



Test & validation (P1)

Conformément à la certification ISTQB
(International Software Testing Qualification Board)

Plan du cours

Introduction

1. Fondamentaux de Test
2. Test pendant le cycle de vie
3. Techniques statiques
4. Technique de conception des tests
5. Gestion des tests
6. Outils de support aux tests

Introduction

Introduction

- 04 Juin 1996 : l'explosion d'Ariane 5 : une catastrophe informatique

causé par un dysfonctionnement informatique (appelé aussi **Bug**)

dû à un **dépassement d'un entier** dans les registres mémoire des calculateurs électroniques utilisés par le pilote automatique



Introduction

Les 10 plus grandes catastrophes
liées à l'informatique d'après Colin
Barker (<http://vudestic.over-blog.com/article-14376549.html>)
:

Introduction

- Un ingénieur /consultant/chef de projet doit être un bon testeur de logiciel
- Les tests garantissent la qualité du logiciel en terme de fiabilité et de performance et le gain en temps et en argent et évite de nombreux problèmes qui peuvent mettre fin à la vie d'un logiciel
- La formation ISTQB :
 - Une grande ouverture d'Esprit sur le concept de tests
 - Maîtrise de mise en œuvre de tout un projet de test
 - Une accréditation internationale
 - Condition d'entrée : ISTQB Foundation Certificate in Software Testing
 - Le contenu : le syllabus et le glossaire

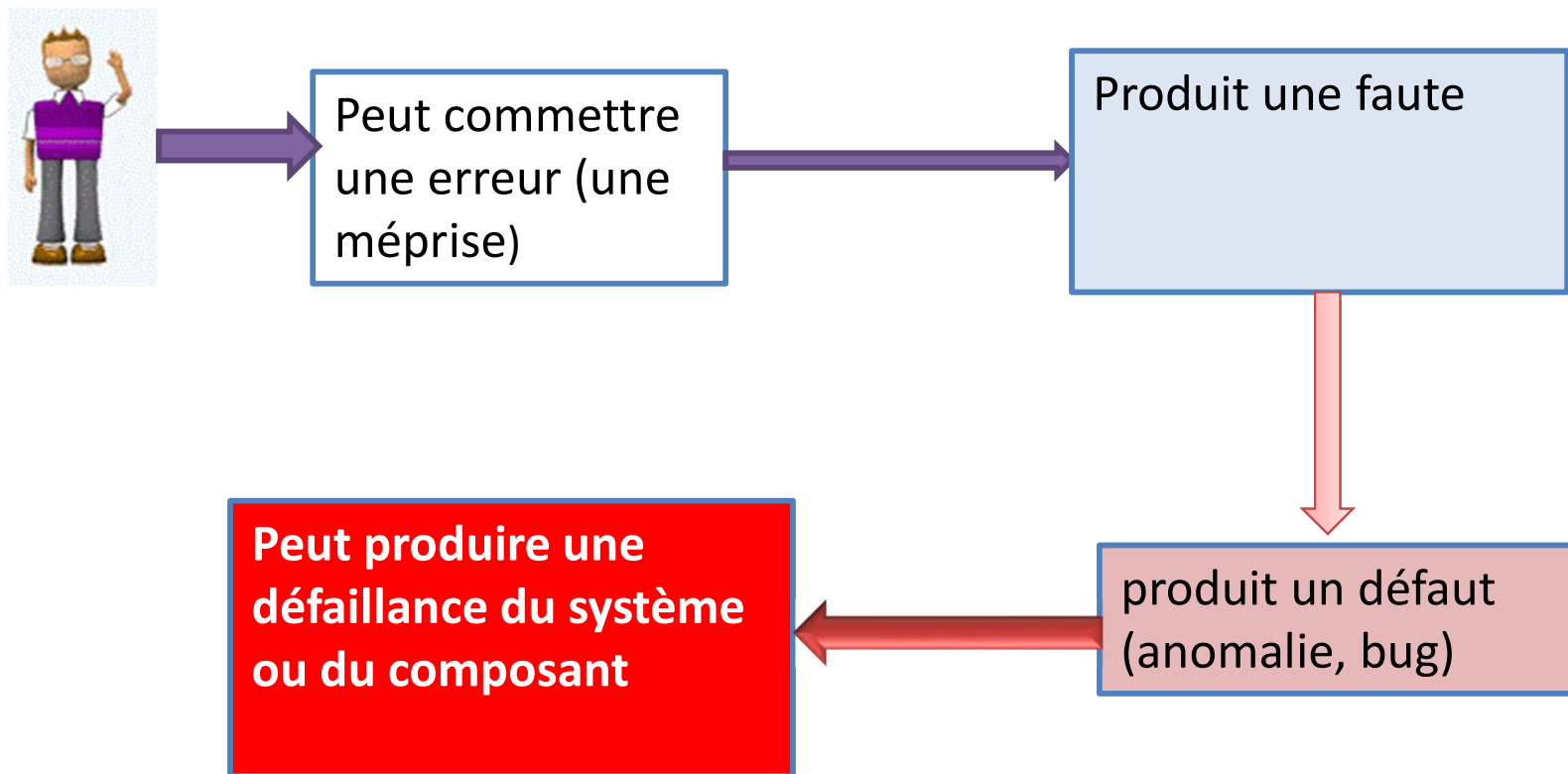
Organisation du module

- Cours :pour la compréhension et la préparation à la certification ISTQB
- TD : exercices et examens blancs
- TP :les outils de tests sur différentes plateformes

1-Les fondamentaux de test

Pourquoi Tester ?

➤ Origine des défauts logiciels :



1-Les fondamentaux de test

1-Les fondamentaux de test

Pourquoi Tester ?

- Un être humain peut faire une erreur (méprise)
- Une erreur peut produire un défaut (bug) dans le code, dans un logiciel ou un système, ou dans un document.
- Si un défaut dans du code est exécuté, le système n'effectuera pas ce qu'il aurait dû faire (ou fera ce qu'il n'aurait pas dû faire), générant une défaillance.
- Des défauts dans les logiciels, systèmes ou documents peuvent générer des défaillances, mais tous les défauts ne le font pas.



1-Les fondamentaux de test

Pourquoi Tester ?

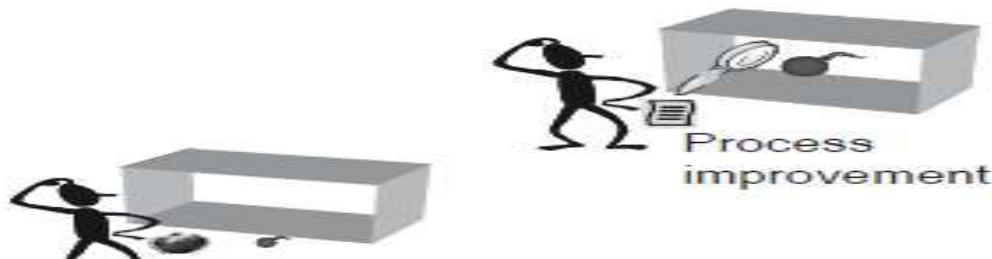
- Les défauts apparaissent parce que les humains peuvent se tromper et à cause des échéanciers serrés, de la complexité du code et des infrastructures, des modifications de technologies et/ou de multiples interactions entre les systèmes
- Les défaillances peuvent être aussi causées par des conditions d'environnement : radiations, magnétisme, champs électromagnétiques et pollution peuvent causer des défauts dans les microprogrammes ou influencer l'exécution des logiciels en modifiant les conditions matérielles.

1-Les fondamentaux de test

Pourquoi Tester ?

