Adicto al verde

@kinisoftware

cas Conferencia Agile Spain 2013 BILBAO 10-11 OCT

theEvnt

tuenti


INFECTIOUS
SUBSTANCE
6

Para empezar

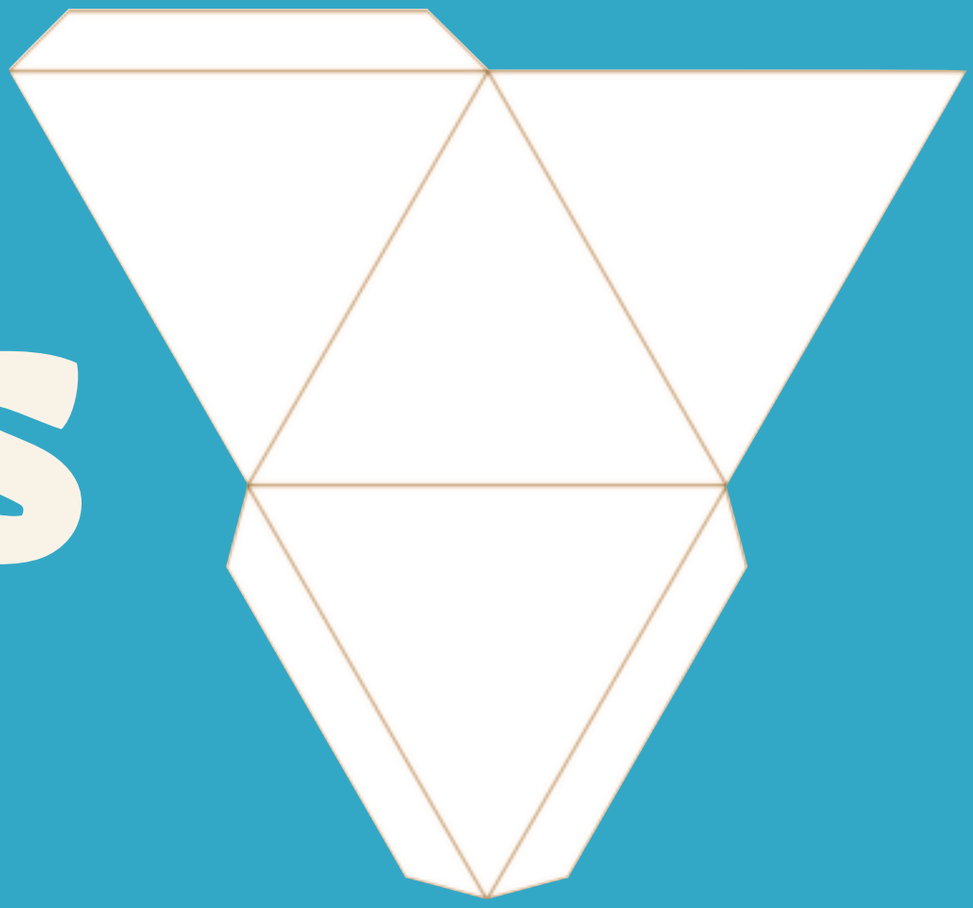
...

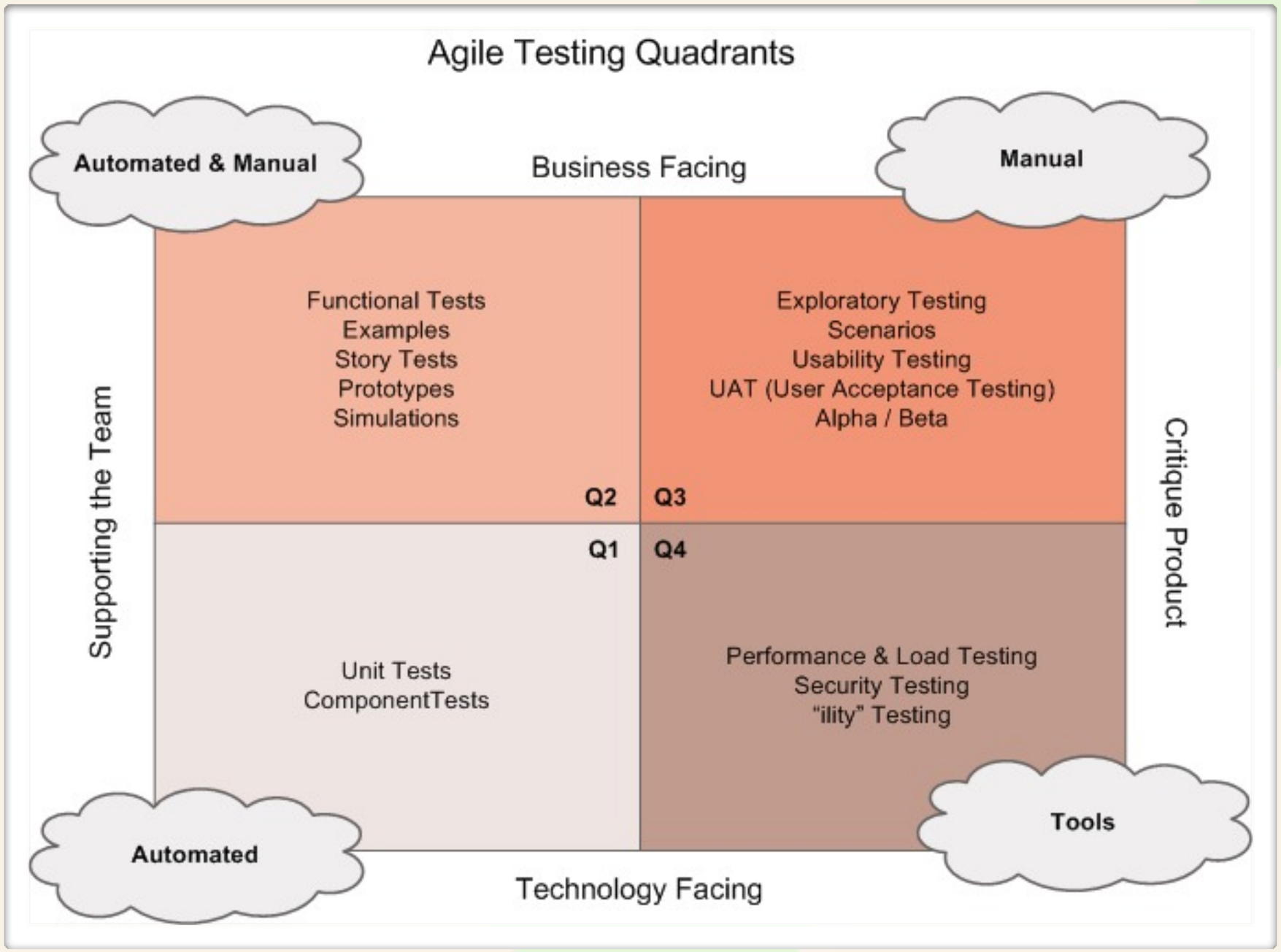
¡un teeeest, un teeeest, un teeeeest!

