SOMMAIRE

TEST AUTOMATIQUE

Dr. Soukeina Ben chikha

INTRODUCTION

- Un des exemples les plus frappants est l’échec du lancement de la fusée **Ariane 5**. 10 ans de recherches et de développement se soldant par un échec cuisant au décollage... Le tout à cause d'un système de guidage hérité d'Ariane 4, et non testé dans des simulations adaptées aux scénarios d'Ariane 5.

INTRODUCTION

Objectif du test logiciel

- En développement, les tests visent à **vérifier que le produit codé fonctionne comme prévu selon des scénarios prédéfinis et représentatifs**.
- Cela permet de garantir la qualité de ce qui est codé, malgré les contraintes du projet, comme les délais ou le budget, par exemple.

TYPES DE TESTS

Tests manuels

- La manière la plus directe de faire des tests, c'est de réaliser des **tests manuels**.
- Emergence du métier de testeur. Les **testeurs**, qui prennent la responsabilité de la qualité d'une application. Ils vérifient les logiciels durant une phase de développement du projet, ou même après, lorsque le logiciel est en **production**.
- Quand le nombre de fonctionnalités augmente, tester manuellement l'exécution d'un produit et de toutes ses fonctionnalités est une activité **répétitive** et **peu créative**.
- De plus, coder une fonctionnalité peut avoir un impact sur une autre fonctionnalité dans le logiciel. C’est pour cela qu’il faudrait retester tout le produit à chaque développement !

TYPES DE TESTS

Tests automatiques

- Actuellement, on produit de nouvelles versions des logiciel plusieurs fois, l'alternative est de développer des **programmes spécifiques** qui vérifient le code de nombreuses façons différentes. C'est ce que l'on appelle des ***tests automatisés***.
- Comme il s'agit de code, l'exécution de ces tests est rapide et peut être répétée à de multiples reprises pour un coût très faible.
- Il existe plusieurs **outils de tests automatisés**, chacun avec ses avantages et ses inconvénients et son contexte.

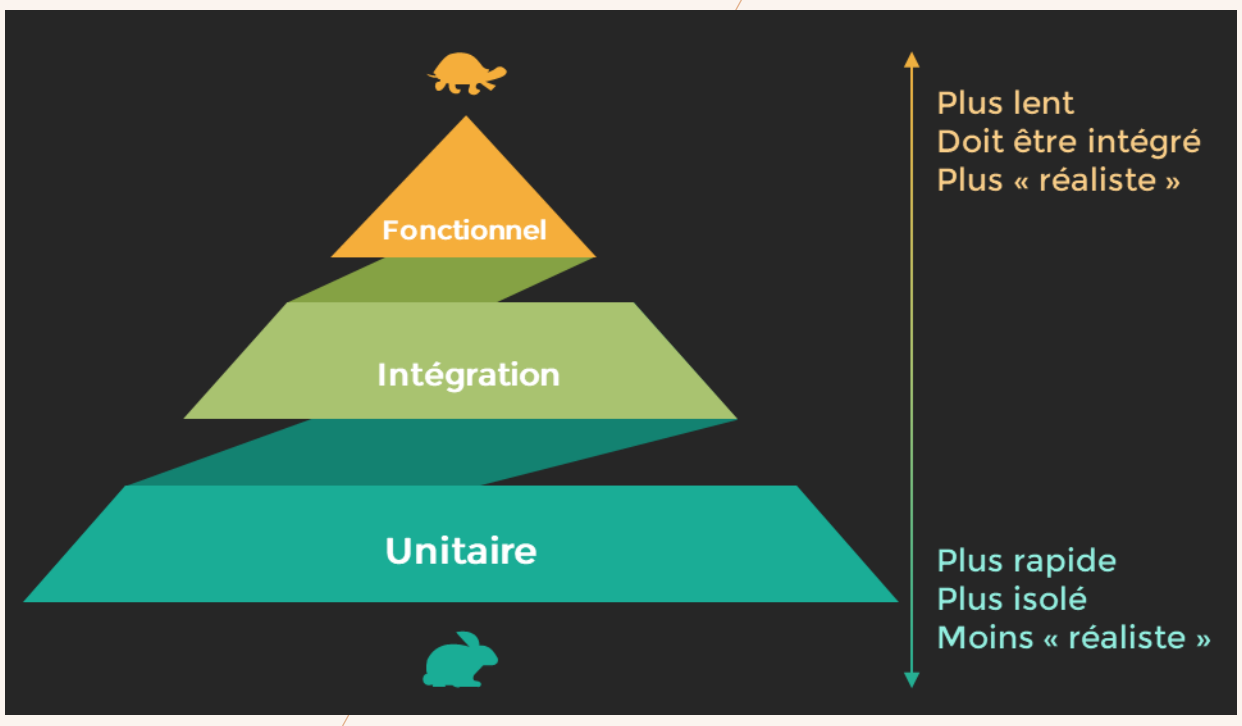
TYPES DE TESTS

Les types de tests automatisés

- Les 3 types de tests automatisés les plus courants sont les tests **unitaires**, **d'intégration** et **fonctionnels**.
- Ces test sont représenté par une **pyramide des tests**,

TYPES DE TESTS

- A la base: il y a davantage de tests rapides et autonomes
- Au milieu de moins en moins à mesure que l'on grimpe vers le sommet.
- Au sommet, les tests sont plus longs et plus complexes à exécuter, tout en simulant des scénarios de plus en plus réalistes.



