Les tests

Tests

Un projet informatique se doit d'être testé :

- Respect des spécifications
- Augmentation de la qualité de l'application
- Faciliter la maintenance
- Faciliter les évolutions futures

Tests

Définition :

« Un test en informatique est une vérification **partielle** du système visant à **valider** le fonctionnement de l'application et d'**identifier ses défauts** et d'**augmenter la qualité** finale du projet »

Tests

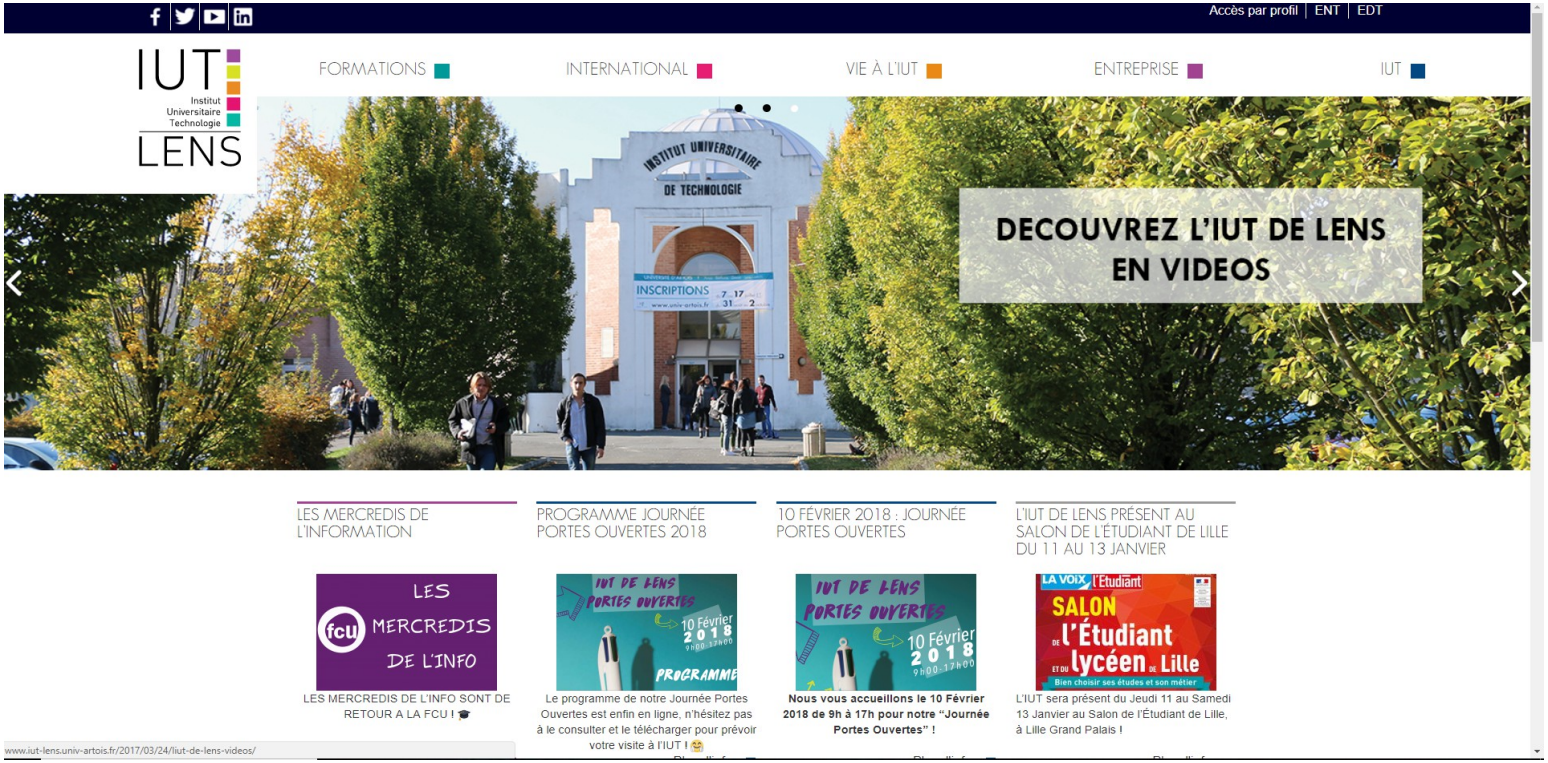
Deux façons de faire des tests :

- Tests manuels
  - Pas de développement supplémentaire
  - Peu être déroulé par n'importe qui
  - Traçabilité peu fiable
  - Long à dérouler
- Tests automatisés
  - Développement supplémentaire
  - Géré par les développeurs
  - Possibilité de loguer les tests
  - Déroulement automatique

Tests manuels

On teste à la main que l'application fonctionne.

