Tests, typologie & concepts

Plan de séance

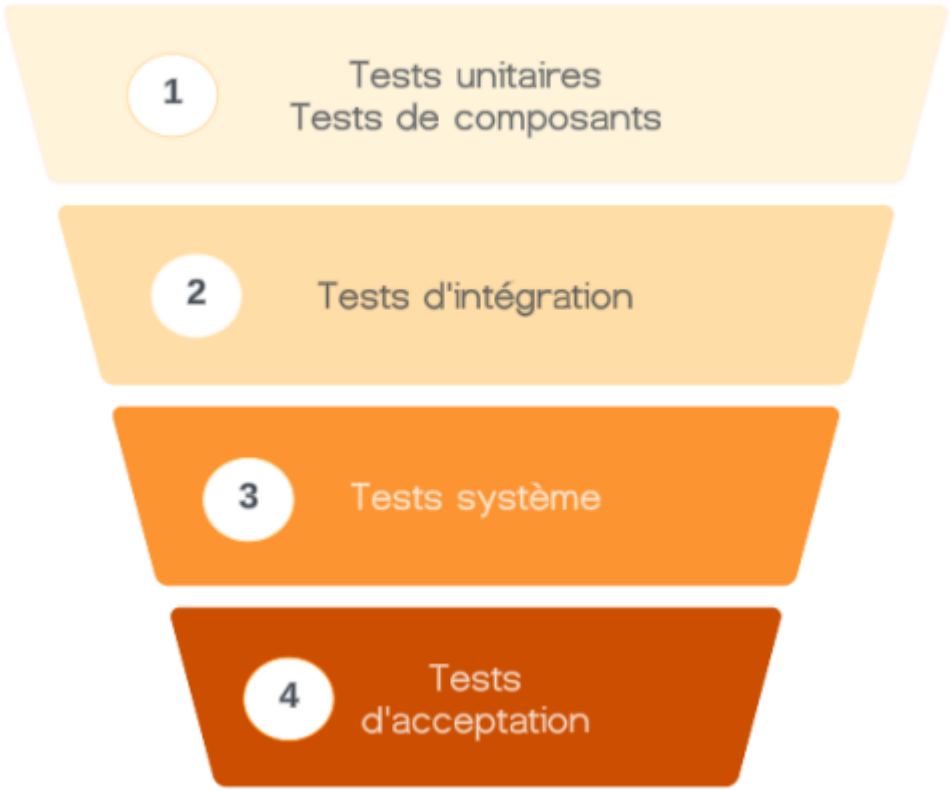
Ils contribuent à la qualité logicielle : QA Quality Assurance

- En vérifiant la satisfaction des exigences fonctionnelles ou techniques.
- En évitant la régression.

Certains doivent être manuels, d'autres peuvent être automatisés.

Niveaux de test

Selon l'ISTQB (International Software Testing Qualifications Board), il existe 4 niveaux de tests, classé selon leur portée (à ne pas confondre avec types de test).



1 - Tests unitaires

Ils sont qualifiés également de tests de composant et permettent de tester le comportement d'une unité de code :

- Méthode (procédure/fonction)
- Service, composant, module

Ils sont entièrement automatisés.

2 - Tests d'intégration

Ils permettent de vérifier l'assemblage de différents composants, et leur bonne interaction.

3 - Tests système

Leurs but est de vérifier que le système (le logiciel ou l'application dans son ensemble) répond aux exigences définies dans les spécifications.

Ils sont avant tout fonctionnels (tests de fonctionnalités) et doivent en partie être automatisés, pour contrôler et éviter la régression.

