



Learn To Change

Tests fonctionnels automatisés

Cédric Rup
Julien Jakubowski
Octo Technology

50 avenue des Champs-Élysées
75008 Paris – France

Organisme de formation N° 11 75 48 936 75

Tél : +33 (0)1 70 36 18 67

Fax : +33 (0)1 58 56 10 01

academy@octo.com - www.octo.academy



▷ Jour 1

- > Généralités sur les tests
- > Focus sur les tests fonctionnels
- > BDD
- > Gherkin

▷ Jour 2

- > Specflow
- > Mises en situation

Les Commandements de cette formation

- ▷ Les horaires tu respecteras
- ▷ La feuille de présence tu signeras
- ▷ Sur ta formation tu te concentreras
- ▷ A ton téléphone et à l'internet mondial tu résisteras
- ▷ A ton formateur des questions tu poseras
- ▷ Ton expérience tu partageras
- ▷ Ton avis tu donneras
- ▷ De la pause tu reviendras
- ▷ Comme Yoda de parler tu n'es pas obligé





Chapitre 1 : Généralités sur les tests

Racontez la situation de test la plus difficile que vous ayez jamais rencontrée.

D'où venaient les difficultés ?

Comment vous en êtes vous sorti ?

Qui ?

Quoi ?

Pourquoi ?

Comment ?

Evaluation ?

1. **Qui: La personne qui effectue les tests**

le développeur
le product owner, le testeur,
un utilisateur

...

2. **Quoi: Que couvre le test**

chaque fonction
les services
de bout en bout

...

3. **Pourquoi: Problèmes potentiels recherchés par les tests**

valeurs extrêmes
régressions
erreurs fréquentes

...

4. **Comment: Activités de test**

tests automatisés
tests exploratoires

...

5. **Evaluation: Critères de passage des test**

comparaison avec un résultat attendu
comparaison avec le résultat d'une exécution précédente

...

- ▷ **Chaque activité de test implique les 5 dimensions**
- ▷ **Chaque technique de test s'envisage dans les 5 dimensions**
- ▷ **Exemple: « tests basés sur les spécifications »**
 - > **couverture**: tester tout ce qui est listé dans ce document de spéc.
 - > **problèmes potentiels**: tester pour tout ce qui dans cette spec. pourrait ne pas être satisfait.
 - > **évaluation**: concevoir les tests en utilisant la spéc. pour déterminer si le test passe ou pas (ex: spéc. exécutable ATDD).

▷ **Tests Utilisateurs**

- Ceux qui utiliseront ou utiliseraient typiquement l'application.

▷ **Alpha Tests**

- Equipe de test (et peut être d'autres personnes intéressées)

▷ **Beta Tests**

- Utilisateurs qui ne font pas partie de l'organisation mais du « marché » pour l'application.

▷ **Tests par un expert du domaine**

▷ **Tests en binômes**

▷ **« eat your own dog food »: l'entreprise utilise des pré-versions de son propre logiciel**

- ▷ Tests de fonction
- ▷ Tests de Fonctionnalité (feature)
- ▷ Tour des menus
- ▷ Tests du domaine
- ▷ Analyse des classes d'équivalence
- ▷ Tests aux limites
- ▷ Tests des meilleurs représentants pour une classe d'équivalence
- ▷ Matrices de champs de saisie
- ▷ Toutes les façons de renseigner un champ
- ▷ Tests de la logique
- ▷ Tests basés sur les états
- ▷ Tests des chemins
- ▷ Couverture des instructions et des branchements
- ▷ Couverture des configurations
- ▷ Tests basés sur les spécifications
- ▷ Tests basés sur les exigences
- ▷ Tests de combinaisons de variables

▷ Contraintes sur les entrées

- > e.g: une entrée est limitée à un entier sur 32 bits. Tester sur des entiers plus grands.

▷ Contraintes sur les sorties

- > e.g: une sortie est limitée à un entier sur 32 bits. Tester un cas où la sortie va être plus grande.

▷ Contraintes de calcul

- > e.g: on multiplie deux entiers. Tester un cas où la multiplication crée un débordement.

▷ Contraintes de mémoire ou de stockage

▷ Contraintes de temps de réponse

▷ Tests de régression

- > prouver qu'un bug est toujours présent; prouver qu'une correction a causé des impacts sur le reste de l'application

▷ Script de test

- > scénario écrit suivi par un testeur (débutant) exécutant manuellement le test

▷ Smoke Test

- > test de non régression cherchant à prouver que le nouveau build n'est pas stable. Si le test ne passe pas, il est inutile de faire les autres tests.