➤ Tests et qualité

- Avec l'aide des tests, il est possible de mesurer la qualité des logiciels en termes de défauts trouvés, pour des caractéristiques et exigences tant fonctionnelles que non-fonctionnelles (p.ex. fiabilité, utilisabilité, rentabilité (coût et bénéfice), maintenabilité(modularité, réutilisabilité ,facilité d'analyse et de modification) et portabilité (multiplateforme, paramétrable)).
- Les tests peuvent augmenter le niveau de confiance en la qualité d'un logiciel s'ils trouvent peu ou pas de défauts. Un test conçu correctement et qui est exécuté sans erreur réduit le niveau de risque général du système. Quand les tests trouvent des défauts, la qualité du système logiciel s'accroît quand ces défauts sont corrigés.



1-Les fondamentaux de test

Que sont les tests?

- perception habituelle :
 - consistent uniquement en l'exécution de tests
 - exécuter le logiciel, est une partie des tests, mais pas l'ensemble des activités de test.
- les tests existent avant et après l'exécution des tests.
- Les activités de test sont:
 - la planification et le contrôle
 - la sélection des conditions de test
 - la conception et l'exécution des cas de tests
 - la vérification des résultats
 - L'évaluation des critères de sortie.
 - la réalisation et la finalisation des activités de clôture définitive à la fin d'une phase de test .

1-Les fondamentaux de test

Que sont les tests?

➤

- la revue des documents (incluant le code source) et les analyses statiques.
 - **Revue** : une évaluation d'un état d'un produit ou projet pour s'assurer des déviations par rapport aux résultats planifiés et recommander des améliorations. Exemples : revue de gestion, revue informelle, revue technique, inspection et relecture ,etc.
 - **Réviseur** : la personne impliquée dans une revue qui identifiera et décrira les anomalies dans le produit ou projet en revue. Les réviseurs peuvent être choisis pour représenter divers points de vue ou rôles dans le processus de revue.

1-Les fondamentaux de test

Que sont les tests?

- **Test statique et test dynamique :**
 - **Test dynamique** : test qui nécessite l'exécution du logiciel d'un composant ou système.
 - **Test statique** : repose sur l'examen manuel (revues) ou l'analyse (analyse statique) du code ou de la documentation du projet sans exécution du code.
 - **Les tests dynamiques et les tests statiques** peuvent être utilisés comme des moyens pour atteindre des objectifs similaires, et fourniront des informations permettant l'amélioration du système à tester et des processus de développement et de test.
- **Tester et déboguer :**
 - Les tests dynamiques peuvent montrer des défaillances causées par des défauts.
 - Déboguer est l'activité de développement qui trouve, analyse et supprime les causes de la défaillance. La responsabilité de chaque activité est différente : les testeurs testent, les développeurs déboguent.

1-Les fondamentaux de test

Les 7 principes généraux de tests

1. **Les tests montrent la présence de défauts**
2. **Les tests exhaustifs sont impossibles** : Plutôt que des tests exhaustifs, nous utilisons l'analyse des risques et des priorités pour focaliser les efforts de tests.
3. **Tester tôt** : Pour trouver des défauts tôt et minimiser les défauts lors des tests dynamiques
4. **Regroupement des défauts** : Un petit nombre de modules contiennent généralement la majorité des défauts détectés lors des tests pré-livraison
5. **Paradoxe du pesticide** : il arrivera que le même ensemble de cas de tests ne trouvera plus de nouveaux défauts : cas de tests doivent être régulièrement revus et révisés
6. **Les tests dépendent du contexte** : ex. les logiciels de sécurité critique seront testés différemment d'un site de commerce électronique.
7. **L'illusion de l'absence d'erreurs** : système conçu inutilisable et ne comble pas les besoins et les attentes des utilisateurs.

Le processus de réalisation des tests



Le processus de réalisation des tests

➤ 1-Planification et contrôle des tests

- définir les objectifs du test et spécifier les activités de test à mettre en œuvre pour atteindre les objectifs de la mission.
- les activités de test devraient être contrôlées tout au long du projet.
- La planification des tests prend en compte le feedback des activités de contrôle et de suivi.

Le processus de réalisation des tests

➤ 2-Analyse et conception des tests

- Représente les activités où les objectifs de test généraux qui seront transformés en des conditions de test et des conceptions de test tangibles.
- Se compose des tâches majeures suivantes:
 1. Réviser les bases du test (telles que les exigences, le niveau d'intégrité logiciel1 (niveau de risque), les rapports d'analyse de risque, l'architecture, la conception et les interfaces).
 2. Evaluer la testabilité des exigences du système.
 3. Identifier et prioriser les conditions de test sur la base de l'analyse des articles de test, la spécification, le comportement et la structure du logiciel.

Le processus de réalisation des tests

➤ 2-Analyse et conception des tests

4. Concevoir et prioriser les tests de haut niveau.
 5. Identifier les données de test nécessaires pour les conditions de test et les cas de test.
 6. Concevoir l'initialisation de l'environnement de test et identifier les infrastructures et outils requis.
 - **Environnement de tests** : un environnement contenant du matériel, des instruments, des simulateurs, des outils logiciels et d'autres éléments de support nécessaires à l'exécution d'un test.
- Créer une traçabilité bidirectionnelle entre les bases de test et les cas de test.
- **Base de tests** : tous les documents à partir desquels les exigences d'un composant ou système peuvent être déduites. La documentation sur laquelle les cas de tests sont basés.

Les étapes d'un projet de test

3-Implémentation et exécution des tests

7. Finaliser, développer et prioriser les cas de test
8. Développer et prioriser les procédures de test, créer les données de test et, éventuellement, préparer les harnais de test et écrire les scripts de tests automatiques
9. Créer des suites de tests à partir des procédures de test pour une exécution rentable des tests
10. Vérifier que les environnements de tests ont été mis en place correctement
11. Vérifier et mettre à jour la traçabilité bidirectionnelle entre les bases de test et les cas de test
12. Exécuter les procédures de test soit manuellement soit en utilisant des outils d'exécution de tests, en suivant la séquence planifiée

Les étapes d'un projet de test

3-Implémentation et exécution des tests

13. Consigner les résultats de l'exécution des tests et enregistrer les identités et versions des logiciels en test, outils de test et TestWare.
14. Comparer les résultats actuels et les résultats attendus.
15. Signaler les divergences comme des incidents et les analyser de façon à établir leur cause (p.ex. défaut dans le code, dans les données de test, dans la documentation de test, ou méprise dans la manière d'exécuter le test)

Les étapes d'un projet de test

3-Implémentation et exécution des tests

- Répéter les activités de test en réponse aux actions prises pour chaque divergence. Par exemple, réexécution d'un test qui était préalablement défaillant de façon à valider une correction (test de confirmation), exécution d'un test corrigé et/ou exécution de tests de façon à s'assurer que des défauts n'ont pas été introduits dans des secteurs non modifiés du logiciel ou que le défaut corrigé n'a pas découvert d'autres défauts (test de régression)

Le processus de réalisation des tests

4-Evaluer les critères de sortie et informer

Evaluer les critères de sortie est l'activité où l'exécution des tests est évaluée en fonction des objectifs définis. Ceci devrait être fait à chaque niveau de test.

