Testing

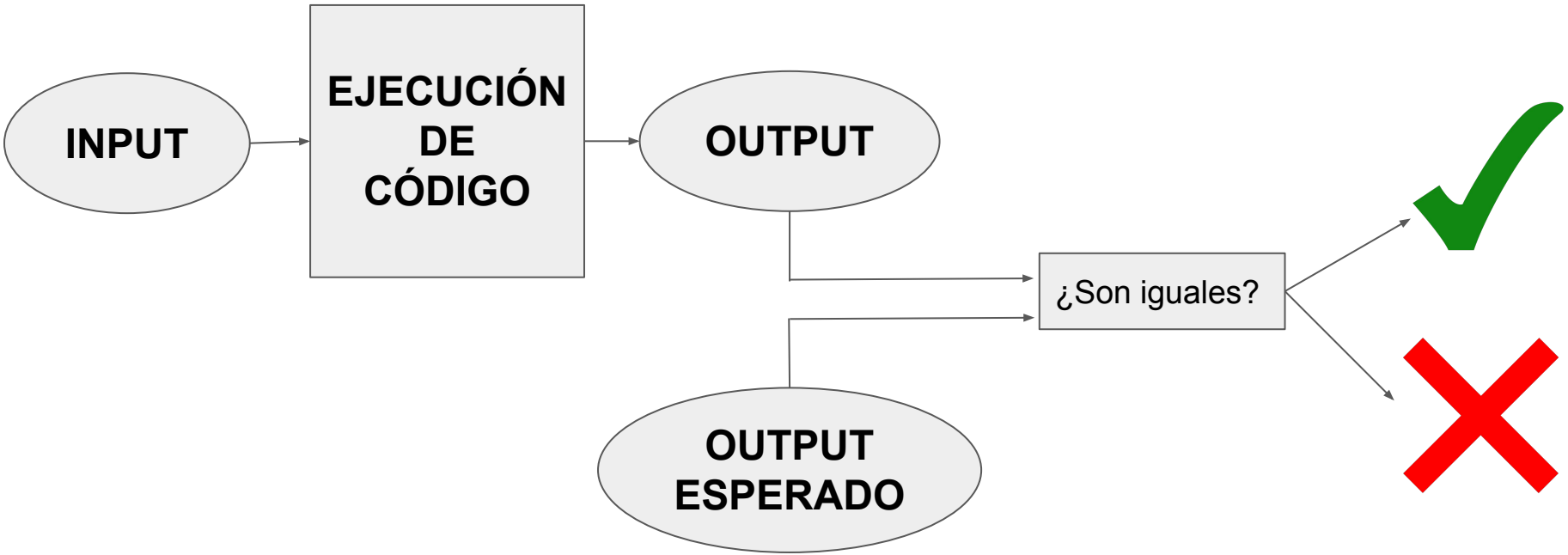
75.41 - Algoritmos y Programación II

2° Cuatrimestre 2019

¿Qué es una test?

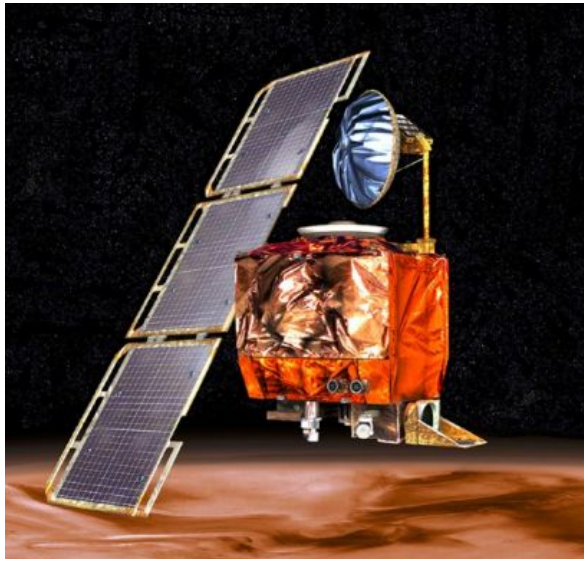
“Un procedimiento destinado a determinar la calidad, rendimiento o confiabilidad de algo, especialmente antes que sea llevado al uso masivo”

