Gestion de Projet Agile

Tests, certifications et plan de tests

Tianxiao.Liu@u-cergy.fr

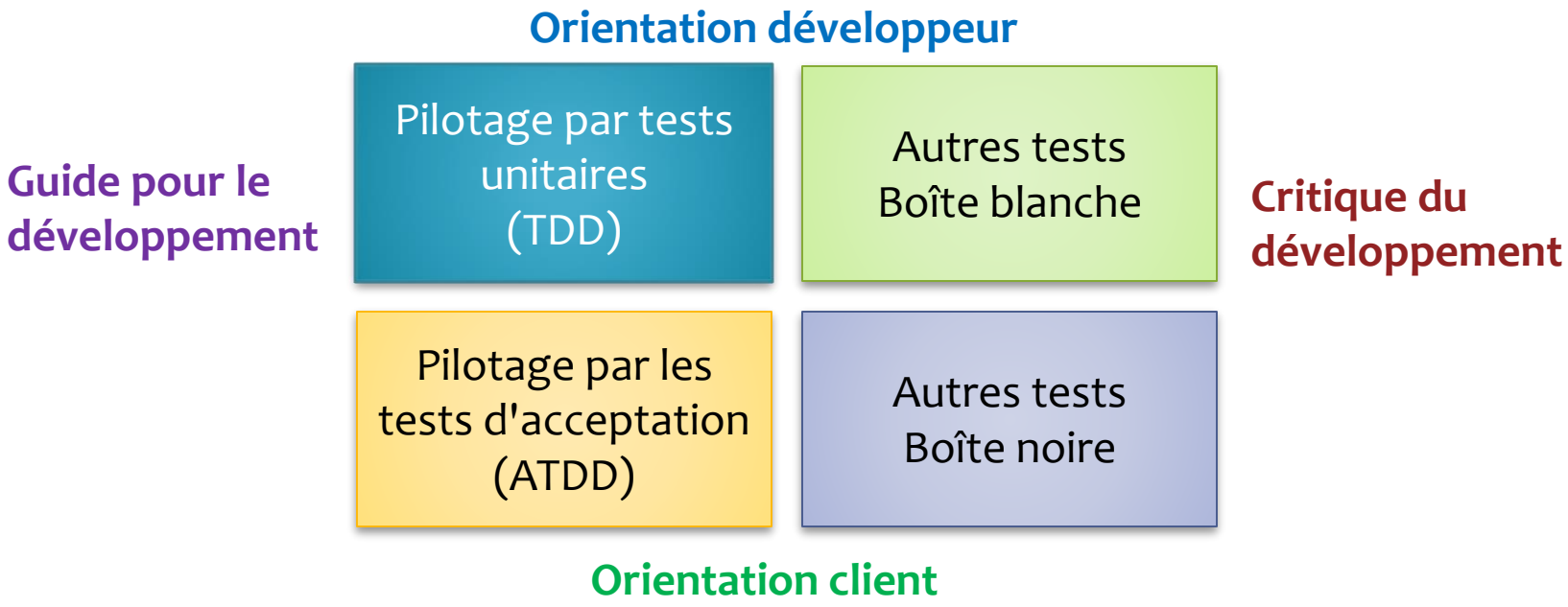
Master IISC pro 2e Année

Plan

- Accepter une story par les tests
- Définir les tests
- Méthode de tests
- Plan de tests

Tests pour accepter les stories

- Principe
  - Tests d'acceptation pour les stories réalisées avec du développement → **certifier** la story