Cette évaluation consiste à réaliser les tâches suivantes :

- Vérifier les registres de tests en fonction des critères de sortie spécifiés dans la planification des tests
- Evaluer si des tests supplémentaires sont requis ou si les critères de sortie doivent être changés.
- Ecrire un rapport de synthèse des tests pour les parties prenantes

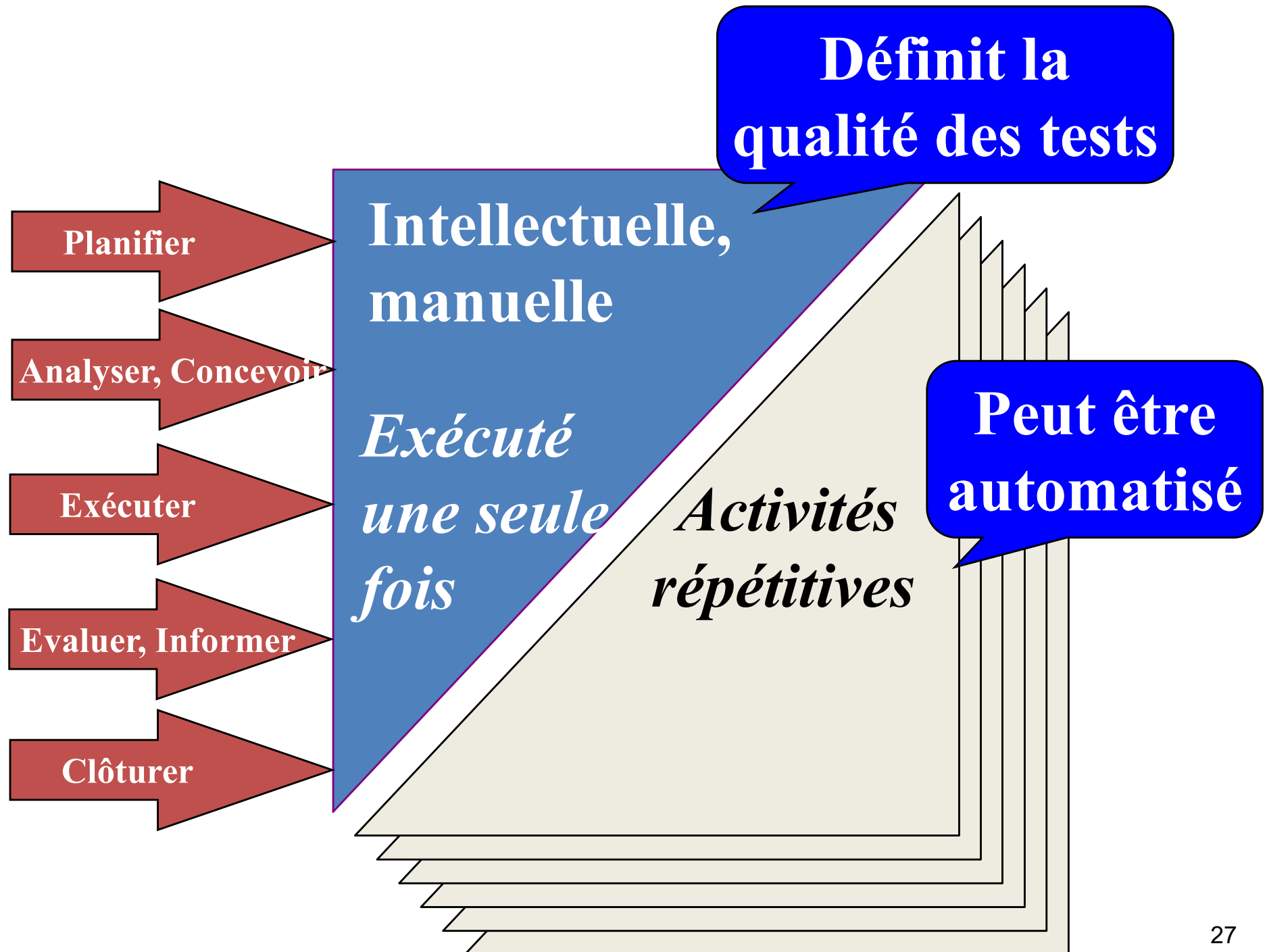
Critères de sortie :

- Ce sont les critères d'acceptation que doit satisfaire un composant ou un système de façon à être accepté par un utilisateur, client ou une autre entité autorisée.
- C'est l'ensemble des conditions génériques et spécifiques, convenues avec les responsables, pour permettre de terminer officiellement un processus.
- Les critères de sortie sont utilisés dans le test pour faire des comptes rendus et pour planifier l'arrêt du test.

Le processus de réalisation des tests

➤ 5-Activités de clôture des tests

- Vérifier les livrables par rapport aux prévisions.
- Clôturer les rapports d'incidents ou créer des demandes d'évolution pour ceux restant ouverts.
- Documenter l'acceptation du système.
- Finaliser et archiver les Testwares, environnements de test et infrastructures de test pour une réutilisation future.
- Fournir les Testwares à l'organisation en charge de la maintenance.
- Analyser les leçons apprises pour identifier les changements nécessaires pour les versions et projets futurs.
- Utiliser l'information collectée pour améliorer la maturité des tests.



Exercices

4. Soit la liste suivante d'activités du processus de test :

1. Analyse et conception
2. Activités de clôture de test
3. Evaluer les critères de sortie et informer
4. Planification et contrôle
5. Implémentation et exécution

Dans quel ordre logique ces activités se déroulent ?

- A. 1, 2 ,3 , 4 et 5
- B. 4, 1, 5 , 3 et 2
- C. 4, 5, 1, 3 et 2
- D. 1, 4, 5, 3 et 2

Exercices

5. Lequel des énoncés suivants décrit un principe clé des tests logiciels?

- A. Les tests automatisés permet d'établir plus de confiance sur la qualité des produits logiciels.
- B. Pour un système de logiciel, il est normalement impossible de tester toutes les combinaisons d'entrée et de sortie.
- C. Le test logiciel exhaustif est, avec suffisamment d'efforts et le soutien de l'outil, faisable pour tous les logiciels.
- D. Le but du test de logiciel est de démontrer l'absence de défauts dans les produits logiciels.

Exercices

6. Quels sont les défauts les moins coûteux à corriger?

- A. Les défauts utilisateurs trouvés par des consommateurs.
- B. Les défauts trouvés dans les fonctionnalités du système rarement utilisés.
- C. Les défauts détectés tôt dans le cycle de vie.
- D. Les défauts mineurs trouvés par les utilisateurs.