Anatomía de una test



Importancia del testing en el software

- Software está en todo:
  - Computadoras, celulares, electrodomésticas, autos, aviones, etc.
  - NASA: *Mars Climate Orbiter I*, pérdida de 328M U$S



- Software es algo que cambia constantemente:
  - Cambios son relativamente fáciles
  - Clientes siempre piden más
  - Necesidad de adaptarse a cambios del entorno
  - Necesidad de solucionar bugs

Ventajas del testing
….y desventajas de su ausencia

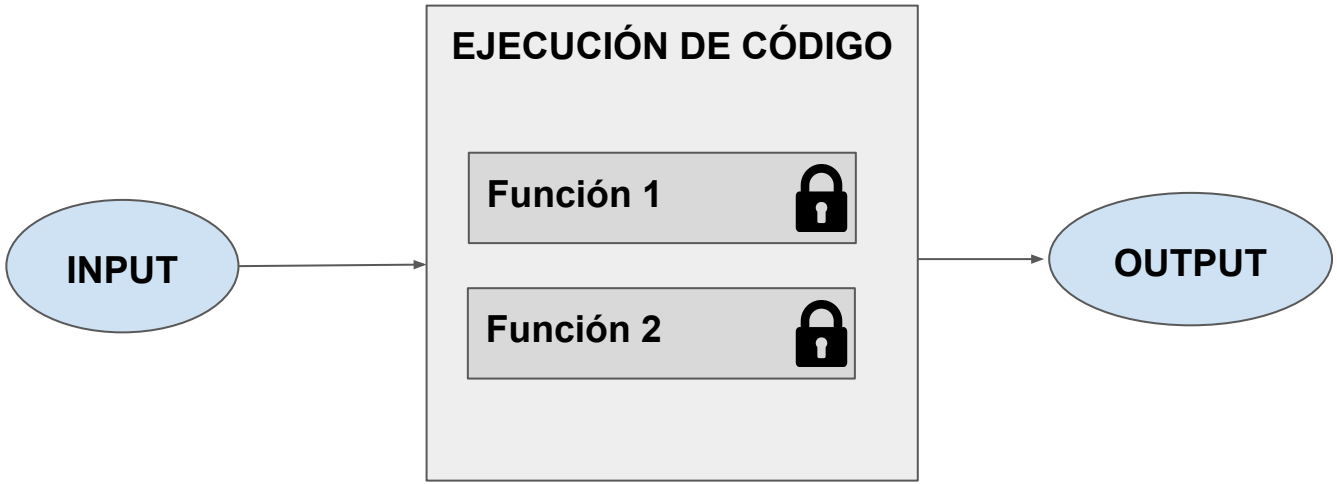
Ventajas del testing / desventajas de su ausencia

- Beneficios (a largo plazo):
  - Verificación de funcionalidad
    - Identificación de bugs
    - Comunicación entre módulos
  - Red de seguridad ante cambios
  - Documentación
- Desventajas de su ausencia:
  - Gerencia y clientes que no ven la importancia del testing
  - Aumento en el costo de mantenimiento
    - Costo de corrección de bugs
    - Costo de agregar nuevas funcionalidades