4 - Tests d'acceptation

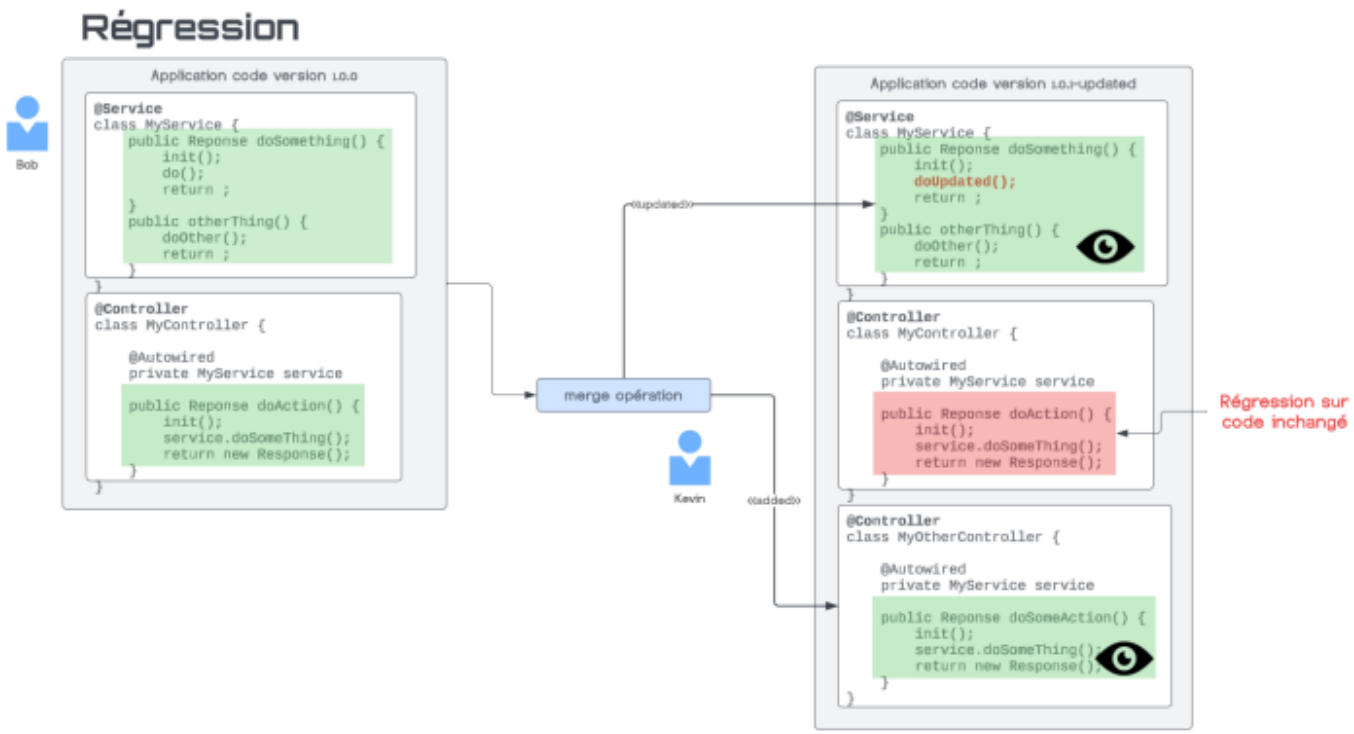
Ce sont les tests métier réalisés par le product Owner et les utilisateurs finaux.

Ils permettent de savoir si le produit livré répond aux exigences métier.

Ce sont des tests manuels.

Concepts clé

1 - Non régression



La régression se produit lorsqu'une évolution du projet :

- ajout de fonctionnalité,
- correction de bug,

- amélioration,

produit des effets négatifs sur une autre partie du code :