▷ Test exploratoire

- > test dans lequel le testeur apprend sur le produit au fur et à mesure de sa recherche des problèmes possibles

▷ Test de Scénario

- > réaliste: correspond à un cas d'usage
- > complexe: impliquant plusieurs fonctionnalités de l'application
- > facile à diagnostiquer: on peut dire aisément si le test passe ou pas
- > significatif: un sponsor exigera une correction de l'application si le test ne passe pas

▷ Test d'installation

- > Installer sur diverses plateformes, à la recherche de différences dans l'installation ou le fonctionnement

▷ Test de charge

- > Mettre l'application face à un système qui l'utilise et le force à utiliser une grande quantité de ressources, à la recherche de points de vulnérabilité qui pourraient être exploités dans d'autres circonstances

▷ Test de longue séquences (ou d'endurance)

- > Utiliser le système pendant des jours voir des semaines, à la recherche de problèmes insidieux comme les fuites mémoires

▷ Test de performances

- > Tester le système afin de déterminer ses temps de réponse, à la recherche de besoins d'optimisation ou d'erreur de programmation

▷ Données auto-vérifiantes

- > Les fichiers de données contiennent l'information permettant de vérifier si les données en sortie sont corrompues.

▷ Comparaison avec des résultats sauvegardés

▷ Comparaison avec une spécification

▷ Cohérence heuristique

- > avec le comportement passé de l'application
- > avec l'image que l'entreprise souhaite projeter via cette application
- > avec des produits comparables
- > avec des déclarations concernant le produit
- > avec les attentes des utilisateurs
- > à l'intérieur du produit, avec des fonctions comparables
- > avec le but de l'utilisateur

▷ Comparaison basée sur un oracle

- > e.g: un programme alternatif calculant les résultats attendus

Comment qualifiez vous :

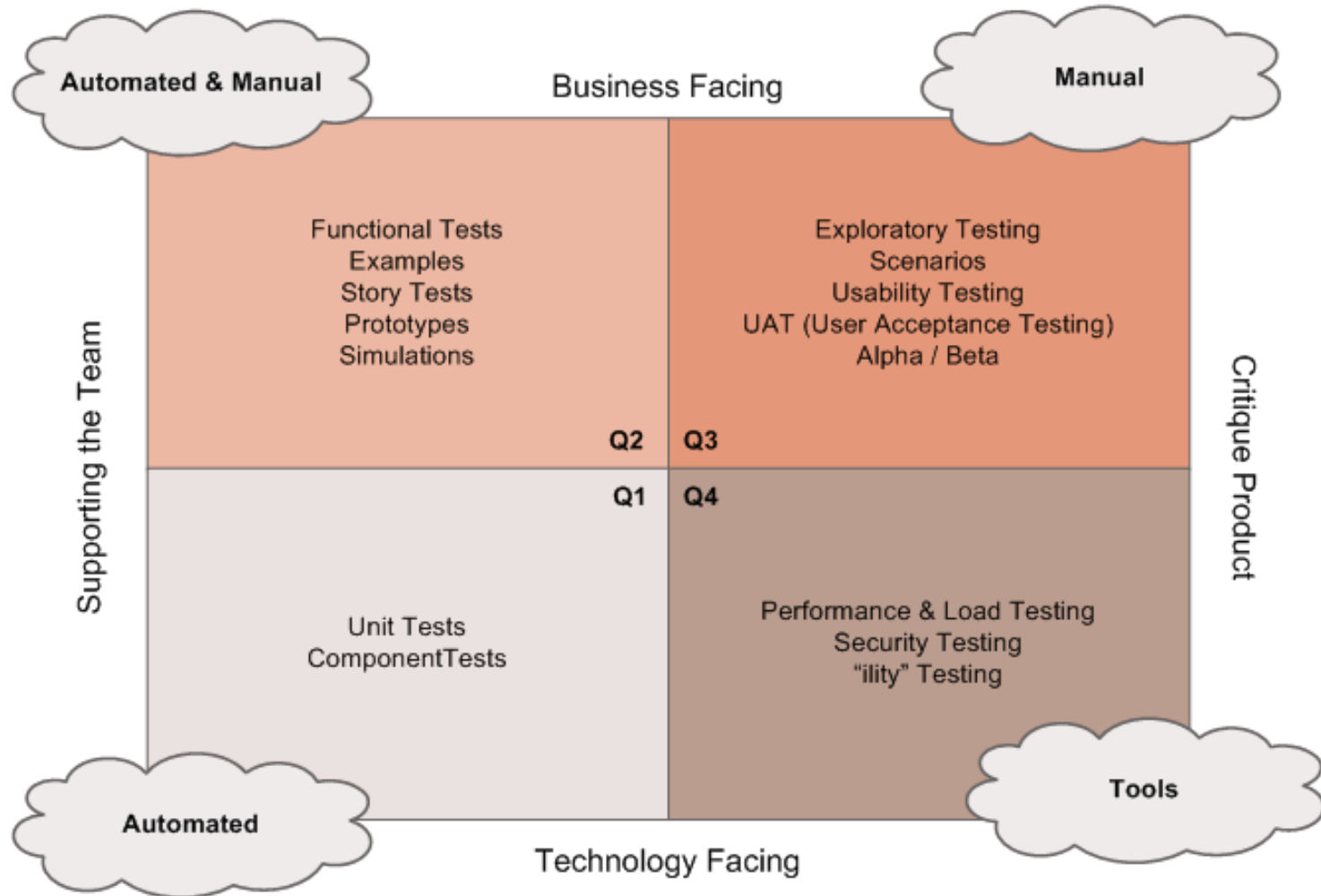
- > les tests d'intégration ?
- > les tests unitaires ?
- > les tests de non régression ?
- > les tests de sécurité ?
- > les tests de charge ?
- > ... ?



