Tests des logiciels (Certification ISTQB)

Département Informatique

2023-2024

Mariem HAOUES

Maître-assistant, FSB

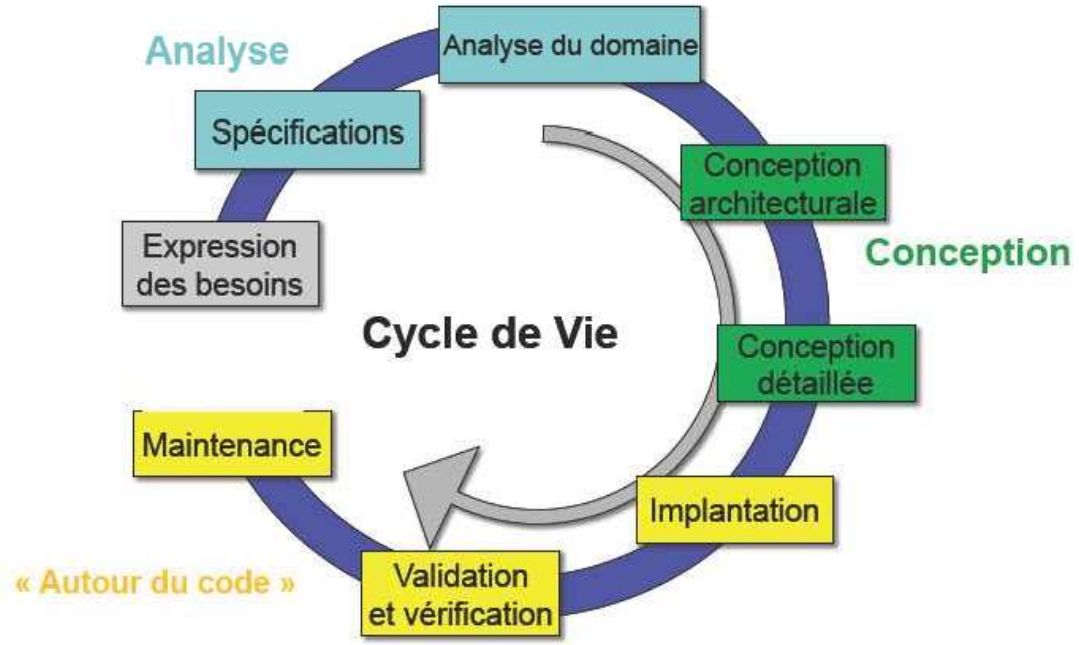
Université de Carthage

Chapitre 2

Tester pendant le cycle de vie du développement logiciel

1. Modèles de développement logiciel

Un modèle de cycle de vie de développement logiciel décrit les types d'activités réalisées à chaque étape d'un projet de développement logiciel, et la façon dont les activités sont reliées entre elles logiquement et chronologiquement



1. Modèles de développement logiciels

- **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
  - 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, et sont impliqués dans la revue des produits d'activités dès que des versions préliminaires sont disponibles

1. Modèles de développement logiciels

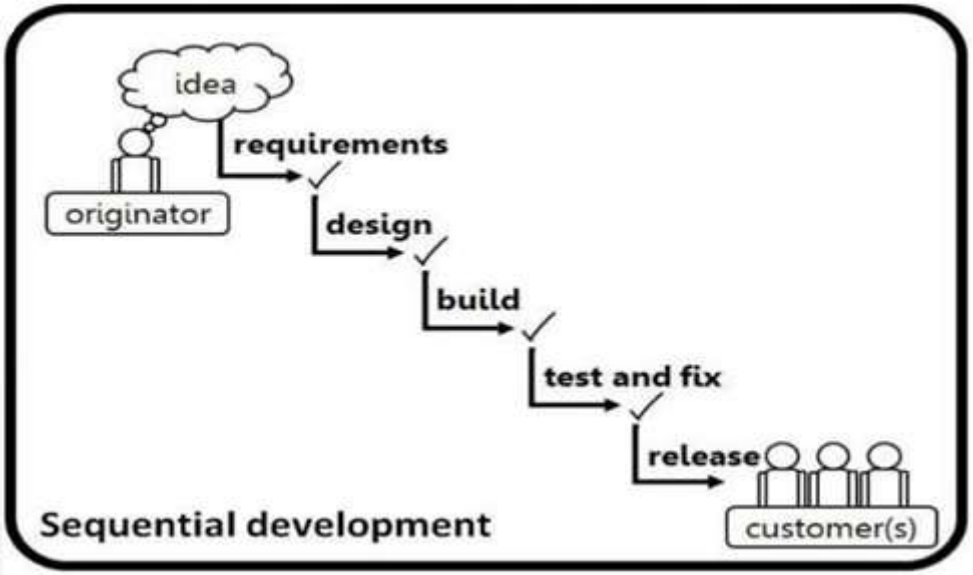
SDLC

Séquentiel : toute phase du processus de développement devrait commencer lorsque la phase précédente est terminée