- *On définit des procédures de vérification et on indique si oui ou non le test est un succès*



Tests manuels

Lister les tests à faire

→ *Passer par un cahier de tests, répertoriant :*

- Le nom du test
- Les actions à dérouler
- Les prérequis
- La date du test
- Le statut du test (OK/KO)
- Un commentaire

Nom	Actions	Prérequis	Date	Statut	Commentaire
Page d'accueil	Aller sur http://qlf.monbeausite.priv.sigs.com	NA	11/02/17	OK	
Page de création de compte	Aller sur https://qlf.monbeausite.priv.sigs.com/create-customer	NA	11/02/17	OK	
Création de compte	Remplir chaque champ du formulaire. Appuyer sur « Créer »	Ne pas avoir l'adresse mail déjà en base de données	11/02/17	KO	Bouton créer ne fonctionne pas

Tests manuels

Avantages :

- *Il n'est pas nécessaire d'avoir des connaissances informatiques*
- *Le fichier est facile à partager. Le client peut valider ce fichier*
- *Mis à part le fichier à initialiser, peu de temps à passer pour définir les tests*

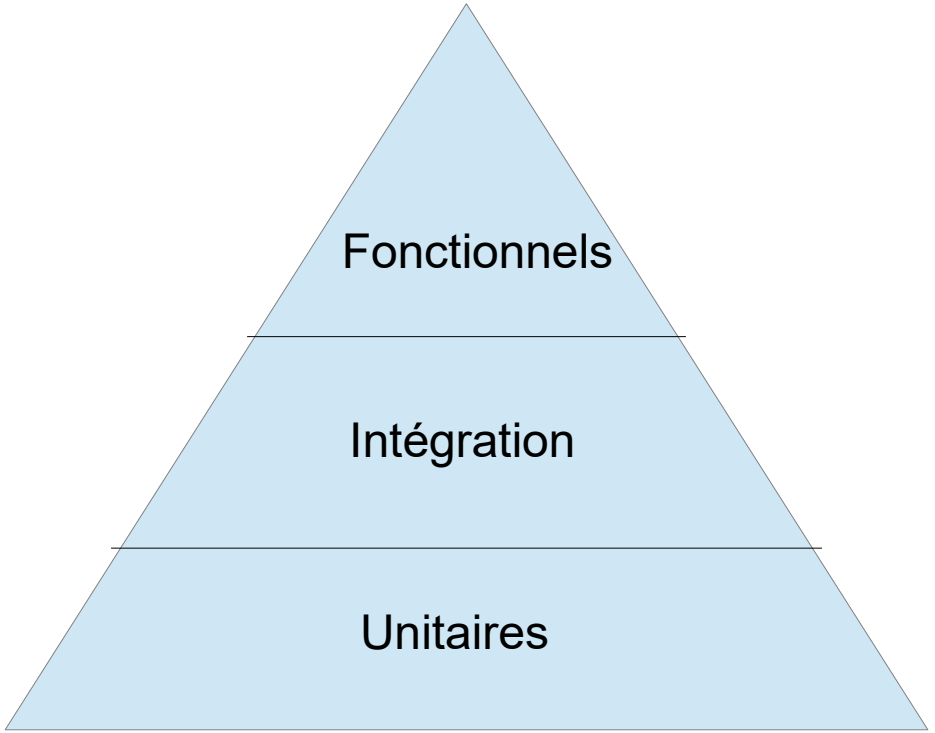
Inconvénients *:*

- *Prennent du temps à dérouler*
- *Demande des ressources humaines*
- *Erreurs de manipulation*

Tests automatiques

Il existe plusieurs types de tests automatiques :

- *Tests unitaires : testent chaque fonction*
- *Test d'intégration : testent que les briques de l'architecture fonctionnent bien ensemble*
- *Test fonctionnels : testent les fonctionnalités de l'application*



Tests unitaires

Les tests unitaires sont la base de la pyramide de tests.