Chapitre 2 : Tests fonctionnels ?

Vérifier le comportement de l'application vis à vis de la spécification

Agile Testing Quadrants



source: Agile Testing – Lisa Crispin, Janet Gregory.



Chapitre 3 : Vers le Behavior Driven Development

Analyser une spécification

Nous allons implémenter un nouveau modèle de réduction sur une collection d'items

Un item seul sera à plein tarif

La commande de deux items différents déclenchera une réduction de 5%

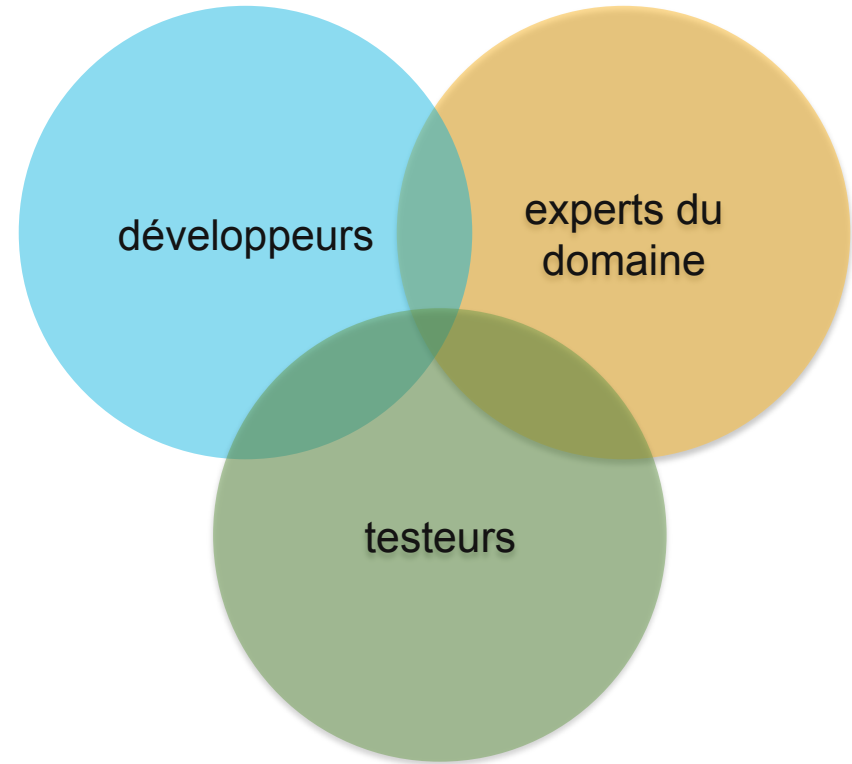
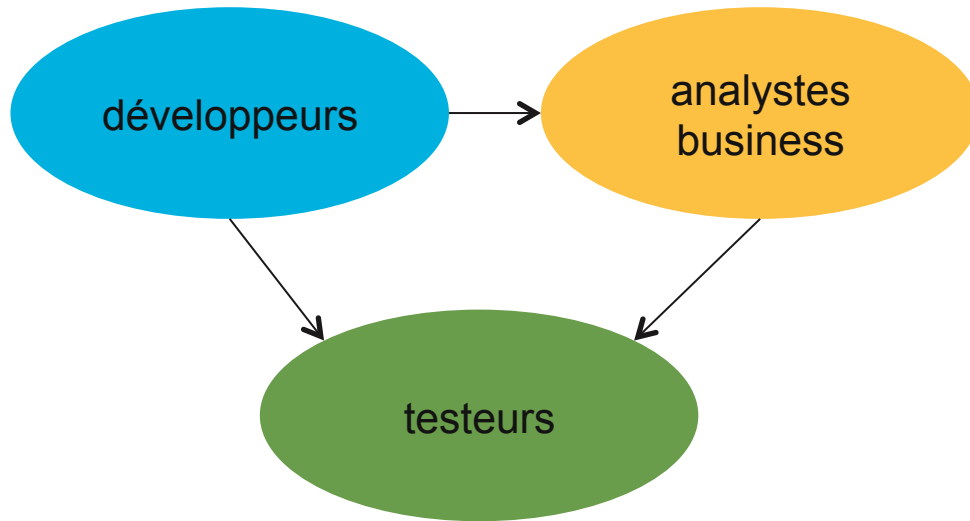
La commande de trois items différents déclenchera une réduction de 10%