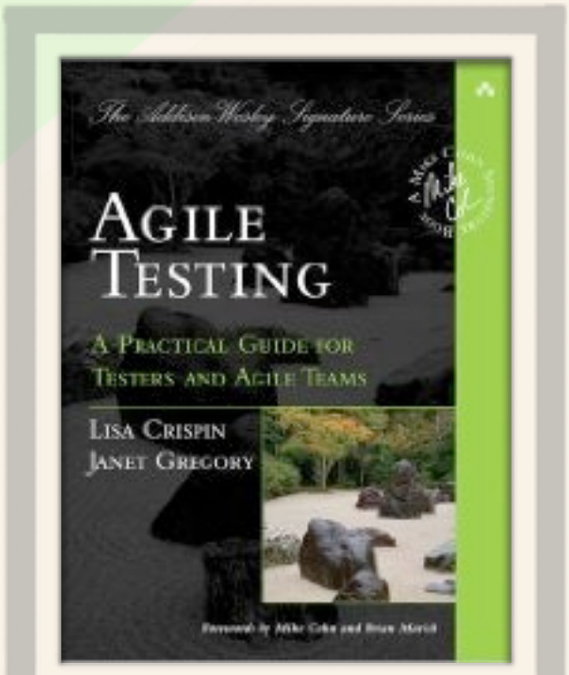
¿Por qué hacer tests?

- Queremos a nuestros usuarios
- Dormir por las noches
- ¿Habré roto algo?
- Pero, si esto antes funcionaba ¿no?

