Tests des Logiciels

ISTQB®

Chapitres

Recap Chap 1

Fondamentaux des tests Q/A?

Recap Chap 1

1- Définition & objectifs de ‘Test’
2- Tester vs. Déboguer
3- Erreur > Défaut > Anomalie
4- Les 7 principes de test
5- Processus de test
6- Psychologie du testeur

Chapitre 2 : Tester pendant le cycle de vie du développement logiciel

1- Modèles de développement logiciels
2- Les niveaux de tests.
3- Types de tests
4- Tests de maintenance

Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Développement de logiciel et tests logiciels:

Une partie importante du rôle d'un testeur est de se familiariser avec les principaux modèles de cycle de vie du développement logiciel afin que les activités de **test adaptées** puissent être réalisées.

Quel que soit le modèle de cycle de vie de développement logiciel, il y a plusieurs caractéristiques de bonnes pratiques des tests :

- Pour chaque activité de développement, il y a une activité de test correspondante
- Chaque niveau de test a des objectifs de test spécifiques à ce niveau (Etude et analyse, Développement, Assemblage des modules, fonctionnalité ..)

Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Développement de logiciel et tests logiciels:

L'analyse et la conception des tests pour un niveau de test donné commencent au cours de l'activité de développement correspondante.

Les testeurs :

- Participent aux discussions pour définir et affiner les exigences et la conception
- Sont impliqués dans la revue des produits d'activités (p. ex. les exigences, la conception, les User Stories, etc.) dès que des versions préliminaires sont disponibles

Quel que soit le modèle de cycle de développement logiciel choisi, les activités de test devraient commencer dès les premières étapes du cycle de vie, en respectant le principe de « **Tester tôt** ».

Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Comment ?

Un modèle de cycle de vie du développement logiciel approprié doit être choisi et adapté en fonction du:

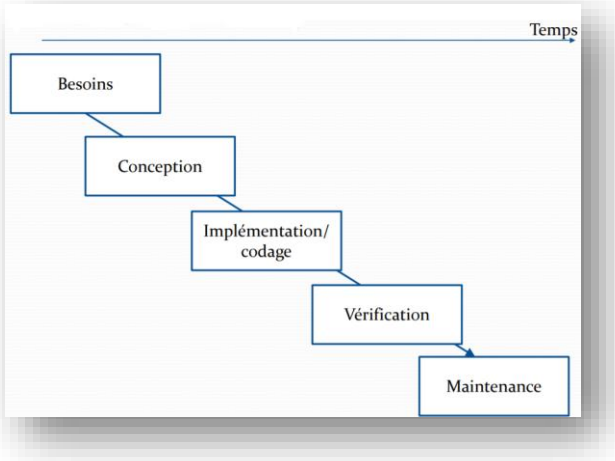
- Contexte/l'objectif du projet
- Caractéristiques du produit
- Type du produit développé
- Priorités métier (ex., délai de mise sur le marché)
- Risques produit et projet identifiés

Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Développement de logiciel et tests logiciels:

Modèle de développement séquentiel « En cascade »

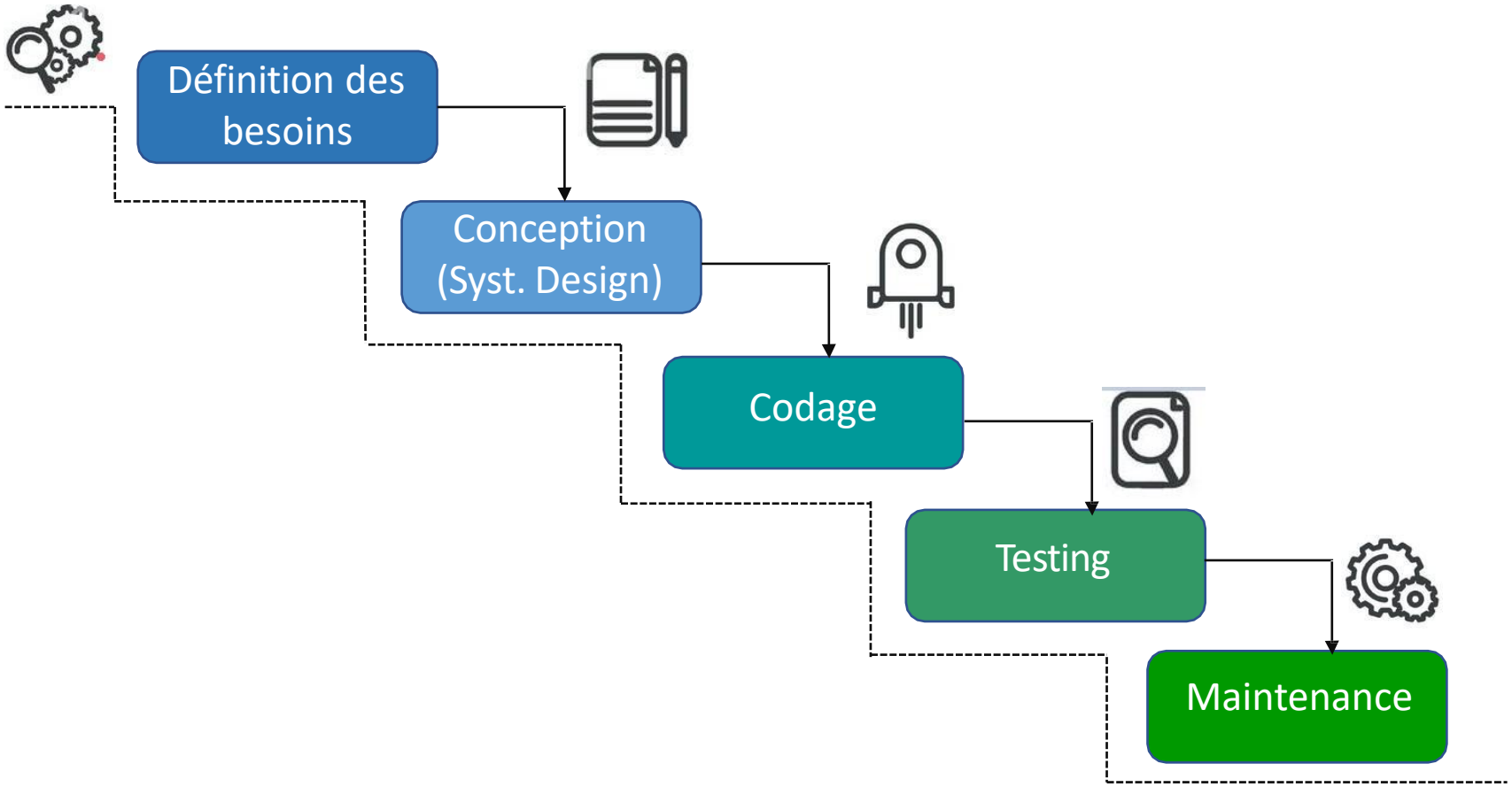


Décrit le processus de développement logiciel comme un flux linéaire et séquentiel d'activités.