La commande de quatre items différents déclenchera une réduction de 20%

La commande de la collection complète de 5 items déclenchera une réduction de 25%

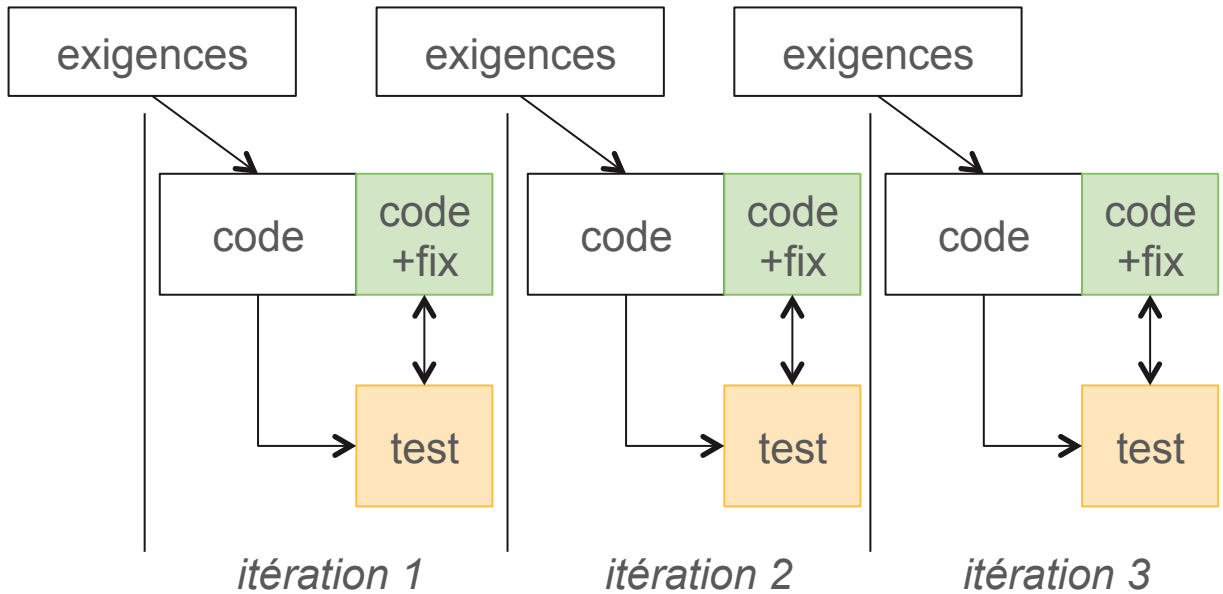


- > Dans les projets en cycle en V, les informations sont échangées de manière le plus souvent formelle.
- > Les activités sont encloses dans la définition des responsabilités.
- > Le document est l'aliment du projet.