- Bug, dysfonctionnement,
- Fonctionnalité non satisfaite,

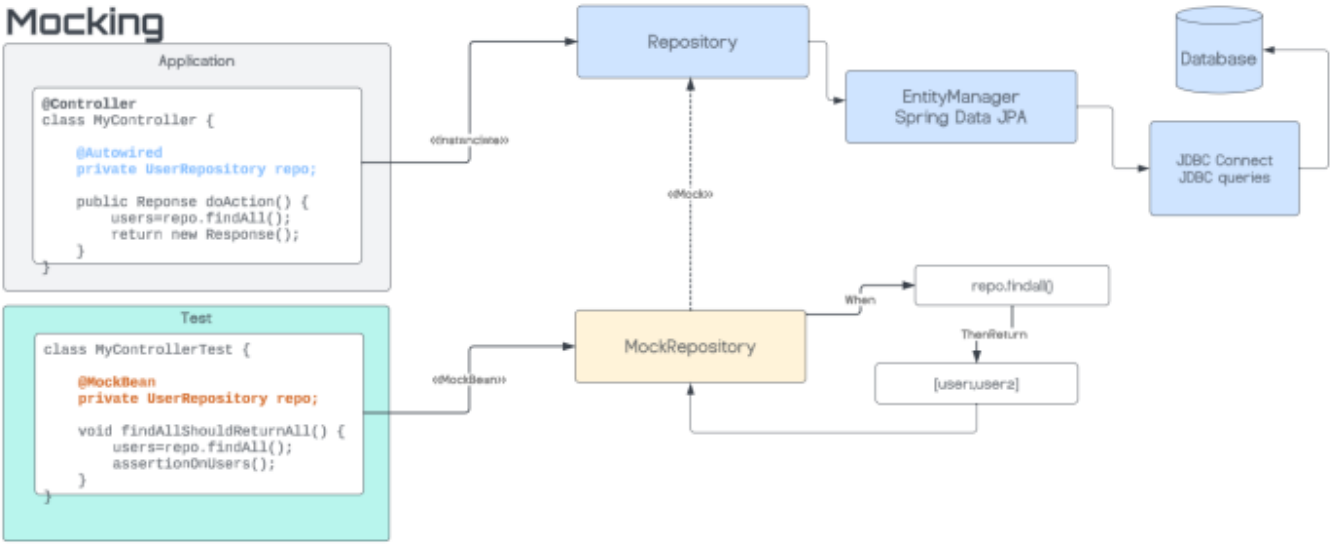
compte tenu des inter-dépendances.

L'absence de régression constitue la Non-régression.

2 - Mocking

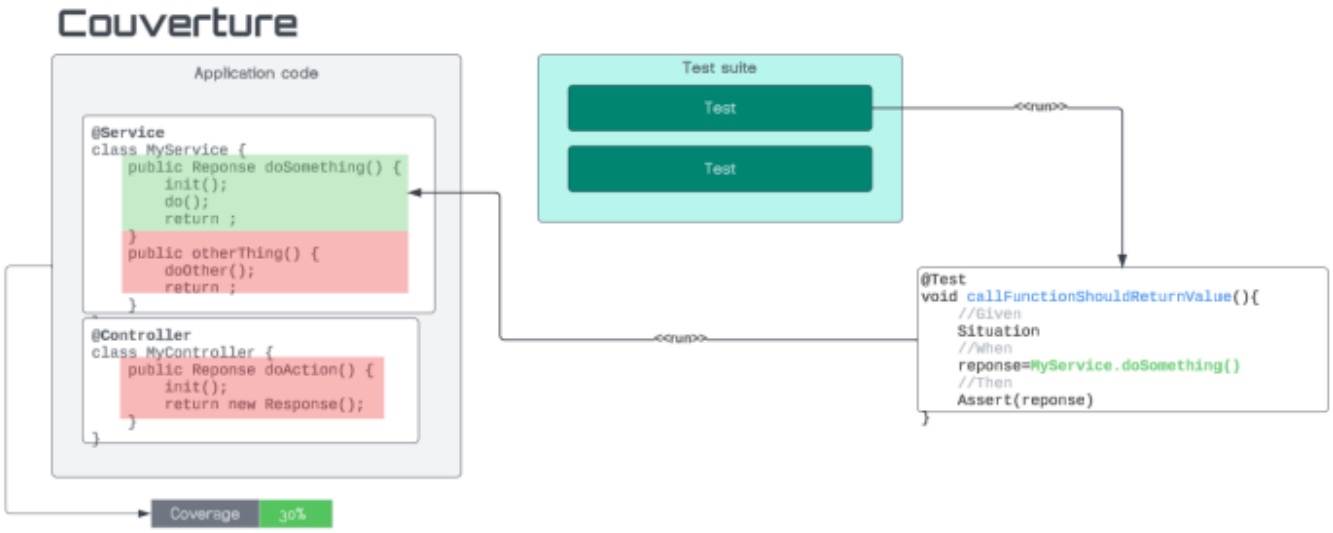
Les inter-dépendances limitent les possibilités de test :

