



vers les exigences exécutables



Business
Services



Olivier BILLARD

Architecte Logiciel

IT&L@BS

olivier1.billard@orange-ftgroup.com

Thierry HENRIO

Programmeur

Organisateur Conférence

Agile France

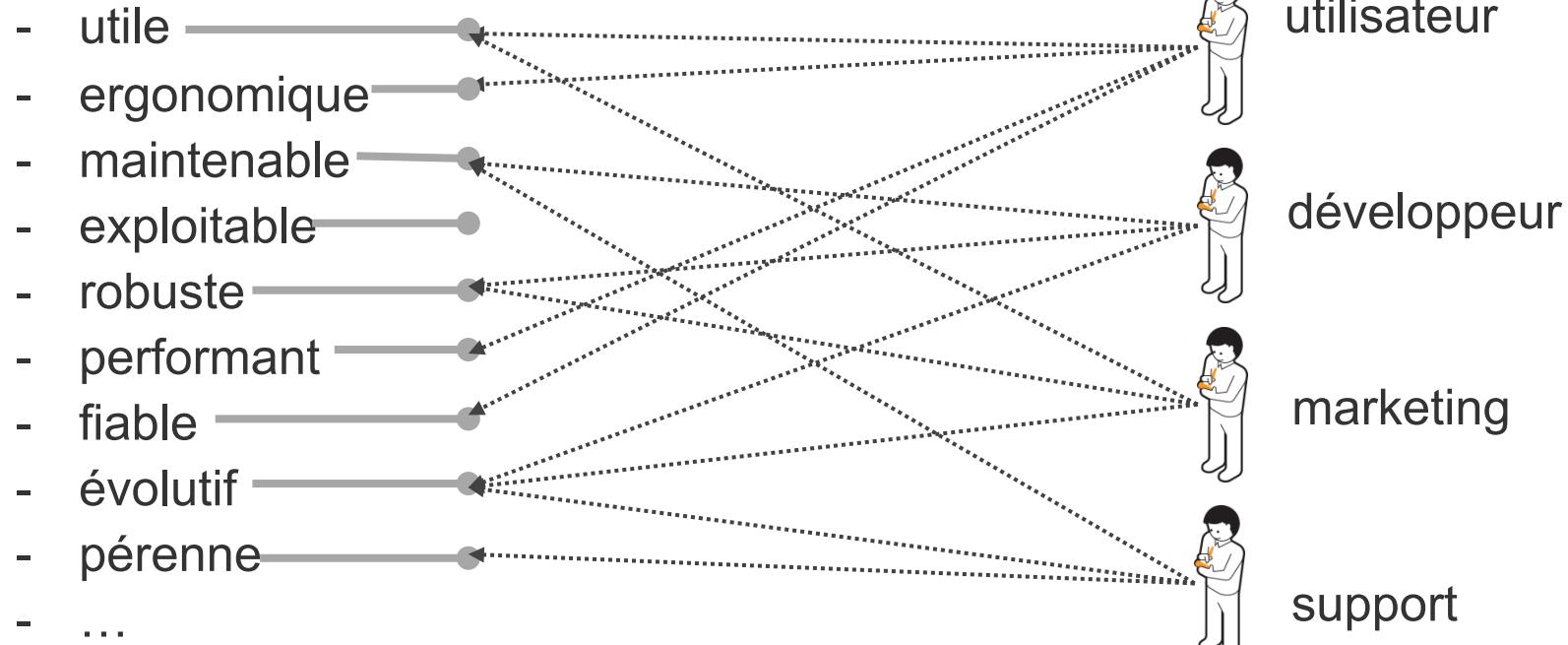


Vers les exigences exécutables

histoire