Ils sont le minimum vital pour s'assurer que l'application fonctionne correctement

Les caractéristiques des tests unitaires sont les suivantes :

- *Ils doivent tester **une** fonction*
- *Ils doivent être **reproductibles***
- *Ils doivent être **compréhensibles***
- *Ils doivent être **documentés***
- *Ils doivent être **indépendants***

Tests unitaires

```
describe('stringUtils',function() {
    it('should a concat of the strings', function() {
        let result = concatStrings('a','b');
        expect(result).to.be.equal('ab');
    }
});
```

```
describe('Users',function() {
    it('should get user toto from database', function() {
        let result = db.get({'id' : 'toto'));
        expect(result.id).to.be.equal('toto');
    }
});
```

Tests unitaires

Comment déterminer le nombre de tests unitaire à développer ?

- *Au moins un test par fonction*
- *Faire suffisamment de tests pour tester toutes les lignes de la fonction*
- *Tester les cas particuliers (valeurs « null », tableaux vides...)*

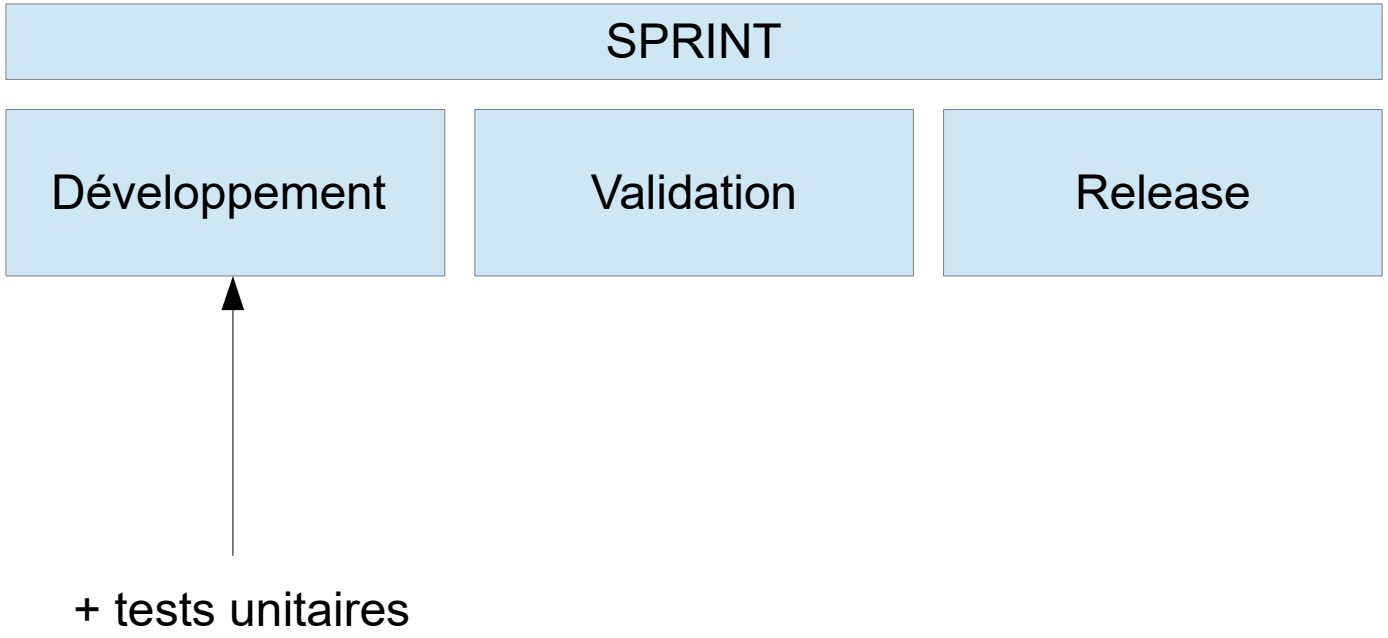
```
function absolute(a) {
   if(a > 0)
      return a;
   else
      return -a;
}
```

Tests unitaires

- *Avantages des tests unitaires*
  - *Généralement rapides à écrire*
  - *Ne nécessite pas que l'ensemble de l'application soit développée*
  - *Donne une base de la solide à l'application*
- *Ce que ne font pas les tests unitaires*
  - *Tester l'interaction entre les fonctions*
  - *Tester les fonctionnalités finales de l'application*

Tests unitaires

- *Place des tests unitaires dans un projet AGILE :*
  En même temps que le développement



