Introducción a TDD

La base de la visión moderna del software

FEB-2017 © mvelosop@gmail.com

¿Quién soy?

- Apasionado de la tecnología y el desarrollo de software
- Desarrollando software desde hace 34 años (1983)
- Graduado de Ingeniería Química hace 33 años (1984)
- Co-propietario de una empresa de software durante 21 años
- Seis años trabajando en desarrollo web con C# y .NET
- Tres años como consultor independiente en desarrollo web
- Un año en Tenerife

@mvelosop; https://github.com/mvelosop; mvelosop@gmail.com

Objetivos de hoy

1. Entender el concepto general y la aplicación del TDD

2. Conocer la visión moderna del software (DevOps)

3. Conocer la importancia del TDD en DevOps

Actividades de hoy

1. Presentar los conceptos básicos de TDD
2. Demostrar la aplicación de TDD en un ejercicio práctico
3. Conversar sobre la aplicación de TDD en la vida real
4. Presentar DevOps

TDD – Conceptos

Test Driven Development

TDD es una metodología de desarrollo

No es una metodología de pruebas

¿Cómo se hace TDD?

Define el objetivo final

Debes saber a dónde quieres llegar antes de comenzar

Identifica pasos pequeños

Comienza por lo más sencillo

Sólo un poco más complejo en cada paso

Avanza paso a paso

Primero la prueba que debe fallar - **RED**

Después el mínimo código necesario para pasar la prueba – **GREEN**

Siguen pasando todas las pruebas anteriores

Mejora en cada paso

Refactoriza para eliminar duplicaciones

Refactoriza para facilitar el mantenimiento

Todas las pruebas deben seguir pasando - **GREEN**

Resumen

1. Definir objetivo final
2. Identificar resultados intermedios
3. Lograr resultados intermedios
   - **RED** – Primero una prueba que falla
   - **GREEN** – El mínimo código necesario para que la prueba pase
4. Mejorar código
   - **REFACTOR** – Para facilitar mantenimiento

- Repetir 2-4 hasta lograr objetivo final

TDD – Demo

StringCalculator by Roy Osherove
http://osherove.com/tdd-kata-1/

Demo – StringCalculator

1. Crear una calculadora simple de strings, con el método

   Int Add(string numbers)

   que retorne la suma de los valores enteros que reciba en un string, con 0, 1 o 2 números, no es necesario validar que sean enteros.

   1. El resultado para la entrada “” debe ser 0.
   2. El resultado para la entrada “1” debe ser 1.
   3. El resultado para la entrada “1, 2” debe ser 3.

Convenciones Recomendadas

```
<Proyecto>.Tests

public class <Clase>_Tests
{
	[Fact]
	public void <Método>_<Condiciones>_<Resultado>()
	{
		// Arrange
		// Act
		// Assert
	}
}
```

Demo – StringCalculator

2. Aceptar cualquier cantidad de valores en la entrada

   Int Add(string numbers)

   Debe manejar una cantidad indeterminada de enteros separados por coma, no es necesario validar que sean enteros.

   - El resultado para la entrada “1, 2, 3” debe ser 6.

Demo – StringCalculator

3. Aceptar un separador adicional

   Int Add(string numbers)

   Debe manejar “newlines” (\n) para separar los números, además de comas.

   - Esta es una entrada válida: “1\n2,3” y debe retornar 6.
   - Esta NO es una entrada válida: “1,\n2” (no es necesario verificarlo)

Demo – StringCalculator

4. Aceptar cualquier separador

   Int Add(string numbers)

   Debe manejar cualquier separador que se indique en la entrada, con una línea inicial y opcional, con el formato:
   "//[separador]\n[números...]"

   - La entrada "//;\n1;2" debe retornar 3
   - Debe seguir manejando todos los casos anteriores

Demo – StringCalculator

5. Validaciones

   Int Add(string numbers)

   No debe permitir valores negativos y debe levantar una excepción con la lista de valores negativos.

