

# conferencia Agile-Spain2010

MADRID 10 - 11 Junio

Haciendo realidad la agilidad



© flioukas



## Behavior Driven Development Aplicado en Acceptance Test

Christian Ramírez Arévalo  
**certum**



# Agenda

- Introducción
- Behavior Driven Development
- El testing y el BDD
- Herramientas
- El BDD y el usuario
- Beneficios reales

# TDD

- Es de las primeras piedras del Agile Development
- Surge TDD como la esperanza para resolver huecos en el diseño
- No es un método de pruebas, es una forma de diseño
- No es fácil de asimilar el concepto y menos de transmitir, en la vida real
- Confusion!!! tener pruebas unitarias no es hacer TDD

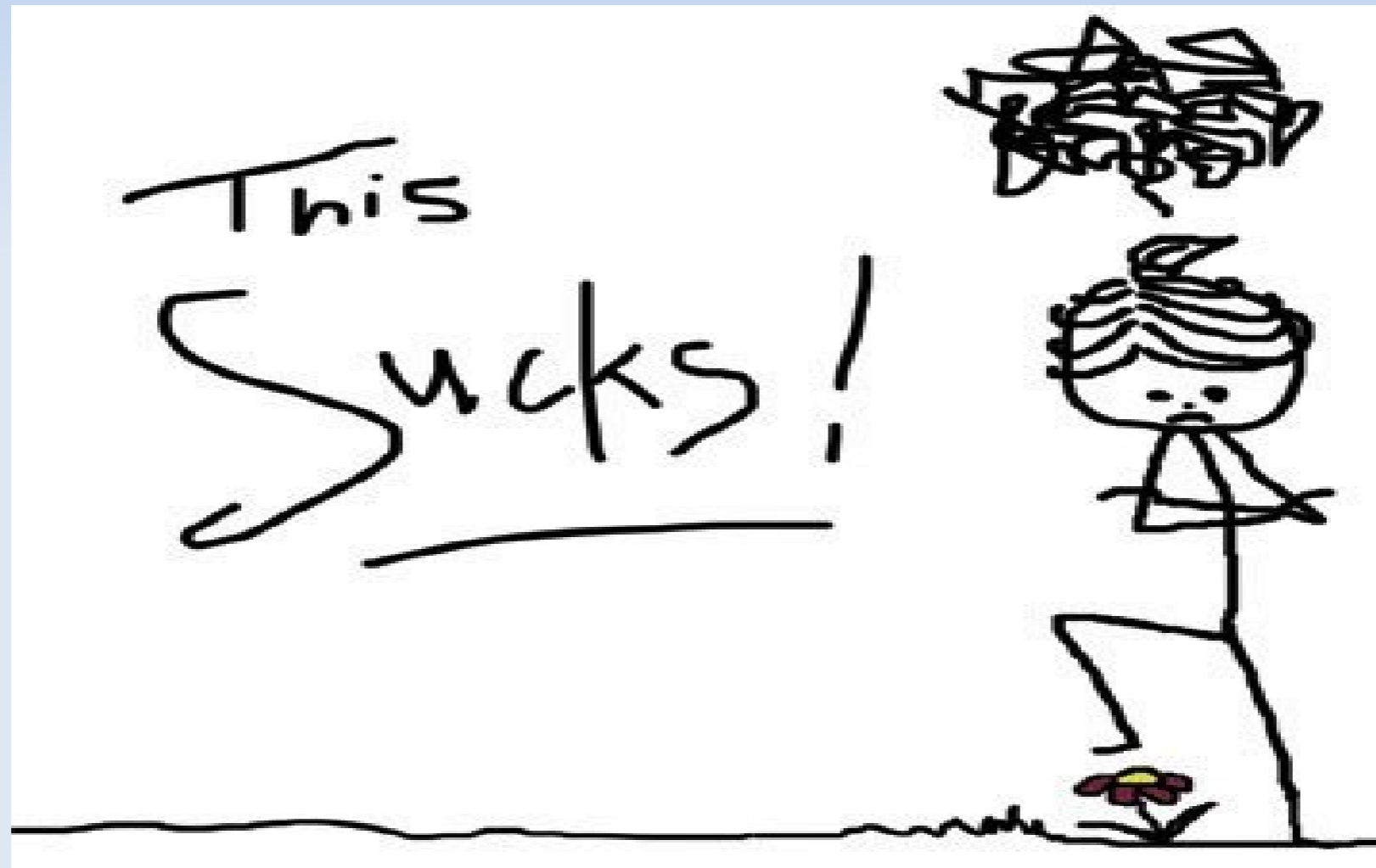
# TDD y el mundo

- Para el usuario de negocio es como hablar con él en CHINO!!
- Para la gente de QA no tiene mucho valor
- Se enfoca en los metodos no en el comportamiento
- No es reutilizable en el dominio de negocio
- A veces entre desarrolladores no se entiende, o ni siquiera es facil de leer