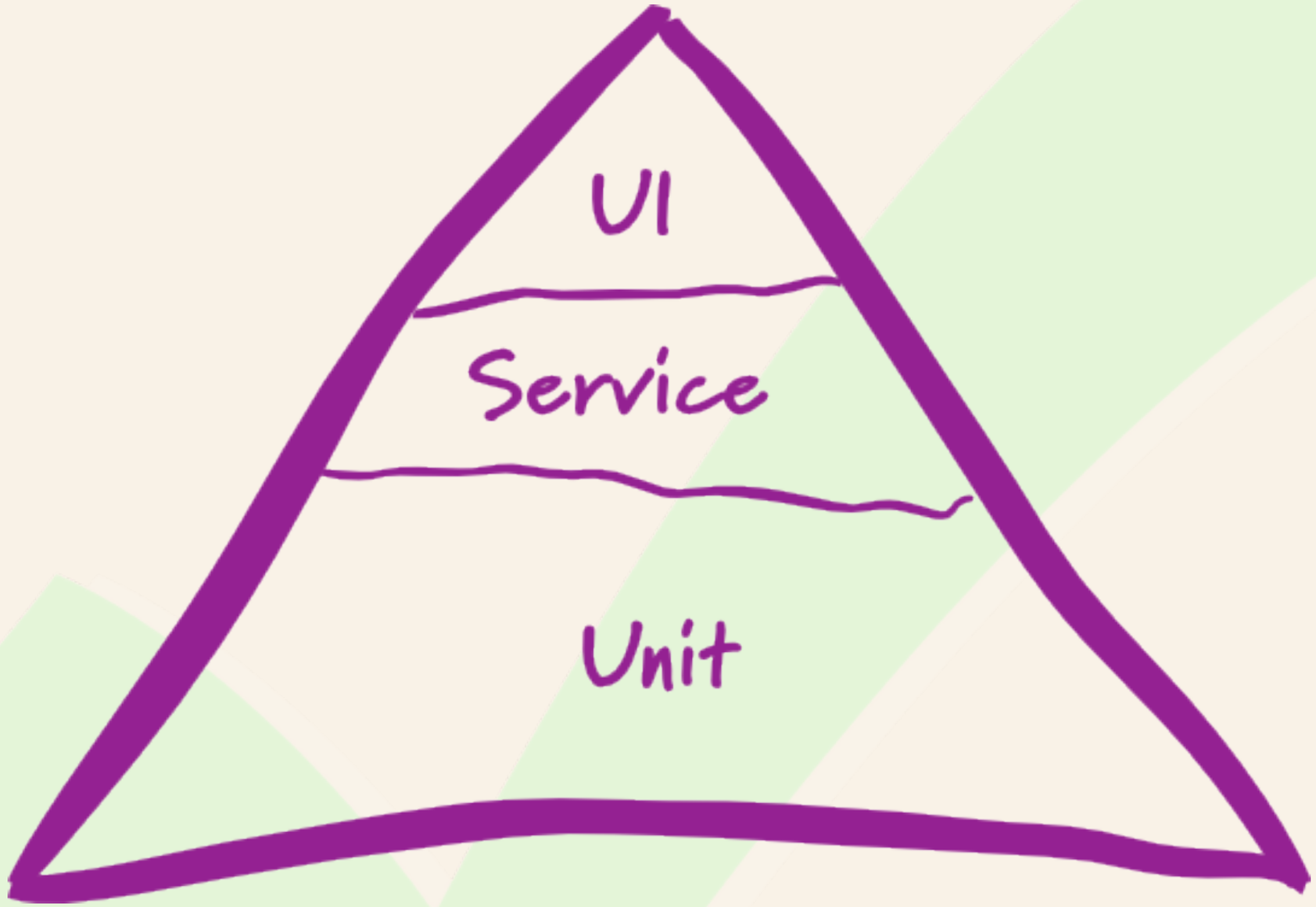
Tipos de Tests







http://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants/



by Martin Fowler

http://martinfowler.com/bliki/TestPyramid.html

Toolbox

- ✓ Repetible
- ✓ Automatizable*
- ✓ Expresivo
- ✓ Fácil de hacer
- ✓ Relevante
- ✓ Self-Verifying

Hamcrest

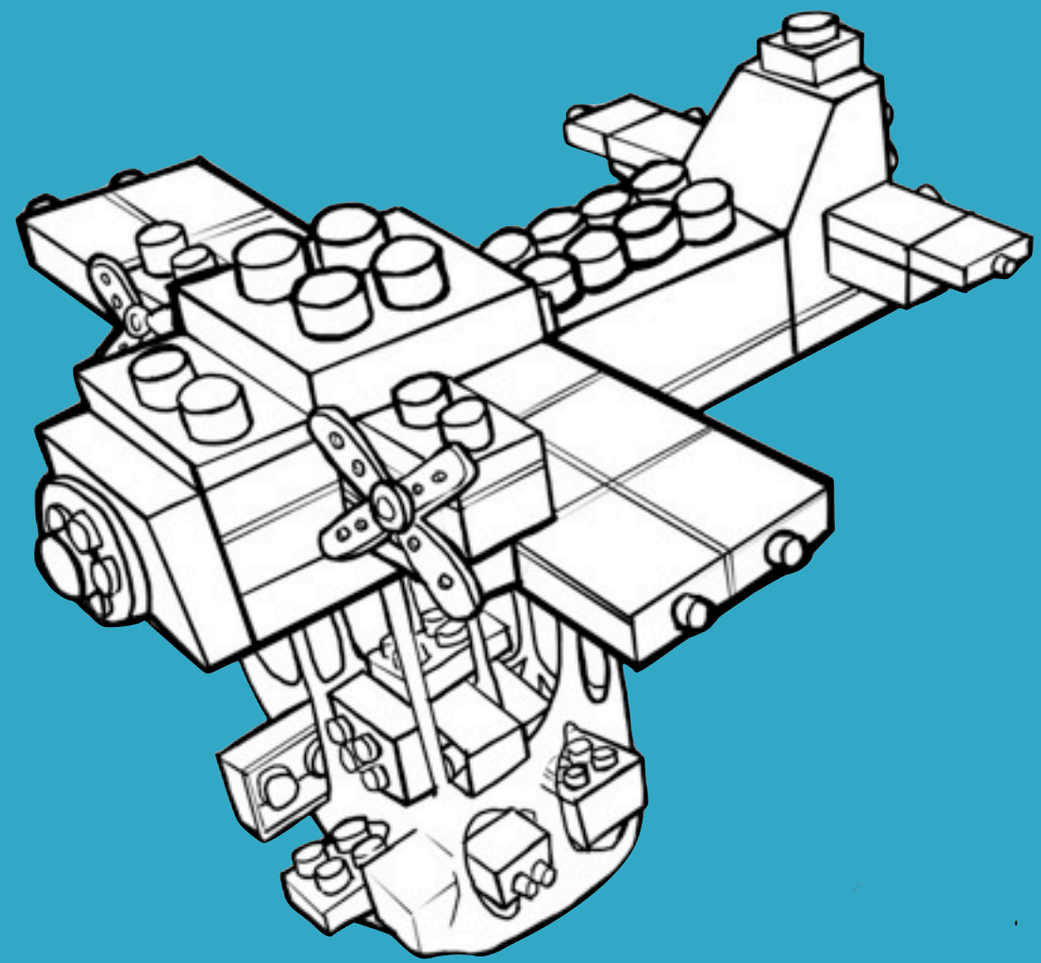
```
assertThat(theBiscuit, equalTo(myBiscuit));
assertThat(theBiscuit, is(equalTo(myBiscuit)));
assertThat(theBiscuit, is(myBiscuit));

assertThat(Math.sqrt(-1), is(notANumber()));

assertThat(myArray, hasItemInArray(item));

assertThat(person.getAge(),
        is(greaterThan(LEGAL_AGE)));
```