Diferentes tipos de testing

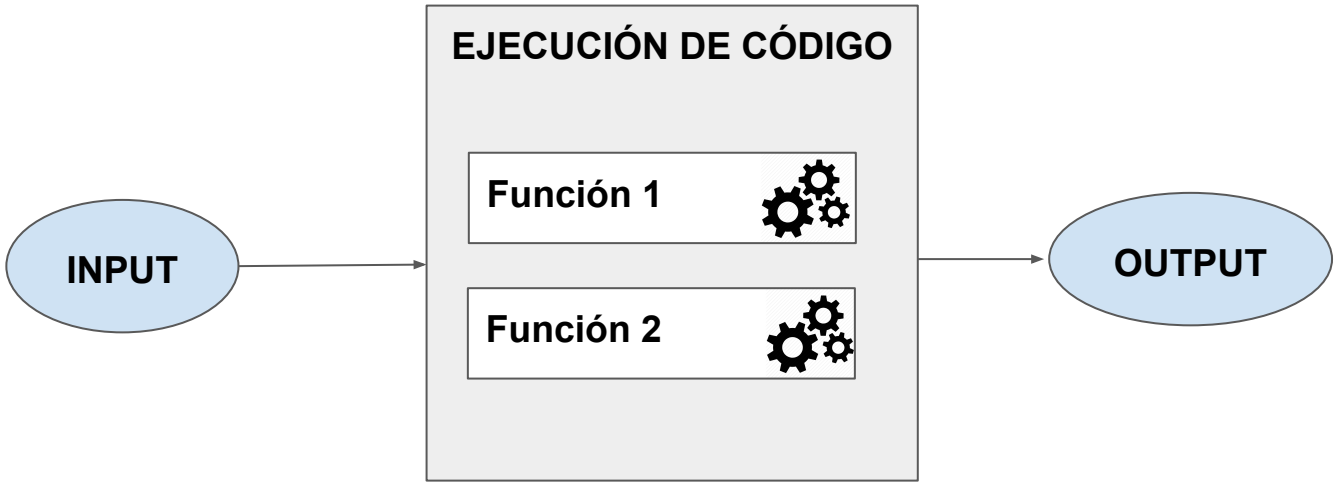
Test de caja negra vs caja blanca

- Tests de caja negra:
  - No importa la implementación
  - Sólo interesan inputs y outputs
  - Casos bordes del dominio



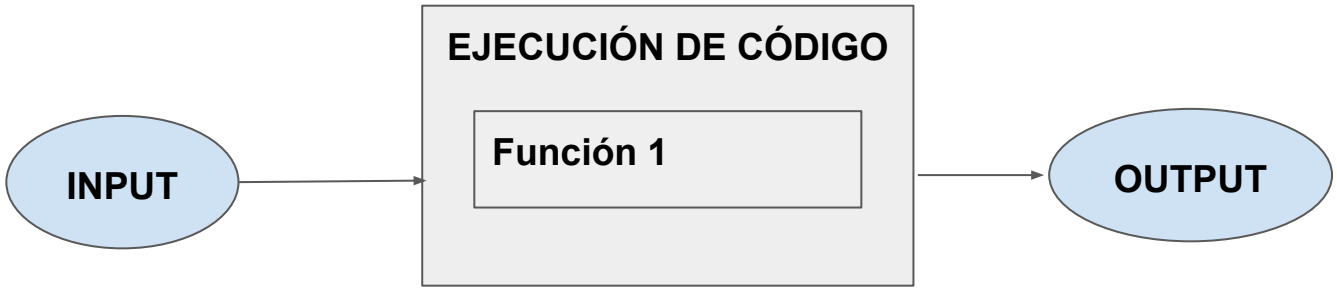
Test de caja negra vs caja blanca

- Tests de caja blanca:
  - Se pone a prueba la implementación
  - Interesa conocer el comportamiento del código ante un input determinado
  - Casos bordes de la implementación