   - La entrada “1, 2, -3, -4” debe indicar “negatives not allowed: -3, -4”

Demo - Comentarios

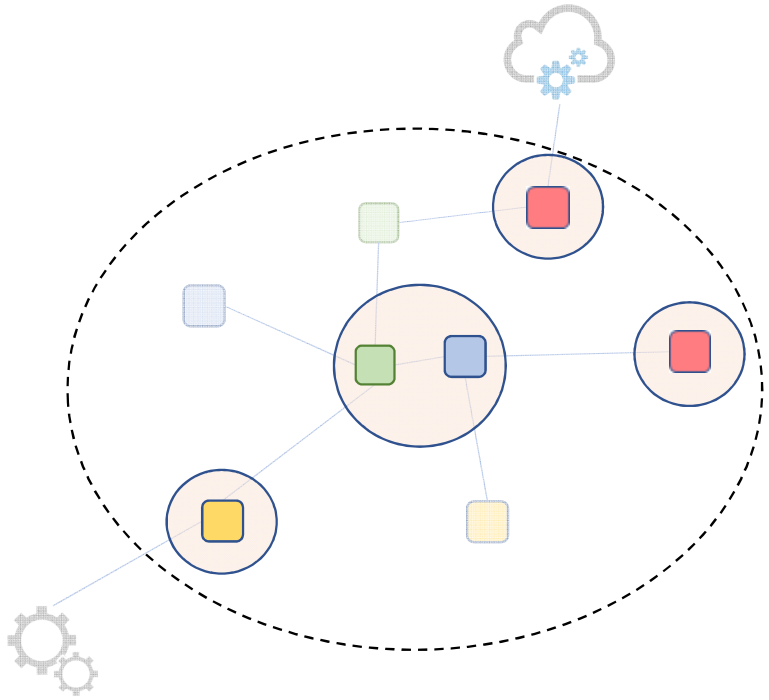
- Estamos seguros de cumplir los requerimientos
- Es fácil agregar complejidad poco a poco
- Es cómodo poder regresar a la última versión correcta
- Las pruebas sirven como documentación