Test de Aceptacion



A formal test conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.

Acceptance tests are different from UnitTests in that UnitTests are modeled and written by the developer of each class, while the acceptance test is at least modeled and possibly even written by the customer.

http://c2.com/cgi/wiki?AcceptanceTest

- ✓ Colaboración con el cliente
- ✓ Lenguaje Natural
- ✓ DSL
- ✓ Definir una historia de usuario
- ✓ ¿UI?

Cucumber

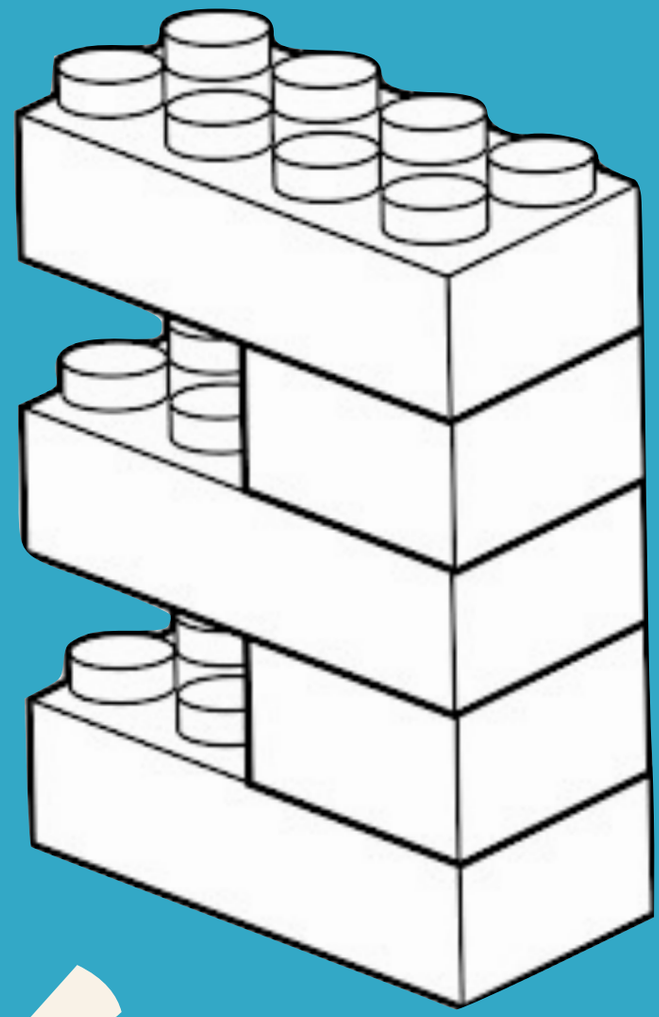
Robotium

http://kinisoftware.com/2012/03/bdd-java-que-framework-uso/

La pregunta de la audiencia:
¿UI vs API/Services?