Itératif & incrémental : les fonctionnalités du logiciel augmentent de façon incrémentale. La taille de ces incréments de fonctionnalités varie, certaines méthodes ayant des étapes plus grandes et d'autres plus petites

1. Modèles de développement logiciels

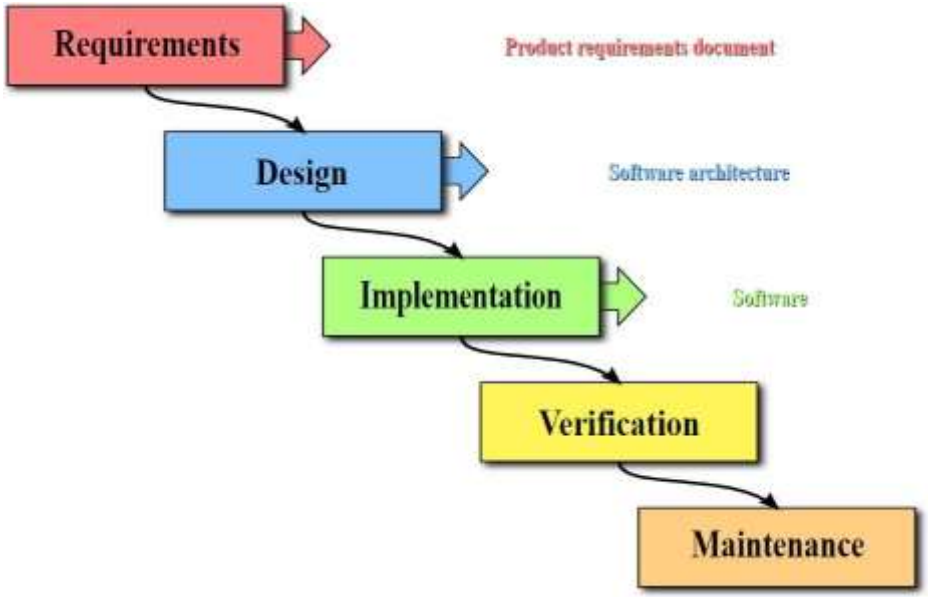
- Un modèle de développement séquentiel décrit le processus de développement logiciel comme un flux linéaire et séquentiel d'activités
- Cela signifie que toute phase du processus de développement devrait commencer lorsque la phase précédente est terminée
- En théorie, il n'y a pas de chevauchement des phases, mais dans la pratique, il est bénéfique d'avoir un retour rapide de la phase suivante



1. Modèles de développement logiciels

Modèle en cascade

- Les activités de développement sont réalisées l'une après l'autre
- Dans ce modèle, les activités de test ont seulement lieu une fois que toutes les autres activités de développement sont terminées



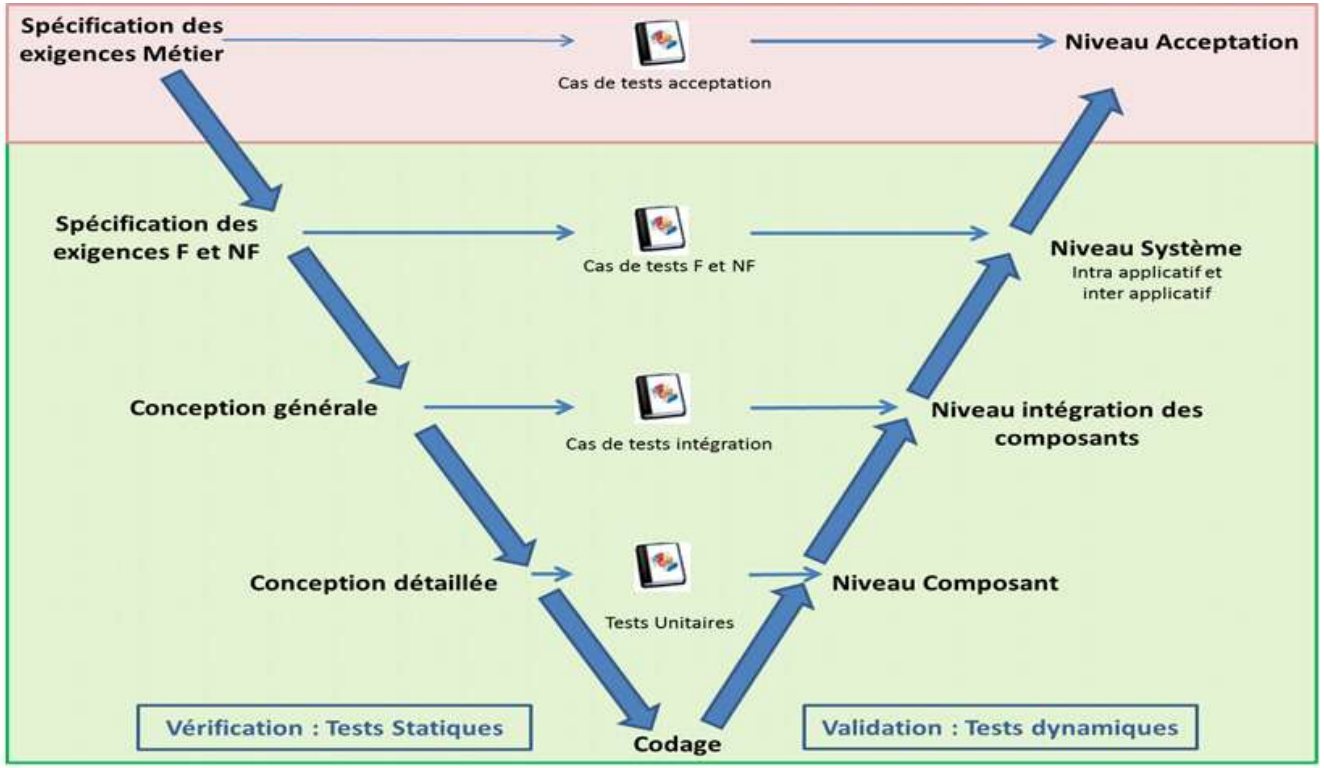
Quand l'utiliser ?