Meta-aprendizaje

- Esta estrategia tiene muchas otras aplicaciones

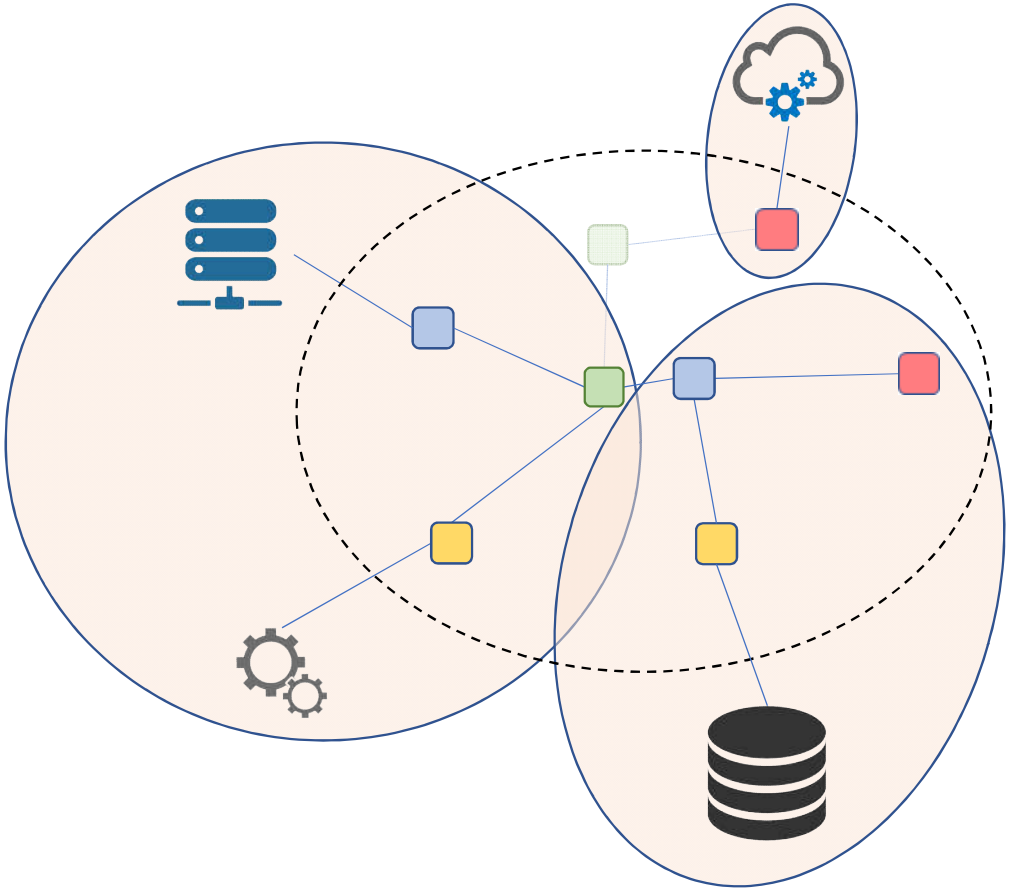
TDD - Alcance

TDD – Pruebas Unitarias



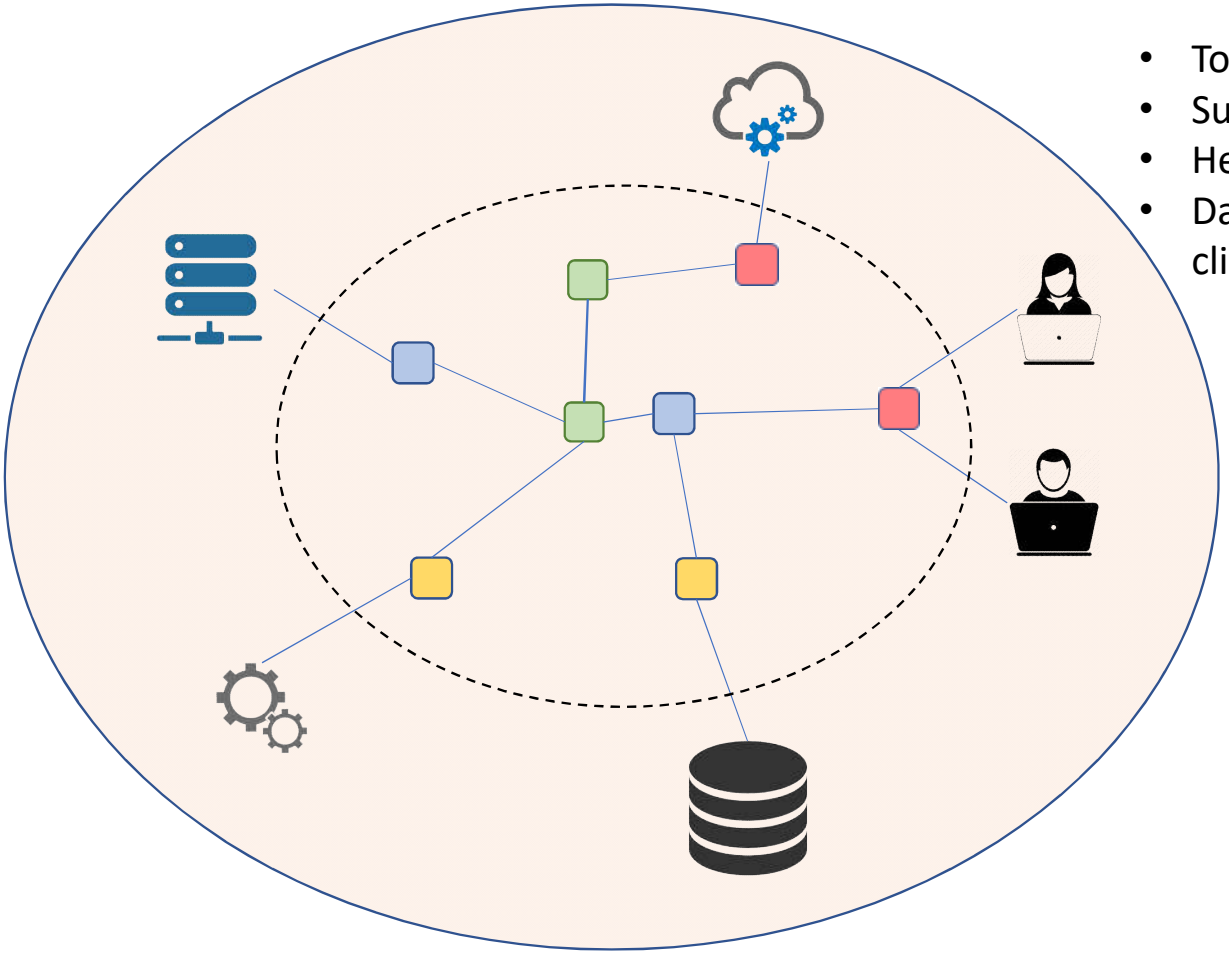
- Uno o varios componentes
- No se incluyen componentes externos
- Maquetas (mocks) para todo lo demás