7. Lequel des problèmes ci-dessous représente une conséquence directe des défaillances logicielles?

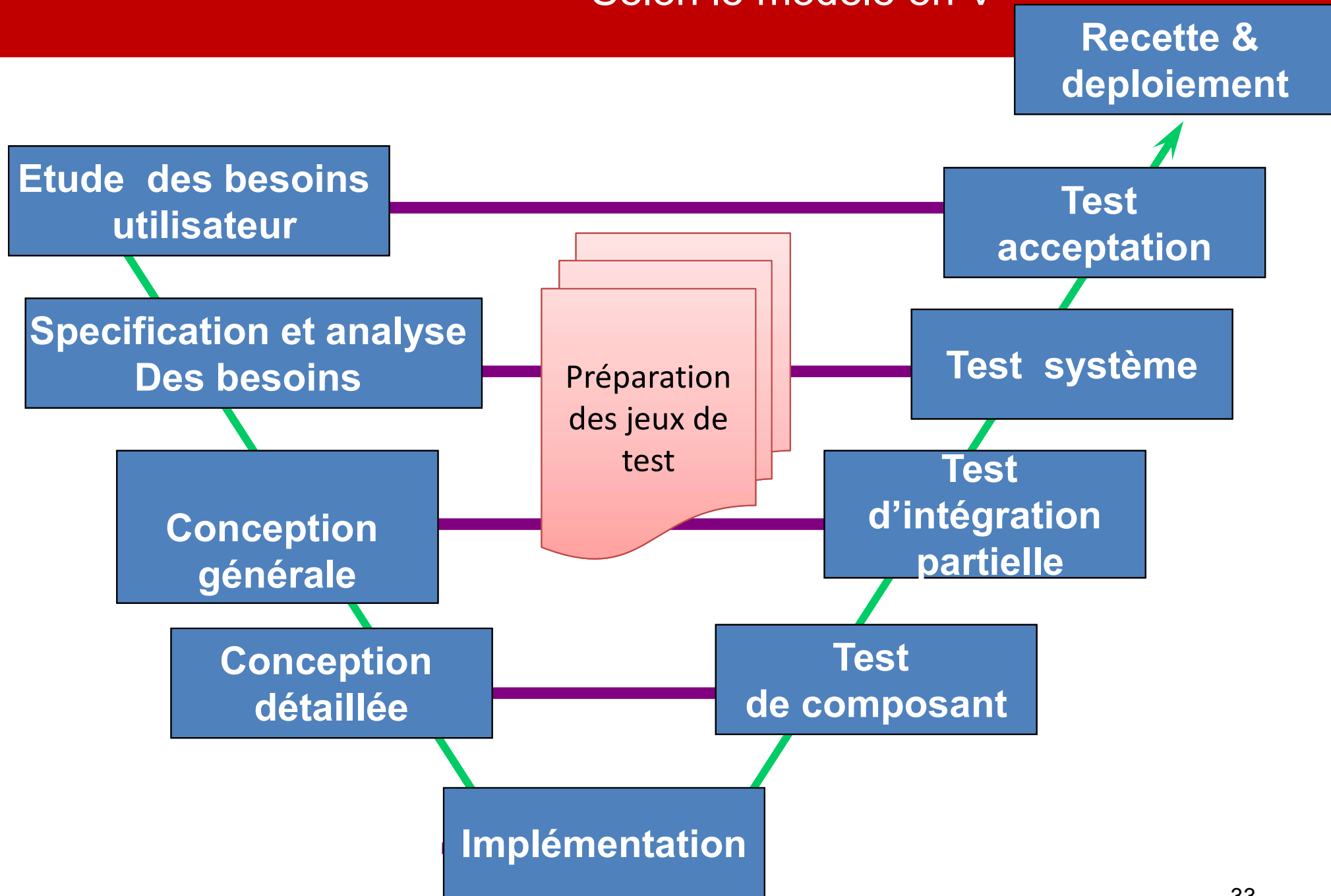
- A. Mauvaise réputation.
- B. Manque de méthodologie.
- C. Formation inadéquate.
- D. Conformité réglementaire.

2-Tests Pendant le cycle de vie

2-Tests Pendant le cycle de vie

- Les tests n'existent pas de façon isolée : les activités de test sont liées aux activités de développement logiciel.
- Les différents modèles de cycle de développement nécessitent des approches de tests différentes.
- 2 modèles:
 - modèle en V
 - Modèle itératif

2-Tests Pendant le cycle de vie Selon le modèle en V



2-Tests Pendant le cycle de vie

- **Le flux descendant:** vise à détailler le produit jusqu'à sa réalisation. Il comprend l'expression des besoins, l'analyse, la conception, puis la mise en œuvre.
 - Pour un logiciel, la mise en œuvre correspond essentiellement à la programmation.
- **le flux ascendant :** vise à valider le produit jusqu'à sa « recette », c'est-à-dire son acceptation par le client. Il comprend principalement une série de tests jusqu'à pouvoir valider que le produit répond au besoin et aux exigences.

2-Tests Pendant le cycle de vie

la dimension système :

- permet de voir le produit du projet (logiciel) comme un « système » fait de plusieurs éléments qui sont des modules ou composants.
- Ceci requiert dans le flux descendant de distinguer une conception générale du système dans son ensemble, et une conception détaillée de chaque composant.
- Dans le flux ascendant, il convient d'effectuer des tests unitaires de chaque composant, d'intégrer le système (c'est-à-dire d'assembler ses composants), puis de faire un test d'integration.

2-Tests Pendant le cycle de vie ...Modèle en V

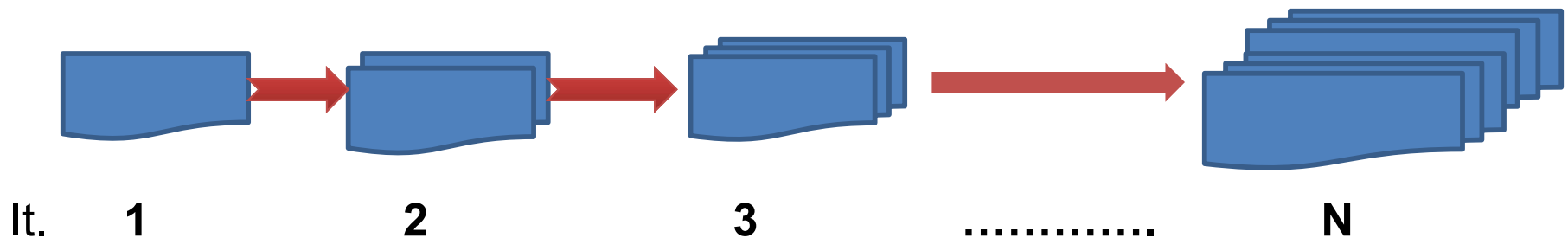
- **Les 4 niveaux de tests :**
 - Tests de composants (unitaires)
 - Tests d'intégration (de plusieurs composants dans un module ou un processus d'un module)
 - Tests système
 - Tests d'acceptation.
- On peut avoir des niveaux de développement et de tests différents, en moins ou en plus en fonction du volume du projet
- les scénarios d'utilisation ,les cas d'emploi, les spécifications d'exigences, les documents de conception et le code sont les bases de tests de un ou plusieurs niveaux de tests.

2-Tests Pendant le cycle de vie

Modèle itératif

➤ Tests d'intégration:

- Développement logiciel = une série de petits fragments à développer.
- Un fragment = exigences, conception, construction et tests
- Un incrément est le système logiciel résultant à chaque itération qui ajoute un fragment développé .il forme un système partiel en croissance.
- Plusieurs niveaux de tests
- Les tests de régression sont de plus en plus importants sur toutes les itérations



2-Tests Pendant le cycle de vie

- A chaque activité de développement, correspond une activité de test.
- Chaque niveau de test a des objectifs de tests spécifiques pour ce niveau.
- L'analyse et la conception des tests pour un niveau de test devraient commencer pendant l'activité correspondante de développement.

➤ Tests de composants

- Recherche des défauts et vérification du bon fonctionnement des modules, objets, classes qui sont testables séparément
- Peuvent inclure des tests fonctionnels, non fonctionnels et structurels
- peut se faire de façon isolée par rapport au reste du système
- Les tests de composants se font généralement avec l'accès au code du logiciel testé et sur un environnement de développement comme un framework de tests unitaire

➤ Tests de composants (suite)

➤ Bases de tests:

- Exigences des composants
- Conception détaillée
- Code

➤ Objets habituels de test:

- Composants
- Programmes
- Conversions de données / utilitaires ou programmes de migration

2-Tests Pendant le cycle de vie

Niveaux de tests

➤ Tests de composants (suite)

- Des bouchons, pilotes et simulateurs peuvent être utilisés.
 - **Bouchon** : une implémentation spéciale ou squelettique d'un composant logiciel, utilisé pour développer ou tester un composant qui l'appelle ou en est dépendant. Cela remplace un composant appelé.
 - **Pilote** : un composant logiciel ou outil de tests qui remplace un composant qui contrôle et/ou appelle un composant ou système.
 - **Simulateur** : un appareil, programme ou système utilisé pendant les tests, qui se comporte ou fonctionne comme un système donné à la réception d'entrées contrôlées .