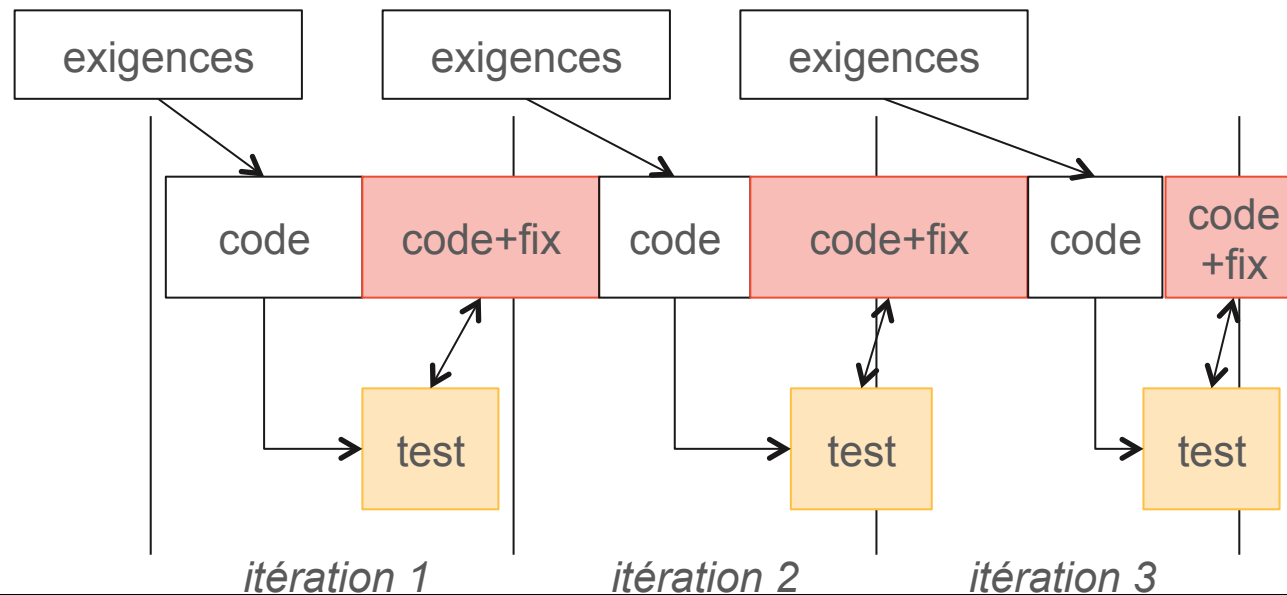


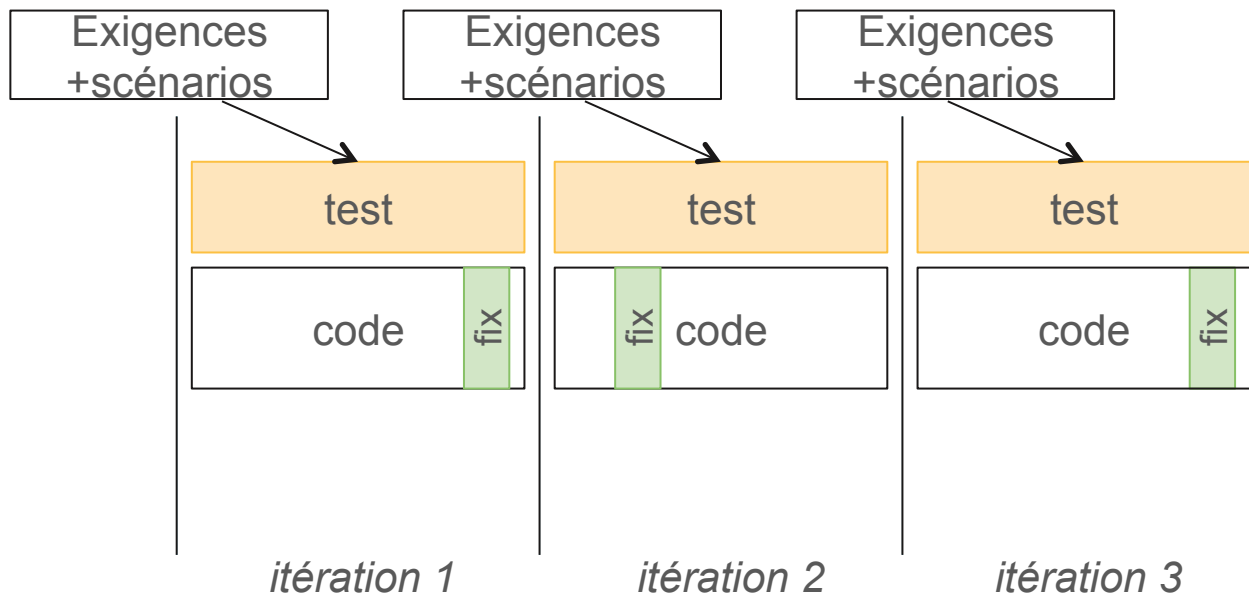
- > En Agile, les informations sont échangées de manière le plus souvent informelle
- > Les activités sont légèrement redondantes entre acteurs du projet.
- > Le document sert de support.