TDD – Pruebas de Integración



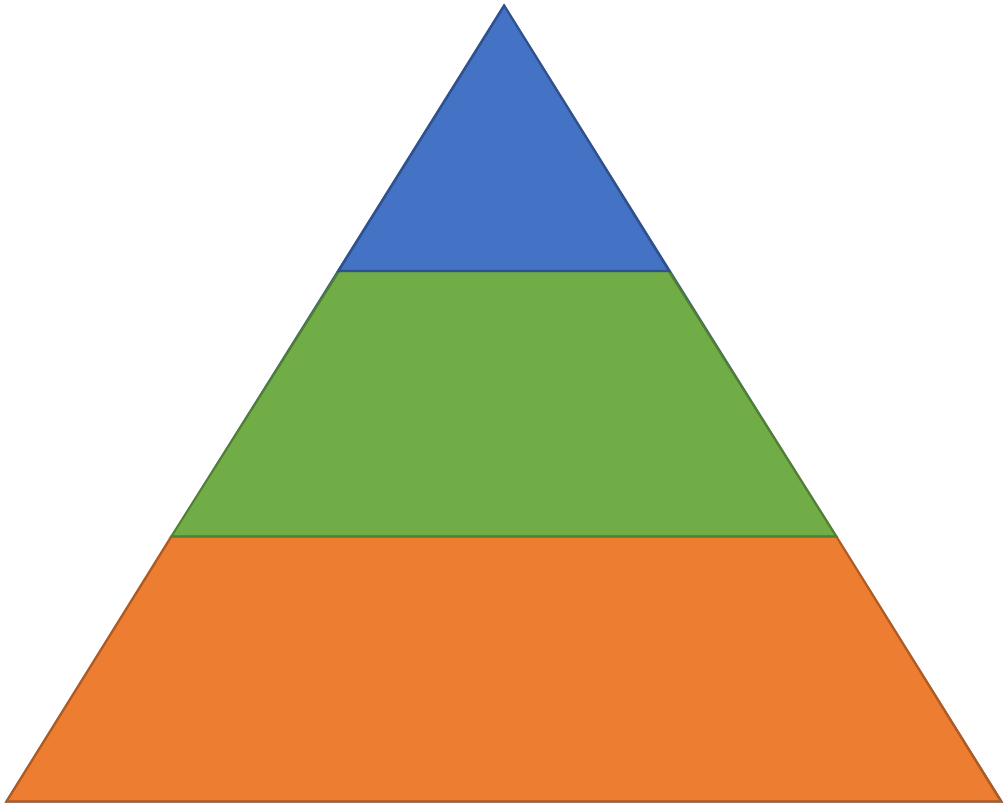
- Uno o varios componentes
- Se incluyen componentes externos
- Herramientas BDD

TDD – Pruebas de Aceptación



- Todo el sistema
- Suele incluir Interfaz de Usuario
- Herramientas BDD
- Datos de prueba suministrados por el cliente

TDD – Distribución



Pruebas de interfaz de usuario

Pruebas de integración

Pruebas unitarias

TDD en la vida real

Mi experiencia particular...