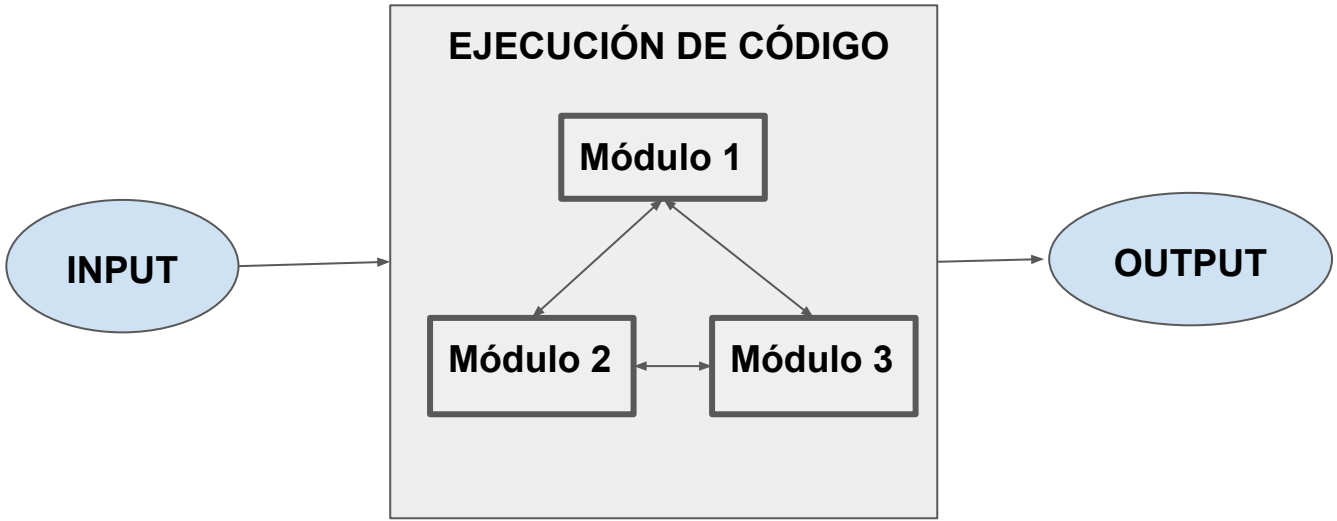
Tests unitarios vs de integración

- Tests unitarios:
  - Se pone a prueba una única función o una pequeña funcionalidad
  - Facilitan el debuggeo de funciones individuales
  - Pequeños indicadores del funcionamiento del código



Tests unitarios vs de integración

- Tests de integración:
  - Se pone a prueba la interacción entre varios módulos
  - Facilitan el debuggeo entre módulos
  - Indican cómo se comporta la aplicación en conjunto



Tests manuales vs automáticas

- Tests manuales:
  - Ejecutadas manualmente
  - Preferible para casos difíciles de automatizar
  - Lentas y propensas a errores

Tests manuales vs automáticas

- Tests automáticas:
  - Las pruebas se escriben en código para que se ejecuten de forma automática
  - Preferible para casos repetitivos
  - Rápidas y repetibles

Anatomía de una prueba automática

```
5   void prueba1() {
6
7       //Codigo que establece el input de la prueba
8
9       //Codigo que llama al codigo bajo prueba
10
11      //Codigo que verifica que el output sea el correcto
12
13  }
```

Anatomía de una prueba automática

```
44  void DadaUnAprendizConVisionesPasadasYFuturas_CuandoSePidenLasVisionesFuturas_EntoncesSeDevuelveUnConjuntoDeVisionesFuturas() {
45
46      //Arrange / Organizar
47      cuervo_aprendiz_t aprendiz;
48      aprendiz.cantidad_visiones = 3;
49      aprendiz.visiones_adquiridas = (vision_t*) malloc(sizeof(vision_t) * aprendiz.cantidad_visiones);
50      aprendiz.visiones_adquiridas[0].epoca = FUTURO;
51      aprendiz.visiones_adquiridas[1].epoca = PASADO;
52      aprendiz.visiones_adquiridas[2].epoca = FUTURO;
53
54      //Act / Actuar
55      conjunto_visiones_t visiones_futuras = obtener_visiones_futuras(&aprendiz);
56
57      //Assert / Verificar
58      vision_t* visiones_esperadas[] = {aprendiz.visiones_adquiridas,
59                                        aprendiz.visiones_adquiridas + 2};
60      ASSERT_EQUALS("Hay dos visiones futuras", 2, visiones_futuras.cantidad_visiones);
61      ASSERT_VECTOR_EQUALS("Los dos vectores son iguales",visiones_esperadas, visiones_futuras.visiones, visiones_futuras.cantidad_visiones);
62  }
63
```