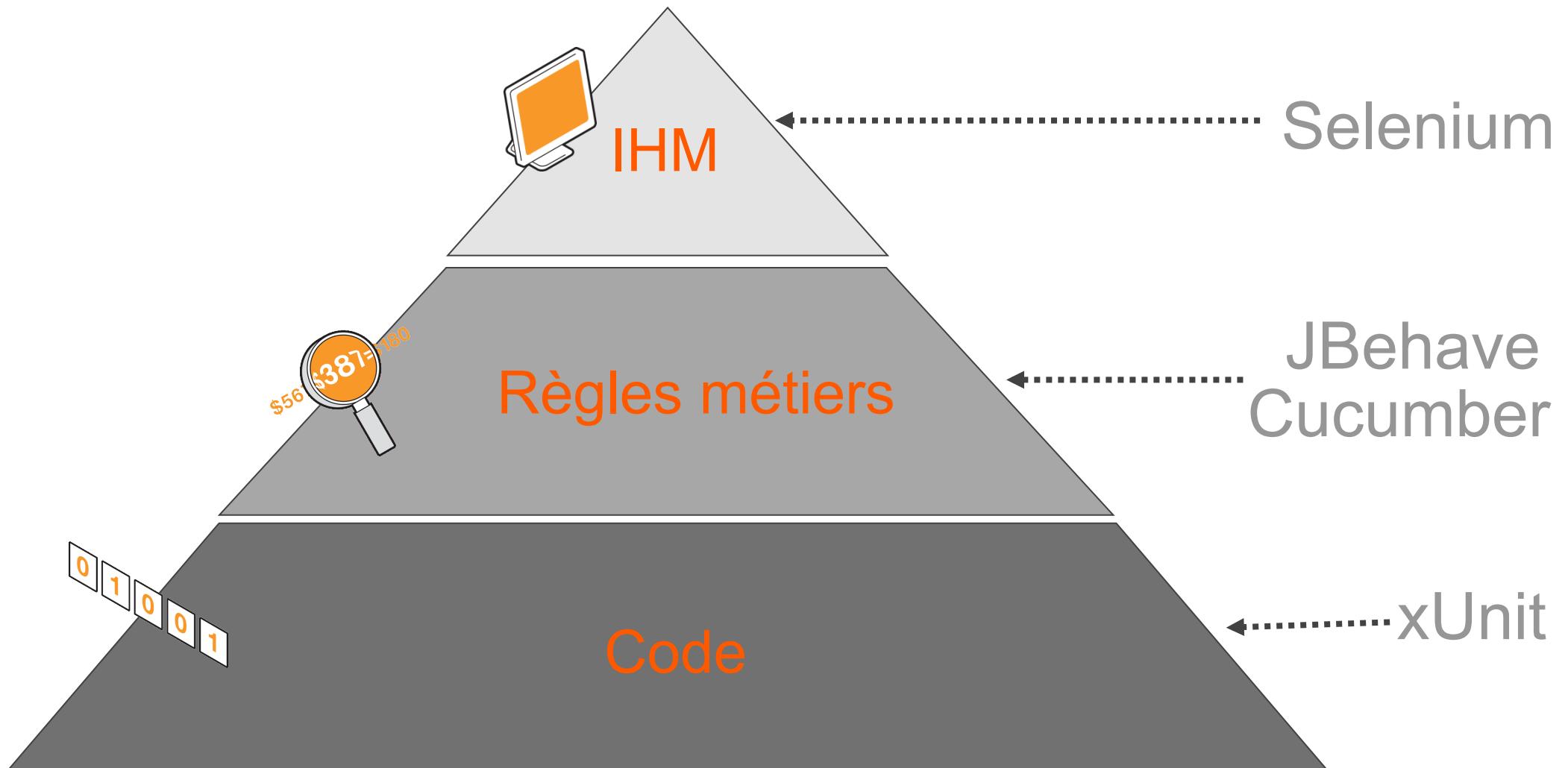
industrialiser les développements pour soutenir la qualité des productions

- > de nombreux critères et manifestations :



- > développer un logiciel de qualité doit être un **principe outillé** et non seulement l' étape d'un projet

un outil de tests ?