- Quand les besoins sont connus et stables
- Quand la technologie à utiliser est maîtrisée
- Lors de la création d'une nouvelle version d'un produit existant
- Lors du portage d'un produit sur une autre plateforme

1. Modèles de développement logiciels

Quand l'utiliser ?

- Quand le produit à developer a de très hautes exigences de qualité
- Quand les besoins sont connus à l'avance
- Les technologies à utiliser sont connues à l'avance



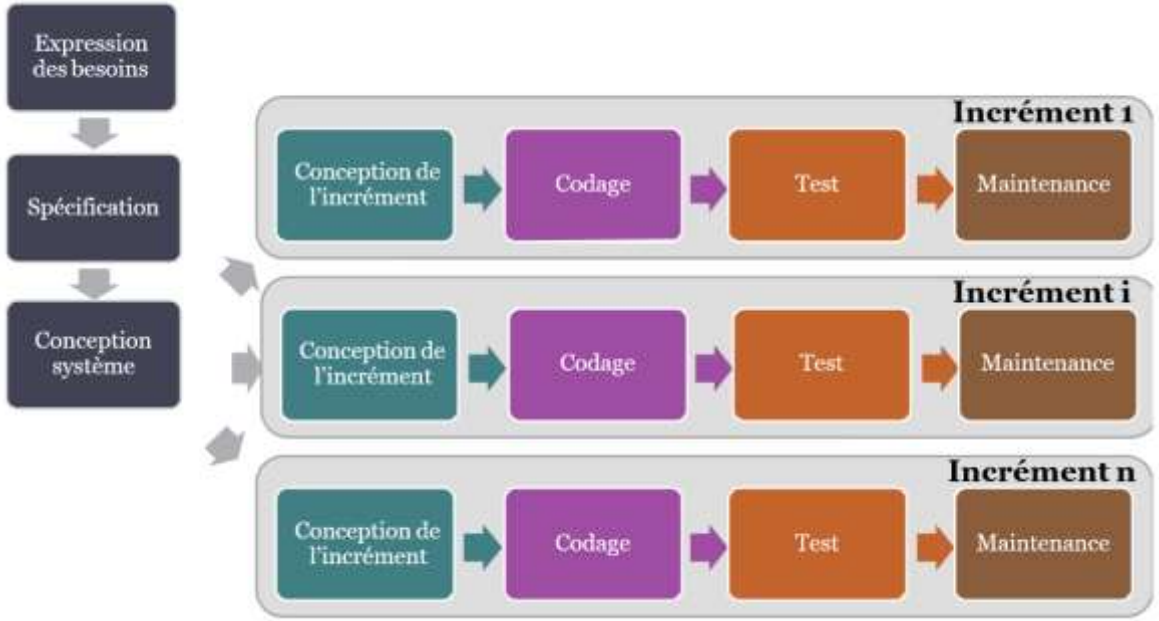
Modèle en V

Contrairement au modèle en 'cascade', le modèle en V intègre le processus de test tout au long du processus de développement, en appliquant le principe de «Tester tôt»

