Test Driven Development

ou comment tester en développant

Frédéric Delorme
frederic.delorme@gmail.com

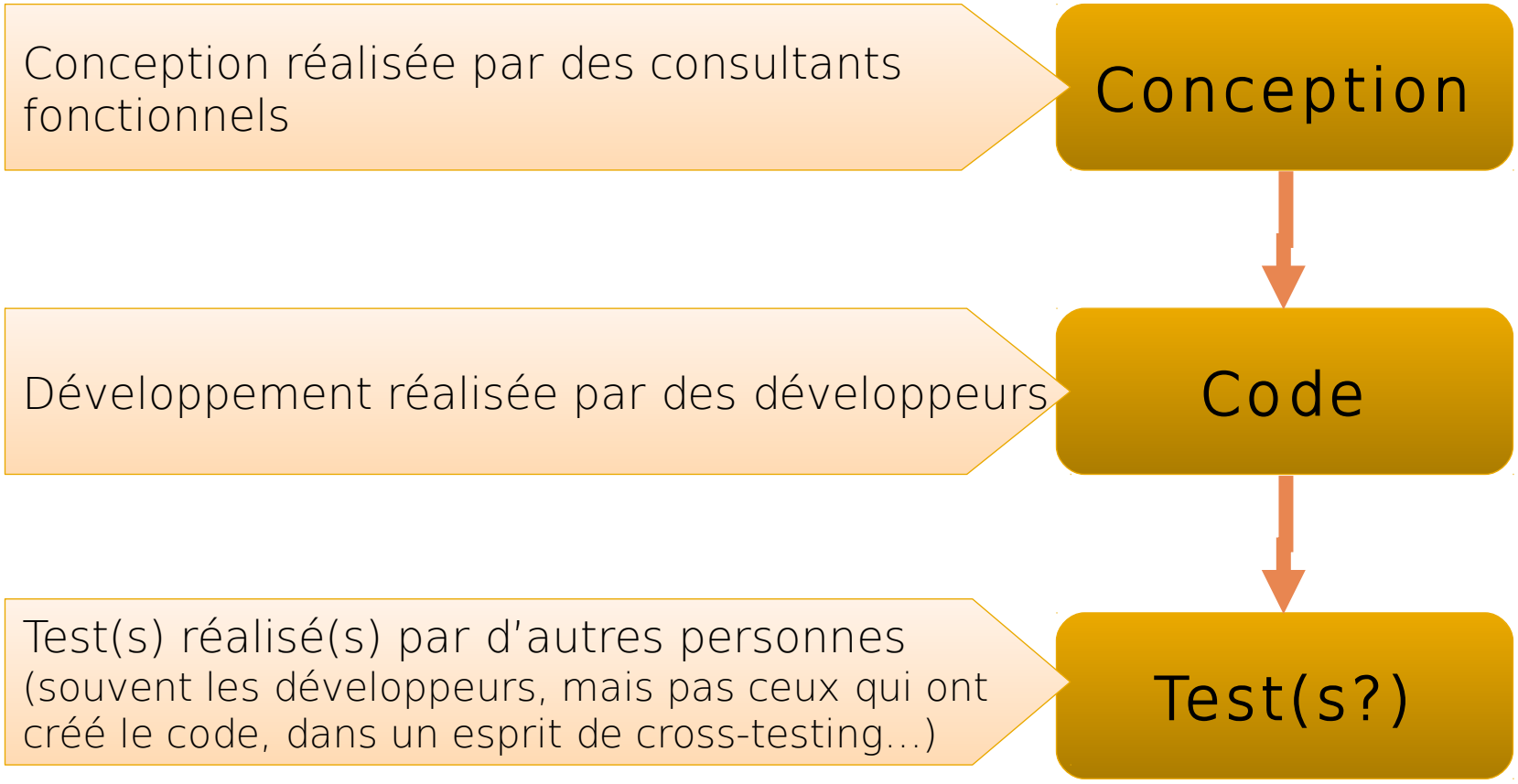
Décembre 2011

ubuntu

TDD: qu'est-ce ?

- La méthode TDD, **T**est **D**riven **D**evelopement ou développement piloté par les tests est une méthode de développement (comme son nom l'indique) **imposant la formalisation du besoin technique** avant son développement, et ce en utilisant des outils **permettant de rejouer et vérifier** à volonté **la conformité de son implémentation**.
  - Sur la plateforme Java, cet outil est souvent **JUnit** (http://www.junit.org).