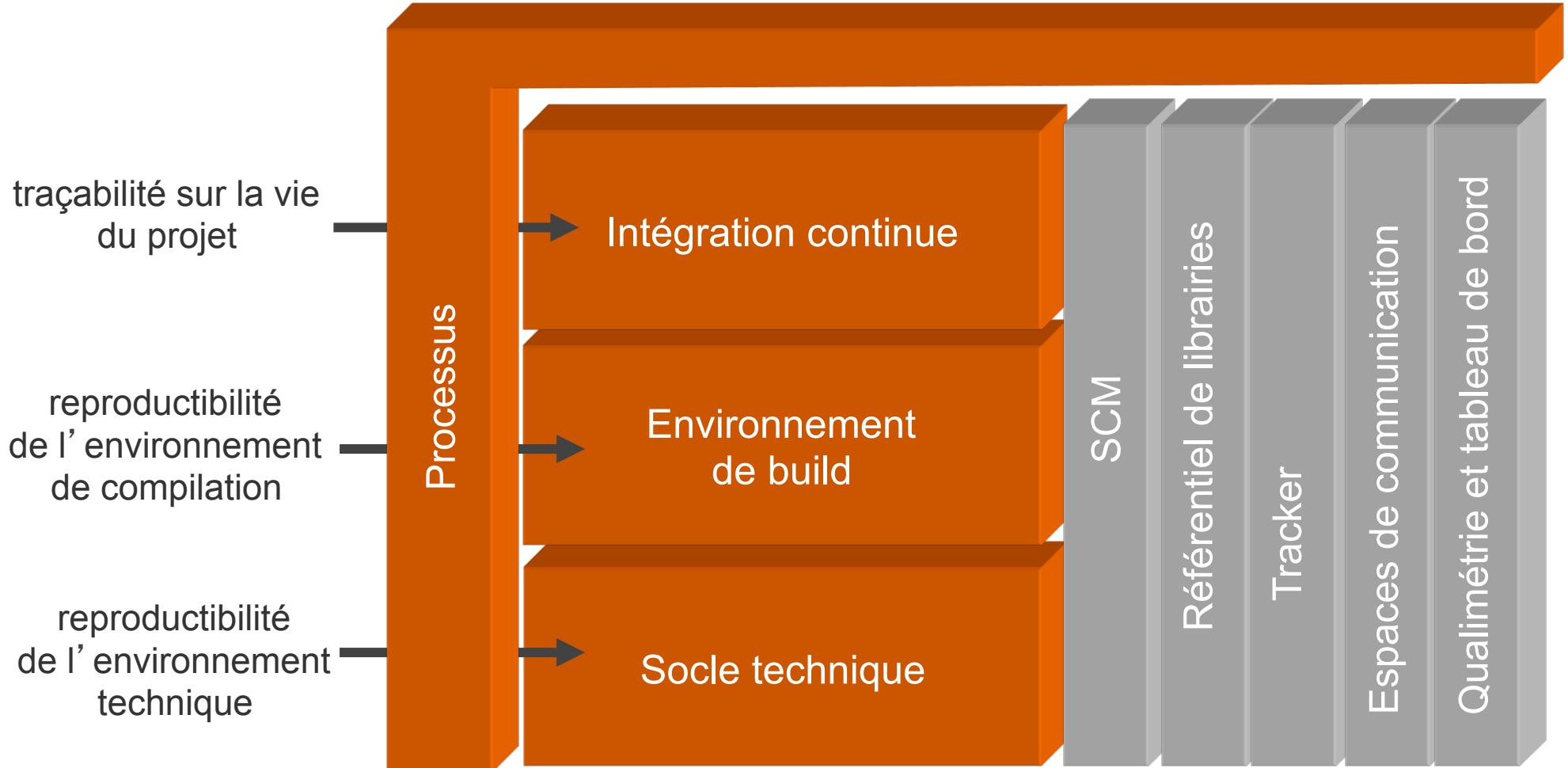
Vers les exigences exécutables

Page 5

Business
Services

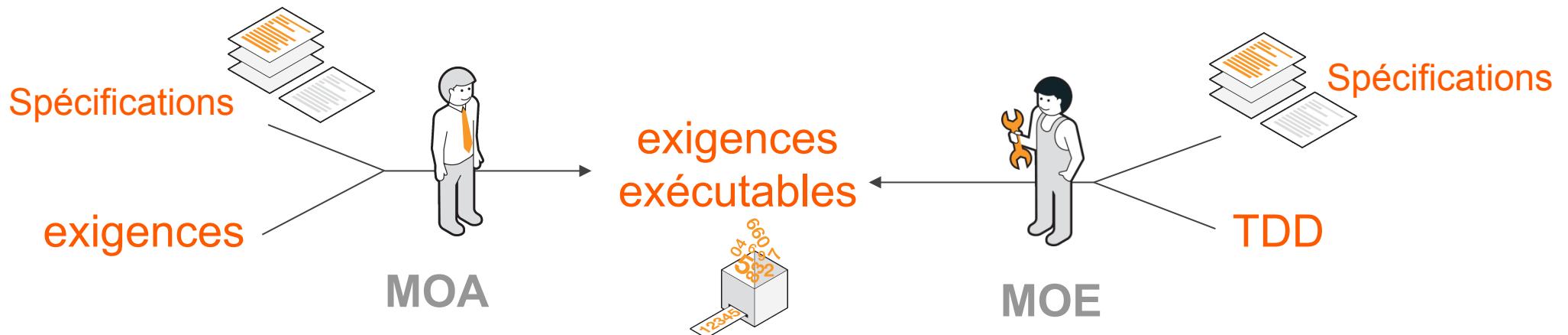


ingrédients et ustensiles d' une forge logicielle



vers les exigences exécutables...