1. Modèles de développement logiciels

Le développement incrémental

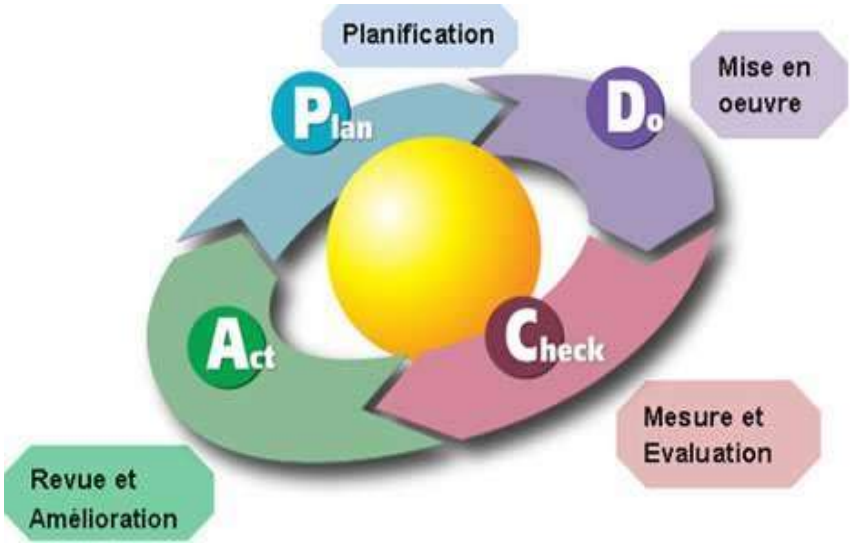
- Le développement incrémental implique la définition des exigences, la conception, le développement et le test d'un système par morceaux, ce qui signifie que les fonctionnalités du logiciel augmentent de façon incrémentale
- La taille de ces incréments de fonctionnalités varie, certaines méthodes ayant des étapes plus grandes et d'autres plus petites
- Les incréments de caractéristiques peuvent être aussi petits qu'une unique modification d'un écran d'interface utilisateur ou qu'une nouvelle option de requête



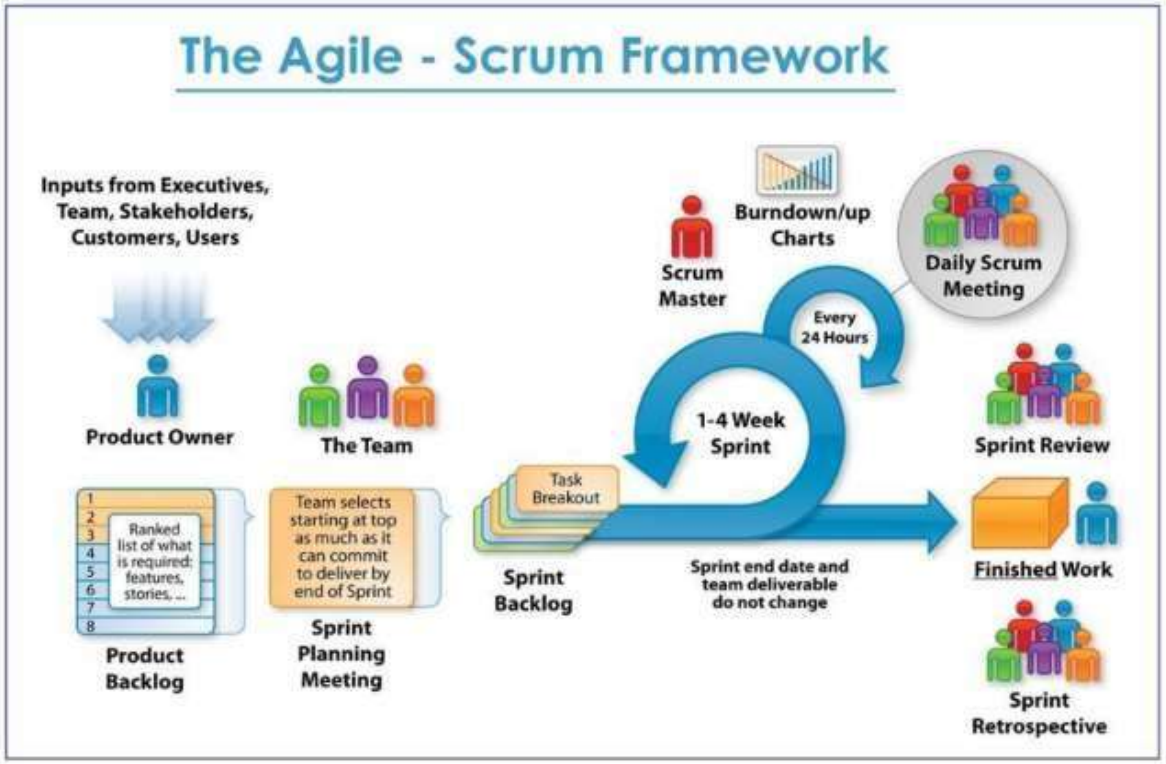
1. Modèles de développement logiciels

Le développement itératif

- Le 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



1. Modèles de développement logiciels



Scrum

Chaque itération a tendance à être relativement courte (p. ex., heures, jours ou quelques semaines), et les incréments de caractéristiques sont proportionnellement petits, de l'ordre de quelques améliorations et/ou deux ou trois nouvelles caractéristiques