➔ **Toute phase du processus de développement devrait commencer lorsque la phase précédente est terminée.**

Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels



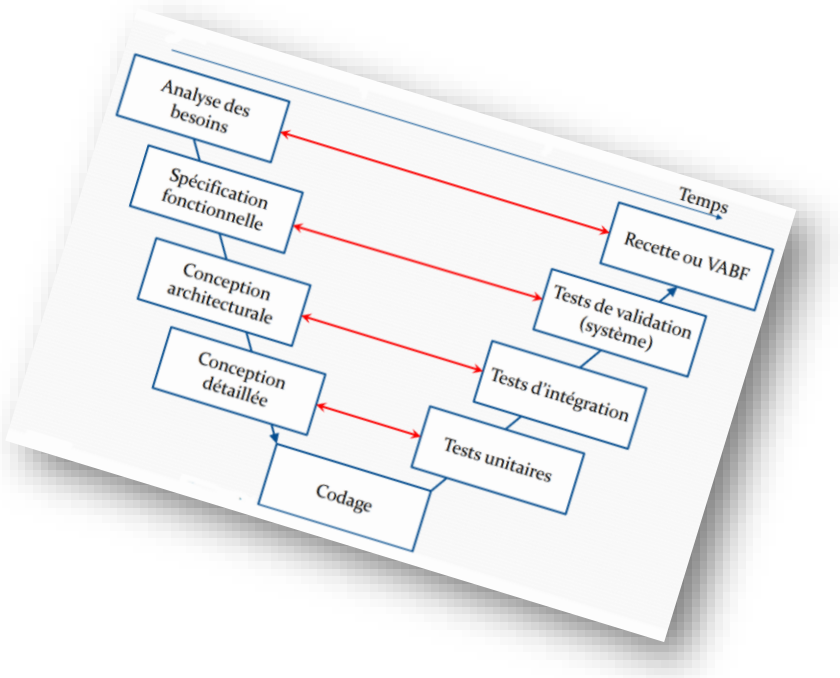
Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Développement de logiciel et tests logiciels:

Modèle de développement en V

Contrairement au modèle en 'cascade', le modèle en V Intègre le processus de test tout au long du processus de développement



➔ **Application du principe de « Tester tôt »**

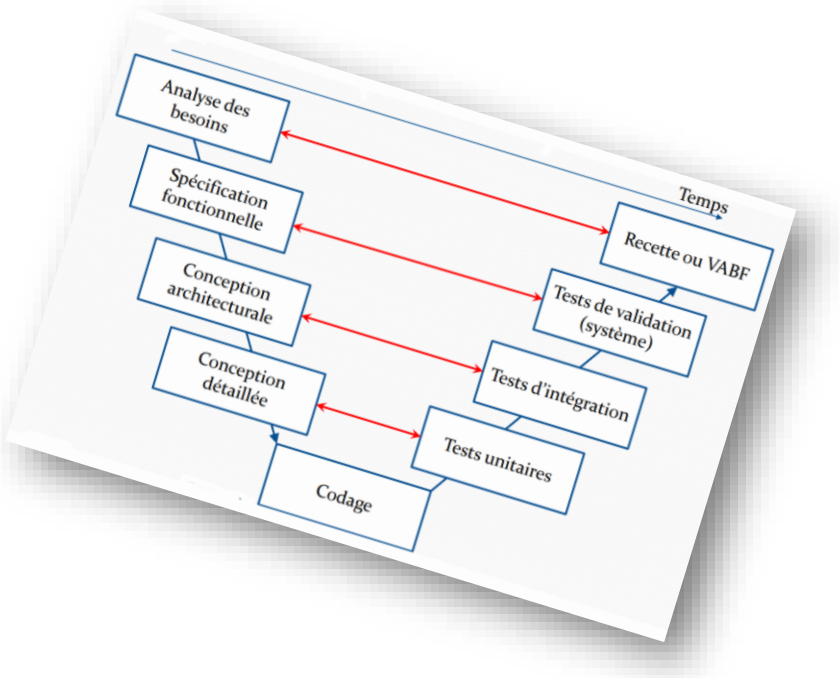
Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Développement de logiciel et tests logiciels:

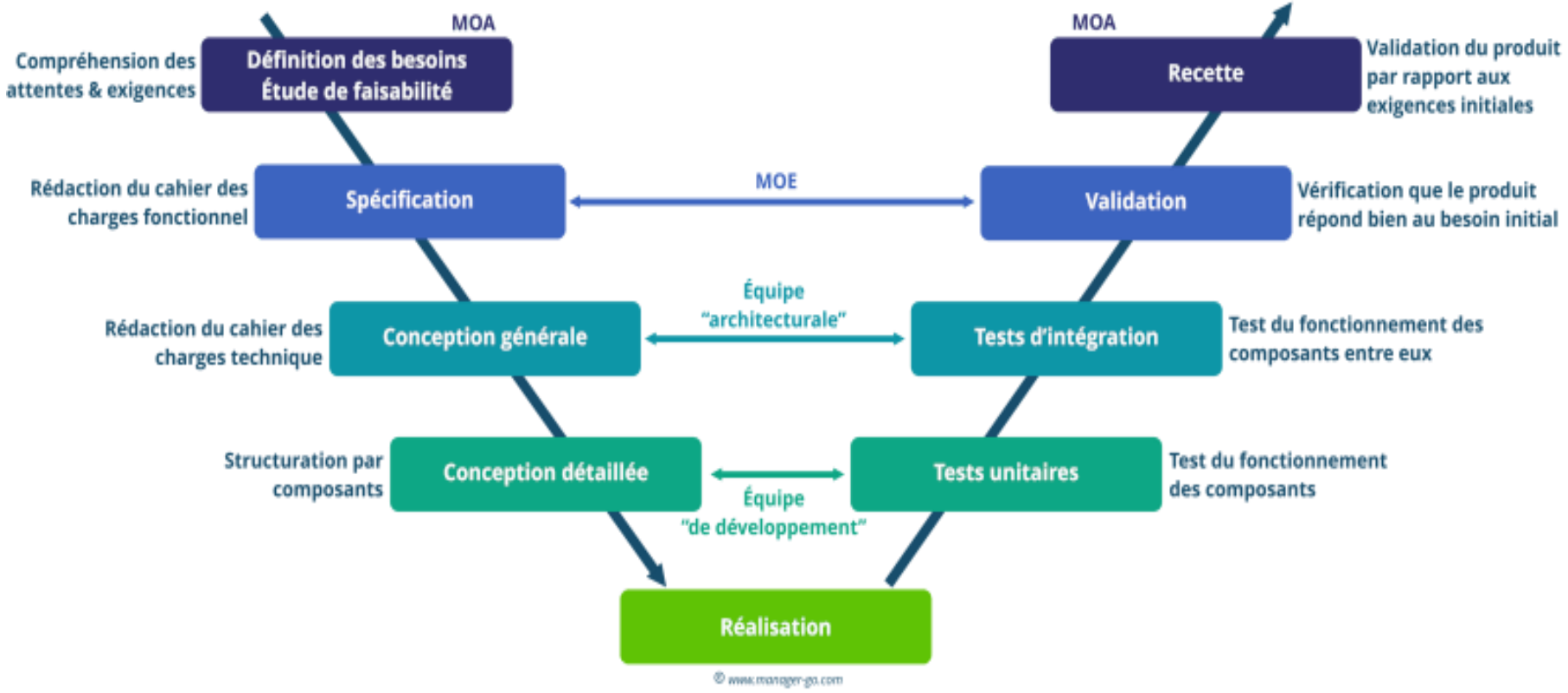
Modèle de développement en V (document)