TDD es una herramienta

No es una religión

El objetivo conseguir resultados

No es hacer pruebas

A veces no conocemos el resultado esperado hasta que hacemos el programa

Y entonces no se puede aplicar TDD

¿Se debe probar “*TODO*”?

Sólo se verifican resultados observables

No se verifica el estado interno de las clases

Las pruebas también son programas que hay que mantener

Compromiso entre riesgo, costo y beneficio

Mi receta

Contexto: Modelo estilo DDD, con código generando usando MDA

1. Definir la historia de usuario
2. Diseñar escenarios para verificar funcionamiento
   (BDD, integración backend)
   - Primero escenarios “felices”
   - Después escenarios de error
3. Implementar componentes complejos
   (TDD, unitarias)
4. Implementar el resto de la funcionalidad
   (BDD, integración backend)

¿Comentarios?

DevOps

La visión moderna del software

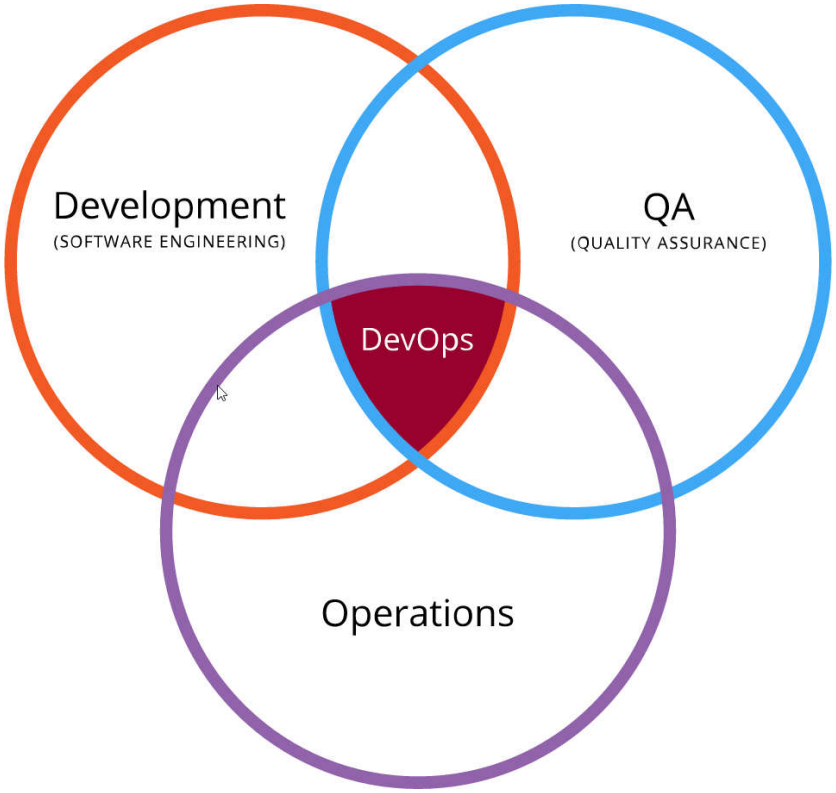
FEB-2017 © mvelosop@gmail.com

El software es un componente para implementar los cambios que necesita la organización

El objetivo final es mucho más que producir software

Es un cambio cultural...
Necesita colaboración entre desarrollo, calidad y operaciones