- ✓ UI
- ✓ Selenium/Webdriver al rescate!
- ✓ Page Objects
- ✓ API

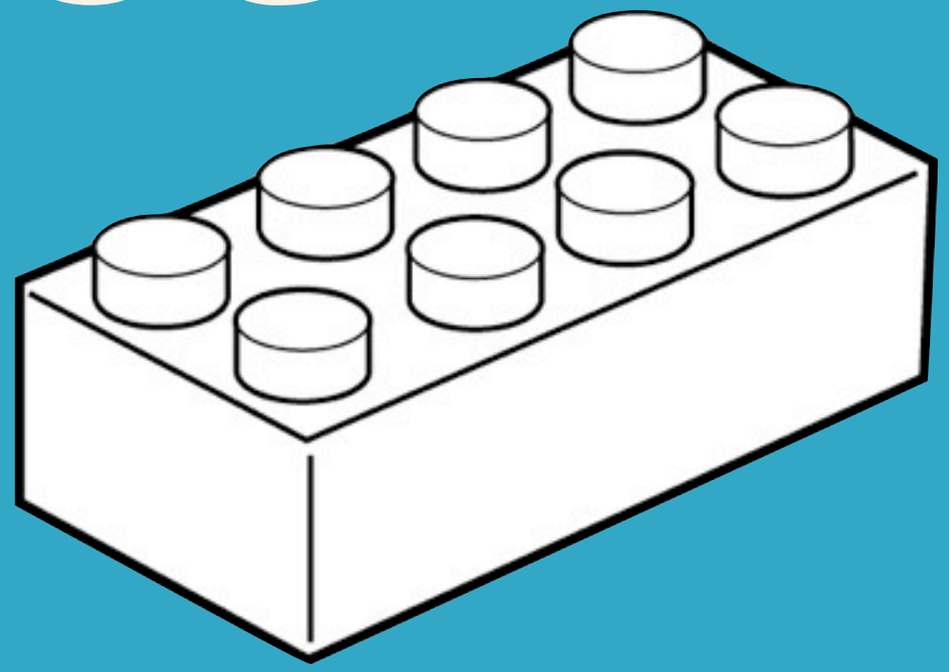
Test de Integración



Un test que combina varias partes de la aplicación de tal forma que buscamos el resultado de la colaboración conjunta. Puede involucrar un escenario “end-to-end” o simplemente varios componentes.

A veces depende del framework que estemos usando...

Test Unitarios



A kind of AutomatedTest, though some would say a better name is DeveloperTest

"Unit" casually refers to low-level test cases written in the same language as the production code, which directly access its objects and members.

Under the strict definition, for QA purposes, the failure of a UnitTest implicates only one unit. You know exactly where to search to find the bug.

http://c2.com/cgi/wiki?UnitTest

F.I.R.S.T.

- ✓ Fast
- ✓ Isolated
- ✓ Repeatable
- ✓ Self-Verifing
- ✓ Timely/Transparent

http://pragprog.com/magazines/2012-01/unit-tests-are-first

Toolbox

Dobles

- ✓ **Stubs** - Respuestas programadas
- ✓ **Mocks** - Expectaciones (Stubs+)
- ✓ **Spies** - Guardan información
- ✓ **Fake** - Similar al recurso de producción
- ✓ **Dummy** - No hace nada at all :)

EasyMock

jMock

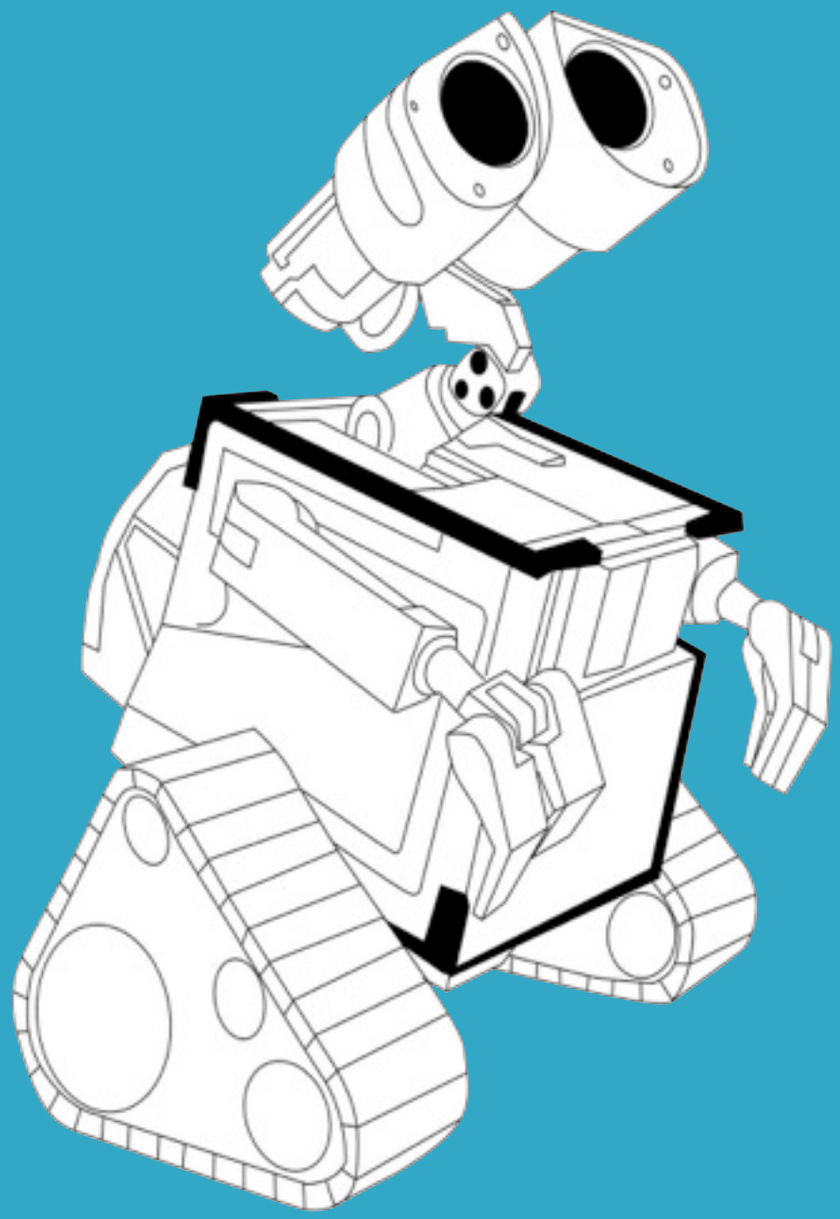
PowerMock

jetty://

REST-assured

ant

Automatización (CI)