1. Modèles de développement logiciels

La livraison continue

- Dans certains cas, les équipes utilisent la livraison continue ou le déploiement continu, les deux impliquant une automatisation significative de multiples niveaux de test dans le cadre de leurs flux de livraison
- De nombreux efforts de développement utilisant ces méthodes incluent également le concept d'équipes auto-organisées
- Les tests de régression sont de plus en plus importants à mesure que le système grossit

1. Modèles de développement logiciels

- En outre, les modèles de cycle de vie du développement logiciel eux-mêmes peuvent être combinés
- Par exemple, un modèle en V peut être utilisé pour le développement et le test des systèmes back-end et de leurs intégrations, tandis qu'un modèle de développement Agile peut être utilisé pour développer et tester l'interface utilisateur front-end (IHM) et les fonctionnalités
- Le prototypage peut être utilisé au début d'un projet, avec un modèle de développement incrémental adopté une fois la phase expérimentale terminée

Quiz 1/2

1. **Les méthodes agiles**
    a. Mettent en avant la souplesse des programmeurs car ils doivent programmer dans différentes postures pour maintenir une forte concentration
    b. Mettent en avant les tests qui doivent être réalisés très régulièrement
    c. Rendent inutiles les tests car le code ne peut contenir des erreurs avec ces techniques
    d. Déconseillent l'utilisation des tests car ceux-ci risquent de « casser » l'aspect agile des méthodes

2. **Les défauts sont introduits**
    a. Tout au long du cycle du projet
    b. Uniquement dès que l'on programme, les défauts ne sont essentiellement que les conséquences d'erreurs de programmation
    c. Éventuellement en conséquence d'une spécification ambiguë
    d. Essentiellement volontairement et de manière malveillante par les équipes projet

Quiz 2/2

3. Quel événement Scrum permet à l'équipe de rétrospective de discuter des améliorations à apporter à son processus ?
    a. Sprint Review
    b. Daily Scrum
    c. Sprint Planning
    d. Sprint Retrospective

4. Quelle est la principale caractéristique du développement itératif ?
    a. Approche linéaire
    b. Approche séquentielle
    c. Répétition cyclique des phases
    d. Livraison unique à la fin du projet

2. Niveaux de test

- Les niveaux de test sont des groupes d'activités de test qui sont organisées et gérées ensemble
- Chaque niveau de test est une instance du processus de test
- Les niveaux de test sont liés à d'autres activités du cycle de vie du développement logiciel
- Les niveaux de test sont caractérisés par les éléments suivants :
  - Objectifs spécifiques
  - Bases de test, référencées pour en dériver des cas de test
  - Objet de test (c.-à-d. ce qui est testé)
  - Défauts et défaillances types
  - Approches et responsabilités spécifiques

- Pour chaque niveau de test, un environnement de test approprié est requis

2. Niveaux de test

Test de composants

Les tests de composants (également connus sous le nom de tests unitaires ou de modules) se concentrent sur des composants qui peuvent être testés séparément

Objectifs	Réduire le risque, Vérifier les comportements fonctionnels et non-fonctionnels, Renforcer la confiance, Trouver des défauts, Empêcher les défauts
Bases de test	Conception détaillée, Code, Spécifications des composants
Objets	Composants, unités ou modules, Code et structures de données, Classes, Modules de bases de données
Défauts et défaillances courants	Fonctionnalité incorrecte, Problèmes de flux de Données, Code et logique incorrects
Approches spécifiques et responsabilités	Fait par les développeurs

2. Niveaux de test

Test d'intégration

Les tests d'intégration se concentrent sur les interactions entre les composants ou les systèmes

Objectifs	Réduire le risque, Vérifier les comportements fonctionnels et non-fonctionnels, Renforcer la confiance, Trouver des défauts, Empêcher les défauts
Bases de test	Conception du logiciel et du système Diagrammes de séquence Spécifications des protocoles d'interface et de communication Cas d'utilisation Architecture au niveau du composant ou du système Workflows Définitions des interfaces externes
Objets	Sous-systèmes Bases de données Infrastructure Interfaces APIs Micro-services
Défauts et défaillances courants	Données incorrectes Mauvais séquencement ou synchronisation des appels d'interfaces Décalages au niveau des interfaces Défaillances dans la communication Défaillances de communication non gérées ou mal gérées
Approches spécifiques et responsabilités	Fait par les testeurs