Une story et ses tests

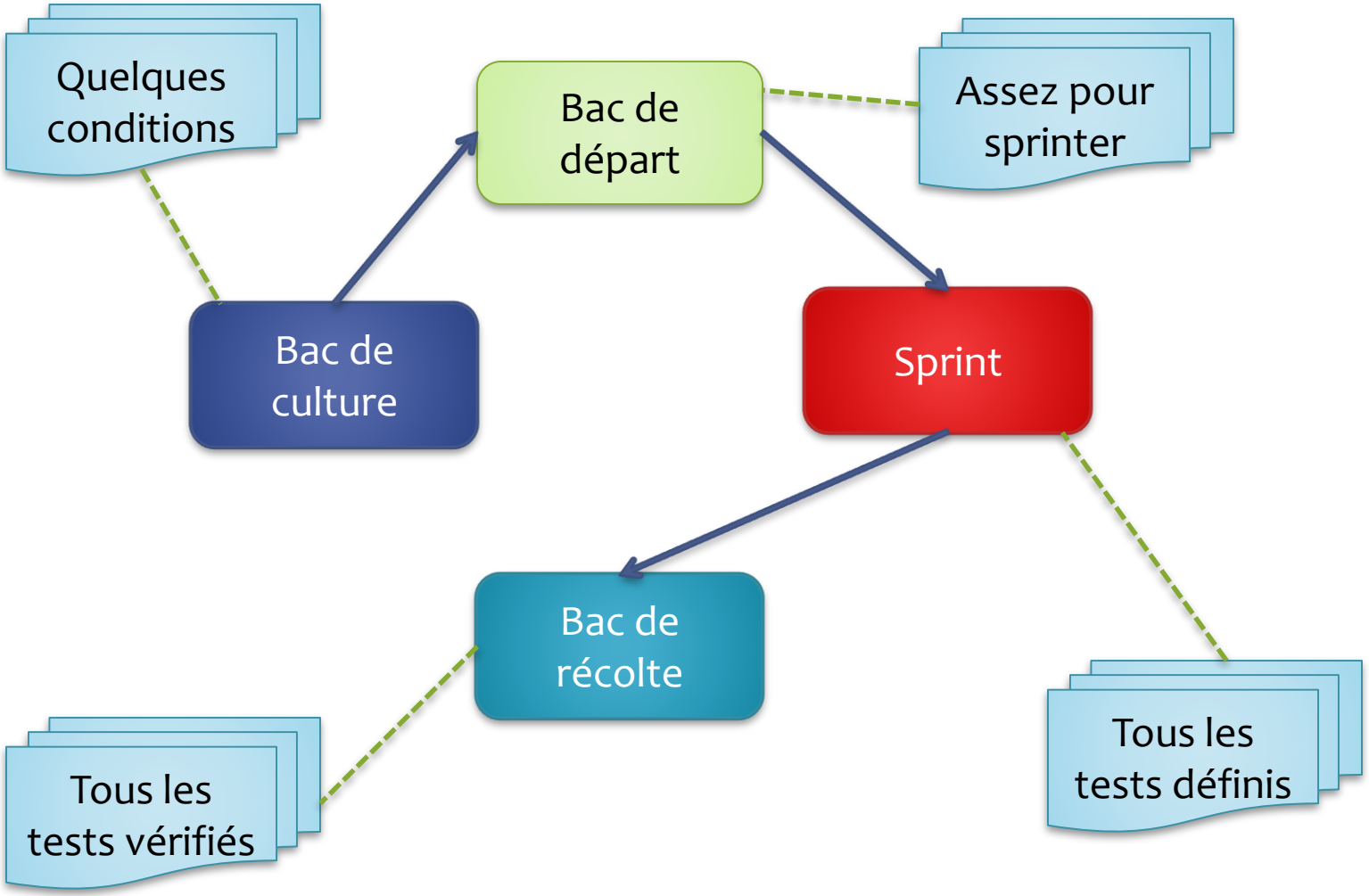
- Principes de base
  - ① Une user story doit posséder au moins une condition d'acceptation.
  - ② A cette condition, sont associés un ou plusieurs tests.
  - ③ Un test d'acceptation décrit l'exécution de la story.

  - **Remarque : Un nombre trop important de tests → la story a une trop grande complexité → à décomposer**

Définition d'un test

- Méthode BDD (***Behavior Driven Development***)
- Chaque test est formalisé avec trois rubriques :
  - Etat du système **avant** l'exécution du test
  - Evénement qui **déclenche** l'exécution de la story
  - Etat du système **après** l'exécution
- Formalisme : ***Given When Then***
  - **Etant donné** un maître de chien de race
  - **Quand** le maître inscrit son chien à une confirmation
  - **Alors** il est informé de son inscription

Etapes pour accepter la story



Bac de culture : un « jardin » où on fait « pousser » les user stories

Quand passer les tests ?

- Principes de base
  - **Conception et réalisation d'une user story → basées sur les tests d'acceptation**
  - Pendant le développement
    - Modification et complétion des tests
    - Ajout de nouveaux tests
  - A chaque nouveau build, repasser tous les tests → éviter les régressions

Intégration continue ?

- Objectif
  - A la première itération, on passe T1 tests.
  - A la deuxième, on passe T2 **nouveaux** tests.

    …
  - On suppose que le nombre moyen de tests par itération est Ti, pour n itérations, on doit passer

    Total = Ti x n x (n+1) / 2

Exemple : Ti = 5 n = 3

5 + (5+5) + (5+5+5) = 30 5 x 3 x (3+1) /2 = 30

L'intégration continue a un coût, et l'automatisation des tests a un autre coût encore plus important.

Quelques conseils

- N’oubliez pas les tranches verticales
  - En référence aux architectures en couches
  - Vertical = à travers toutes les couches
  - Ex. de l'IHM jusqu'à la base de données
- Le test n'est plus une phase
  - ~~*En cascade*~~ → ***Agile***
- Se servir des tests pour communiquer !
  - Pont entre métier (client) et développement
- Rendre explicite le travail de test

Plan de tests à multi-dimensions

- Dimension *story* (quoi tester?)
  - Tester pour accepter les user stories
  - Relier les stories → dépendance ? Intégration ?
  - Ainsi que les features

- Dimension *type*
  - **Test unitaires** : réalisés par les développeurs, souvent longs, fastidieux, donc automatisés
  - **Tests fonctionnels** : une plus grande granularité que les tests unitaires, automatisable ou non.
  - **Tests de performance** : tests non fonctionnels
  - **Tests de scénarios** : véritables tests d'un point de vue de l'utilisateur

Plan de tests à multi-dimensions

- Dimension "temps"
  - La date et la fréquence de tests
- Dimension "stratégie"
  - Qui écrit (définit) et effectue les tests ?
  - En cas de "K.O.", que faire ?
- Dimension "environnement"
  - Définir une configuration

Remarque : plan de test évolutif → mise à jour fréquente