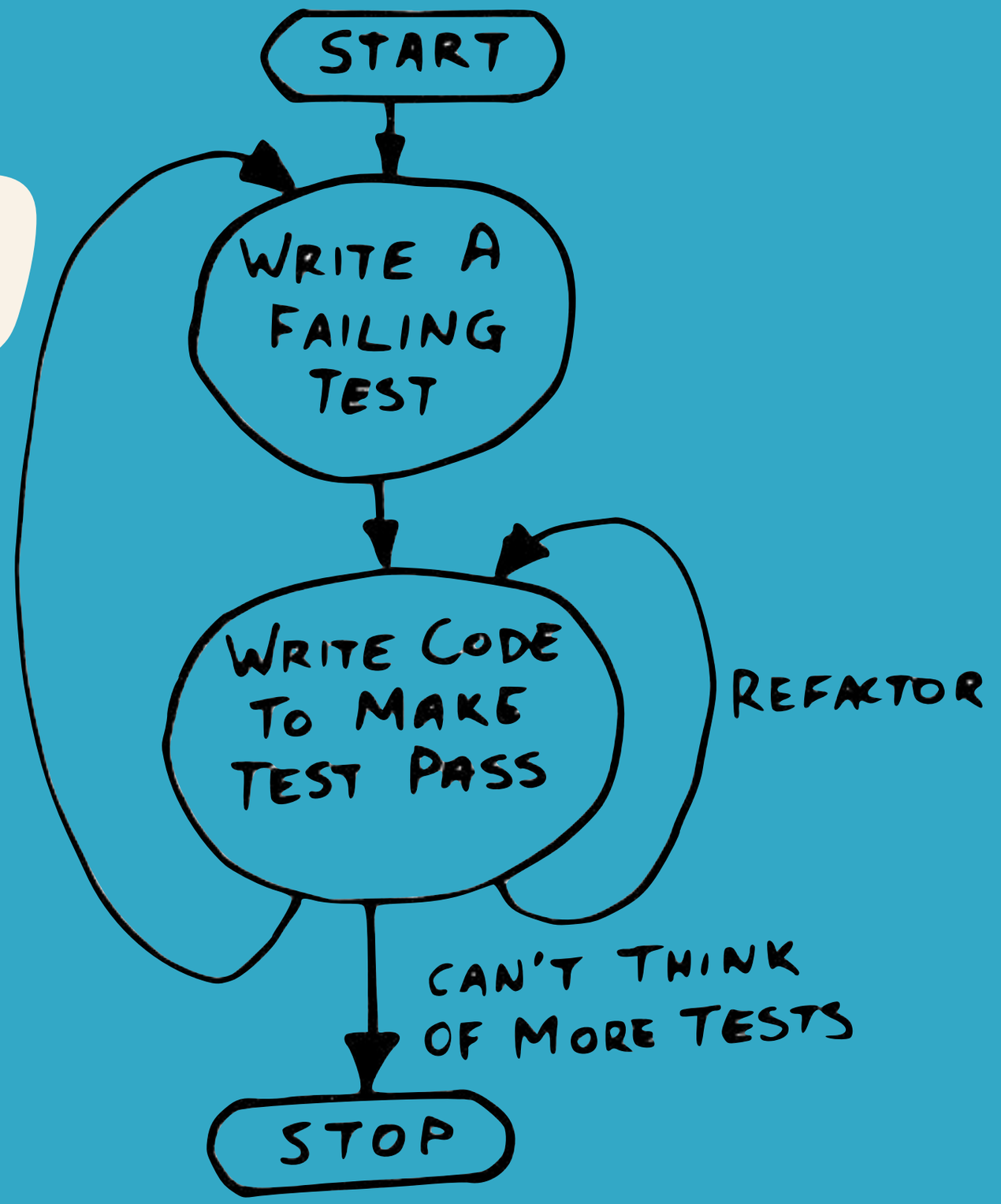
- ✓ Precisión (vs posibles errores en tareas manuales)
- ✓ Perfomance (+tests en el mismo tiempo y en diferentes entornos, escenarios costosos, ...)
- ✓ Feedback asíncrono
- ✓ Construir el proyecto con sólo un comando
- ✓ Detección temprana de conflictos
- ✓ Métricas
- ✓ ... y un largo etcétera