L'isolation est difficile, voir impossible dans le cadre de tests unitaires et de composants. Les dépendances introduisent une complexité difficile à reproduire dans le cadre des tests. Les dépendances peuvent provoquer des faux-positifs ou des faux-négatifs dans les tests. Le mocking permet de résoudre ces problèmes, en utilisant des objets factices dans le cadre des tests, reproduisant les caractéristiques minimales des objets réels.



3 - Couverture

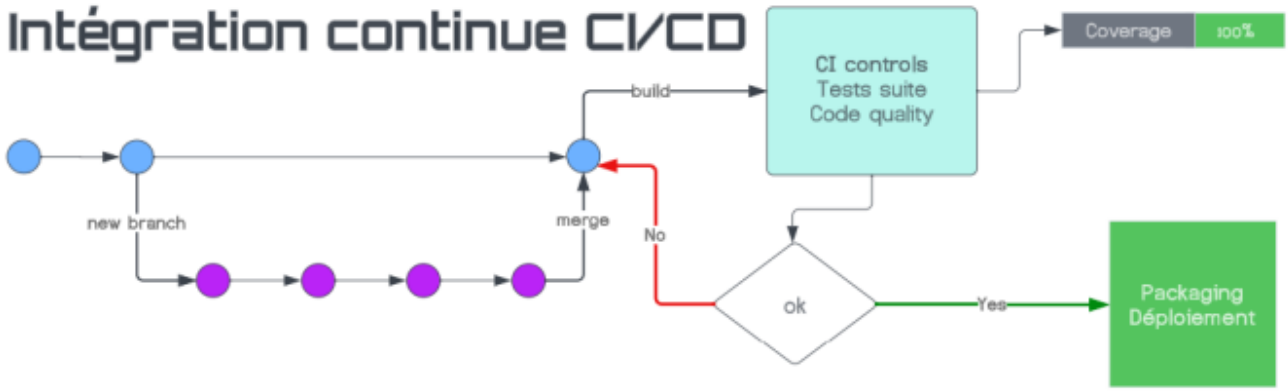
Elle définit le degré de couverture du code par les tests (exprimé en %).



4 - Intégration continue

Principe et outils permettant d'effectuer en continu (à chaque merge) les opérations de :

- test
- vérification de code
- packaging
- déploiement



Applications

Analogie (Estimé : 45 mn)



Par équipes de projet :

Depuis le document Tests logiciels- analogie

1. Lire les consignes
2. Compléter le document

Spring Implémentation (Estimé 3h00 et +)



Par équipes de projet (1 seul fork par équipe):

Lire le document Tests Spring

1. Créer un fork du repository Spring-tests
2. Créer :
    1. Test unitaire sur HelloService
    2. Test d'intégration avec Mocking sur UserRepository
    3. Test système avec MockMvc
    4. Test système avec Selenium
3. Pour chaque type de test, sur une nouvelle branche :
    1. Créer une classe factorisant les manipulations courantes (requête, récupération du contenu…)
    2. Evitant les imports statiques

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/qa/tests

Last update: **2023/12/18 16:20**