Mike Cohn, **pyramide des tests**,

Tests unitaires

- **Les tests unitaires** testent de "**petites" unités** de code. Ils testent que chaque **fonctionnalité** extraite de manière isolée se comporte comme attendu.
- Ils sont très rapides et faciles à exécuter. Si vous avez *cassé* quelque chose, vous pouvez le découvrir **vite et tôt**.
- Les bons tests unitaires sont **stables**, c'est-à-dire que le code de ces tests n'a pas besoin d'être modifié, même si le code de l'application change pour des raisons techniques. Ils sont donc **rentables**, car ils ont été écrits une seule fois, mais exécutés de nombreuses fois.

Tests d'intégration

- Les tests d'intégration vérifient si les unités du code **fonctionnent ensemble** comme prévu – avec le fait que les tests unitaires soient réussis.
- Les tests d'intégration vérifient les interactions entre les unités, ce qui augmente la confiance concernant le bon fonctionnement de l'application finale.
- Ils peuvent nécessiter **l'exécution de composants extérieurs** (base de données, services web externe, etc.). Le lancement de ces composants et l'interaction entre les unités de code développées rendent ces types de test plus lents et potentiellement moins stables. Mais on teste des scénarios plus proches de l'utilisation finale de l'application.

Tests fonctionnels

- Les tests fonctionnels visent à simuler le comportement d'un utilisateur final de l'application. L'ensemble du code développé est pris comme une **boîte noire** à tester, sans connaissance des briques et des unités de code qui la composent. Les simulations obtenues sont donc les plus proches de l'application utilisée dans des conditions réelles.
- Ces tests nécessitent toute l'infrastructure nécessaire à l'application. Ces tests sont les plus lents à exécuter, et **testent une partie beaucoup plus grande du code développé**.
- Cela crée une plus forte dépendance qui rend les tests **moins stables**, donc **moins rentables**.
- Potentiellement, une modification simple de l'interface utilisateur (la couleur d'un bouton) pourrait nécessiter de recoder le test fonctionnel associé.

TYPES DE TESTS

- Écrire et exécuter des tests est **un investissement** qui a un coût. L'automatisation des tests n'a de réelle valeur ajoutée que si l'on arrive à exécuter ces tests sans les modifier de nombreuses fois, et la pratique montre qu'il est plus facile de rentabiliser un test unitaire qu'un test fonctionnel.
- De plus, **plus un bug est détecté tôt, moins son coût de correction est élevé**.
  - Pour qu'une fontionnalité de code développée puisse être couverte par un test fonctionnel, il faut développer le code de l'interface utilisateur associée ainsi que le code d'accès aux données nécessaires. Si l'unité de code présente des bugs, sa détection sera alors beaucoup plus tardive, et son coût beaucoup plus élevé.

TYPES DE TESTS

- Les bonnes pratiques visant à coder beaucoup de tests unitaires (en particulier en *TDD)* poussent le développeur à coder des unités de code plus robustes et autonomes. Ces pratiques favorisent aussi une détection très tôt des bugs, ces derniers ont donc moins d'impact en termes de coût sur le projet.

MAUVAISES PRATIQUES

- Mais, de nombreuses équipes de développement, soumises à des contraintes fortes de délais et de coûts, ne respectent pas ces bonnes pratiques
- Dans ce cadre, le produit développé est beaucoup moins **maintenable,** les tests deviennent obsolètes très vite et on retourne à des pratiques de tests manuelles coûteuses et peu efficaces.

FINALITÉS DU TEST

Tester pour faire face à l'inattendu

- Il est facile d'imaginer le **bon scénario.**
- La réalité correspond rarement à nos attentes. Il y aura toujours des **scénarios alternatifs**, des comportements utilisateurs inattendus, des problèmes réseau, un service web externe non disponible. Il y aura souvent plus de mauvais chemins que ce que vous pouvez envisager.
- Le risque zéro n'existe pas. Réduire le risque en combinant les scénarios attendus, les cas **limites courants**, et les risques de non-disponibilité des services extérieurs.

FINALITÉ DES TESTS

Testez pour faciliter la maintenance

- Les tests vont **faciliter la correction et la maintenance du code.**
  - Si un bug bloquant a lieu après la mise en production de votre produit, disposer de tests existants permet de vérifier très rapidement si les corrections apportées cassent quelque chose dans le code existant.
  - Ils testent la **non-régression** de votre produit. La mise en production de la correction s'effectue en confiance et cette réactivité donne aussi confiance au client.
- La régression, c'est lorsqu'une fonctionnalité existante se déroulait correctement et ne fonctionne plus suite à une correction ou à une mise à jour.

FINALITÉ DES TESTS

Faciliter le travail d’équipe

- Les cas de tests décriront le comportement attendu du logiciel.
- La lecture des tests permet de comprendre non seulement ce que le code est censé faire, mais aussi **comment il fonctionne**
- Ce qui permet le reprise après une période avec le même développeur ou un autre..

RECAP

- **L'automatisation des tests** .
- **Les tests unitaires**
- **Les tests d'intégration**.
- **Les tests fonctionnels**
- Au-delà de la vérification, les tests permettent, entre autres, de mieux faire face à **l'inattendu**, d'améliorer la réactivité de la **maintenance**, et de mieux **collaborer** entre développeurs.

MERCI