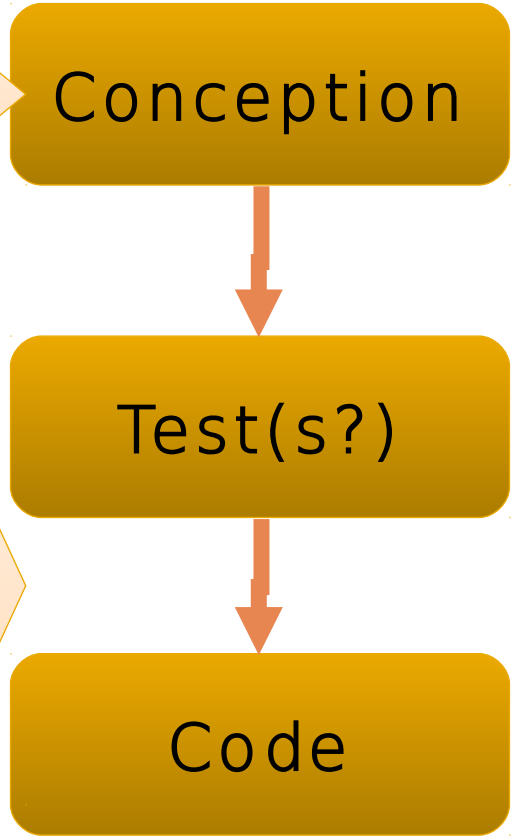
Habitude du développeur



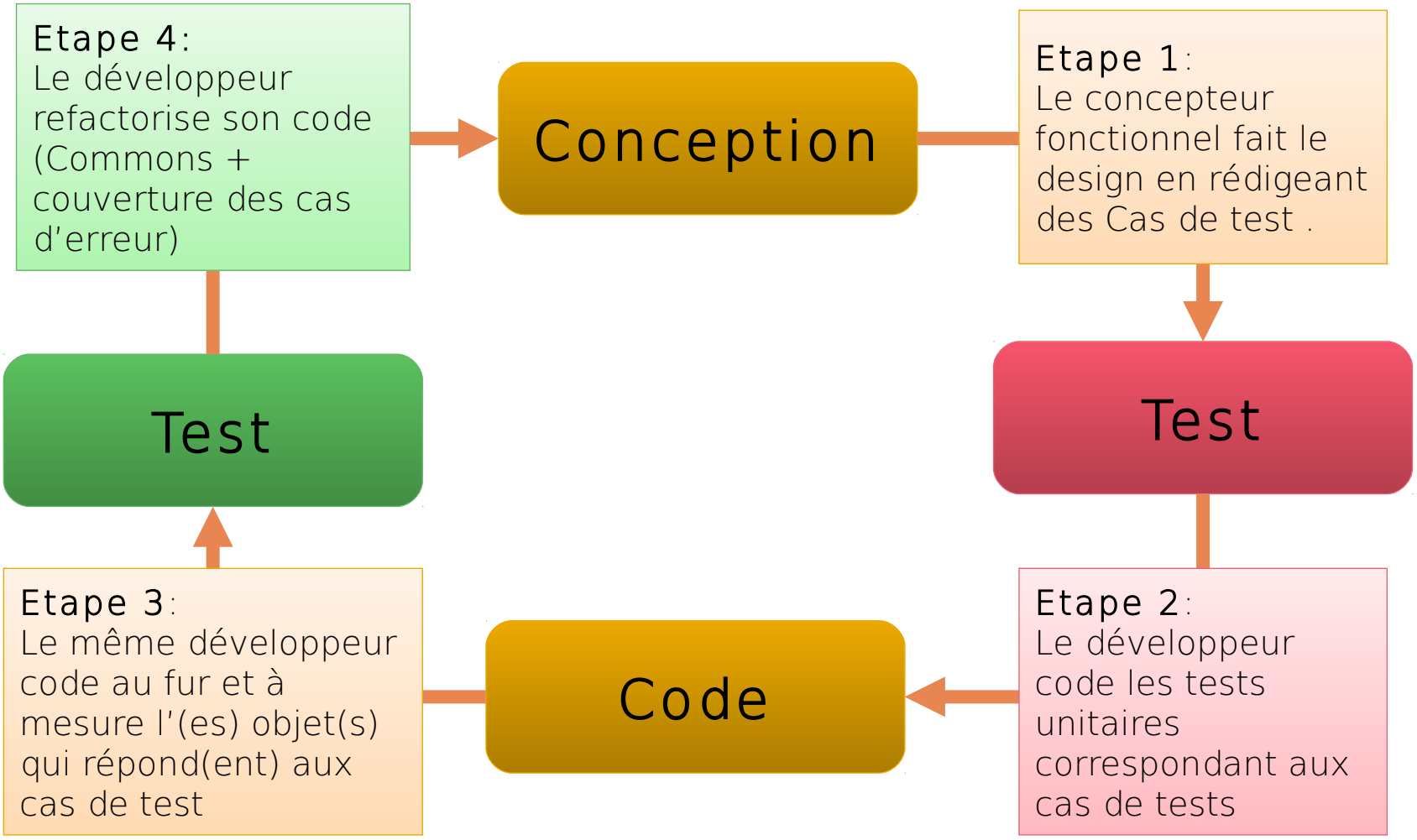
La vision TDD

Conception réalisée par des consultants fonctionnels

Tests et Développement (dans l'ordre) réalisés par la même personne pour un même composant !