- > deux approches des exigences exécutables :
 - **partir de la MOA** : écrire ses spécifications sous la forme d'exigences et les tests associés qui les valident
 - **partir de la MOE** : vient du monde du Test Driven Development et permet de prendre en compte les spécifications sous forme d'exigences exécutables

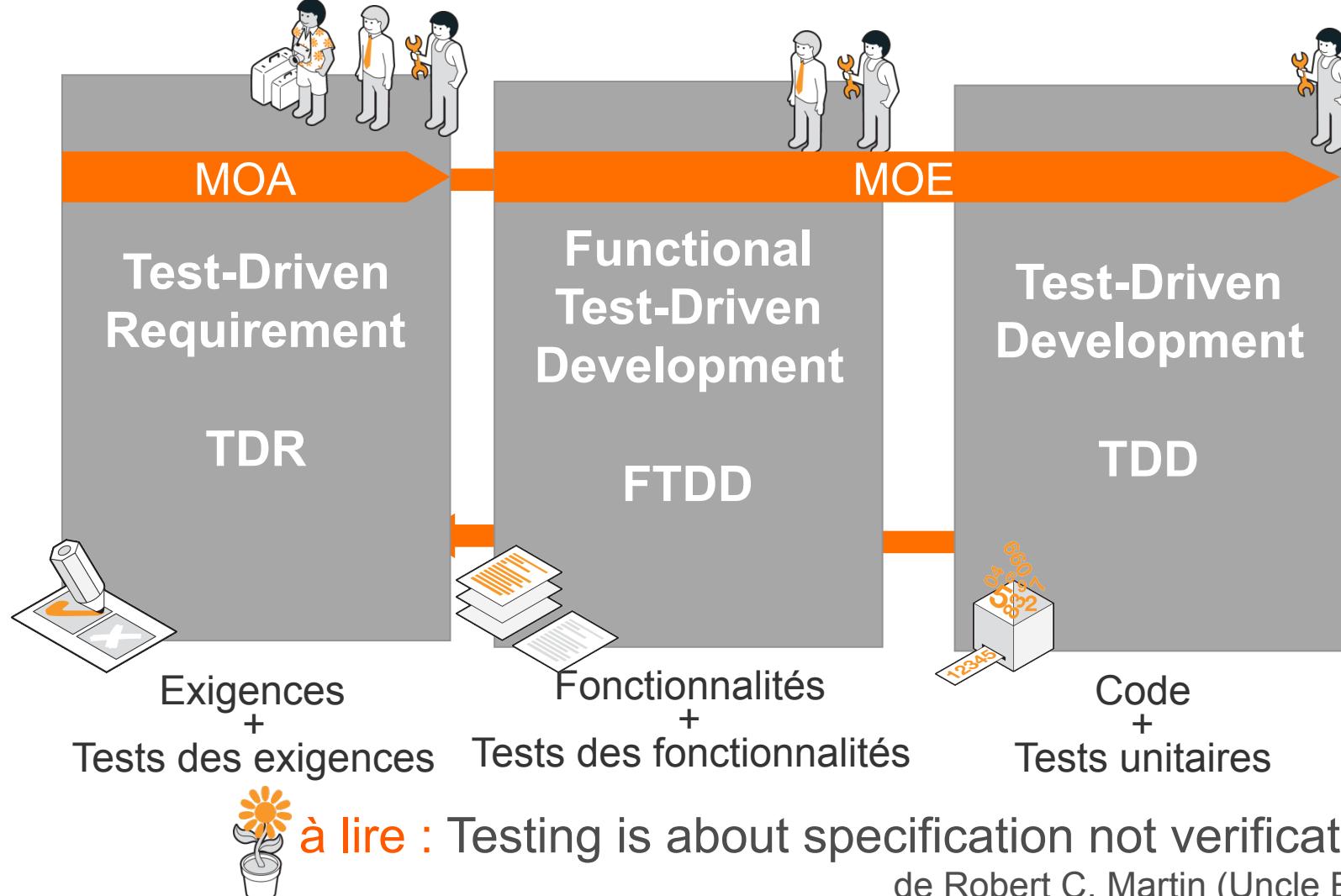




Vers les exigences exécutables

les principes et la théorie

Test Driven Requirements (TDR) exigences pilotées par les tests



Vers les exigences exécutables

Page 9

Business
Services



Test Driven Requirements (TDR) exigences pilotées par les tests

- > produire les spécifications sous la forme d'un ensemble d'exigences et des tests qui les valident
 - ces tests sont écrits sous la forme de tableau (simplicité de la structure) afin de valider la compréhension des uns et des autres.