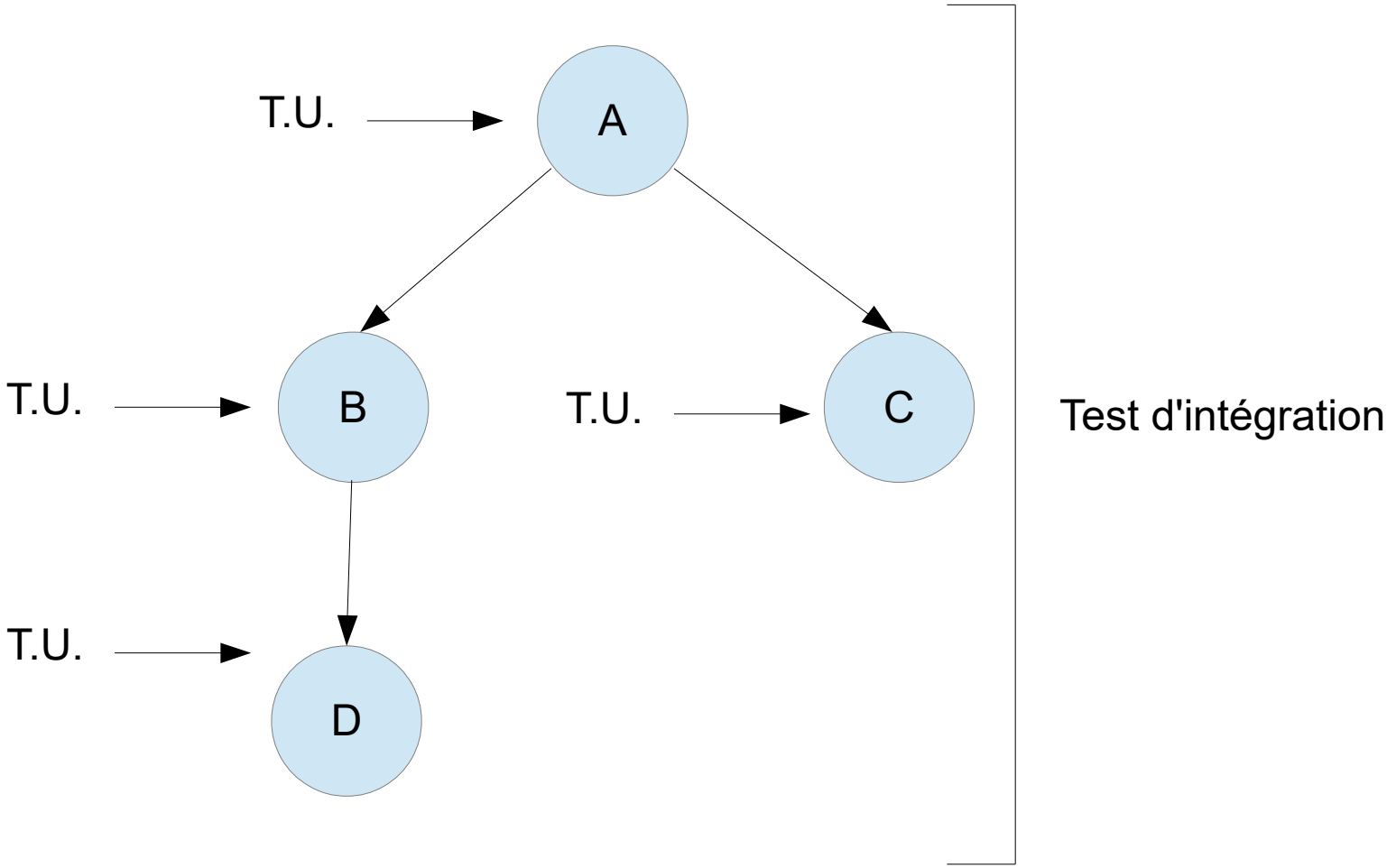
Tests d'intégration

- *Les tests d'intégrations visent à tester que les fonctions développées fonctionnent bien entre elles.*

Les caractéristiques des tests d'intégration sont les suivantes :

- *Ils doivent tester une **fonctionnalité***
- *Ils doivent être **reproductibles***
- *Ils doivent être **compréhensibles***
- *Ils doivent être **documentés***
- *Ils doivent être **indépendants***

Tests d'intégration



Tests d'intégration

```
describe('Users',function() {
    it('should get user toto from database', function() {
        chai.request(app)
        .get('/user/toto')
        .end(function (err, res) {
            expect(res).to.have.status(200);
        });
    }
});
```