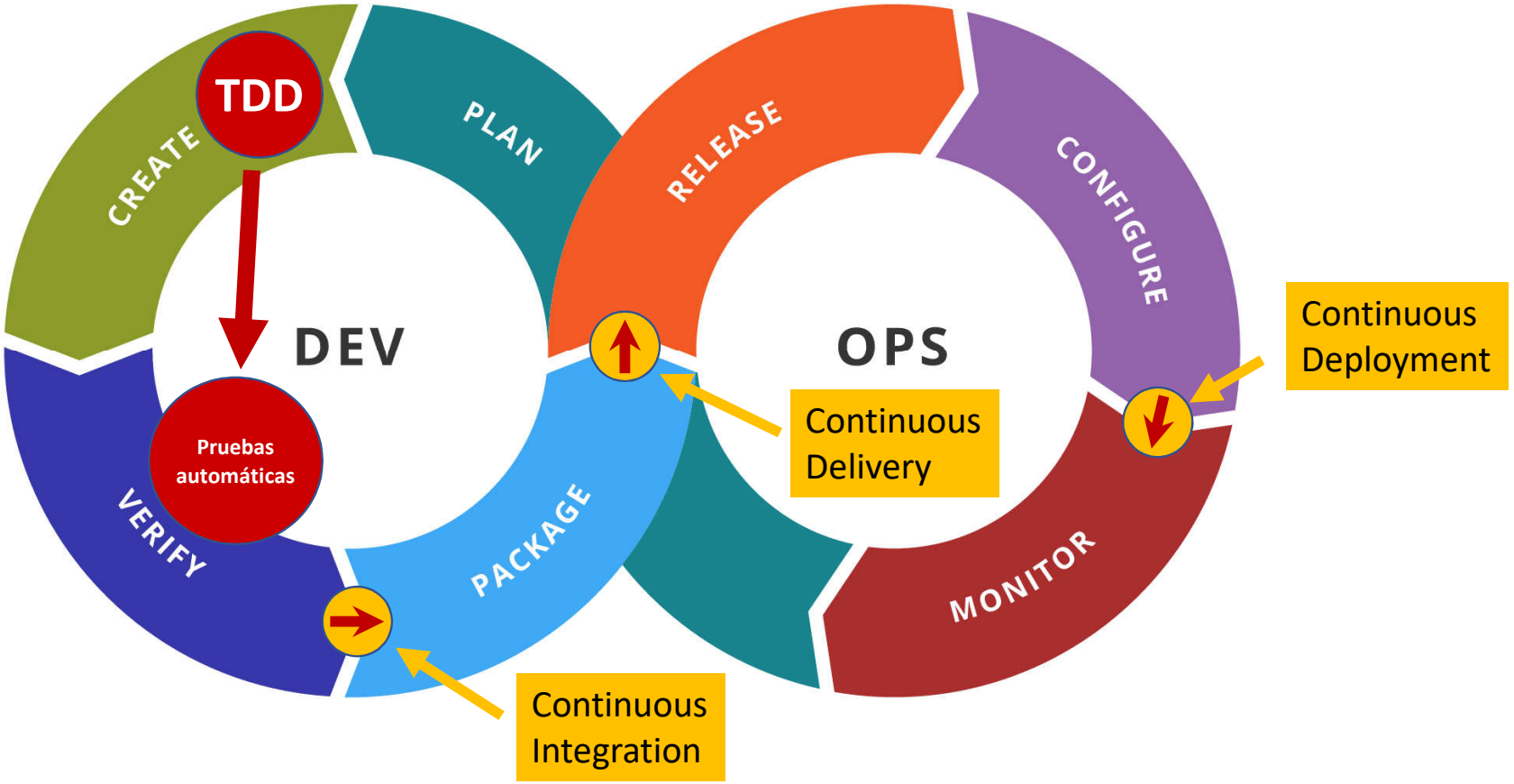
DevOps – La visión moderna del software



DevOps es un proceso de cambio continuo de la organización

Que utiliza el software como herramienta

DevOps – La visión moderna del software



Herramientas

Frameworks de pruebas

- C#: NUnit, xUnit.net, SpecFlow (BDD)
- Java: JUnit, JBehave (BDD)
- Javascript: Mocha, Karma, Protractor, Jasmine (BDD)

Mocking frameworks

- C#: Moq, NSubstitute, MS Fakes
- Java: Mockito, PowerMock
- Javascript: JsMockito, Jasmine

Integración con navegadores

- Selenium (C#, Java, Javascript, etc.)

Gracias...

@mvelosop; https://github.com/mvelosop; mvelosop@gmail.com