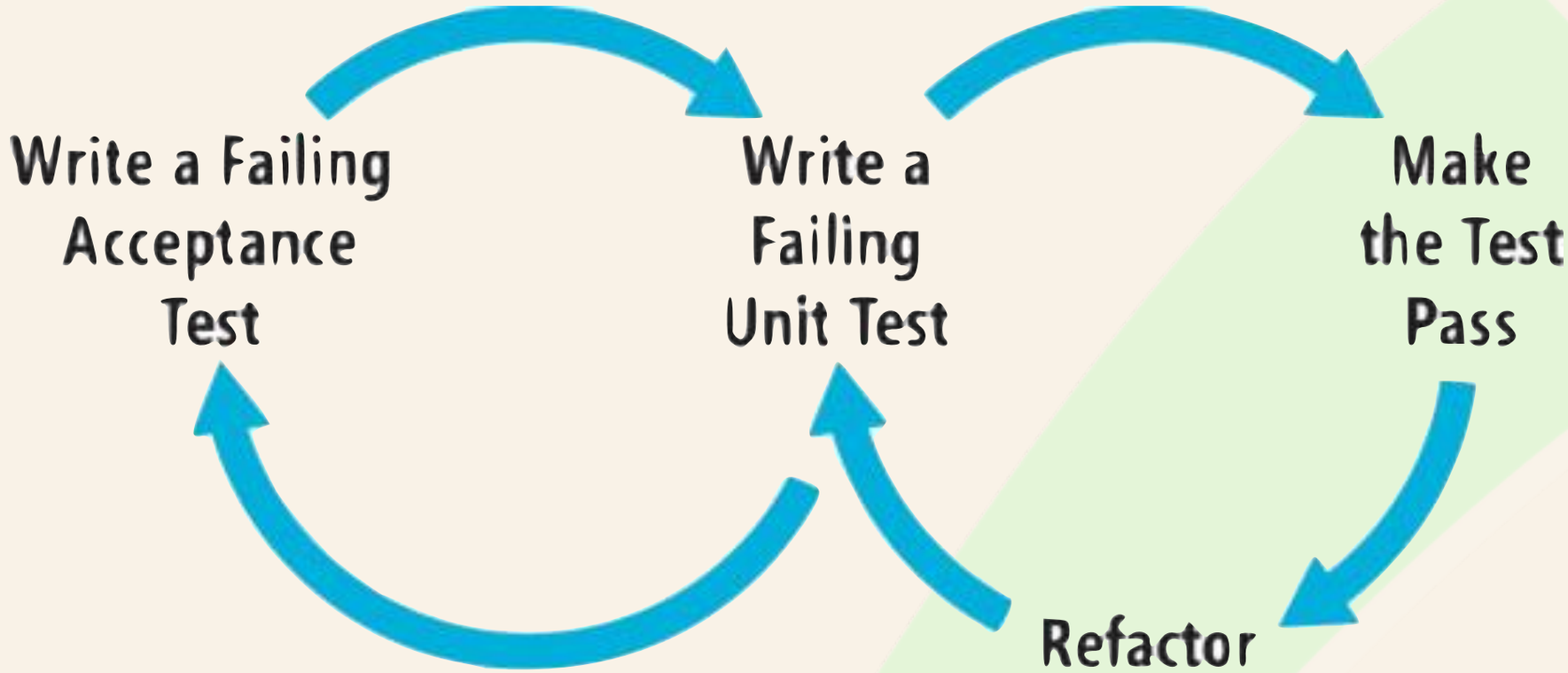
<APACHE ANT>

maven

gradle

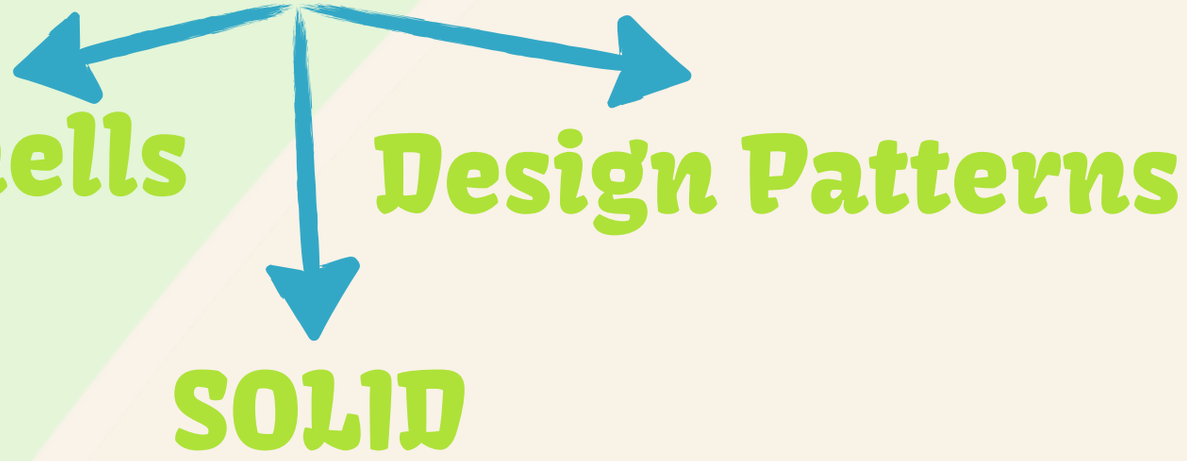
BDD/TDD


START
WRITE A FAILING TEST
WRITE CODE TO MAKE TEST PASS
REFACTOR
CAN'T THINK OF MORE TESTS
STOP



Simple Design:

- ✓ Passes its tests
- ✓ Minimized duplication
- ✓ Maximize clarity
- ✓ Has fewer elements



http://www.jbrains.ca/permalink/the-four-elements-of-simple-design

¿Y mañana en el curro?