Inclut des niveaux de test associés à chaque phase de développement correspondante.



Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

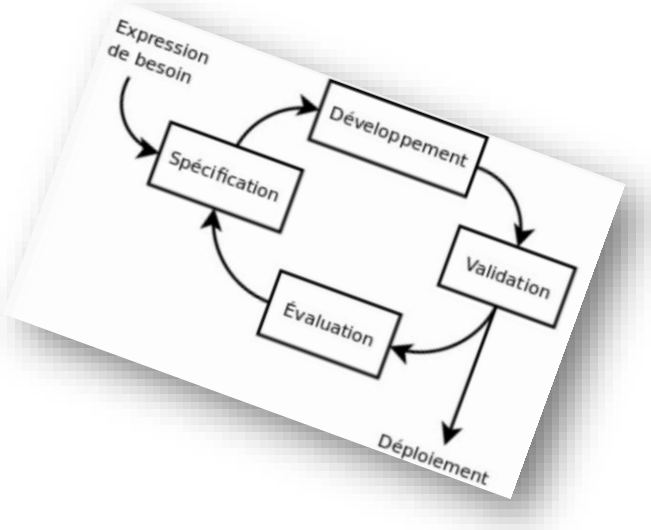


Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Développement de logiciel et tests logiciels:

Modèle de développement itératif



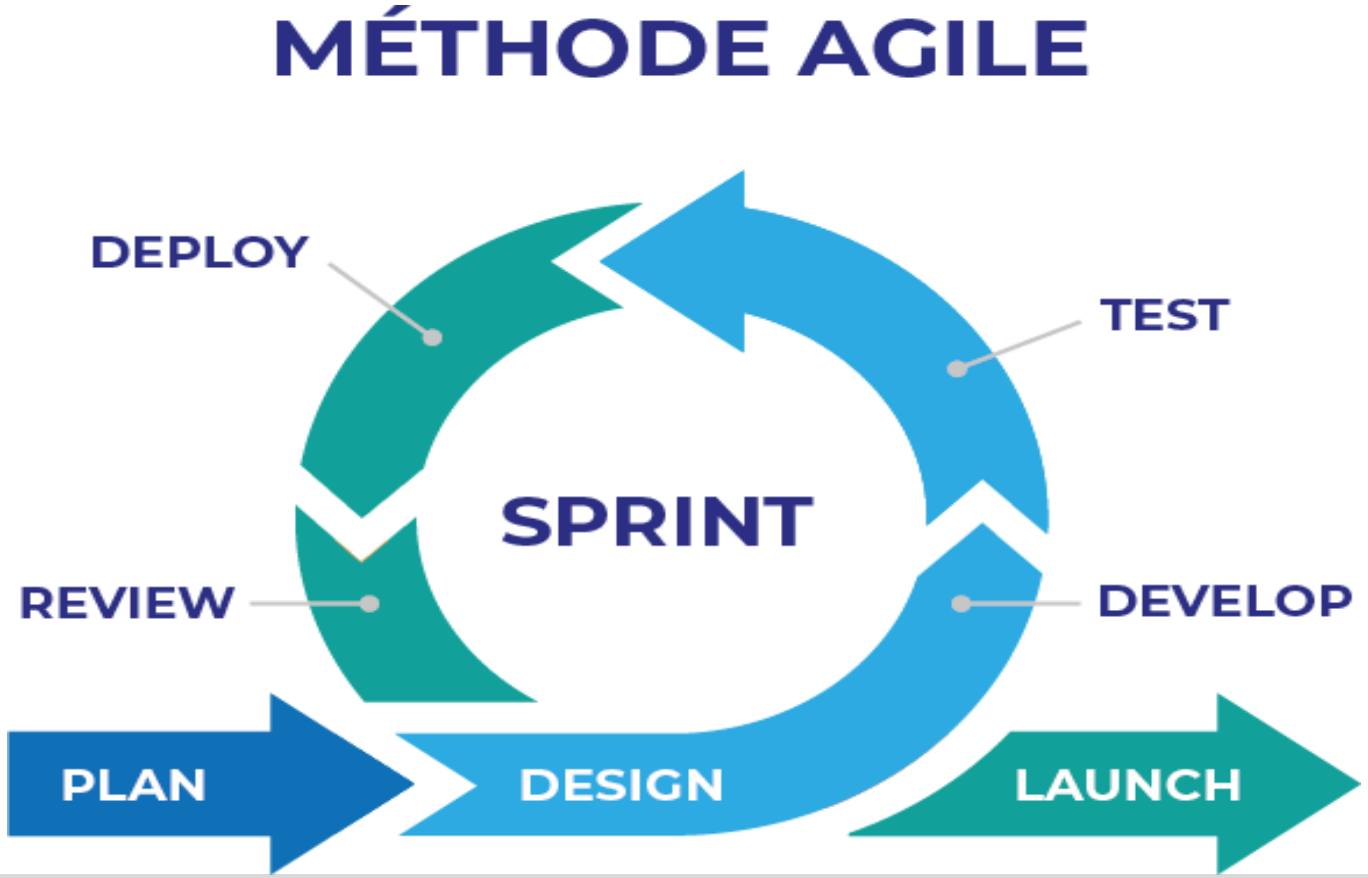
Se déroule lorsque des groupes de caractéristiques sont spécifiés, conçus, développés et testés ensemble dans une série de cycles, souvent d'une durée fixe.

