Agile Testing

Rocket Test your Products

Vision du Sprint

Savoir tester de façon agile et à 100% une application.

Le Product Backlog


Introduction
Tests Unitaires
Tests Fonctionnels
Tests D’acceptation
Gestion documentaire des tests
Reporting
Tests innovants

Introduction

Introduction

Le testeur en tant que rôle n'est plus optionnel

Identifie les risques suffisamment tôt et rend plus efficace les devs

L'activité de test devient une activité continue, structurante et importante

Nouvelle génération

Aident les décideurs à répondre à cette question: Sommes-nous en train de développer le bon produit ?

Comprennent le métier de leurs clients, les enjeux et priorités

Aident l'équipe à construire un logiciel de meilleure qualité

Nouvelle génération

Aident l'équipe à rester sur la bonne voie (minimiser les risques, renforcer les bonnes pratiques, ...)

C'est la voix du client en spécifiant son besoin au travers d'une description opérationnelle complémentaire aux spécifications

Testeur Agile VS Testeur Classique

Testeur Agile	Testeur Classique
Est membre à part entière de l'équipe	Fait partie de la cellule qualité
Collabore avec les développeurs	Détient le pouvoir de validationet contrôle le travail des développeurs
Répond au changement	Applique le contrat
Écrit des tests qui sont la spécification	Conçois des cas de test relatifs aux spécifications
Intervient tout au loang du projet	Intervient après les développements lors des phases de test
Adapte sa stratégie	Ecris un plan de test
Donne le feed back (points forts et points faibles du produit développé)	Cherche et décris les défauts

Développeur

=

Testeur

Les tests Agiles

Représente l'ensemble des méthodes de test, tests exploratoires, ATDD, TDD, BDD

THE TESTING Manifesto

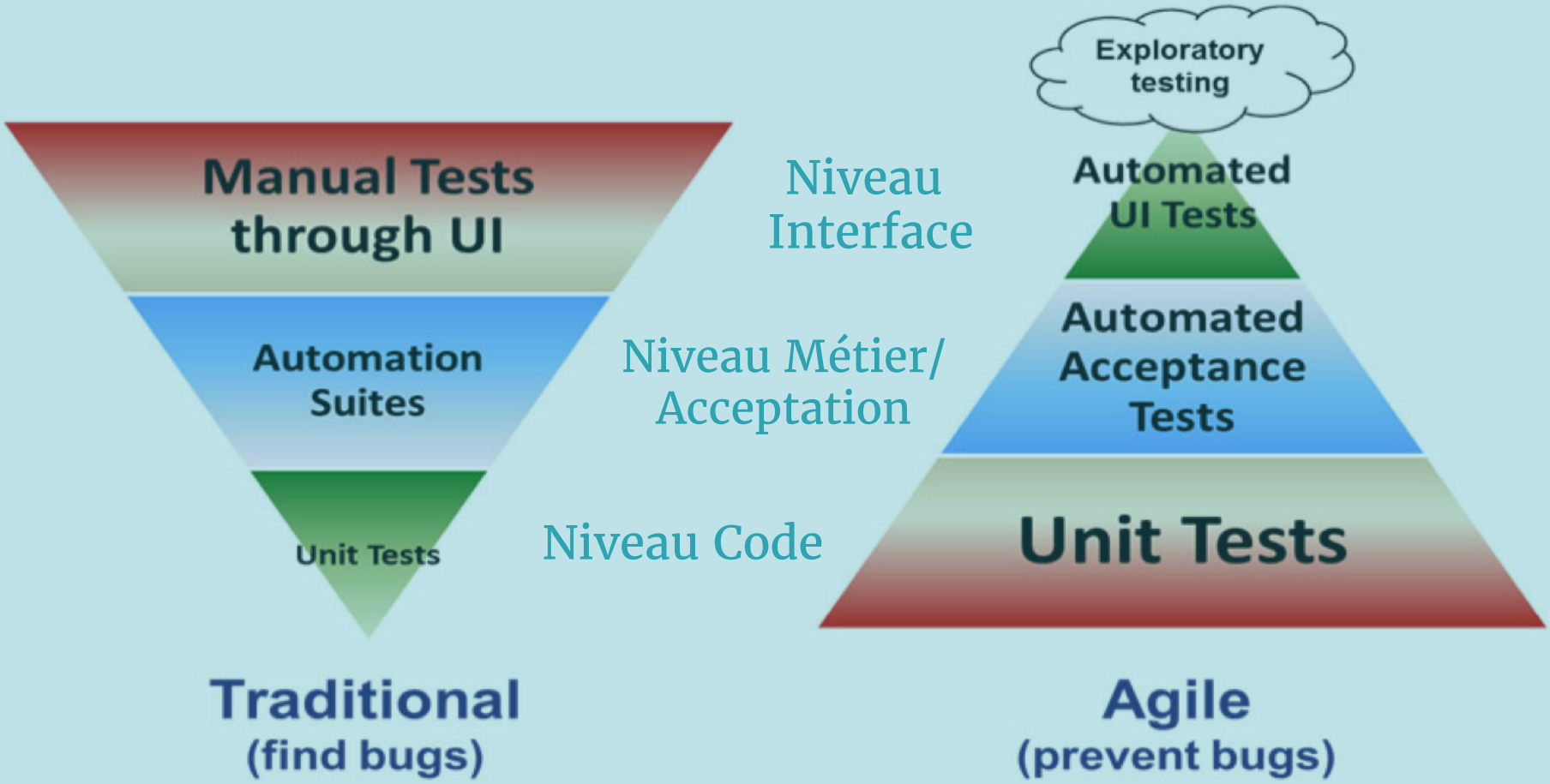
we value:

Testing throughout **OVER** testing at the end	Preventing bugs **OVER** finding bugs	Testing understanding **OVER** checking functionality	Building the best system **OVER** breaking the system	Team responsibility for quality **OVER** tester responsibility

www.GrowingAgile.co.za @growingAgile

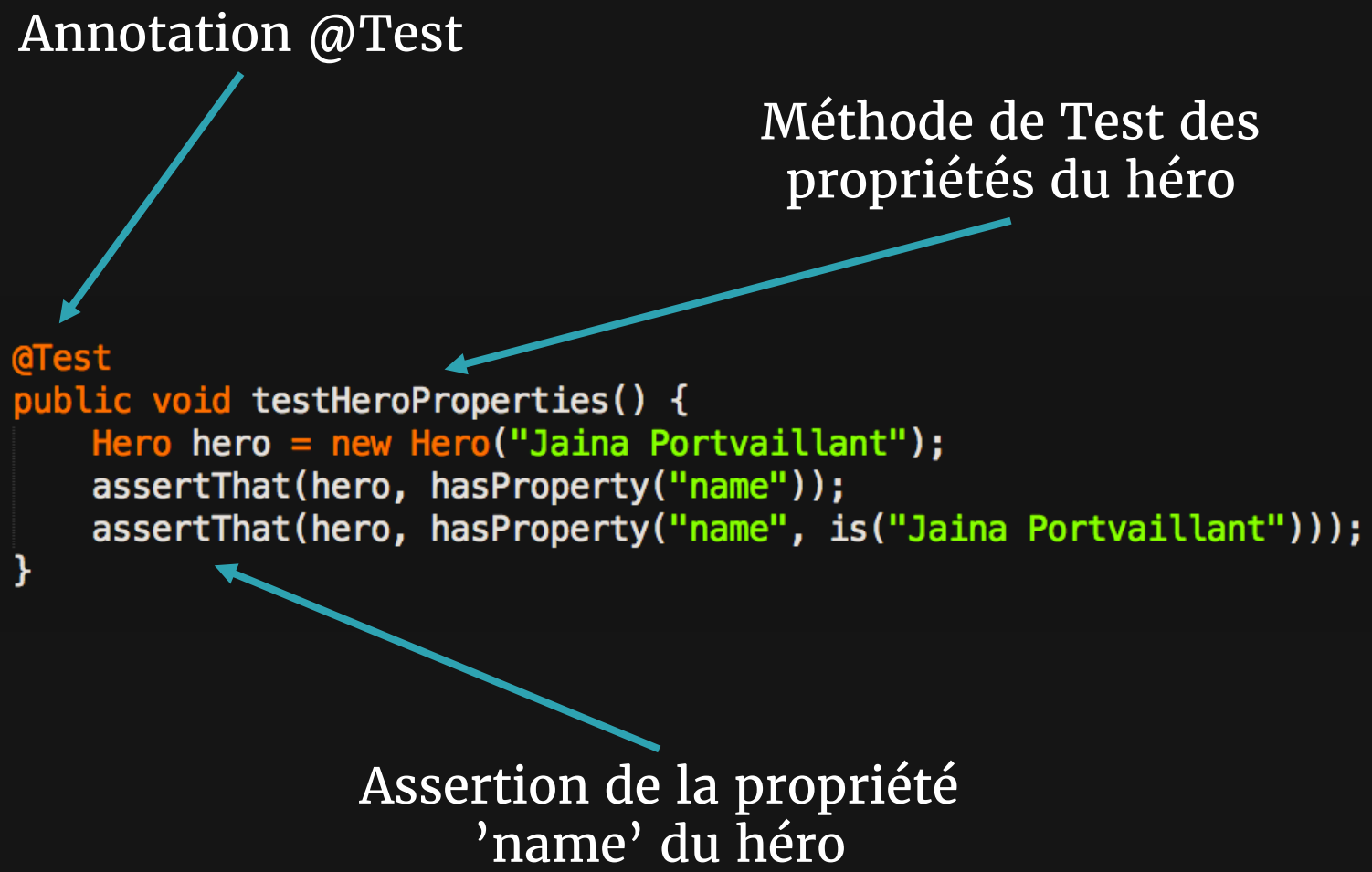
Les tests Agiles

L'approche «Classique» et les modèles «Agiles» ont une conception différente des tests


Manual Tests through UI
Automation Suites
Unit Tests
Traditional
(find bugs)
Niveau Interface
Niveau Métier/ Acceptation
Niveau Code
Exploratory testing
Automated UI Tests
Automated Acceptance Tests
Unit Tests
Agile
(prevent bugs)

Les Tests Unitaires

Définition: Procédure permettant de **vérifier** le bon fonctionnement d'une partie précise d'un **logiciel** ou d'une portion d'un programme.


Annotation @Test
Méthode de Test des propriétés du héro
@Test
public void testHeroProperties() {
Hero hero = new Hero("Jaina Portvaillant");
assertThat(hero, hasProperty("name"));
assertThat(hero, hasProperty("name", is("Jaina Portvaillant")));
}
Assertion de la propriété 'name' du héro

Assertion: **Proposition** que l'on avance comme **vraie**.

```
// Vérifie que c'est égal
assertEquals(strFrenchHello, "bonjour");

// Vérifie que c'est plus grand
assertGreater(125, 180);

// Vérifie que c'est plus petit
assertLesser(125, 18);

// Vérifie que ce n'est pas plus grand
assertNotGreater(125, 18);

// Vérifie que ce n'est pas plus petit
assertNotLesser(125, 180);

// Vérifie que ce n'est pas pareil
assertNotSame("expected", "actual");

// Vérifie que c'est pareil
assertSame("abc", "abc");
```


JUnit

JUnit, quel intérêt ?

Le framework de test Java n°1

Existe pour pratiquement tous les langages (xUnit: phpUnit, nUnit, ...)

Méthodes transposables dans d'autres langages

Le Projet

«RPG-Console»