- ✓ Empieza a escribir tests, de alguno de los tipos que hemos visto
- ✓ Intenta automatizar la construcción del proyecto y la ejecución de test
- ✓ Si puedes haz BDD/TDD
- ✓ Estudia sobre la parte de “Refactoring”
- ✓ Evangeliza

¡Gracias!

@kinisoftware

cas BILBAO 10-11 OCT
Conferencia Agile Spain 2013

tuenti

Engineering Jobs


theEvnt

Recursos (1)

BDD + UI + Selenium - http://thomassundberg.wordpress.com/2012/11/01/a-jsf-web-application/

Page Objects - https://code.google.com/p/selenium/wiki/PageObjects

Mockito Examples - http://gojko.net/2009/10/23/mockito-in-six-easy-examples/

Libro TDD de Carlos Ble - http://www.dirigidoportests.com/el-libro

Code Smells - http://www.codinghorror.com/blog/2006/05/code-smells.html

Code Smell - http://c2.com/cgi/wiki?CodeSmell

xUnit Patterns - http://xunitpatterns.com/

Refactoring Catalog - http://www.refactoring.com/

Recursos (II)

Hamcrest for iOS - https://github.com/hamcrest/OCHamcrest

Mockito for iOS - https://github.com/jonreid/OCMockito

Acceptance/Integration testing iOS - https://github.com/kif-framework/KIF

TDD for iOS - http://www.amazon.com/Test-Driven-iOS-Development-Developers-Library/dp/0321774183

Megalibro - http://www.amazon.com/Growing-Object-Oriented-Software-Guided-Tests/dp/0321503627

Esta presentación - https://www.dropbox.com/s/658csb9s1xppfts/AdictoAlVerde.pdf

Page Objects

```
public class LoginPage {
   // set up and stuff

    public HomePage loginAs(String username, String password) {
        // ... clever magic happens here
    }

    public LoginPage loginAsExpectingError(String username, String password) {
        //  ... failed login here, maybe because one or both of the username and password are
wrong
    }

    public String getErrorMessage() {
        // So we can verify that the correct error is shown
    }
}

public void testLoginPageWithInvalidUsernameContainsTheExpectedErrorMessage() {
    LoginPage loginPage = new LoginPage(driver);
    String actualErrorMessage = loginPage.loginAsExpectingError(INVALID_USERNAME, ANY_PASSWORD);
    assertThat(actualErrorMessage, is(EXPECTED_LOGIN_ERROR_MESSAGE_WITH_INVALID_USERNAME));
}
```

Historia/Escenario (JBehave)

```
Scenario: Add task in empty to-do list

Given an empty to-do list
When you add the task: do laundry
Then the to-do list contains the task: do laundry
And the number of tasks into to-do list should be 1
```

```
public class AddingTasksSteps {

    private ToDoList toDoList;

    @Given("an empty to-do list")
    public void createEmptyToDoList() {
        toDoList = new ToDoList();
    }

    @When("you add the task: $task")
    public void addTask(String task) {
        toDoList.addTask(new Task(task));
    }

    @Then("the to-do list contains the task: $task")
    public void toDoListMustContainOneTask(String task) {
        assertThat(toDoList.nextTask(), is(new Task(task)));
    }

    @Then("the number of tasks into to-do list should be $numberOfTasks")
    public void theNumberOfTasksIntoToDoListShouldBe(int numberOfTasks) {
        assertThat(toDoList.getNumberOfTasks(), is(numberOfTasks));
    }
}
```

Historia/Escenario (Spock)

```
class AddingTasksSpecification extends Specification {

    def "Add task in empty to-do list"() {
        setup:
        def toDoList = new ToDoList()
        def taskDescription = "do laundry"

        when:
        toDoList.addTask(new Task(taskDescription))

        then:
        toDoList.getNumberOfTasks() == 1
        toDoList.nextTask() == new Task(taskDescription)
    }
}
```

Mockito

```
@Test
public void serviceShouldFindTheUserExpectedById() {
  //Arrange
        User expectedUser = mock(User.class);
        UserDao dao = mock(UserDao.class);
  UserService service = new UserService();
        service.setUserDao(dao);

  //Act
  when(dao.findById(USER_ID)).thenReturn(expectedUser);
  User userFound = service.findUserById(USER_ID);

  //Assert
 assertThat(userFound, is(expectedUser));
}
```

```
@Test
public void serviceShouldValidThePassword() {
  //Arrange
        User user = mock(User.class);
        UserDao dao = mock(UserDao.class);
  UserService service = new UserService();
        service.setUserDao(dao);

  //Act
  when(dao.findById(anyString()).thenReturn(user);
  when(user.isValid()).thenReturn(true);
  ... = service.isUserValid(USER_ID, ANY_PASSWORD);

  //Assert
 verify(dao).isValidPassword(user, ANY_PASSWORD);
 verify(user).isValid();
}
```