Les itérations peuvent impliquer des changements sur les caractéristiques développées dans les itérations précédentes, ainsi que des changements sur le périmètre du projet.

Chaque itération délivre un logiciel opérationnel qui est un sous-ensemble croissant de l'ensemble global des caractéristiques jusqu'à ce que le logiciel final soit livré ou que le développement soit arrêté.

Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels



Tester pendant le cycle de vie du développement logiciel

Modèles de développement logiciels

Modèles de cycle de vie du développement logiciel en contexte

Selon le contexte du projet, il peut être nécessaire de **combiner** ou de **réorganiser** les niveaux de test et/ou les activités de test.

Les modèles de cycle de vie du développement logiciel eux-mêmes peuvent être combinés.

Par exemple:

- Le modèle en V peut être utilisé pour le développement et le test des systèmes back-end et de leurs intégrations
- Le modèle de développement Agile peut être utilisé pour développer et tester l'interface utilisateur front-end (IHM) et les fonctionnalités.

Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test de composants

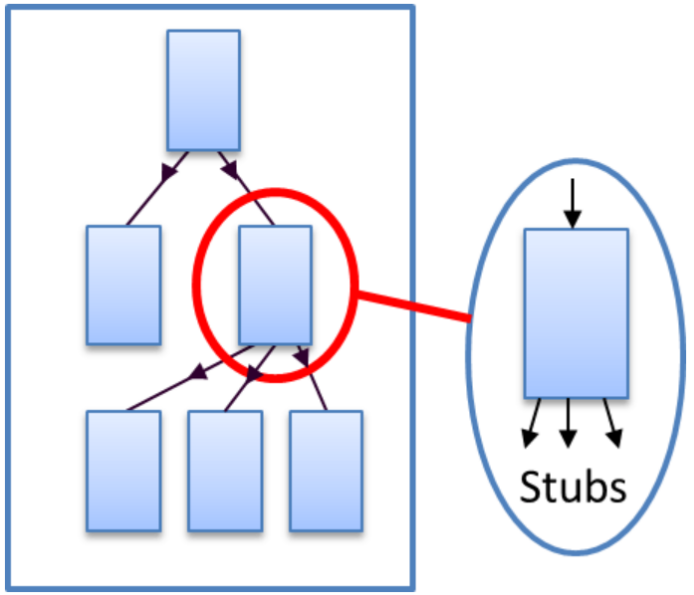
Aussi appelé « Test unitaires »

L'objectif principal des tests unitaires est de séparer chaque partie du programme afin de tester si chaque partie du logiciel fonctionne correctement sans aucune erreur.

Grâce à l'isolement de chaque partie, il peut facilement déterminer le comportement exact du code en fonction des attentes.

Appelée aussi « unité » ou « module »

Cible : les méthodes et les classes



Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Exemples de produits d'activités qui peuvent être utilisés comme bases de test pour les tests de composants :

- Conception détaillée
- Code
- Modèle de données
- Spécifications des composants

Objets de test

Les objets de test habituels pour les tests de composants sont :

- Composants, unités ou modules
- Code et structures de données
- Classes
- Modules de bases de données.

Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Défauts et défaillances courants ?

Des exemples de défauts et de défaillances courants pour les tests de composants :

- Fonctionnalité incorrecte (par exemple, non conforme aux spécifications de conception)
- Problèmes de flux de données
- Code et logique incorrects

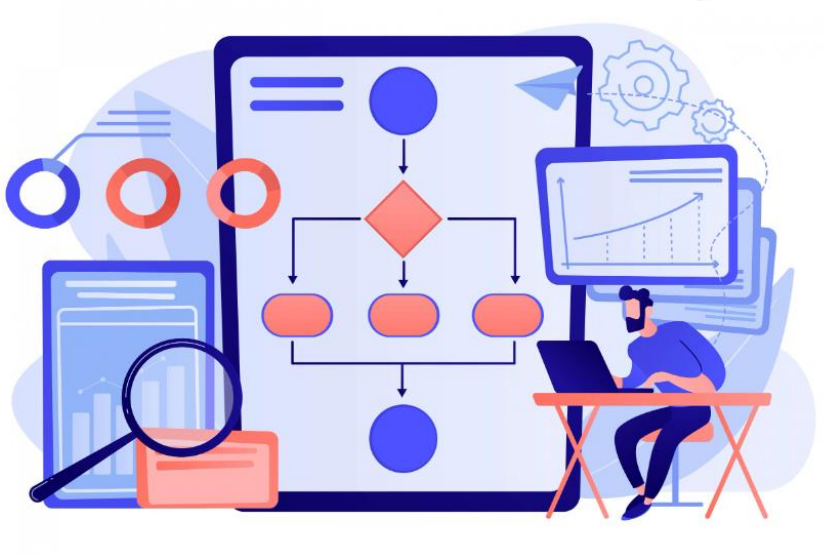
Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test d'intégration

L'intégration a pour but de:

- Valider que toutes les parties développées indépendamment (composants) fonctionnent bien ensemble.
- Vérifier que les composants s'intègrent bien avec d'autres composants ou systèmes
- Vérifier que le produit est compatible avec l'environnement logiciel et matériel du client



Intégration continue: fusion des tests unitaires et des tests d'intégration.

Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test d'intégration

Il comporte 2 types de tests :

Les tests de validité – le nouveau module répond à ce qui a été demandé lorsqu'on l'intègre dans le système global

Les tests de non-régression – L'intégration du nouveau module n'induit pas de régressions fonctionnelles (modification du comportement des modules existants) ou non fonctionnelles (instabilité) du système dans sa globalité

Cible : les composants et le système global

Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test d'intégration

Défauts et défaillances courants

Des exemples de défauts et de défaillances courants pour les tests d'intégration de composants :

- Données incorrectes, données manquantes ou encodage incorrect des données.
- Mauvais séquencement ou synchronisation des appels d'interfaces.
- Décalages au niveau des interfaces.
- Défaillances dans la communication entre les composants.

Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test système

- Tester le système dans son ensemble: tous les modules / composants sont intégrés afin de vérifier si le système est conforme au exigences spécifiées.
- Tester le système sur son futur site d'exploitation dans des conditions opérationnelles et au- delà (surcharge, défaillance matérielle,…)
- Tester la fiabilité et la performance de l'ensemble du système, tant au niveau fonctionnel que structurel, plutôt que de ses composants.
- S'assure que le système complet, matériel et logiciel, correspond bien à la définition des besoins tels qu'ils avaient été exprimés.