Les tests doivent rester indépendants de leur environnement

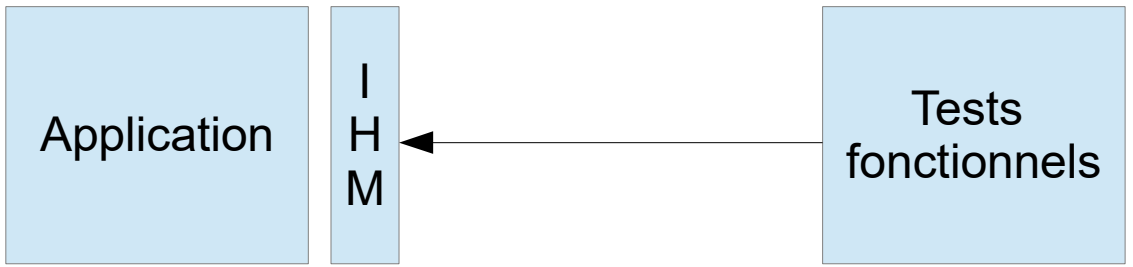
Nous verrons en TP comment s'assurer de cela

Tests fonctionnels

- *Il s'agit du haut de la pyramide des tests. Leur objectif est de simuler les tests manuels dans leur approche*

Les caractéristiques des tests d'intégration sont les suivantes :

- *Ils doivent tester les **cas d'utilisation***
- *Ils doivent être **reproductibles***
- *Ils doivent être **compréhensibles***
- *Ils doivent être **documentés***
- *Ils doivent être **indépendants***



Tests fonctionnels

Chaque test est une interaction usuelle dans l'application et prévue dans le cahier des charges

*Les tests fonctionnels se regroupent en **scénarios** sur le même modèle que les tests fonctionnels manuels*

Cas de tests fonctionnels : un site e-commerce

Tests fonctionnels possibles :
- *Se rendre sur la page de création de compte et créer un compte*
- *Naviguer dans les rayons et ajouter un produit au panier*
- *Passer une commande*

Tests fonctionnels

Chaque test est une interaction prévue dans l'application. Les tests fonctionnels doivent vérifier que l'application valide le cahier des charges initial

Cas de tests fonctionnels : un site e-commerce

Tests fonctionnels possibles :
- Se rendre sur la page de création de compte et créer un compte
- Naviguer dans les rayons et ajouter un produit au panier
- Passer une commande en étant connecté
- Passer une commande sans être connecté

Tests fonctionnels

Avantages des tests fonctionnels :

Meilleurs tests pour tester ce que fait vraiment l'application

Validation possible du cahier des charges avec suffisamment de tests

Mais :

- *Les plus longs à développer*
- *Les plus difficiles à maintenir*

Tests fonctionnels

- *Tests automatisés :*
  - *Demandent le plus d'investissement durant le projet*
  - *Permettent des économies d'échelle*
  - *Permettent de sécuriser l'intégration continue*

Les tests dans un projet informatique

Les tests dans un projet informatique

Les tests manuels

- Dans un projet qui repose uniquement sur des tests manuels, l'essentiel des tests se fait en fin de sprint ou en fin de projet

- Chaque échec dans un cahier de tests est répertorié pour que les personnes en charge du développement corrige l'application

- Il existe de nombreux outils pour historiser l'ensemble des erreurs trouvées :

  https://kanbanflow.com/

  https://trello.com/

  ...

Les tests dans un projet informatique

L'intégration continue

L'intégration continue, permet de rendre automatique, la compilation, les tests et la livraison de l'application sur les environnement

L'intégration continue cherche à de détecter au plus tôt des régressions sur une application