2. Niveaux de test

Test système

Les tests système se concentrent sur le comportement et les capacités d'un système ou d'un produit entier, en considérant souvent les tâches de bout en bout que le système peut exécuter et les comportements non-fonctionnels qu'il présente pendant l'exécution de ces tâches

Objectifs	Réduire le risque Vérifier les comportements fonctionnels et non-fonctionnels Renforcer la confiance Trouver des défauts Empêcher les défauts
Bases de test	Spécifications des exigences système et logicielles Rapports d'analyse des risques Cas d'utilisation Epics et User Stories Modèles de comportement du système Diagrammes d'états Manuels système et manuels d'utilisation
Objets	Applications Systèmes Matériel/Logiciel Systèmes d'exploitation Système sous test, Configuration du système et données de configuration
Défauts et défaillances courants	Calculs incorrects Comportement fonctionnel ou non-fonctionnel incorrect ou inattendu Flux de contrôle et/ou de données incorrects Réalisation incorrecte et incomplète des tâches fonctionnelles 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
Approches spécifiques et responsabilités	Fait par les testeurs

2. Niveaux de test

Test d'acceptation

Les tests d'acceptation, comme les tests système, se concentrent généralement sur le comportement et les capacités d'un système ou d'un produit entier

Objectifs	Vérifier si les comportements fonctionnels et non-fonctionnels Établir la confiance Valider que le système est complet et qu'il fonctionnera comme attendu
Bases de test	Processus métier Exigences utilisateur ou métier Réglementations, contrats légaux et normes Cas d'utilisation Exigences système Documentation système ou utilisateur Procédures d'installation Rapports d'analyse des risques
Objets	Systèmes sous test Configuration du système et données de configuration Processus métier pour un système entièrement intégré Systèmes de récupération et sites sensibles Processus d'exploitation et de maintenance Formulaires Rapports Données de production existantes et modifiées
Défauts et défaillances courants	Les workflows du système ne répondent pas aux exigences métier ou utilisateurs Les règles métier ne sont pas correctement implémentées Le système ne satisfait pas aux exigences contractuelles ou réglementaires Les défaillances non-fonctionnelles
Approches spécifiques et responsabilités	Fait par les testeurs/des clients/ des utilisateurs métier/ des Product Owners ou des exploitants d'un système

2. Niveaux de test

Test d'acceptation

Alpha-test (alpha testing): test effectué en phase de développement, avant la distribution du produit (→ alpha-versions du produit)

Bêta-test (beta testing): test effectué après l'alpha-test, en distribuant le produit (→ des bêta-versions) à un groupe limité d'utilisateurs avertis

Quiz 1/2

1. L'objectif principal des tests d'acceptation est ?
    a) S’assurer du fonctionnement des interactions et de l’interface entre les différents composants
    b) Rechercher des défauts dans les composants logiciels testables
    c) Confirmer que le produit final correspond bien aux besoins des utilisateurs finaux
    d) Aucune de ces réponses
2. Quel niveau de test est principalement axé sur le renforcement de la confiance plutôt que sur la recherche de défauts ?
    a) Tests unitaires
    b) Tests d'intégration
    c) Tests d'acceptation
    d) Test du système
3. Le beta testing est :
    a) Réalisé par les clients sur leur propre site
    b) Réalisé par les clients sur le site du développeur logiciel
    c) Réalisé par une équipe de test indépendante
    d) Utile pour tester un logiciel développé pour un client ou utilisateur spécifique

Quiz 2/2

4. Tester un sous-programme, que ce soit pour des aspects fonctionnels ou non fonctionnels, est l'objectif principal de quel niveau de test ?

    a) Test de composant

    b) Test d'intégration de composants

    c) Test système

    d) Test d'acceptation

5. Quelles sont les caractéristiques des bonnes pratiques suivantes à appliquer quel que soit le modèle de cycle de développement ?