2-Tests Pendant le cycle de vie ...Niveaux de tests

➤ Tests d'intégration

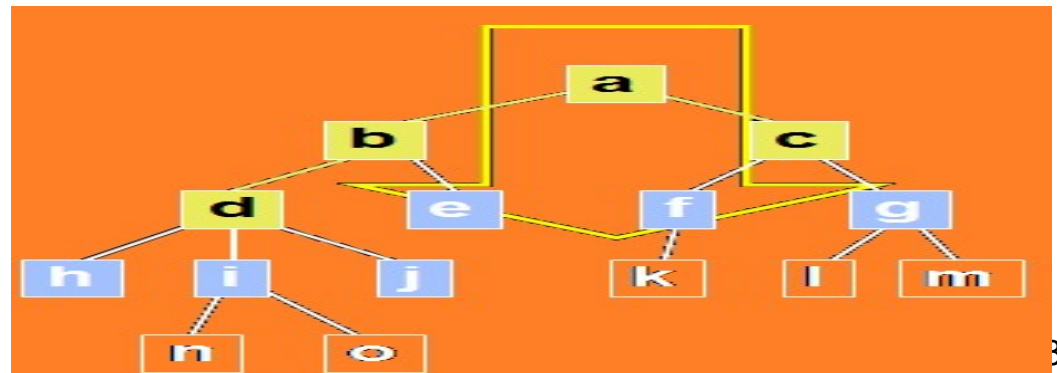
- Tests d'intégration des composants testant les interactions entre les composants logiciels
- Tests d'intégration système testant l'intégration entre les différents systèmes ou entre logiciel et matériel
- Peuvent inclure des tests fonctionnels, non fonctionnels et structurels
- Plus le nombre de composants est élevé, plus il devient difficile d'isoler les défauts

2-Tests Pendant le cycle de vie ...Niveaux de tests

➤ Stratégie d'intégration :

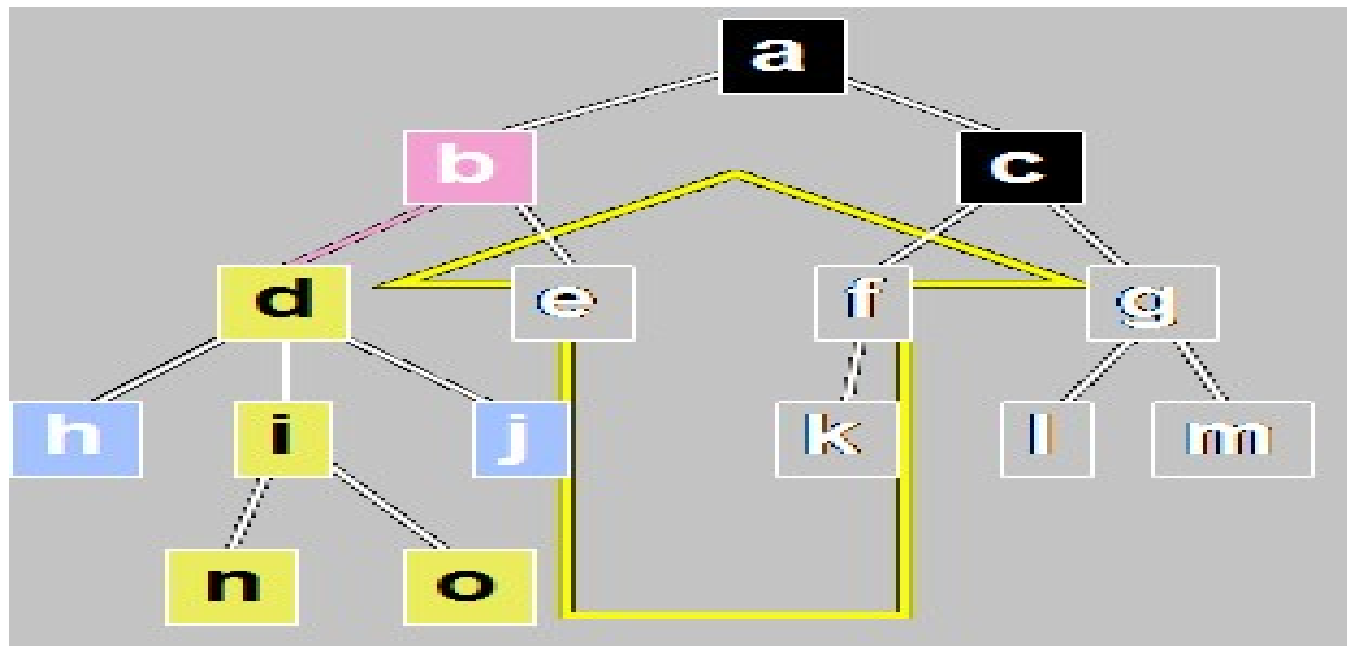
- **Big-Bang** : l'intégration se fait en une seule fois
- **Incrémentale Top-Down** : approche progressive d'intégration où le composant en haut de la hiérarchie est testé en premier lieu. Les composants testés seront utilisés pour tester les composants de niveau inférieur. On réitère jusqu'à tester le composant de plus bas niveau.

- a puis a+b, a+b+c, a+b+c+d, etc...
- Besoin de faire appel aux composants de bas niveaux
- Nécessité de simuler les composants manquants



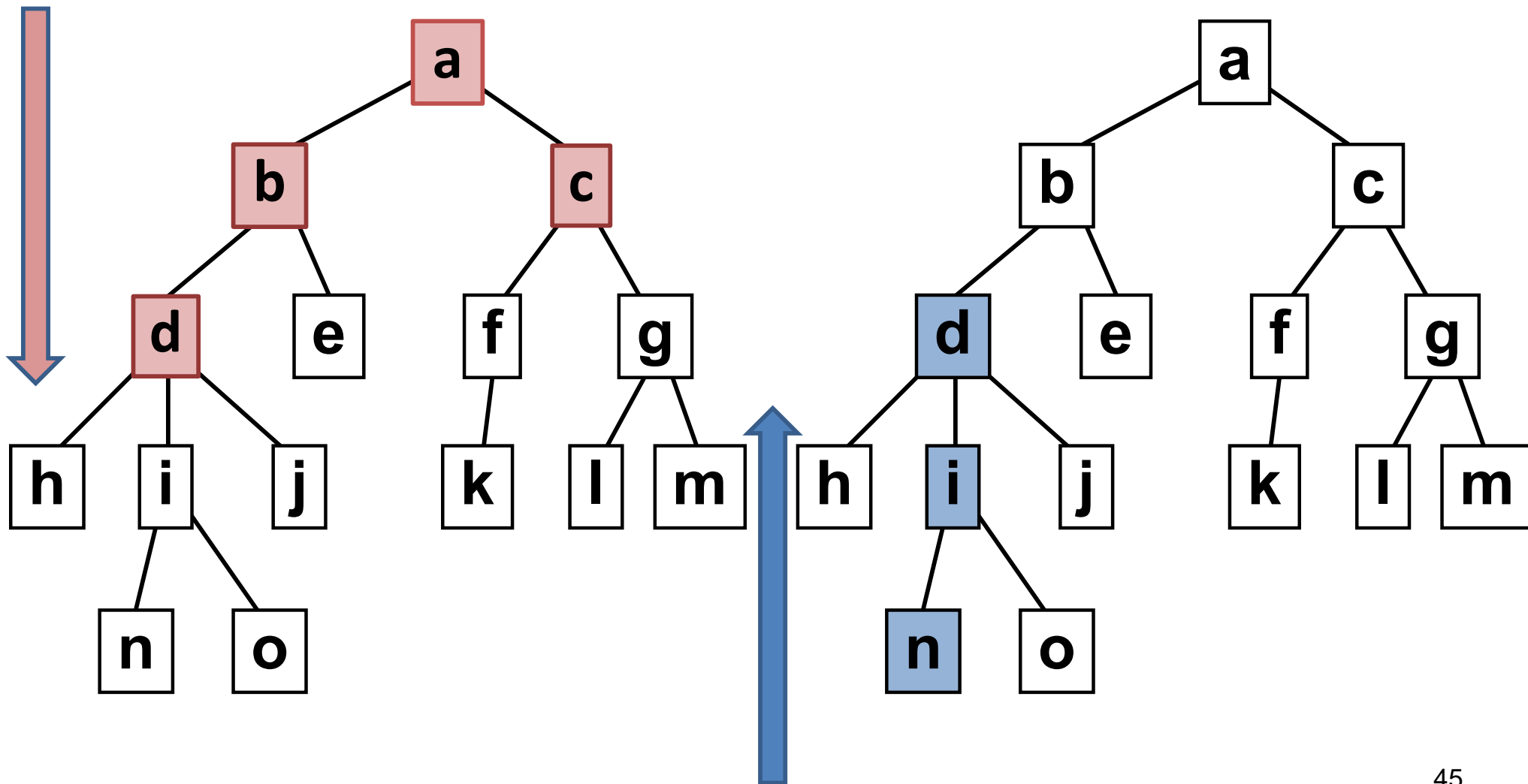
2-Tests Pendant le cycle de vie ...Niveaux de tests