Cible : le système global sur l'architecture informatique du client

Tester pendant le cycle de vie du développement logiciel

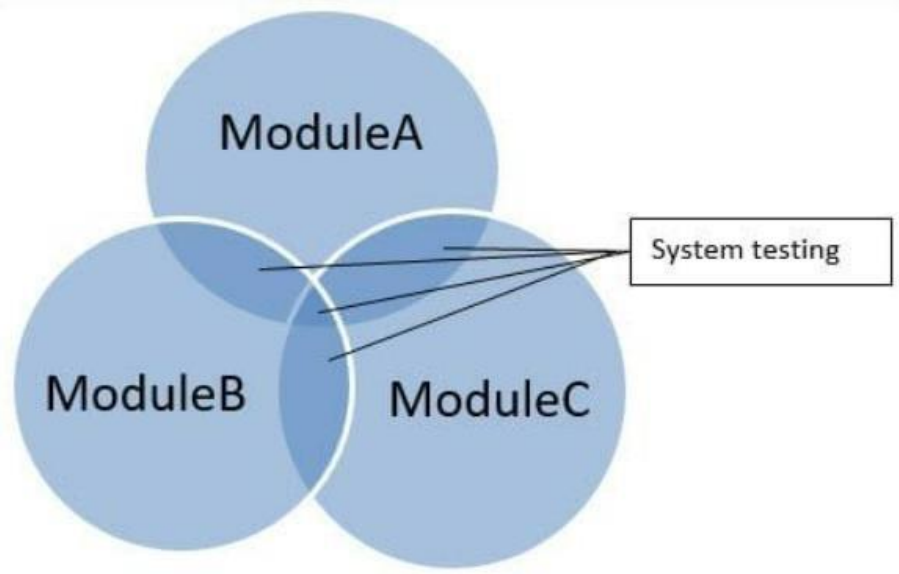
Niveaux de test

Test système

Soit une application comporte trois modules: A, B et C.

Les tests effectués en combinant les modules A et B ou B et C ou A et C sont appelés tests d'intégration.

L'intégration des trois modules et leur test en tant que système complet s'appelle Test du système.



Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test système

S'oriente vers les spécifications non fonctionnelles composé de plusieurs tests :

- **Tests de stress** : tester le produit au-delà des attentes non fonctionnelles du client. Par exemple, un système de sauvegarde qui doit tout sauvegarder deux fois par jour, le tester en mode trois fois par jour.

- **Tests de performance** : évaluation par rapport à des exigences de performances données. Par exemple, un moteur de recherche doit renvoyer des résultats en moins de 30 millisecondes.

- **Tests d'utilisabilité** : vérifier les exigences d'apprentissage demandées à un utilisateur normal pour pouvoir utiliser le produit.

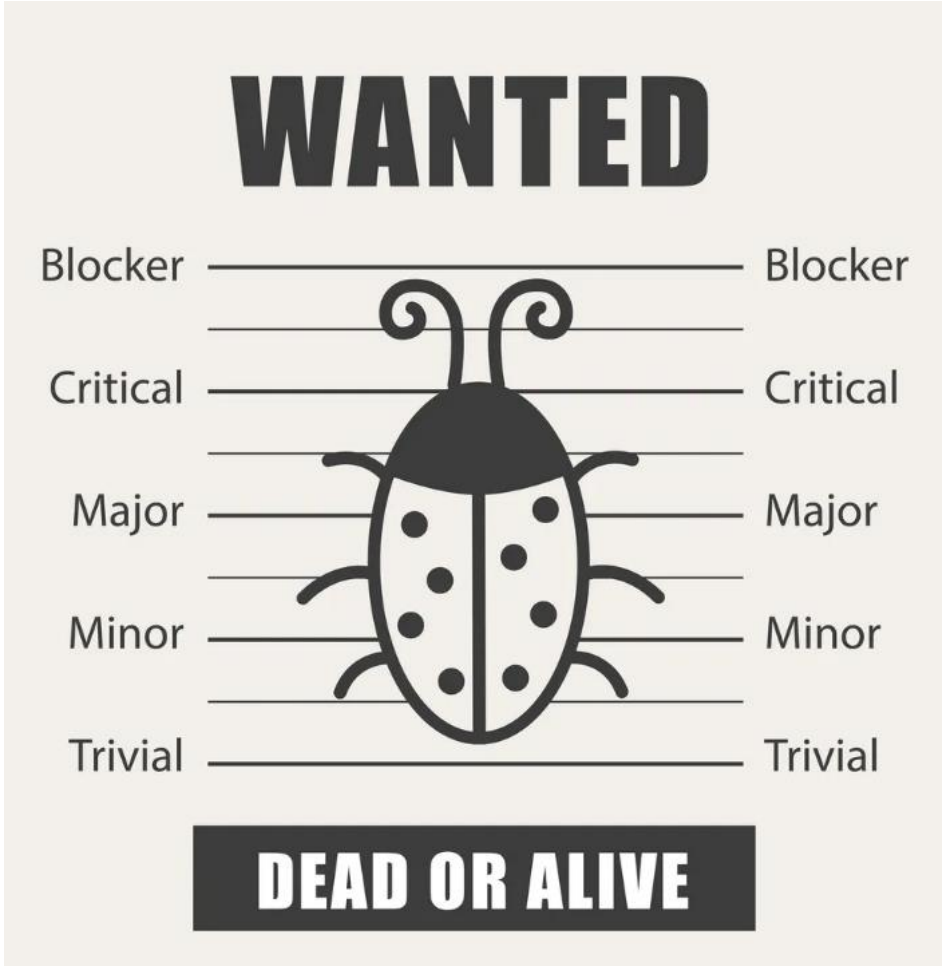
Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Défauts et défaillances courants

Des exemples de défauts et de défaillances courants pour les tests system :

- Calculs incorrects
- Comportement fonctionnel ou non-fonctionnel du système incorrect/inattendu
- Flux de contrôle et/ou de données incorrects au sein du système
- Réalisation incorrecte et incomplète des tâches fonctionnelles de bout en bout
- Incapacité du système à fonctionner correctement dans le ou les environnements de production
- Incapacité du système à fonctionner selon la description faite dans les manuels système et utilisateur



Tester pendant le cycle de vie du développement logiciel

Niveaux de test

Test d'acceptation

UAT (user acceptance testing, ou recette utilisateur)