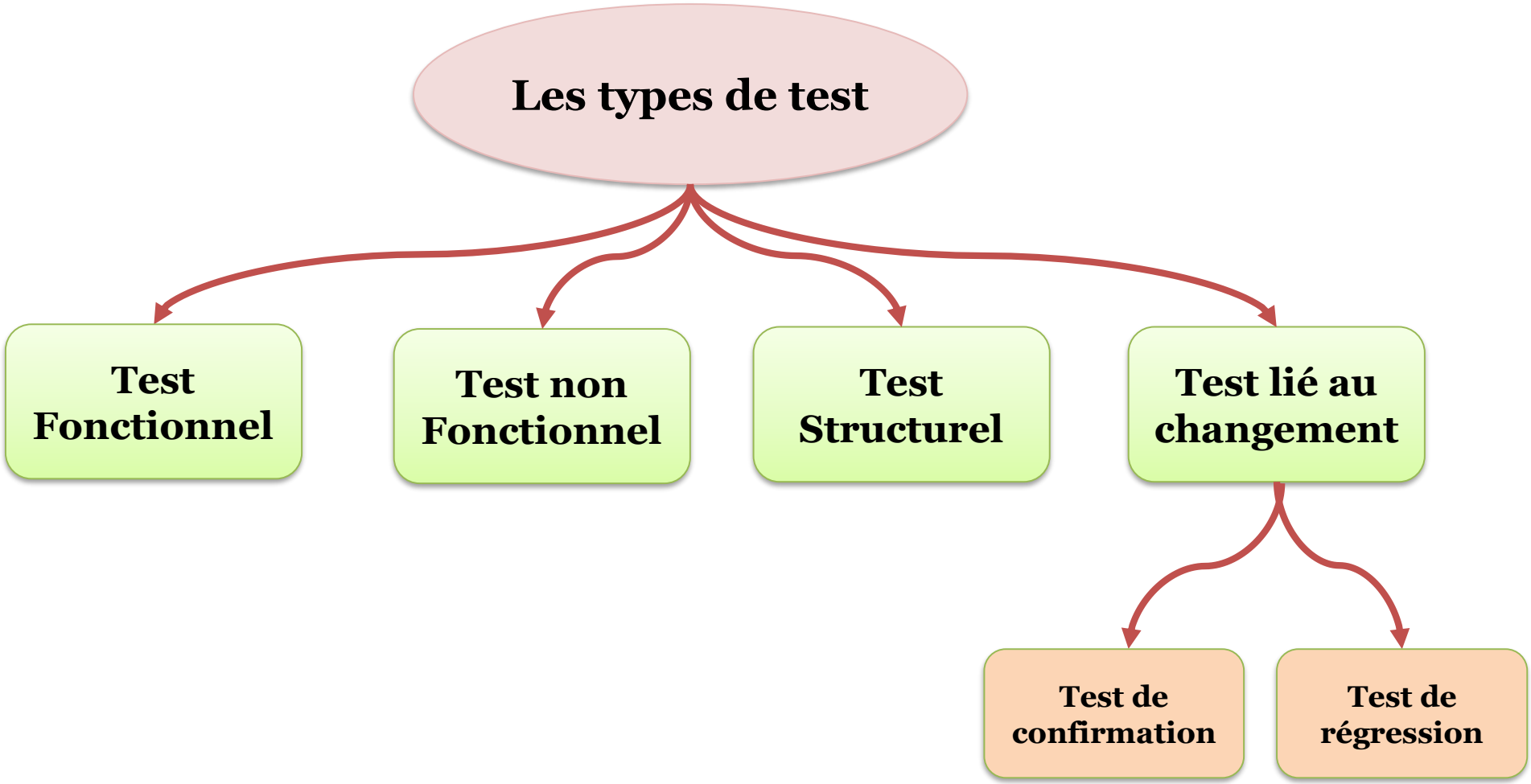
    a) Les tests d'acceptation sont toujours le dernier niveau de tests

    b) Tous les niveaux de tests sont planifiés et complétés pour chaque caractéristique de développement

    c) Les testeurs sont impliqués dès que le premier bout de code peut être exécuté

    d) Pour chaque activité de développement correspond une activité de tests

3. Types de test



3. Types de test

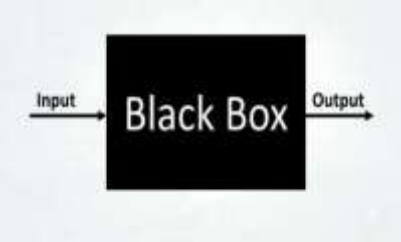
- Un type de test est un groupe d'activités de test visant à tester des caractéristiques spécifiques d'un système logiciel ou d'une partie d'un système, sur la base d'objectifs de test spécifiques

Tests fonctionnels

- Les tests fonctionnels d'un système impliquent des tests qui évaluent les fonctionnalités que le système devrait réaliser
- Les exigences fonctionnelles peuvent être décrites dans des produits d'activités, tels que :
  - les spécifications des exigences métier
  - les épics
  - les User Stories
  - les cas d'utilisation
  - les spécifications fonctionnelles
  - elles peuvent ne pas être documentées

3. Types de test

Tests fonctionnels



- Les fonctionnalités sont "ce que" le système doit faire
- Les tests fonctionnels devraient être effectués à tous les niveaux de test
- Les tests fonctionnels prennent en compte le comportement du logiciel, de sorte que des techniques boîte-noire peuvent être utilisées pour dériver des conditions de test et des cas de test pour la fonctionnalité du composant ou du système