Buenas prácticas: nombre descriptivo

GIVEN / DADO

WHEN / CUANDO

THEN / ENTONCES

```
44  void DadaUnAprendizConVisionesPasadasYFuturas_CuandoSePidenLasVisionesFuturas_EntoncesSeDevuelveUnConjuntoDeVisionesFuturas() {
45
46      //Arrange / Organizar
47      cuervo_aprendiz_t aprendiz;
48      aprendiz.cantidad_visiones = 3;
49      aprendiz.visiones_adquiridas = (vision_t*) malloc(sizeof(vision_t) * aprendiz.cantidad_visiones);
50      aprendiz.visiones_adquiridas[0].epoca = FUTURO;
51      aprendiz.visiones_adquiridas[1].epoca = PASADO;
52      aprendiz.visiones_adquiridas[2].epoca = FUTURO;
53
54      //Act / Actuar
55      conjunto_visiones_t visiones_futuras = obtener_visiones_futuras(&aprendiz);
56
57      //Assert / Verificar
58      vision_t* visiones_esperadas[] = {aprendiz.visiones_adquiridas,
59                                        aprendiz.visiones_adquiridas + 2};
60      ASSERT_EQUALS("Hay dos visiones futuras", 2, visiones_futuras.cantidad_visiones);
61      ASSERT_VECTOR_EQUALS("Los dos vectores son iguales",visiones_esperadas, visiones_futuras.visiones, visiones_futuras.cantidad_visiones);
62  }
63
```

Buenas prácticas: estructura de la test

GIVEN / DADO

WHEN / CUANDO

THEN / ENTONCES

```
44  void DadaUnAprendizConVisionesPasadasYFuturas_CuandoSePidenLasVisionesFuturas_EntoncesSeDevuelveUnConjuntoDeVisionesFuturas() {
45
46      //Arrange / Organizar
47      cuervo_aprendiz_t aprendiz;
48      aprendiz.cantidad_visiones = 3;
49      aprendiz.visiones_adquiridas = (vision_t*) malloc(sizeof(vision_t) * aprendiz.cantidad_visiones);
50      aprendiz.visiones_adquiridas[0].epoca = FUTURO;
51      aprendiz.visiones_adquiridas[1].epoca = PASADO;
52      aprendiz.visiones_adquiridas[2].epoca = FUTURO;
53
54      //Act / Actuar
55      conjunto_visiones_t visiones_futuras = obtener_visiones_futuras(&aprendiz);
56
57      //Assert / Verificar
58      vision_t* visiones_esperadas[] = {aprendiz.visiones_adquiridas,
59                                        aprendiz.visiones_adquiridas + 2};
60      ASSERT_EQUALS("Hay dos visiones futuras", 2, visiones_futuras.cantidad_visiones);
61      ASSERT_VECTOR_EQUALS("Los dos vectores son iguales",visiones_esperadas, visiones_futuras.visiones, visiones_futuras.cantidad_visiones);
62  }
63
```

Reporte de la ejecución:

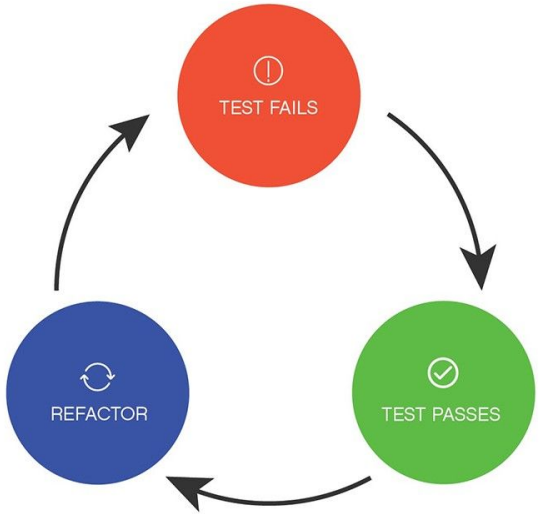
```
44  void DadaUnAprendizConVisionesPasadasYFuturas_CuandoSePidenLasVisionesFuturas_EntoncesSeDevuelveUnConjuntoDeVisionesFuturas() {
45
46      //Arrange / Organizar
47      cuervo_aprendiz_t aprendiz;
48      aprendiz.cantidad_visiones = 3;
49      aprendiz.visiones_adquiridas = (vision_t*) malloc(sizeof(vision_t) * aprendiz.cantidad_visiones);
50      aprendiz.visiones_adquiridas[0].epoca = FUTURO;
51      aprendiz.visiones_adquiridas[1].epoca = PASADO;
52      aprendiz.visiones_adquiridas[2].epoca = FUTURO;
53
54      //Act / Actuar
55      conjunto_visiones_t visiones_futuras = obtener_visiones_futuras(&aprendiz);
56
57      //Assert / Verificar
58      vision_t* visiones_esperadas[] = {aprendiz.visiones_adquiridas,
59                                        aprendiz.visiones_adquiridas + 2};
60      ASSERT_EQUALS("Hay dos visiones futuras", 2, visiones_futuras.cantidad_visiones);
61      ASSERT_VECTOR_EQUALS("Los dos vectores son iguales",visiones_esperadas, visiones_futuras.visiones, visiones_futuras.cantidad_visiones);
62  }
63
```

```
Juanma@Juanma-PC /cygdrive/c/Users/Juanma/Downloads/Algoritmos II 2C 2019/Testing
$ ./testing.exe
TEST: "Hay dos visiones futuras" PASSED
TEST: "Los dos vectores son iguales" PASSED.
```