ColumnFixture	tableau permettant de spécifier des données en entrée et des résultats attendus
ActionFixture	tableau permettant de spécifier un enchaînement d'actions. Ce format de tableau permet de décrire des scénarios de tests
RowFixture	tableau permettant de spécifier des lignes de résultats. Ce format de tableau est généralement utilisé pour faire des initialisations ou pour vérifier des résultats de recherches

- > l'écriture de tests est un outil de recueil du besoin.
- > le principe : pilotage des exigences par les tests.
 1. valorisation des exemples fournis par la MOA
 2. écriture des suites de tests,
 3. écriture des cas d'utilisation ou des règles métier correspondants

Test Driven Requirements (TDR) exigences pilotées par les tests

- > des exemples d' outils
 - Fit
 - FitNesse + Fit
 - FitNesse + Slim
 - GreenPepper (Pyxis Technologies)
 - Twist (ThoughtWorks)
 - ...

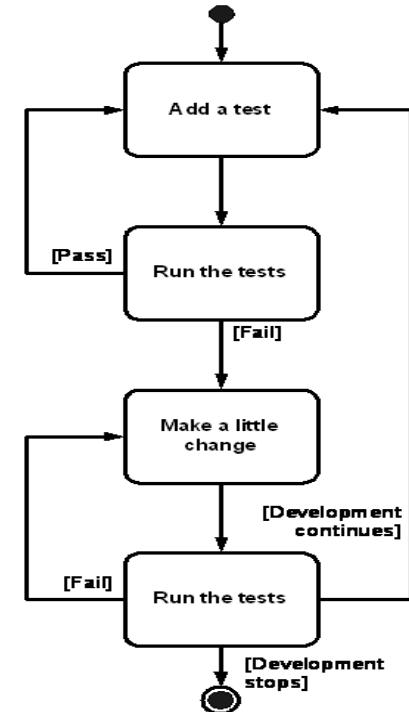


Vers les exigences exécutables

Introduction au BDD

Behaviour Driven Development (BDD) approche comportementale du test logiciel

- > point de départ du BDD : le Test-Driven Development (TDD)
 - « avant d'écrire la moindre ligne de code, on se doit d'écrire un test qui ne passe pas »
 - les motivations sont saines : produire du code utile, simple, sans superflu en concrétisant l'implémentation du test
 - dans les faits, on arrive (bien) à produire le code, mais la construction des tests est une opération pas totalement triviale
- > le BDD propose d'apporter une réponse en établissant des tests basés sur le comportement de l'application
 - par extension, on peut proposer une organisation des tests en s'intéressant aux fonctionnalités de l'application exprimée sous forme d'exemples.



Behaviour Driven Development (BDD) approche comportementale du test logiciel

- > le BDD est complémentaire au TDD, centré sur ce que devrait faire l' application
 - vise à outiller la conception de l' application tout comme sa documentation
 - utilisation d' un vocabulaire commun et simpliste
 - réduire au maximum les malentendus
 - assurer que chacun est bien sur le même cas étudié avec les mêmes mots utilisés.
- > les comportements contribuent à produire des critères d' acceptabilité
 - gestion de la non régression tout comme pour les tests unitaires
- > les comportements sont décrits en termes de scénario sous la forme suivante :
 - **Given** : le contexte initial
 - **When** : un évènement arrive
 - **Then** : qui produit un certain résultat attendu
- > points clefs du BDD :
 - les exigences sont des comportements
 - l' analyse se fait par un langage uniforme pour toute l' équipe
 - les critères d' acceptabilité sont exécutables

Behaviour Driven Development (BDD) approche comportementale du test logiciel

- > des solutions pour la plupart des plates-formes
 - C : CSpec
 - C++ : CppSpec Spec-CPP
 - C# .Net : NSpec
 - .Net : NBehave, NSpecify
 - Delphi - dSpec
 - Groovy - GSpec, easyb, tspec
 - Java - JBehave, JDave, beanSpec, Instinct, Cucumber
 - Javascript - JSSpec
 - PHP - PHPSpec
 - Python - Specipy, spec plugin for nose
 - Ruby - RSpec, Shoulda, test-spec & bacon, Cucumber