# ¿Qué hace?

```
public void testAla()
{
    String address = "ala@missiondata.com";
    boolean expectedResult = true;
    boolean actualResult =
Validator.isValidEmailAddress(address);
    assert( "\ntestAla, EXPECTED: " + expectedResult + ", GOT:
" + actualResult,
        (expectedResult==actualResult) );
}
```

# En resumen...



# Vamos un poco mas lejos ...

**TDD → BDD**

# ¿Beer Driven Development?



# Behavior Driven Development

- BDD original por Dan North
- BDD gira entorno al comportamiento del sistema
- Todo tiene comportamiento
- BDD ayuda a incrementar el testability

# Behavior Driven Development

- Utiliza un DSL para plantear el comportamiento de las aplicaciones
- Domain-Specific Language (DSL)
- DSL, entre lenguaje de programación y lenguaje de script
- DSL permite que la solución(requerimiento) sea expresada en el idioma y nivel de abstracción del dominio del problema

# Behavior Driven Development

- La convencion Given/When/Then es la idea central
- Humanos causa → efecto
- Software Engineering Entrada → Proceso → Salida
- Cambia tu forma de pensar
- If/And/Then ¿te parece conocido?

# Behavior Driven Development

- Se puede ver como una FSM
- Matemáticamente, es posible demostrar que esta completo y es consistente nuestro requerimiento
- Mejor DSL, mejor automatización, mejores requerimientos
- Uso de un muy específico y pequeño vocabulario, para evitar malos entendidos

# Behavior Driven Development

- También describe: rol, característica y beneficio
- Utiliza las historias como unidad mínima de funcionalidad y por lo tanto también de entrega
- Las historias deben ser producto de conversaciones
- Ayuda a delimitar el alcance

# Estructura de una historia

Title (one line describing the story)

Narrative:

As a [role]

I want [feature]

So that [benefit]

Acceptance Criteria: (presented as Scenarios)

Scenario 1: Title

Given [context]

And [some more context]...

When [event]

Then [outcome]

And [another outcome]...

# BDD y el testing

- Requerimientos no son ambiguos
- Se define el criterio de aceptación de forma inherente en las historias
- Diseño reutilizable en el dominio
- Documentación de la aplicación
- Visibilidad, solo se puede probar lo que se ve

# BDD y el testing

- Automatización sencilla
- Código de pruebas automatizadas reutilizable
- Mejor feedback en Continuous Testing
- E2E claro para el usuario
- Funciona para web, cliente-servidor y escritorio
- Caso y escenario de pruebas en texto plano → pruebas automatizadas

# BDD y el testing

- Solo hay errores de programación no de entendimiento
- Las pruebas de aceptación se acortan en tiempo
- Pruebas de usuario guiadas
- El usuario puede probar con todos los datos con que siempre soñó hacerlo
- Diseño y código de pruebas automatizadas más fácil de mantener

# BDD y el usuario

- Expertos del dominio pueden por si mismos entender, validar, modificar, incluso hasta escribir los requerimientos en el DSL especificado
- Negocio y TI se refieren al mismo sistema de la misma forma
- Ve exactamente lo que pidio

# Pruebas de aceptación

- Propiedad de los clientes
- Escrito en conjunto con los clientes, desarrolladores y analistas de prueba
- Enfoque en el Qué y no sobre el Cómo
- Expresada en lenguaje de dominio del problema
- Conciso, preciso y sin ambigüedades
  
- BDD se vislumbra como la solución natural

# Pruebas de aceptación

- Aplicación
  - Crear la historia de la característica (Requerimiento)
  - Traducir empleando el DSL especificado
  - Automatizar la prueba funcional
  - Codificar la funcionalidad que pase la prueba

# Herramientas

- Interpretan scripts en lenguaje natural
- Core en la definición de los pasos
- API
- Data-driven mediante tablas

# Herramientas

- Pionera
  - Jbehave
- La más completa
  - Cucumber → Mundo ruby
  - DSL → Gherkin
  - Más de 30 idiomas
  - Funciona con ruby, Java, .Net, Python, etc.
  - Pensada para desarrollo pero la usamos para testing
- Enfocada a testing
  - Twist
  - Thoughtworks
  - Entorno colaborativo
  - Pensada para testing
  - IDE

# Beneficios

- Mejora la calidad, productividad, confiabilidad, mantenibilidad, portabilidad y reutilización
- Validación a nivel de dominio
- Ahorros de tiempo en UAT

# Desventajas

- Proliferación de DSL similares no estandarizados
- Es necesario aprender un nuevo lenguaje
- Limitado en su aplicación, pruebas funcionales
- No es recomendable para pruebas no funcionales
  - Requerimiento ambigüo
  - Technical story
- Equipos TDD renuentes

# ¿Preguntas?

# Contacto

- Ing. Christian Ramírez
- CERTUM, México D.F.
- [www.certum.com](http://www.certum.com)
- Email: [chramirez@certum.com](mailto:chramirez@certum.com)
- Twitter: [@chrix2](https://twitter.com/chrix2)
- Telefono: 52-55-55-57-53-70

