Pourquoi les tests unitaires ?

Problèmes courants qui arrivent lors du développement d’un produit ou d’un service :

* Une partie de l’équipe n’est pas présente au moment des spécifications 🡪 eg Demande dans la terminologie Agile , besoins ou histoires utilisateur
* La plupart des tests sont manuels, ou alors absence de tests !
* Présence de tests automatisés, mais qui ne détectent pas vraiment les problèmes
* Ecriture et exécution des tests en fin de cycle 🡪 trop tard pour être utile au projet !
* Il y a toujours quelque chose de plus urgent à faire que des tests
* Les équipes sont séparées entre test, développement et analyse fonctionnelle et ne sont donc pas synchronisées 🡪 interdit en Agile
* Incapacité à refactorer le code (refactoring = réécriture du code pour meilleure évolution, qualité, performance et lisibilité) par peur de casser des fonctionnalités
* Coûts de maintenance trop élevés
* Temps de développement pour livrer le produit trop long
* Les clients n’ont pas le sentiment que le produit livré correspond à leurs attentes et demandes
* La documentation n’est jamais à jour
* Vous êtes effrayés de déployer le produit en production car le résultat est imprévisible (ou inconnu)
* Incapacité de déployer en production car les tests de non régression mettent trop de temps à s’exécuter
* L’équipe passe trop de temps à se demander ce qu’une *méthode* ou un *classe* fait réellement

Les ***tests unitaires*** sont une partie de la démarche permettant d’apporter des solutions à ces problèmes courant et récurrents.

Les méthodes classiques de développement (cycle en V) sont inadaptées à de nombreux projets modernes qui demandent des itérations courtes pour maîtriser le tryptique coût – délai – qualité des produits logiciels 🡪 méthodes Agile.

Le développement conduit par les tests (TDD : Test Driven Development) est

Réf :

<https://www.agilealliance.org/glossary/tdd/>

<http://referentiel.institut-agile.fr/tdd.html>

Ce que doivent être les tests unitaires ?

Un bon ***test unitaire*** doit avoir les caractéristiques suivantes :

* Automatisé et répétable
* Facile à implémenter
* Pertinent pour le produit demain (donc durable)
* N’importe qui doit pouvoir l’exécuter (pousse bouton)
* Rapide
* Consistent : résultat identique à contexte identique
* Isolé : s’exécute indépendamment des autres tests
* En cas d’échec (***fails*** ou ***error***), on doit pouvoir facilement détecter le problème pour trouver une solution