Vers les exigences exécutables

Produits, solutions et tendances

outillage du TDR GreenPepper



- > solution d' outillage du TDR proposée par Pyxis Technologies
- > L' idée est de :
 - permettre aux clients, testeurs et développeurs de partager une vue commune de ce que le logiciel devrait faire
 - puis comparer la cible avec ce qu' il fait effectivement à un instant donné
- > Un wiki (Confluence) pour capturer et exécuter les exigences
- > Support Eclipse/Maven, Visual Studio/Nant, Jenkins/Hudson

 **Use Case 4**
[View](#) [Edit](#) [Attachments \(0\)](#) [Info](#)
[Browse Space](#) [Add Page](#) [Add News](#)
Added by [Louis Thibault](#), last edited by [Francois Beauregard](#) on Aug 09, 2007 ([view change](#))Labels: (None) [EDIT](#)**GreenPepperized** - CONFIGUREThis is the Implemented version. Source Requirements: (None) [EDIT](#)**Execute**  For [Bank in Java [EDIT](#)]

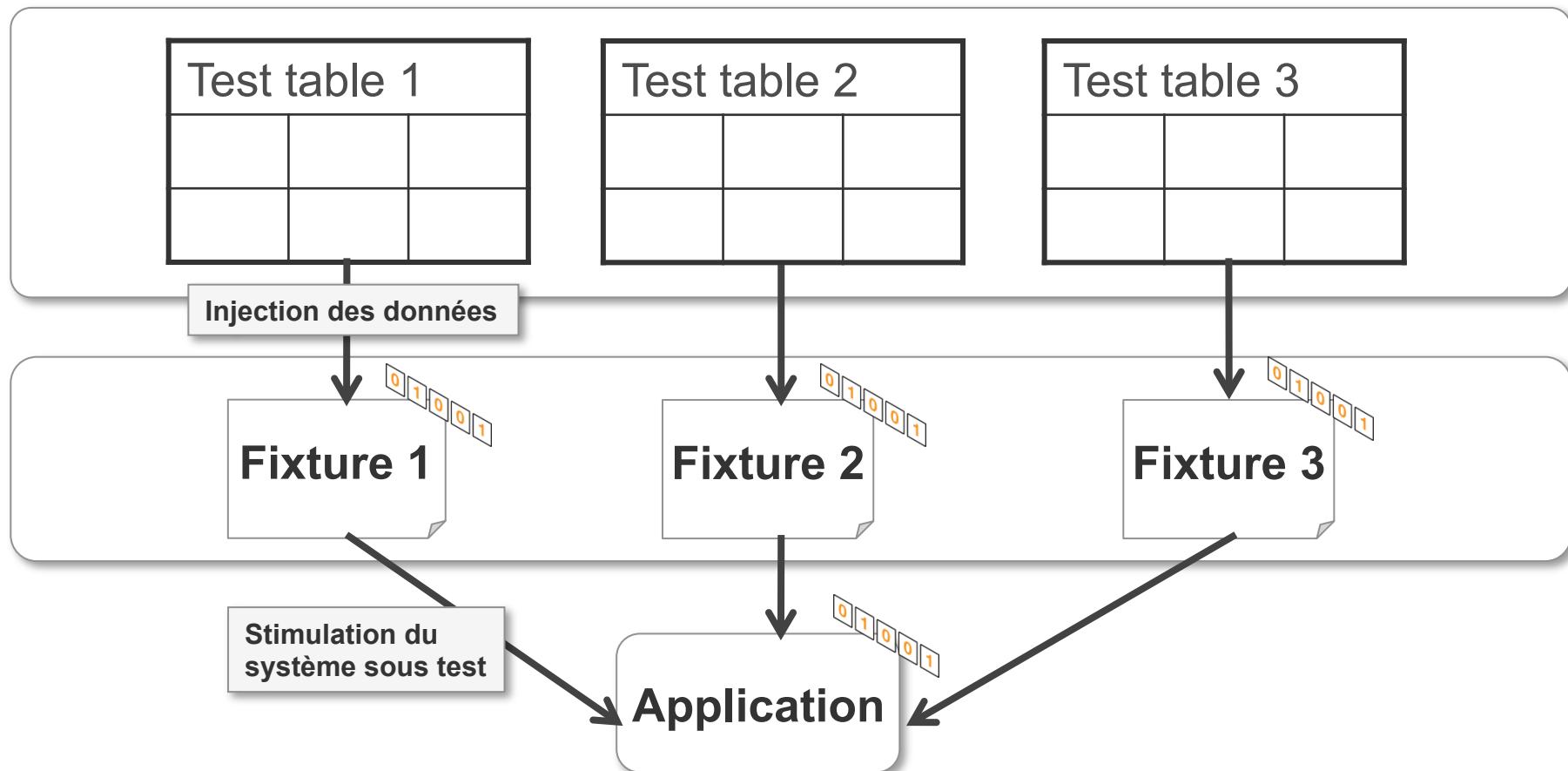
Rights: 17 Wrongs: 1 Errors: 0

Use Case #4:	The customer is allowed to withdraw money within certain rules.
Actor:	Customer
Business rules:	<ul style="list-style-type: none"> The customer is allowed to withdraw if the balance of the account is sufficient (including the withdraw fees) The customer is allowed to withdraw a maximum of 200.00\$ from an ATM and Interact for savings and checking account.

BUSINESS RULES: Limit of withdrawal

Rule for	Limit of withdrawal	
Account Type	Withdraw Type	Limit?
Savings	ATM	\$200.00
Savings	Interact	\$200.00
Savings	Personal Check	nothing
Checking	ATM	\$200.00
Checking	Interact	Expected: \$300.00 Received: \$200

outillage du TDR GreenPepper, écriture du code de liaison



outillage du TDR

GreenPepper, conclusion

- > bon outil, stable, facilitant la coopération entre les différents acteurs par l' intermédiaire du Wiki
- > bonne intégration avec les outils des développeurs (intégration Jira)
- > mais versionnement des tests séparé des sources

outillage du TDR

les autres solutions

- > FitNesse (Robert C. Martin, Micah Martin et Michael Feathers)



<http://fitnesse.org>

- un des tous premiers outils de TDR, basé sur un wiki
- moins riche et plus « sobre » que GreenPepper
- peu ou pas d' intégration avec les environnement de développement