> Sur le papier..

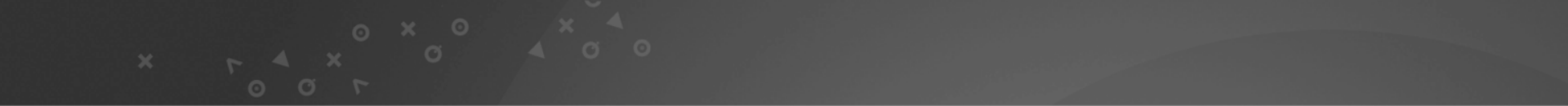


> En pratique:





- ▷ **Combiner Tests et Code tout au long de l'itération**
- ▷ **Facteurs de succès :**
 - > Approche « Test First »
 - > Communication en osmose
 - > « Les trois amis » (PO / Testeur / Développeur)
 - > Binômage
 - > Le « bug tracker » est un outil de documentation pas de communication.



I believe that the hardest part of software projects, the
Most common source of project failure, is
communication with the customers and the users of a
software

By providing a clear yet precise language to deal with
domains, a DSL can help improve this communication

Martin Fowler

De la spécification

ambiguë

▷ Langage naturel

▷ Forme libre

▷ Démarche top-down

à la story

univoque

▷ Vocabulaire partagé

▷ Expression standardisée

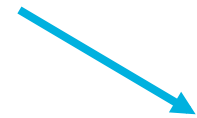
▷ Construction collaborative

Relation humaine

Solution technique

Spécification

exécutable, répétable, automatisable, autoportante



Transparence

Économie

Non-régression

Documentation

Chapitre 4 :

Gherkin

Feature / Fonctionnalité
Scénario

Given / Etant donné que
When / Quand
Then / Alors

Fonctionnalité : Déstockage d'inventaire

Scénario : Déstockage lors d'une commande avec un seul type d'article

- > **Etant donné que** le stock de concombres est de 3 unités
- > **Quand** je vends 2 concombres
- > **Alors** le stock de concombres est de 1 unité

Exprimer en Gherkin la
spécification sur les réductions

Exprimer en Gherkin les
règles du jeu Polar Bears

Chapitre 4 :

Specflow

- > Fonctionnement
- > Hello World
- > Test Jaune
- > Test Rouge
- > Test Vert
- > Paramétrisation
- > Tables
- > Contexte
- > Outlines
- > Hooks
- > Structure des features
- > Structure des steps
- > Partage des informations au cours d'un scénario

Nous allons implémenter un nouveau modèle de réduction sur une collection d'items

Un item seul sera à plein tarif

La commande de deux items différents déclenchera une réduction de 5%

La commande de trois items différents déclenchera une réduction de 10%

La commande de quatre items différents déclenchera une réduction de 20%

La commande de la collection complète de 5 items déclenchera une réduction de 25%



Chapitre 5 : Mises en situation



Le but est de réaliser une MMF avant sa date de release

Une MMF est composée de stories

Une story met un nombre de jours à être réalisée

Un développeur fait une action par jour

- Travailler sur une story
- Corriger un bug
- Faire un dojo

Une fois qu'un dev travaille sur une story, il doit la terminer avant de passer à autre chose