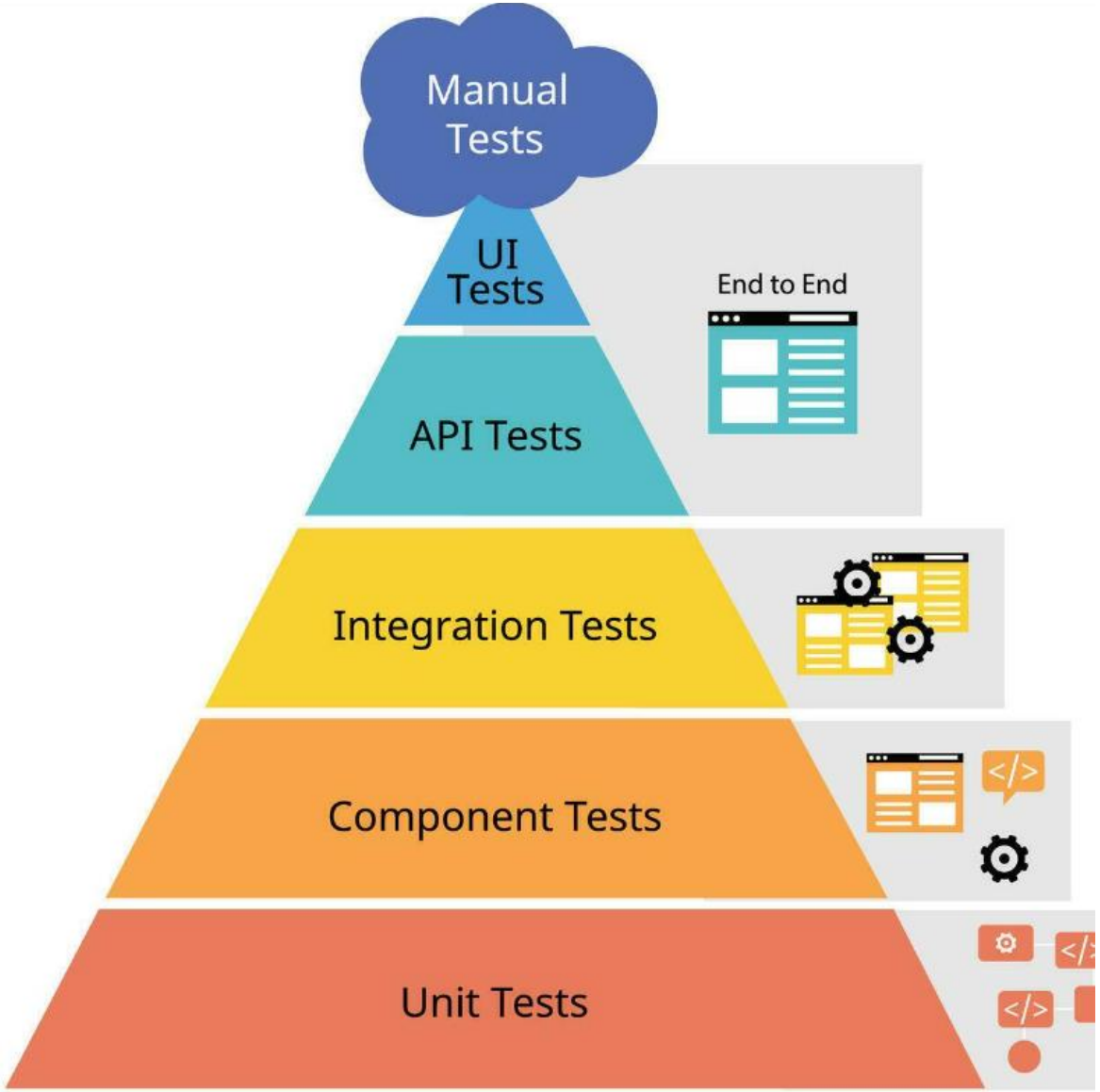
Les tests d'acceptation sont des tests formalisés par le client, ils sont exigés par le client pour qu'il accepte le produit.

Test d'acceptation est la dernière étape avant le mise en œuvre opérationnelle du système.

→ On teste le système dans les conditions définies par le futur utilisateur, plutôt que par le développeur.

Les tests d'acceptation révèlent souvent des omissions ou des erreurs dans la définition de besoins.

→ Des erreurs de validation et non de vérification

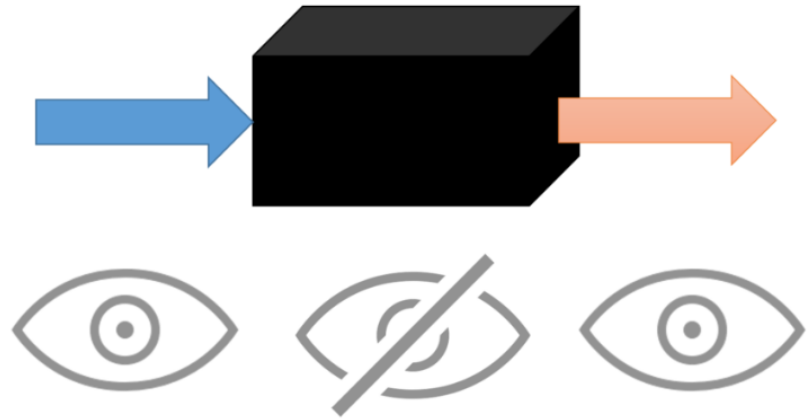

Manual Tests
UI Tests
End to End
API Tests
Integration Tests
Component Tests
Unit Tests

Tester pendant le cycle de vie du développement logiciel

Types de test

Tests fonctionnels Ou test boîte noir

- Utilisés pour vérifier les fonctions d'une application logicielle conformément aux spécifications des exigences.
- Impliquent principalement des tests de boîte noire et ne dépendent pas du code source de l'application.
- Consistent à vérifier l'interface utilisateur, la base de données, les API, les applications client / serveur ainsi que la sécurité et la fonctionnalité du logiciel testé.
- Les tests fonctionnels peuvent être effectués manuellement ou en utilisant l'automatisation



Tester pendant le cycle de vie du développement logiciel

Types de test

Tests non-fonctionnels

- Vérifient des propriétés qui ne sont pas directement liées à une utilisation du code.
- Il s'agit de vérifier des caractéristiques telles que la sécurité ou la capacité à monter en charge (performance).

Les tests non-fonctionnels permettent plutôt de répondre à des questions tel que:« Est-ce que cette classe peut être utilisée par 1000 threads en même temps sans erreur ? ».

Tester pendant le cycle de vie du développement logiciel

Types de test

Tests non-fonctionnels

Test de performance

- Permet d'évaluer la capacité du programme à fonctionner correctement par rapport aux critères de flux de données et de temps d'exécution.

- Ces tests doivent être précédés tout au long du cycle de développement du logiciel d'une analyse de performance, ce qui signifie que les problèmes de performances doivent être pris en compte dès les spécifications.