- > Twist (Thoughtworks)

<http://studios.thoughtworks.com/twist-agile-test-automation>



- produit collaboratif de plate-forme de tests fonctionnels
- environnement riche pour la création, l'exécution et le maintien de tests
- prometteur et à suivre.

outillage du BDD

JBehave, la présentation

- > un outil Open Source dédié au monde Java
- > écrit par Dan North le père du Behaviour Driven Development
- > une surcouche de JUnit
- > une librairie permettant d'exécuter les scénarios de validation des comportements
 - au travers d'une tâche Ant,
 - par un plugin Maven,
 - éventuellement au travers d'un exécuteur de tests JUnit (comme Surefire pour Maven ou ceux fournis dans les IDE modernes).



<http://jbehave.org>

outillage du BDD

JBehave, le principe

- > dans un fichier texte on écrit les différents *scenarii* qui vont définir les critères d'acceptabilité d'une *story* ;
- > on associe une classe java de *behaviour* à cette *story* ;
- > on exécute la *story* ce qui devrait mettre dans l'état *pending* pour les étapes non implémentées ;
 - littéralement 'en suspens', cet état indique que l'étape en question n'est pas réalisée
 - n'est pas bloquant dans le processus de construction
- > produire les étapes pour avancer dans le scénario.
 - lorsqu'une étape passe en erreur on corrige l'implémentation du système testé

outillage du BDD

JBehave, écriture de scénario

- > une écriture simple qui se fait en mode plein texte en utilisant les mots clefs suivants :
 - **Story** : pour décrire de façon informelle l'exigence
 - **Scenario** : pour définir un scénario en lui donnant une description courte et précise
 - **Given** : pour définir le contexte d'exécution
 - **When** : pour définir les évènements
 - **And** : pour ajouter un évènement
 - **Then** : pour définir une étape de validation
- > les mots clés sont internationalisables

outillage du BDD

JBehave, mise en oeuvre



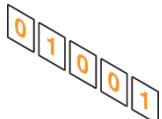
Écriture d' une story et ses scénarios

user_logs_in_successfully.story

Given I am not logged in

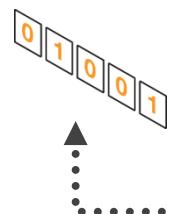
When I log in as Dave with a password JBehaver

Then I should see a message, "Welcome, Dave!"



