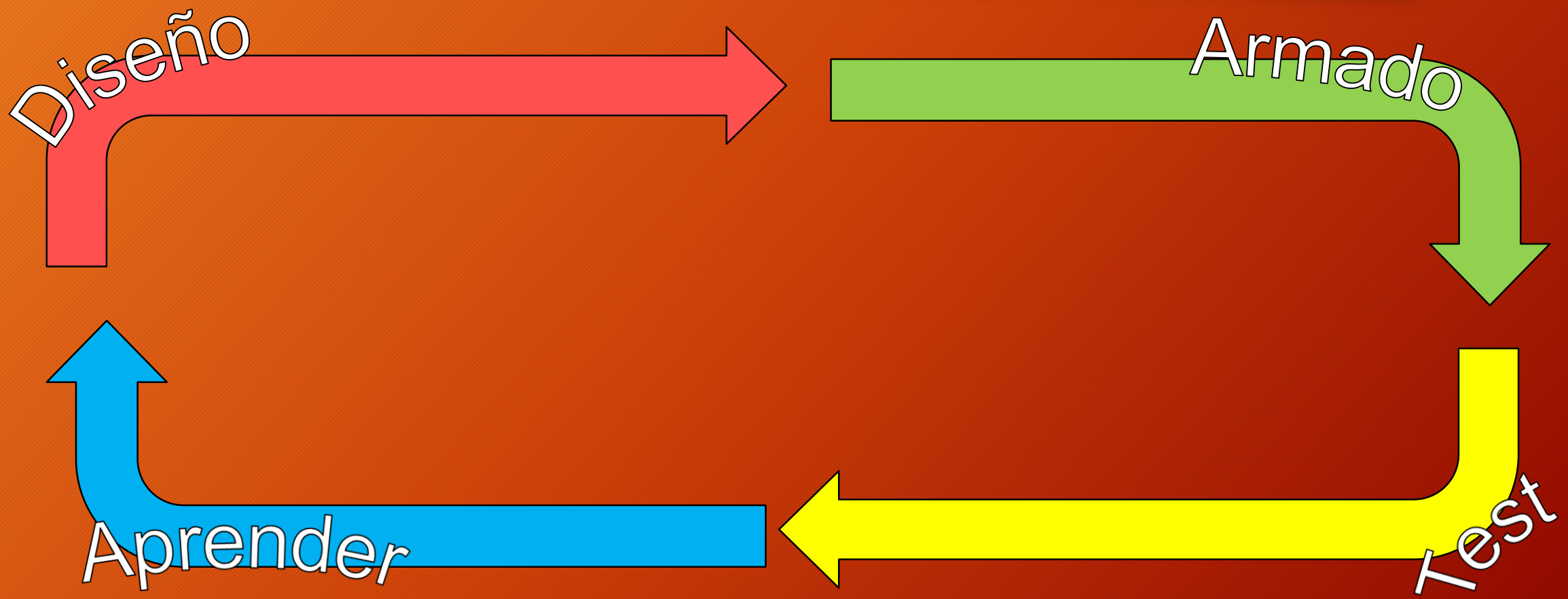
Test Unitarios

16

Programación II y Laboratorio de Computación II

Edición 2018

Ciclo de vida de los sistemas



Test Unitarios

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

- Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

Pruebas Integrales

- Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo.

Pruebas funcionales

- Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.

- Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.

Implementación de Test Unitarios en C#

Nuevo proyecto

- Para poder realizar un test unitario dentro del entorno de Visual Studio debemos generar un nuevo proyecto:
  - Click derecho sobre la solución, Agregar, Nuevo
  - En el menú lateral izquierdo seleccionar Test
  - El proyecto es del tipo Test Unitario
  - Darle un nombre y aceptar el cuadro de dialogo

Patrón AAA

- El patrón AAA (Arrange, Act, Assert) es una forma habitual de escribir pruebas unitarias para un método en pruebas.
  - La sección Arrange de un método de prueba unitaria inicializa objetos y establece el valor de los datos que se pasa al método en pruebas.
  - La sección Act invoca al método en pruebas con los parámetros organizados.
  - La sección Assert comprueba si la acción del método en pruebas se comporta de la forma prevista.

Patrón AAA

```
[TestMethod]
public void Withdraw_ValidAmount_ChangesBalance()
{
    // arrange
    double currentBalance = 10.0;
    double withdrawal = 1.0;
    double expected = 9.0;
    var account = new CheckingAccount("JohnDoe", currentBalance);
    // act
    account.Withdraw(withdrawal);
    double actual = account.Balance;
    // assert
    Assert.AreEqual(expected, actual);
}
```

Patrón AAA

- Clase Assert:
  - Explícita para determinar si el método de prueba se supera o no.
  - Cumple su tarea a través de métodos estáticos.
  - Estos métodos analizan una condición True – False.

Patrón AAA

- El Assert también puede ser manejado desde los atributos o etiquetas del método

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Withdraw_AmountMoreThanBalance_Throws()
{
    // arrange
    var account = new CheckingAccount("John Doe", 10.0);
    // act
    account.Withdraw(20.0);
    // assert es manejado en el ExpectedException
}
```

Patrón AAA

- A través de estos atributos también se puede manejar un tiempo máximo para la prueba.

```
[TestMethod]
[Timeout(2000)]  // Milliseconds
public void UnTest()
{
    //...
}
[TestMethod]
[Timeout(TestTimeout.Infinite)]  // Milliseconds
public void OtroTest()
{
    //...
}
```

Ejecución de las Pruebas

- Abrir la ventana del Explorador de Pruebas:
  - Ir al menú Pruebas / Ventanas / Explorador de Pruebas
- Ejecutar Una prueba:
  - Click derecho sobre la prueba deseada
  - Ejecutar
- Ejecutar todas las pruebas juntas:
  - Ejecutar Todas...