Le cycle initial du TDD

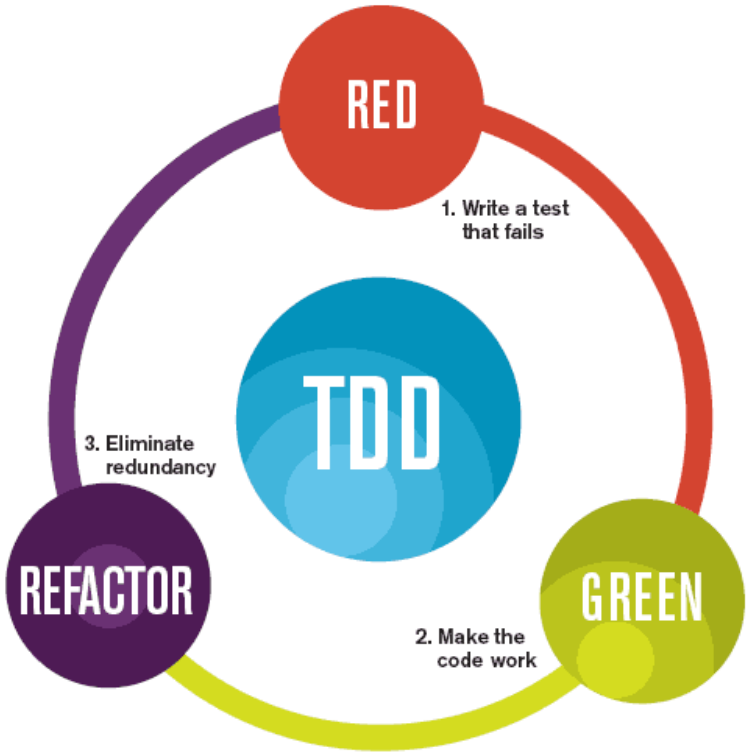


Démarche (1/2)

- Ici, le développeur écrit un peu de code de test et ensuite implémente le métier.
- Le code métier est soumis aux tests puis il est corrigé jusqu'à ce que ses tests soient validés.
- Le développeur procède éventuellement ensuite à un `refactor` du code qui est un autre principe de l'*Extreme Programing* permettant d'améliorer la qualité interne du code en procédant à plusieurs opérations (renommage de méthodes, suppression de codes inutiles,...).

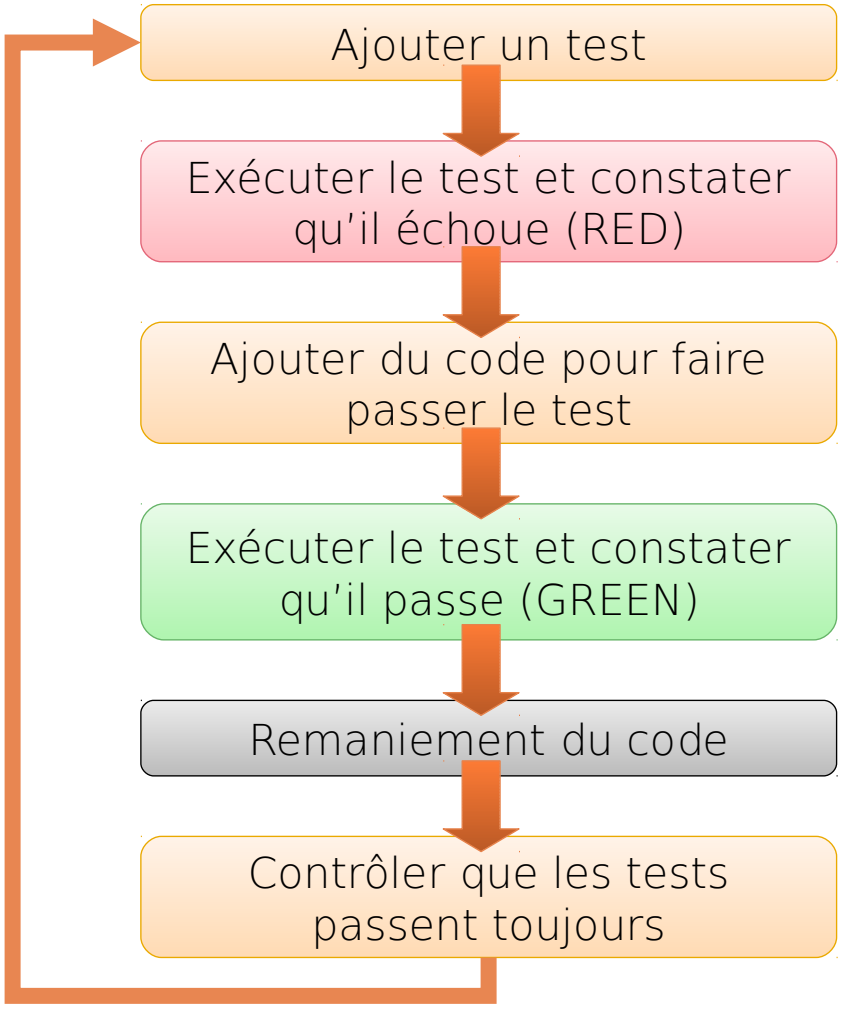
Démarche (2/2)