Tester pendant le cycle de vie du développement logiciel

Types de test

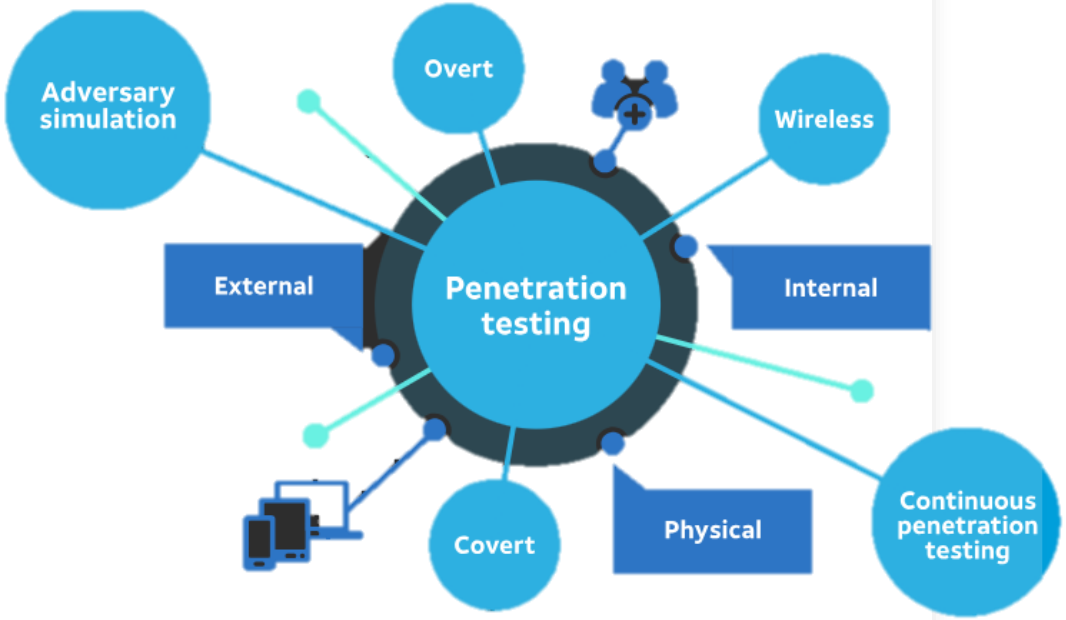
Tests non-fonctionnels

Test de sécurité

Les tests de sécurité et de contrôle d'accès portent sur deux domaines clés de la sécurité :

- La sécurité au niveau de l'application incluant l'accès aux fonctions de traitement de données.
- La sécurité au niveau du système, incluant la connexion à distance au système.

Tester la manière avec laquelle le système protège contre les accès interne ou externes pas autorisés.

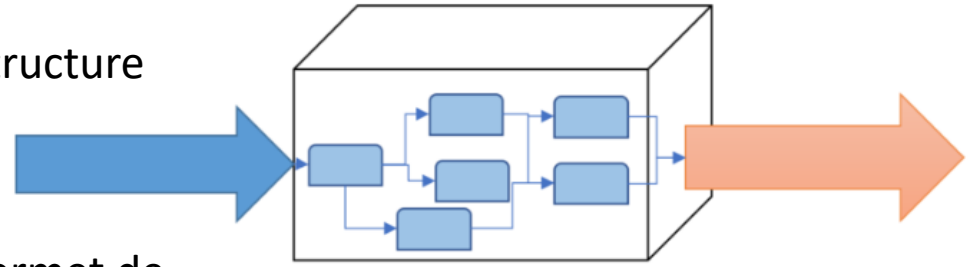


Tester pendant le cycle de vie du développement logiciel

Types de test

Tests boîte blanche

La technique de la boite blanche est une stratégie de test basée sur la structure interne du programme.



Dans ce type de test, les testeurs accèdent au code source ce qui leur permet de fixer

- des valeurs d'entrées afin d'assurer que :

  Toutes les conditions d'arrêt de boucle ont été vérifiées .

  Toutes les branches d'une instruction conditionnelle ont été testées

  Les structures de données internes ont été testées (pour assurer la validité).

Tester pendant le cycle de vie du développement logiciel

Types de test

Tests liés aux changements (test & re-test)

Test de confirmation:
Tests dynamiques effectués après la correction de défauts dans le but de confirmer que les défaillances causées par ces défauts ne se produisent plus.

Tests de régression :
Consiste à effectuer des essais sur les parties (composants) qui ne sont pas touchées directement par une modification.
Vise à s'assurer qu'il n'y a pas eu d'effets secondaires imprévus lors des modifications.
La définition officielle est : '*Des tests sélectifs d'un système ou composants pour vérifier que les modifications n'ont pas entrainé des effets inattendus*"

Tester pendant le cycle de vie du développement logiciel

Types de test

Types de test et niveaux de test

Les niveaux de test:

Ces tests ont pour but de décomposer un logiciel selon 4 niveaux:

1. Le niveau composant (la plus petite brique)
2. Le niveau intégration (la liaison entre ces briques: le ciment)
3. Le niveau système (le contenu de la construction)
4. Le niveau acceptation (la correspondance au besoin utilisateur)

Tester pendant le cycle de vie du développement logiciel

Types de test

Types de test et niveaux de test

Les types de test:

Groupe d'activités de test dont l'objectif est de tester un composant ou système sur un ou plusieurs attributs liés entre eux.

Un type de test est focalisé sur un objectif de test spécifique (ex. : test de fiabilité, d'utilisabilité, de régression, etc) et peut couvrir un ou plusieurs niveaux de test et une ou plusieurs phases de test

Tester pendant le cycle de vie du développement logiciel

Types de test

Types de test et niveaux de test

Les types Vs les niveaux de test:

Les niveaux de tests sont un moyen de structurer et d'organiser ses tests.