Une story bien réalisée amène un point de dev

Certaines stories ne peuvent être réalisées qu'avec la compétence nécessaire

- MainFrame-> Legacy
- Front -> AngularJS

Le dev peut dépenser des points de dev pour acquérir une compétence ou des objets

A la fin de la journée, le Crocto mange le développeur ayant le moins de point de dev

La partie se termine si il n'y a plus de développeur ou si il n'y a plus de story ;o)



`https://github.com/jak78/SpecflowDojoSkeleton/tree/debut_formation`

Bien prendre la branche « `debut_formation` »



ONLINE
SHOPPING

US 01: En tant que client je commande mes courses sur internet afin de gagner du temps

▷ Critères de succès

- > je crée une nouvelle commande
- > je sélectionne des articles (je peux modifier leur quantités) qui s'insèrent dans ma commande
- > je choisis une date de livraison
- > je valide et je paye ma commande
- > je reçois un mail de confirmation

US 02: En tant que client je paye avec un chèque cadeau, afin de profiter du cadeau qui m'a été fait

▷ Critères de succès

- > je crée une nouvelle commande et sélectionne des articles (je peux modifier leur quantités) qui s'insèrent dans ma commande
- > je saisis un numéro de chèque cadeau
- > le montant à payer de ma commande est diminué du montant du chèque cadeau

US 03: En tant que client je profite d'une remise sur le transport afin de faire des économies

▷ Critères de succès

- > je crée une nouvelle commande et sélectionne des articles, pour un montant > 200€
- > les frais de transport sont réduits de 20%
- > On applique ensuite une réduction croissante par tranche de 100 €

US 04: En tant que client je paye avec un coupon afin de faire des économies

▷ Critères de succès

- > je crée une nouvelle commande et sélectionne des articles
- > sur la ligne correspondant à l'article du coupon, je clique sur « coupon »
- > je renseigne le numéro de coupon
- > le prix de mon article est diminué du montant du coupon

US 05: En tant que client je parraine un nouveau client afin de faire des économies