Écriture d' une classe Java pour la story

```
class MyBehaviour extends JUnitScenario
```

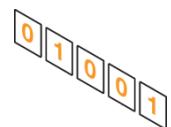


```
@Given("I am not logged in")
public void logout () { ... }

@When("I log in as $username with a password $password")
public void enterUsernameAndPassword(String username, String password) { ... }

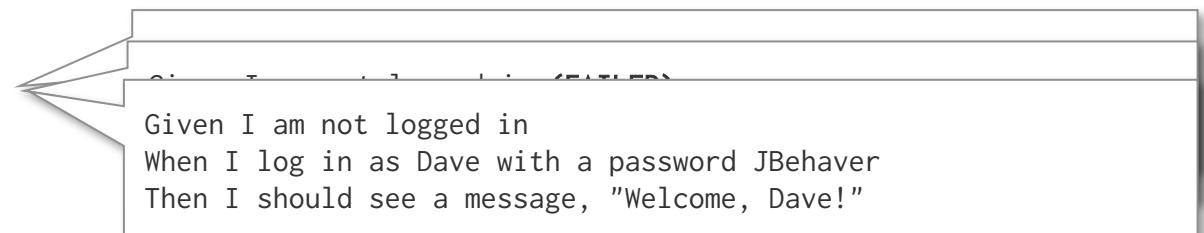
...
```

s rapports
de la story



Écriture d' une ou plusieurs classes Java pour les étapes du
scénario

```
class MySteps extends Steps
```



Vers les exigences exécutables

Page 26

outillage du BDD

JBehave, conclusion

- > un outil tourné vers le développeur, léger avec peu de contraintes
 - fichier texte pour les scénarios, exécution JUnit,...
- > un outil qui outille mal la partie MOA avec une absence d'éditeur graphique,
 - gestion en configuration via le SCM
- > un point positif à retenir, contrairement à la plupart des frameworks BDD, les scenarii sont proprement séparés du code.
- > rapports quasiment inexistant, mais facilement intégrable dans une usine logicielle.



Vers les exigences exécutables

éléments de conclusion



éléments de conclusion

- > BDD et TDR permettent de se concentrer sur les besoins du client
 - dans une optique et avec un outillage pour les développeurs dans le cadre du BDD
 - dans une optique et avec un outillage pour les fonctionnels dans le cadre de TDR
- > dans les faits...
 - si la MOA et la MOE sont « proches », le TDR paraît une approche intéressante en tant qu'outil de communication et outil de travail permettant de formaliser les exemples que doit satisfaire le logiciel développé
 - FIT(NESSE), GreenPepper, Twist
 - si la MOA et la MOE sont plus distantes, et communiquent au travers de documents contractuels, spécification par exemple, BDD permet au développeur de se concentrer efficacement sur le logiciel à développer en « sortant » proprement les tests fonctionnels du code de l' application
 - JBehave, NBehave, RSpec,

éléments de conclusion

- > présence de deux approches pour l'intégration des tests fonctionnels dans le cycle de vie du projet.
 - la première venant de la MOA, a mis en œuvre des outils collaboratifs autour des wikis pour communiquer et échanger avec la MOE.
 - la deuxième approche est partie elle de la MOE, en proposant de généraliser le Test Driven Development pour aller des tests unitaires vers les tests fonctionnels, et donc permettre à la MOA d'intégrer le cycle de développement du logiciel.
- > attention, l'utilisation de wikis, apportent une solution pour l'aspect collaboratif entre les deux entités MOA et MOE avec un certain nombre d'inconvénients :
 - mélange de la présentation et des tests.
 - introduction d'un système qui vit en dehors de l'outil de gestion de configuration, et qu'il va donc falloir synchroniser avec le cycle de vie du logiciel.

éléments de conclusion

- > des concepts et des outils qui restent agnostiques des environnements d'exécution (.NET, Java, Ruby, ...)
- > des concepts émergents et une adoption récente de la communauté (taille des communautés, listes de diffusion ou nombre de téléchargements)
- > excellente réactivité des développeurs de ces solutions.

merci



références recommandables

- > introducing BDD de Dan North
 - <http://dannorth.net/introducing-bdd>
- > site dédié à la théorie du BDD
 - <http://behavior-driven.org/>
- > « Testing is about specification not verification » de Robert C. Martin
 - <http://c-spin.net/meet0303.html>
- > blog de Eric Lefevre
 - <http://ericlefeuvre.net/wordpress/>
- > fitnesse.org
- > fitnesse.info
- > jbehave.org