- **Incrémentale Bottom-Up** : approche progressive d'intégration ou les composants plus bas niveau sont testés en premier
 - Composant n, puis n+i, puis n+i+o, n+i+o+d, etc...
 - Besoin de drivers de configuration
 - Besoin aussi de stub (mockup) pour certains composants



2-Tests Pendant le cycle de vie ...Niveaux de tests

Tests d'intégration : Top-down vs Bottom-up.



➤ Tests système

- Les tests systèmes traitent le comportement d'un système/produit complet .
- L'environnement de test devrait correspondre le maximum possible à la cible finale .
- Les tests système devraient examiner à la fois les exigences fonctionnelles et non-fonctionnelles du système .

➤ Tests d'acceptation

- Les objectifs des tests d'acceptation sont de prendre confiance dans le système ou dans des caractéristiques non-fonctionnelles du système.
- Souvent de la responsabilité des clients ou utilisateurs d'un système
- Les formes des tests d'acceptation incluent:
 - Tests d'acceptation utilisateurs
 - Tests (d'acceptation) opérationnelle
 - Tests d'acceptation contractuelle et réglementaire
 - Tests alpha et beta
 - Les Alpha tests sont exécutés sur le site de l'organisation effectuant le développement mais pas par les équipes de développement.
 - Les Béta tests ou tests sur le terrain sont exécutés par des personnes sur leurs sites propres.

- **Tests fonctionnels** : Tests basés sur une analyse des spécifications d'une fonctionnalité d'un composant ou système

Que fait le système ?

- **Tests non fonctionnels** : Les tests non-fonctionnels incluent, mais pas uniquement, les tests de performances, tests de charge, tests de stress, tests d'utilisabilité, tests de maintenabilité, tests de fiabilité et les tests de portabilité.

Comment le système fonctionne ?

➤ Tests de maintenance

Les tests de maintenance sont effectués sur un système opérationnel existant et sont déclenchés par des modifications, migrations ou suppression de logiciels ou de systèmes. Ils incluent:

- Les tests des modifications
- Les tests de non régression

Tests liés au changement

- **Tests de confirmation (ou Retest)**

Tests qui exécutent des cas de test qui ont été en échec la dernière fois qu'ils furent exécutés, de façon à vérifier le succès des actions de correction.

- **Tests de non régression**

Tests d'un programme préalablement testé, après une modification, pour s'assurer que des défauts n'ont pas été introduits ou découverts dans des parties non modifiées du logiciel.



Test & validation (P2)

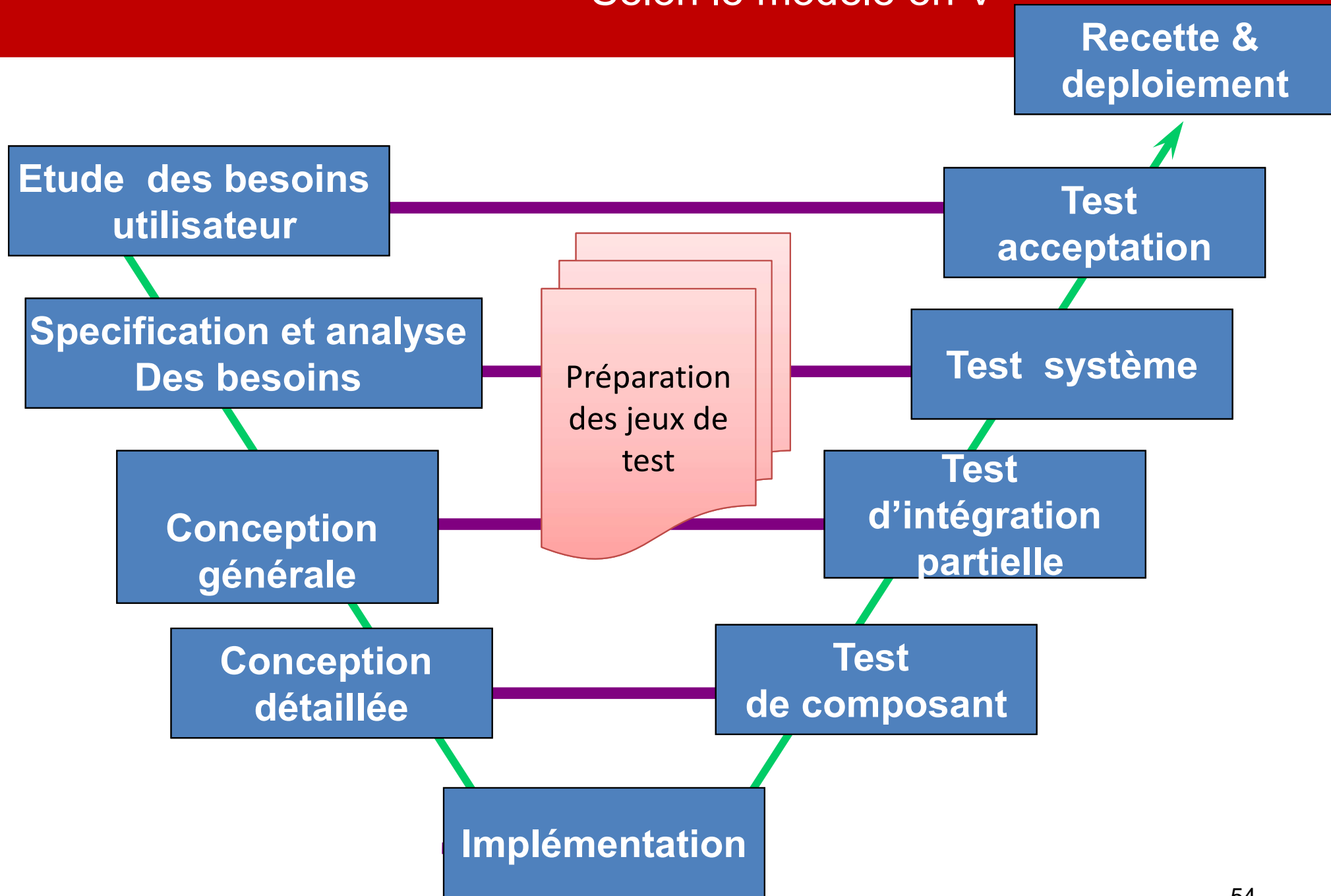
Conformément à la certification ISTQB
(International Software Testing Qualification Board)

2-Tests Pendant le cycle de vie

2-Tests Pendant le cycle de vie

- Les tests n'existent pas de façon isolée : les activités de test sont liées aux activités de développement logiciel.
- Les différents modèles de cycle de développement nécessitent des approches de tests différentes.
- 2 modèles:
 - modèle en V
 - Modèle itératif

2-Tests Pendant le cycle de vie Selon le modèle en V



2-Tests Pendant le cycle de vie

- **Le flux descendant:** vise à détailler le produit jusqu'à sa réalisation. Il comprend l'expression des besoins, l'analyse, la conception, puis la mise en œuvre.
 - Pour un logiciel, la mise en œuvre correspond essentiellement à la programmation.
- **le flux ascendant :** vise à valider le produit jusqu'à sa « recette », c'est-à-dire son acceptation par le client. Il comprend principalement une série de tests jusqu'à pouvoir valider que le produit répond au besoin et aux exigences.

