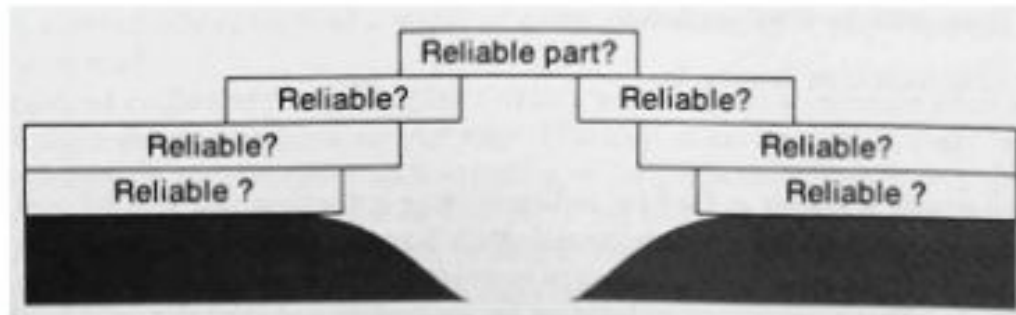
Anuncios curso

- **Coevaluación E1:** contestaron 36 / 90
- **Notas E1:** se publicarán hoy lunes 5 de octubre
- **Actividad 2:** se entrega este martes 6 de octubre
- **Entrega 2:** se entrega el miércoles 14 de octubre
 - No habrá clases este miércoles 7 de octubre
- **Charla NotCo:** miércoles 14 de octubre
- **Actividad 3:** se entrega el martes 20 de octubre

1. Anuncios curso
2. Pruebas unitarias

Pruebas unitarias

- Se centran en verificar las unidades más pequeñas del *software* (componentes y módulos).
- Se pueden realizar antes, durante o después de la codificación.
- Se debe tener control de los resultados esperados (*inputs y outputs*)



Tests unitarios: Beneficios

- Permiten hacer cambios al código de manera segura
- Ayudan a entender el diseño y funcionalidades a desarrollar
- Sirven como apoyo a la documentación (son ejemplos de uso)

Tests unitarios: Costos

- Consumen más tiempo en el corto plazo
 - Diseñarlos
 - Implementarlos
 - Mantenerlos
- No todas las pruebas agregan el mismo valor
- No representan ni garantizan la calidad del *software*

Mocks y Stubs

Mocks:

Son imitaciones de objetos, de las cuales se espera que ciertos métodos sean invocados durante un *test*. De no ser así el *test* falla.

Stubs:

Son imitaciones de objetos que proveen resultados predefinidos para ciertas invocaciones sobre ellos. Son útiles para probar de forma aislada los componentes.

También existen conceptos similares:

- *doubles, dummies, fakes*

Test unitarios: estructura

1. Nombre del *test*
2. *Setup* general de las pruebas
3. *Setup* particular de un *test*
4. Ejecución del método a probar
5. Validación de resultados a través de *asserts*
6. *Teardown*

Coverage aplicado

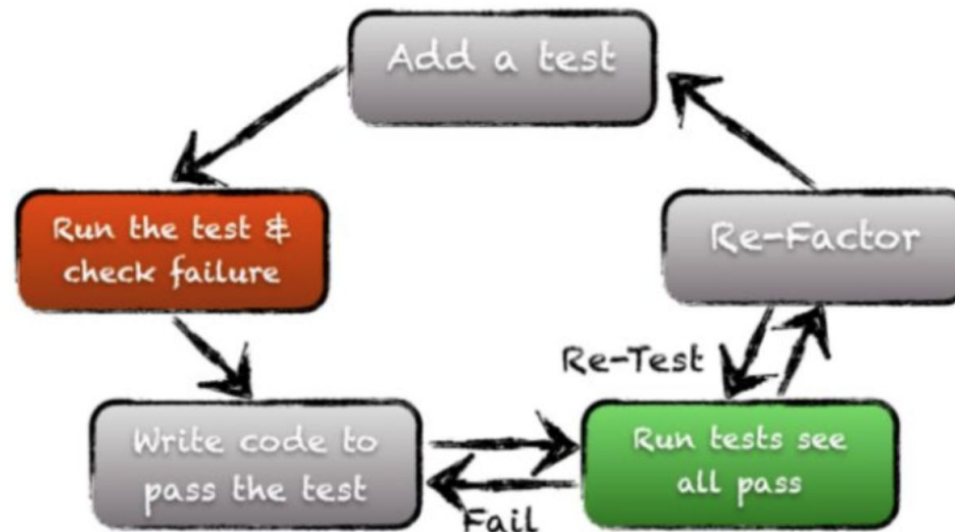
Distintos criterios de medición computables:

- ***Function coverage***: proporción de métodos que son invocados.
- ***Statement coverage***: proporción de instrucciones ejecutadas.
- ***Branch coverage***: proporción de caminos independientes recorridos.
- ***Condition coverage***: proporción de predicados/cláusulas probados.

Test Driven Development

Metodología basada en desarrollar código en pequeños ciclos iterativos que incluyen:

- Diseñar *tests* para un requerimiento
- Desarrollar código hasta que los *tests* pasen
- Mejorar implementación



Test Driven Development

Beneficios:

- Garantiza que toda línea de código tenga *tests* asociados.
- Implica un análisis del diseño del código al momento de crear los *tests*.

Desventajas:

- Genera muchos *tests* que quedan obsoletos rápidamente.
- Puede que no se justifique probar todo el código exhaustivamente.

Recomendaciones al *testear*

- Nombres descriptivos para *tests* y variables
 - Apoyo a la documentación y más fáciles de mantener
- Valores definidos explícitamente al momento de comparar *outcomes*
- Un objetivo claro y definido por *test*
 - Conocido como “un *assert*” (aunque no necesariamente)
- Evitar pruebas redundantes y operaciones costosas (como escribir en base de datos)
- No depender de condiciones externas al código
 - Por ejemplo: API externa, valores de *DateTime.now*
- Si un código es difícil de *testear* es mejor rediseñarlo

Flaky test

- Son *tests* que su resultado depende de factores externos al código.
 - Para código idéntico en distintas ejecuciones del *test* este puede fallar o aprobar.
- Esto se puede deber a:
 - Concurrencia de procesos
 - Dependencia entre *tests*
 - Dependencia en API externa o *Time*
- Son difíciles de detectar e identificar si el problema es del *test* o del código asociado.

Lectura complementaria

- [Martin Fowler - UnitTest](#)
- [Flaky Tests - A War that Never Ends](#)
- [Test Flakiness – Methods for identifying and dealing with flaky tests](#)
- [Flaky tests](#)



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 12

Pruebas Unitarias

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

5 de octubre de 2020