- Décomposée en trois phases appelé **RGR**
  (aussi appelé la **Mantra**).
  Les deux premières phases sont nommées d'après la couleur de la barre de progression dans les outils de tests unitaires comme Junit
- R (Red): écrire un code de test et les faire échouer
- G (Green) : écrire le code métier qui valide le test
- R (Refactor) : remaniement du code afin d'en améliorer la qualité



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

En clair


Ajouter un test
Exécuter le test et constater qu'il échoue (RED)
Ajouter du code pour faire passer le test
Exécuter le test et constater qu'il passe (GREEN)
Remaniement du code
Contrôler que les tests passent toujours

Avantages : Tests écrits !

- La mise en place de la méthode TDD offre de nombreux avantages au sein du développement d'un logiciel.
- Voici les avantages apportés par l'emploi de cette méthode

- **Les tests unitaires sont réellement écrits**
  On remarque généralement que les tests unitaires sont souvent remis à plus tard (par manque de temps ou d'intérêt). Le fait de commencer par rédiger les tests permet de s'assurer que les tests seront écrits.

Avantages : Satisfaction

- La satisfaction du développeur permet d'obtenir un code plus cohérent
  En suivant la méthode traditionnelle le développeur écrit une unité et va ensuite procéder aux tests afin de s'assurer que ce qu'il a codé est valide. Cette méthode est peu satisfaisante pour un développeur car il est déjà conscient que le code qu'il a écrit est juste. A l'inverse, en appliquant la méthode TDD, il va commencer par rédiger les tests puis ensuite passer à l'implémentation de l'unité afin de faire passer les tests. Ce procédé est plus satisfaisant et motivant car il s'agit d'une sorte de défis de faire valider les tests, C'est une sensation d'accomplissement. Il est important de ne pas ignorer les aspects psychologiques si on veut s'assurer que le travail réalisé par les développeurs soit propre et efficace.

Avantages : Clarification

- **Clarification des détails de l'interface et du comportement**
  En effet, lorsque le développeur écrit du code de test pour tester une implémentation qui n'existe pas encore, il va devoir penser aux détails de la méthode dont il a besoin pour écrire la spécification.

  Aussi, il va s'interroger sur le nom de la méthode, sa(ses) valeur de retour, ses paramètres, comportement,...

  Cela permet de clarifier la conception et d'écrire seulement du code utile.

Avantages : Rejouabilité

- **Vérification démontrable, répétable et automatisé**
  Le fait de disposer d'un grand nombre de test permet de s'assurer de la solidité et garantie du code.

Avantages: Non régression

- **Non présence de régression**
  Lorsqu'un développeur modifie une méthode existante, il peut relancer les tests unitaires afin de s'assurer que sa modification n'a pas impacté l'existant et donc cela offre un feedback immédiat.

Comment procède-t-on ?

- Dans le monde Java: utilisation d'API
  - JUNIT
    - Pour le code des classes de tests unitaires
  - DBUNIT
    - Pour le setting de données dans une base
  - HSQLDB (par exemple)
    - Comme moteur de base de données en mémoire (pour le temps d'exécution des tests)
    - Chaque test peut avoir son jeu de données isolé.

Travaux pratiques

- Exemple 1 :
  Implémentation d'une classe de service de compte bancaire (basique)

- Exemple 2:
  Implémentation d'une DAO pour un objet persistant et ajout d'un jeu de données de test.