2-Tests Pendant le cycle de vie

la dimension système :

- permet de voir le produit du projet (logiciel) comme un « système » fait de plusieurs éléments qui sont des modules ou composants.
- Ceci requiert dans le flux descendant de distinguer une conception générale du système dans son ensemble, et une conception détaillée de chaque composant.
- Dans le flux ascendant, il convient d'effectuer des tests unitaires de chaque composant, d'intégrer le système (c'est-à-dire d'assembler ses composants), puis de faire un test d'integration.

2-Tests Pendant le cycle de vie ...Modèle en V

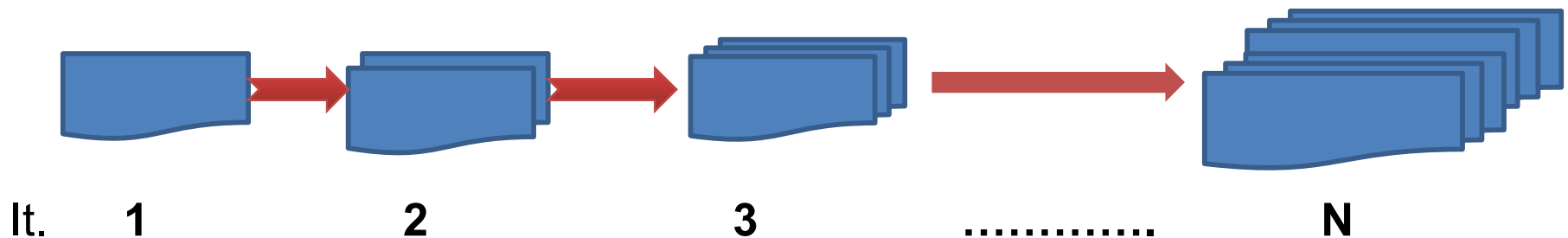
- **Les 4 niveaux de tests :**
 - Tests de composants (unitaires)
 - Tests d'intégration (de plusieurs composants dans un module ou un processus d'un module)
 - Tests système
 - Tests d'acceptation.
- On peut avoir des niveaux de développement et de tests différents, en moins ou en plus en fonction du volume du projet
- les scénarios d'utilisation ,les cas d'emploi, les spécifications d'exigences, les documents de conception et le code sont les bases de tests de un ou plusieurs niveaux de tests.

2-Tests Pendant le cycle de vie

Modèle itératif

➤ Tests d'intégration:

- Développement logiciel = une série de petits fragments à développer.
- Un fragment = exigences, conception, construction et tests
- Un incrément est le système logiciel résultant à chaque itération qui ajoute un fragment développé .il forme un système partiel en croissance.
- Plusieurs niveaux de tests
- Les tests de régression sont de plus en plus importants sur toutes les itérations



2-Tests Pendant le cycle de vie

- A chaque activité de développement, correspond une activité de test.
- Chaque niveau de test a des objectifs de tests spécifiques pour ce niveau.
- L'analyse et la conception des tests pour un niveau de test devraient commencer pendant l'activité correspondante de développement.

➤ Tests de composants

- Recherche des défauts et vérification du bon fonctionnement des modules, objets, classes qui sont testables séparément
- Peuvent inclure des tests fonctionnels, non fonctionnels et structurels
- peut se faire de façon isolée par rapport au reste du système
- Les tests de composants se font généralement avec l'accès au code du logiciel testé et sur un environnement de développement comme un framework de tests unitaire

➤ Tests de composants (suite)

➤ Bases de tests:

- Exigences des composants
- Conception détaillée
- Code

➤ Objets habituels de test:

- Composants
- Programmes
- Conversions de données / utilitaires ou programmes de migration

2-Tests Pendant le cycle de vie

Niveaux de tests

➤ Tests de composants (suite)

- Des bouchons, pilotes et simulateurs peuvent être utilisés.
 - **Bouchon** : une implémentation spéciale ou squelettique d'un composant logiciel, utilisé pour développer ou tester un composant qui l'appelle ou en est dépendant. Cela remplace un composant appelé.
 - **Pilote** : un composant logiciel ou outil de tests qui remplace un composant qui contrôle et/ou appelle un composant ou système.
 - **Simulateur** : un appareil, programme ou système utilisé pendant les tests, qui se comporte ou fonctionne comme un système donné à la réception d'entrées contrôlées .

2-Tests Pendant le cycle de vie ...Niveaux de tests

➤ Tests d'intégration

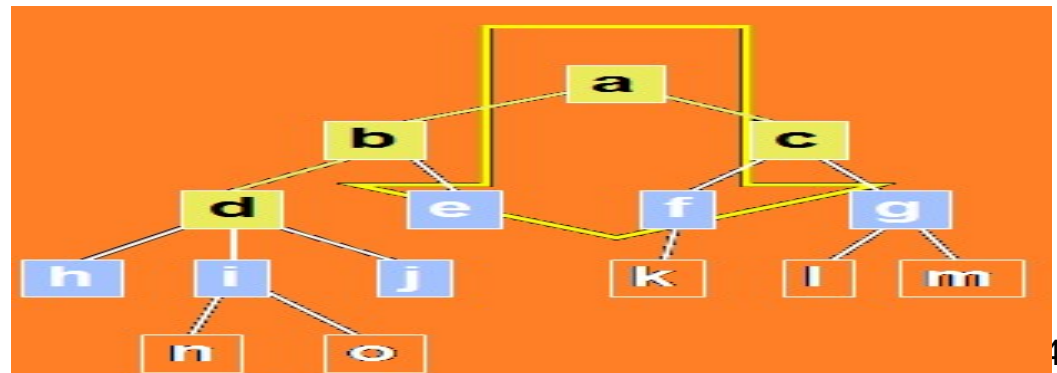
- Tests d'intégration des composants testant les interactions entre les composants logiciels
- Tests d'intégration système testant l'intégration entre les différents systèmes ou entre logiciel et matériel
- Peuvent inclure des tests fonctionnels, non fonctionnels et structurels
- Plus le nombre de composants est élevé, plus il devient difficile d'isoler les défauts

2-Tests Pendant le cycle de vie ...Niveaux de tests

➤ Stratégie d'intégration :