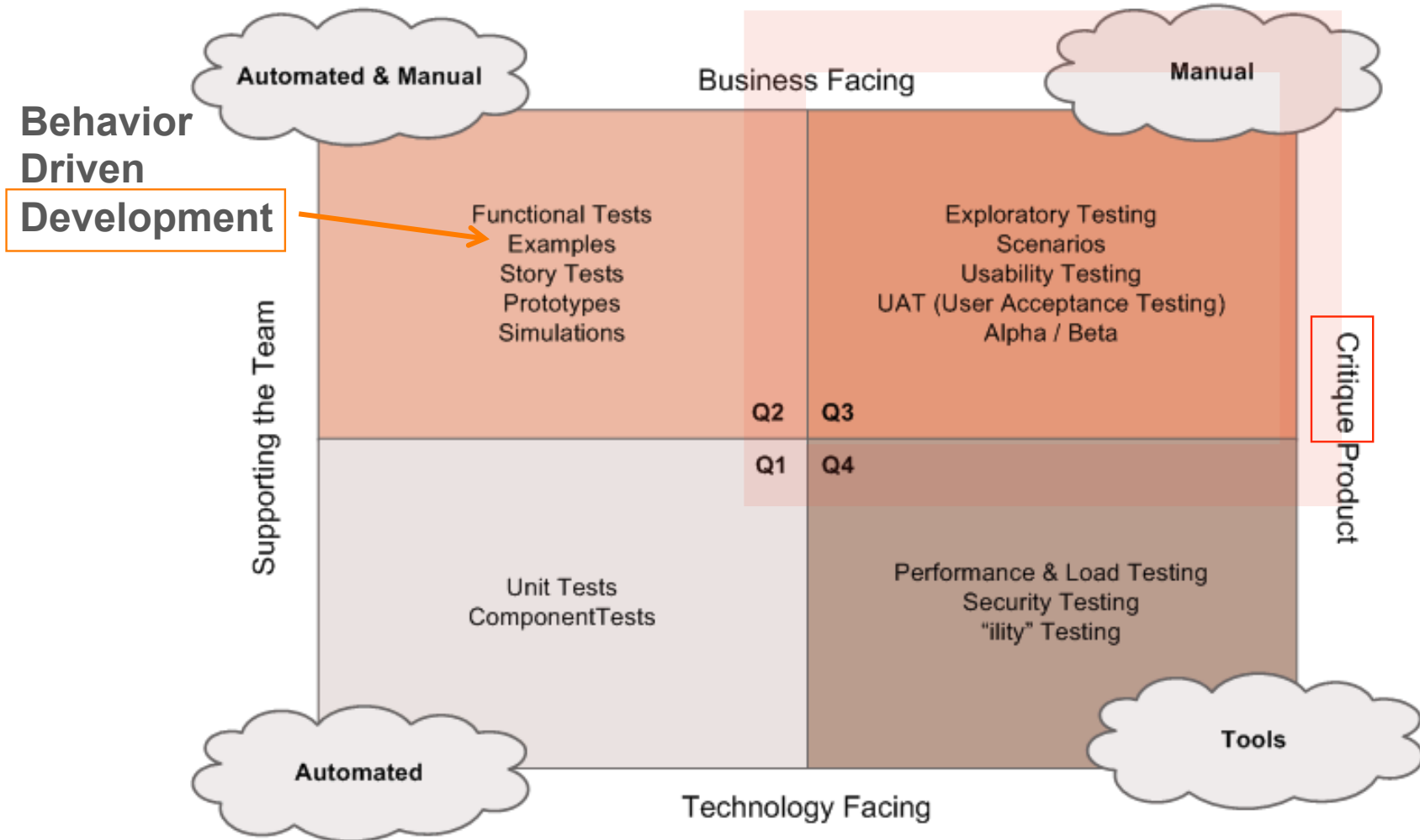
▷ Critères de succès

- > je sélectionne la fonction « parrainer » dans le menu principal
- > je renseigne les coordonnées du client que je parraine
- > après confirmation du client, mes dix prochaines commandes bénéficient d'une remise parrainage de 5%



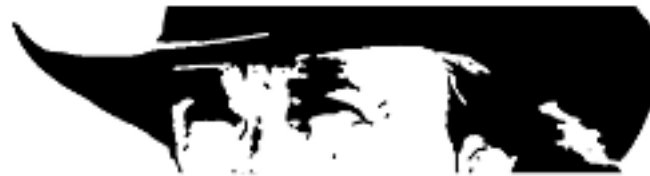
Chapitre 6 : Tests Fonctionnels pour Critiquer le Produit

Agile Testing Quadrants



source: Agile Testing – Lisa Crispin, Janet Gregory.

Votre entreprise veut intégrer un élément différenciant dans sa 'homepage' :



T H E I N T E R N E T C H U C K N O R R I S D A T A B A S E

Chuck Norris types with one finger. He points it at the keyboard and the keyboard does the rest.

<http://www.icndb.com/api/>

C'est techniquement possible:

C# API Wrapper for the Internet Chuck Norris Database: <http://www.icndb.com/api/>

```
var response = ChuckNorris.API.Random(exclude: new string[] {"explicit"});  
var joke = response.Result;  
  
Console.WriteLine(joke.Text);
```

Question: est-ce que cela peut aller en production ?

(le PDG est d'accord avec l'idée).

- ✓ **Etudiez** la spécification de la base de données Chuck Norris Internet.
- ✓ **Relevez** tous les éléments de fonctionnalités qui sont à tester.
- ✓ **Elaborez** un **plan de tests**.
- ✓ **Positionnez** chaque test de votre plan sur les 5 dimensions du test.

1. Qui: La personne qui effectue les tests

le développeur
le product owner, le testeur,
un utilisateur

...

2. Quoi: Que couvre le test

chaque fonction
les services
de bout en bout

...

3. Pourquoi: Problèmes potentiels recherchés par les tests

valeurs extrêmes
régressions
erreurs fréquentes

...

4. Comment: Activités de test

tests automatisés
tests exploratoires

...

5. Evaluation: Critères de passage des test

comparaison avec un résultat attendu
comparaison avec le résultat d'une exécution précédente

...

- Comment limiter le nombre de tests ?
- Quels tests devraient être automatisés ?
- Comment automatiser les tests ?
- Mettez en œuvre les tests
- Décrivez les bugs rencontrés.

Apportez vos propres tests:

- Soit un scénario utilisateur sur lequel il faut écrire une spécification par l'exemple.
- Soit une spécification pour laquelle il faut élaborer des tests fonctionnels.

- Qu'est-ce que j'ai appris ?
- Qu'est-ce que ça change ?
- Quelle est la prochaine étape ?