El testing en las metodologías de trabajo

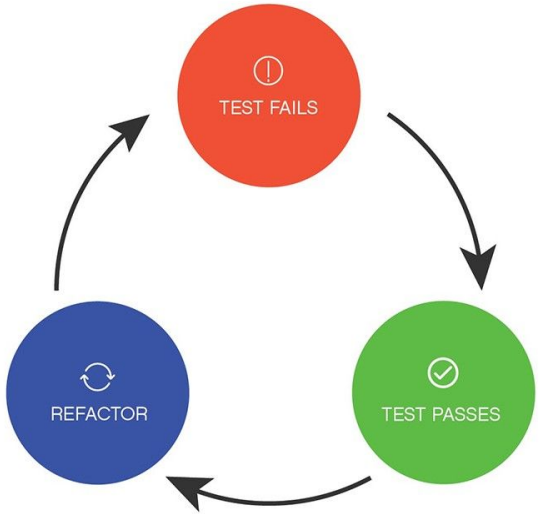
TDD: Test Driven Development

- Desarrollo guiado por las pruebas
  - Escribir una test ANTES que el código de la aplicación, la test falla
  - Implementar el mínimo código necesario para que esa test (y las anteriores) pasen
  - Refactorizar el código para que cumpla con las buenas prácticas de programación



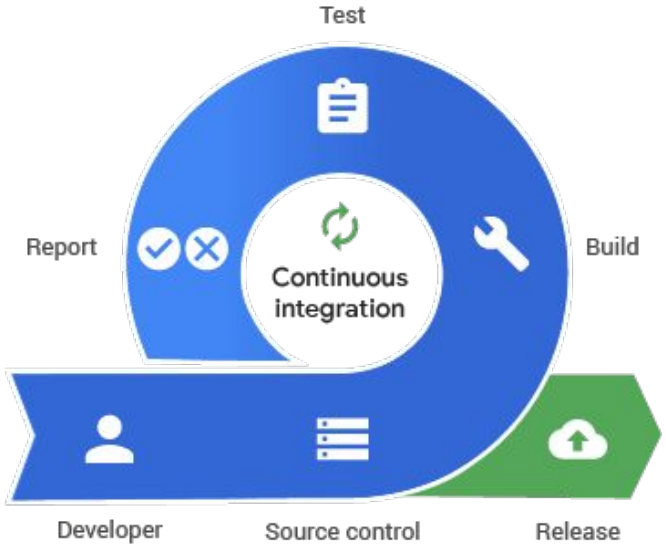
TDD: Test Driven Development

- Ventajas:
  - Código escrito pensando en cómo se va a usar
  - Código escrito está cubierto por pruebas
  - Código bien modularizado gracias a la refactorización



Tests y Continous Integration

- Integración Continua:
  - Metodología para sincronizar el trabajo entre varias personas
  - Seguridad al fusionar los cambios
  - Seguridad al trabajar sobre el código de otra persona



Repaso:

- Qué es el testing y su importancia
- Ventajas del testing y desventajas de su ausencia
- Diferentes tipos de testing
- Anatomía de una test automática
- Testing en las metodologías de trabajo