L'automatisation des tests en est donc un concept central

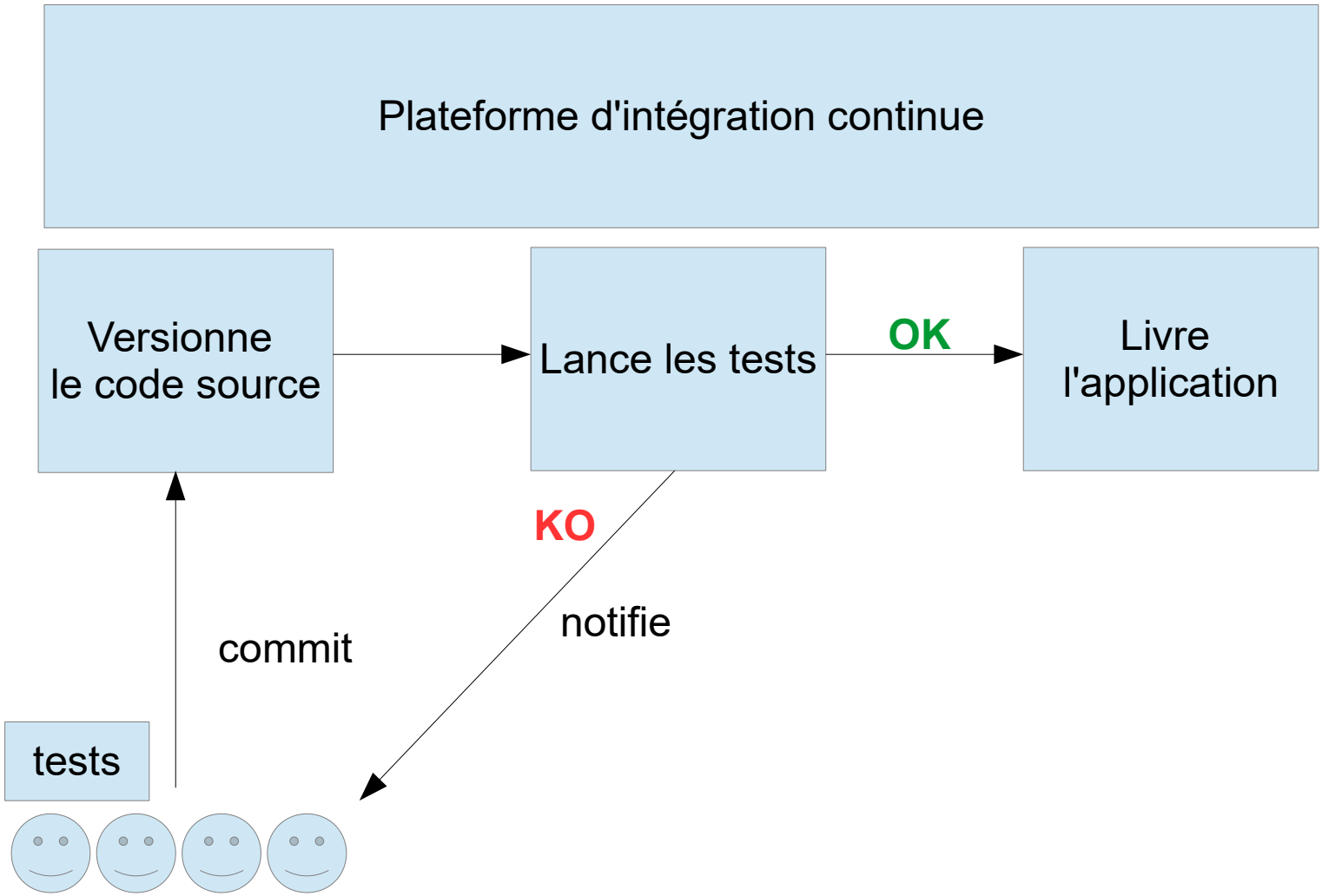
Les tests dans un projet informatique

L'intégration continue permet une forte réactivité quant aux régressions de l'application.

Possibilité de planifier des compilations et l'exécution de tests à heure fixe où dès qu'un changement a été détecté sur le code source

Si une erreur dans les tests à été détectée, l'équipe de développement est prévenue

Les tests dans un projet informatique



Les tests dans un projet informatique

Objectif :

- *Éviter d'enregistrer de régressions dans le dépôt commun*
- *Si jamais cela devait arriver, on notifie l'équipe qui prend les actions nécessaires au plus vite pour rétablir la situation*

Les tests dans un projet informatique

Test-driven development (TDD)

Le développement dirigé par les tests (TDD) est une méthode de développement informatique où les tests sont la première chose définie

La méthode a les règles suivantes :

- *On démarre avec une fonctionnalité inexistante*
- *On écrit un test de cette fonctionnalité*
- *On s'assure que le test échoue sur l'erreur attendue*
- *On écrit juste assez de code pour passer le test*
- *On refactorise notre code pour le rendre plus simple et lisible si nécessaire*

Les tests dans un projet informatique

Objectifs de la méthode TDD

- *Réduire de manière significative le nombre de bugs*
- *Éviter les biais de confirmations (écrire un test qui valide notre code plutôt qu'un code conforme à ce qu'il devrait faire)*
- *Permet de s'astreindre à tester l'ensemble de son code pour éviter les erreurs détectées en fin de projet*

Les tests dans un projet informatique