Tests non-fonctionnels

- Les tests non-fonctionnels d'un système évaluent les caractéristiques des systèmes et des logiciels comme l'utilisabilité, la performance ou la sécurité
- Le test non-fonctionnel est le test de "comment" le système se comporte
- Les tests non-fonctionnels peuvent et devraient souvent être effectués à tous les niveaux de test

3. Types de test

Tests boîte-blanche

- Les tests boîte-blanche sont des tests basés sur la structure ou l'implémentation interne du système
- La structure interne peut comprendre le code, l'architecture, les flux de travail et/ou les flux de données au sein du système

Tests liés aux changements

- Lorsque des modifications sont apportées à un système, que ce soit pour corriger un défaut ou en raison d'une fonctionnalité nouvelle ou modifiée, des tests devraient être effectués pour confirmer que les modifications ont corrigé le défaut ou implémenté la fonctionnalité correctement, et n'ont pas causé de conséquences préjudiciables inattendues

3. Types de test

Test de confirmation

- Après la correction d'un défaut, le logiciel doit être testé
- A minima, les étapes pour reproduire le(s) défaut(s) causé(s) par le défaut doivent être ré-exécutées sur la nouvelle version du logiciel
- Le but d'un test de confirmation est de confirmer que le défaut d'origine a été réparé avec succès

Tests de régression

- Il est possible qu'une modification apportée à une partie du code, qu'il s'agisse d'une correction ou d'un autre type de modification, puisse accidentellement affecter le comportement d'autres parties du code
- Les tests de régression sont des tests visant à détecter de tels effets de bord involontaires
- Les suites de tests de régression sont exécutées plusieurs fois et évoluent généralement lentement, raison pour laquelle les tests de régression sont de bons candidats pour l'automatisation
- L'automatisation de ces tests devrait commencer tôt dans le projet
- Des tests de confirmation et de régression sont effectués à tous les niveaux de test

4. Tests de maintenance

- Une fois déployés dans les environnements de production, les logiciels et les systèmes doivent être maintenus
- Les tests de maintenance se concentrent sur le test des changements apportés au système, ainsi que sur le test des parties inchangées qui auraient pu être affectées par les changements
- L'analyse d'impact évalue les changements qui ont été apportés pour une version de maintenance afin d'identifier les conséquences imprévues
- L'analyse d'impact peut également aider à identifier 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
- L'analyse d'impact peut être effectuée avant qu'un changement ne soit apporté, pour aider à déterminer si le changement devrait être apporté

Facteurs déclencheurs pour la maintenance

Quiz 1/2

1. Pendant quel niveau de test les tests non fonctionnels doivent-ils être exécutés ?
    a) Unité et intégration uniquement
    b) Test du système uniquement
    c) Intégration, système et acceptation uniquement
    d) Unitaire, intégration, système et acceptation seulement
2. Lequel des énoncés suivants est **VRAI** ?
    a) Le but des tests de régression est de vérifier si la correction a été effectuée avec succès, tandis que le but du test de confirmation est de confirmer que la correction n'a pas d'effets de bord
    b) Le but des tests de régression est de vérifier si la nouvelle fonctionnalité fonctionne, alors que le but des tests de confirmation est de vérifier si le défaut d'origine a été corrigé
    c) Le but des tests de régression est de détecter les effets de bord sur le produit, tandis que le but des tests de confirmation est de vérifier si le défaut d'origine a été corrigé
    d) Aucune de ces réponses
3. Les tests non fonctionnels comprennent ?
    a) Tester pour voir où le système ne fonctionne pas correctement
    b) Tester les attributs de qualité du système, y compris la performance et la fiabilité
    c) Tester des fonctions qui ne devraient pas exister
    d) Aucune de ces réponses

Quiz 2/2

4. Quel test ci-dessous est considéré comme un test fonctionnel ?
    a) Mesurer le temps de réponse sur un système de réservation en ligne
    b) Vérifier l'effet de volumes élevés de trafic dans un système de centre d'appels
    c) Vérifier la concordance des informations affichées sur l'écran de réservation en ligne et du contenu de la base de données avec les informations figurant dans la lettre adressée aux clients
    d) Évaluer la facilité d'utilisation du système

5. Supposons que vous testiez un système bancaire en ligne. Vous recevez un projet des spécifications techniques décrivant l'architecture du système au niveau des interfaces avec d'autres applications. Dès que vous recevez ce document, vous concevez des tests pour couvrir tous les messages et échanges d'interfaces, afin de réaliser simultanément des tests de bout en bout et des composants du système pendant les phases d'intégration et de tests système de systèmes. Quel type de test réalisez-vous ?
    a) Tests structurels
    b) Tests d'intégration
    c) Tests système
    d) Tests fonctionnels