Jeu de rôle minimaliste

Base pour créer un moteur de jeu en Java mais surtout pour le tester !!

Code sur mon Github (adesousa)

Liens pratiques

Doc JUnit

http://junit.org/junit4/

Hamcrest

https://code.google.com/archive/p/hamcrest/wikis/Tutorial.wiki

Conseils TU

http://blog.xebia.fr/2008/04/11/les-10-commandements-des-tests-unitaires/

Mon GitHub

https://github.com/adesousa

Installation & Configuration

```
## Junit lib & Hamcrest lib Download
cd /Library/Java/
mkdir JUNIT
cd JUNIT/
sudo wget https://github.com/junit-team/junit4/releases/download/
    r4.12/junit-4.12.jar
sudo wget http://central.maven.org/maven2/org/hamcrest/hamcrest-
    all/1.3/hamcrest-all-1.3.jar
sudo chown $USER:$USER /Library/Java/JUNIT/*
sudo chmod 755 /Library/Java/JUNIT/*

## Junit & Hamcrest Configuration
vim ~/.bash_profile
export CLASSPATH=/Library/Java/JUNIT/junit-4.12.jar:/Library/Java
    /JUNIT/hamcrest-all-1.3.jar:.
source ~/.bash_profile
```

```
## First of all, you need to compile your src classes and test
cases. For example (from src folder):


    ### Classes
    cd src/
    javac -d ../bin/ codingfactory/rpgconsole/enemy/*.java
    javac -d ../bin/ codingfactory/rpgconsole/hero/*.java
    javac -d ../bin/ codingfactory/rpgconsole/game/*.java


    ### Tests
    cd src/ (important, pour éviter des erreurs de compilation)
    javac -d ../bin/ HeroTest.java
```

```
## Play with the Java App (from bin folder)
cd bin/
java bin/codingfactory.rpgconsole.game.Starter


## Then run your test cases. (from bin tests folder):
cd bin/
java org.junit.runner.JUnitCore test.HeroTest
```

Live Coding

Action !

Testez «RPG-Console»

Jouez au RPG

Appropriez-vous son code, modifiez ce que vous voulez.

Code disponible sur mon github (adesousa)

Objectif

Couverture de code à 100%

Chaque méthode doit être testée au moins par un assert