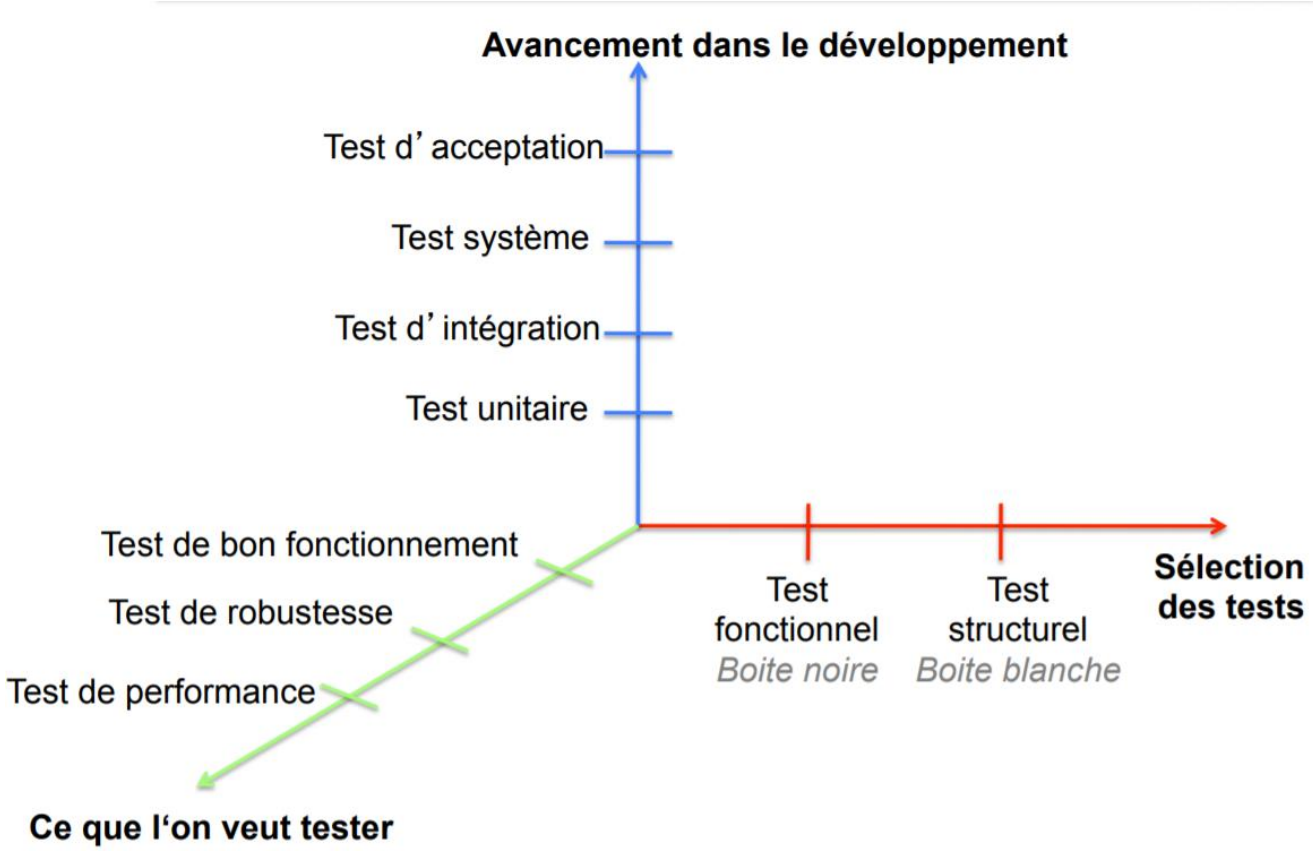
Les types de tests c'est tous les tests possibles.

On peut prendre les différents « types de test » et définir à quels niveaux ils se peuvent généralement se situer:

- **Les tests de régression** → peuvent/doivent être effectués sur tous les niveaux.
- **Les tests boites blanches** → peuvent être effectués sur tous les niveaux même si en général on les voit moins au niveau des tests d'acceptation.

Tester pendant le cycle de vie du développement logiciel

Les types Vs les niveaux de test



Tester pendant le cycle de vie du développement logiciel

tests de maintenance

Une fois déployés dans les environnements de production, les logiciels et les systèmes doivent être maintenus.

→ Des changements de diverses sortes sont presque inévitables dans les logiciels et les systèmes livrés pour:

- Corriger des défauts découverts lors de l'utilisation opérationnelle
- Ajouter de nouvelles fonctionnalités
- Supprimer ou modifier des fonctionnalités déjà livrées

La maintenance est également nécessaire pour préserver ou améliorer les caractéristiques de qualité **non-fonctionnelles** du composant ou du système pendant toute sa durée de vie, en particulier la **performance**, la **compatibilité**, la **fiabilité**, la **sécurité** et la **portabilité**

Tester pendant le cycle de vie du développement logiciel

tests de maintenance

Facteurs déclencheurs pour la maintenance

Il y a plusieurs raisons pour lesquelles la maintenance logicielle, et donc les tests de maintenance, ont lieu, à la fois pour les changements planifiés et non planifiés:

- Modification, comme des améliorations planifiées (p. ex., basées sur les versions)
- Les changements correctifs et d'urgence
- Les changements de l'environnement opérationnel (comme des mises à niveau planifiées du système d'exploitation ou de la base de données)

Tester pendant le cycle de vie du développement logiciel

tests de maintenance

Facteurs déclencheurs pour la maintenance

Il y a plusieurs raisons pour lesquelles la maintenance logicielle, et donc les tests de maintenance, ont lieu, à la fois pour les changements planifiés et non planifiés:

- Les correctifs pour les défauts et les vulnérabilités
- Migration d'une plate-forme à une autre, qui peut nécessiter des tests opérationnels du nouvel environnement ainsi que du logiciel modifié, ou des tests de conversion de données lorsque les données d'une autre application seront migrées dans le système en cours de maintenance.
- Déclassement, par exemple lorsqu'une application arrive en fin de vie.

Tester pendant le cycle de vie du développement logiciel

tests de maintenance

Analyse d'impact pour la maintenance

- Evalue les changements qui ont été apportés pour une version de maintenance afin d'identifier les conséquences prévues ainsi que des effets secondaires attendus et possibles d'un changement

- Identifie les zones du système qui seront affectées par le changement.

- Identifie l'impact d'un changement sur les tests existants. Les effets secondaires et les zones affectées dans le système doivent être testés pour les régressions, éventuellement après la mise à jour des tests existants affectés par le changement.

- L'analyse d'impact peut être effectuée avant qu'un changement ne soit apporté, pour aider à déterminer si le changement devrait être apporté en fonction des conséquences potentielles dans d'autres parties du système.

Tester pendant le cycle de vie du développement logiciel

tests de maintenance

Analyse d'impact pour la maintenance

L'analyse d'impact peut être difficile si :

- Les spécifications (p. ex., exigences métier, User Stories, architecture) sont obsolètes ou manquantes
- Les cas de test ne sont pas documentés ou sont obsolètes
- La traçabilité bidirectionnelle entre les tests et les bases de test n'a pas été maintenue
- Le support de l'outillage est faible ou inexistant
- Les personnes impliquées n'ont pas de connaissance du domaine et/ou du système
- Une attention insuffisante a été accordée à la maintenabilité du logiciel au cours du développement