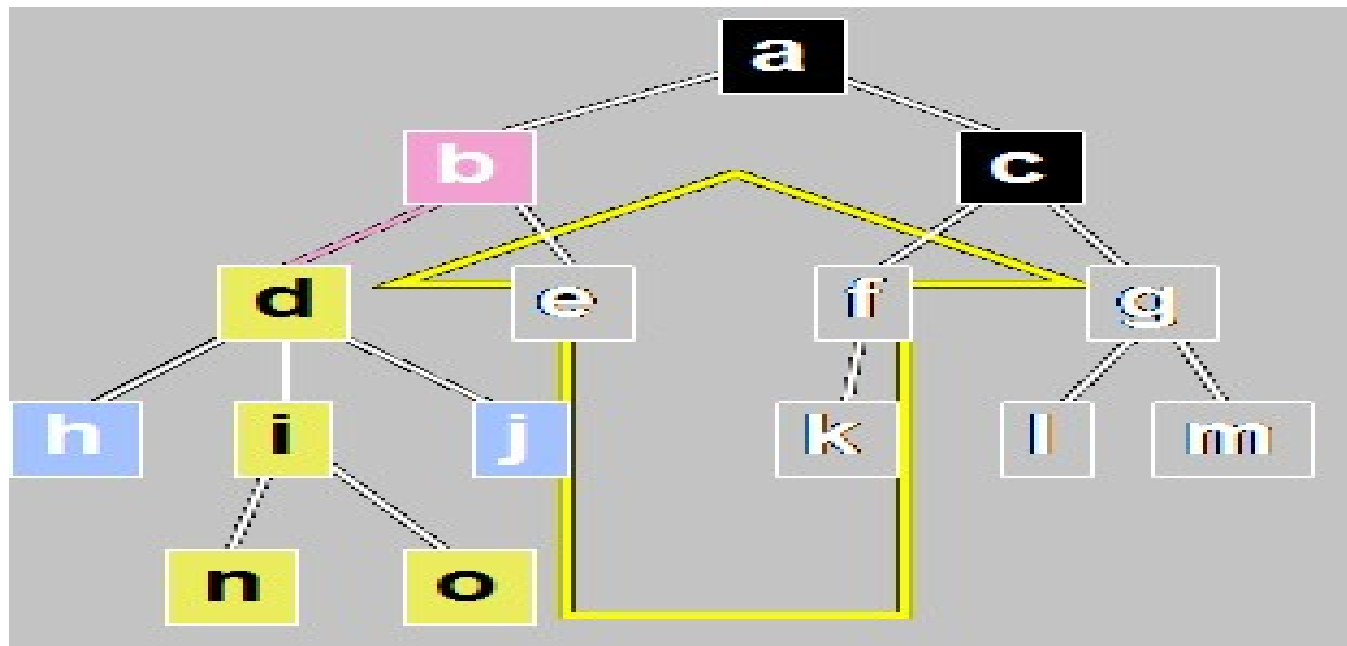
- **Big-Bang** : l'intégration se fait en une seule fois
- **Incrémentale Top-Down** : approche progressive d'intégration où le composant en haut de la hiérarchie est testé en premier lieu. Les composants testés seront utilisés pour tester les composants de niveau inférieur. On réitère jusqu'à tester le composant de plus bas niveau.

- a puis a+b, a+b+c, a+b+c+d, etc...
- Besoin de faire appel aux composants de bas niveaux
- Nécessité de simuler les composants manquants



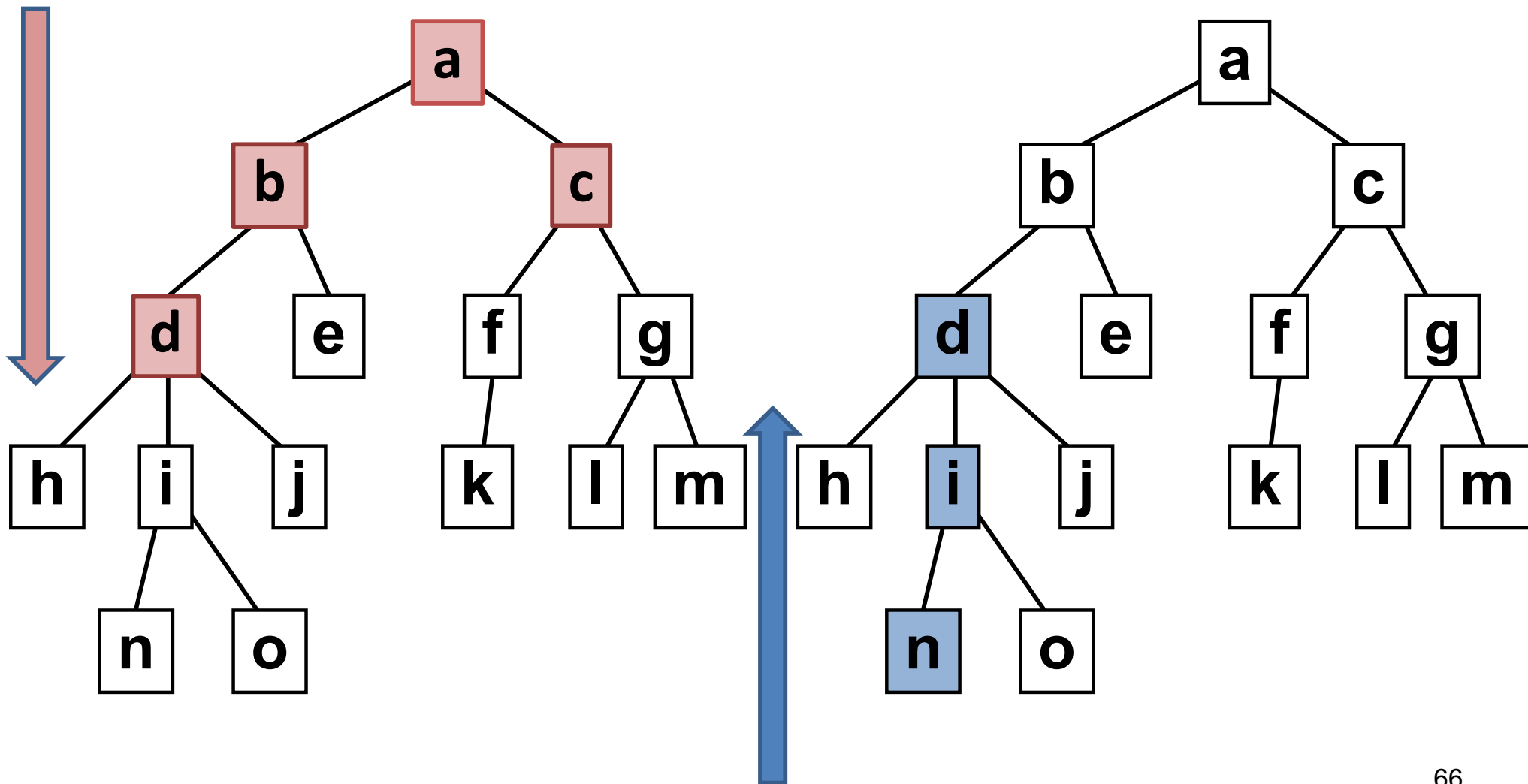
2-Tests Pendant le cycle de vie ...Niveaux de tests

- **Incrémentale Bottom-Up** : approche progressive d'intégration ou les composants plus bas niveau sont testés en premier
 - Composant n, puis n+i, puis n+i+o, n+i+o+d, etc...
 - Besoin de drivers de configuration
 - Besoin aussi de stub (mockup) pour certains composants



2-Tests Pendant le cycle de vie ...Niveaux de tests

Tests d'intégration : Top-down vs Bottom-up.



➤ Tests système

- Les tests systèmes traitent le comportement d'un système/produit complet .
- L'environnement de test devrait correspondre le maximum possible à la cible finale .
- Les tests système devraient examiner à la fois les exigences fonctionnelles et non-fonctionnelles du système .

➤ Tests d'acceptation

- Les objectifs des tests d'acceptation sont de prendre confiance dans le système ou dans des caractéristiques non-fonctionnelles du système.
- Souvent de la responsabilité des clients ou utilisateurs d'un système
- Les formes des tests d'acceptation incluent:
 - Tests d'acceptation utilisateurs
 - Tests (d'acceptation) opérationnelle
 - Tests d'acceptation contractuelle et réglementaire
 - Tests alpha et beta
 - Les Alpha tests sont exécutés sur le site de l'organisation effectuant le développement mais pas par les équipes de développement.
 - Les Béta tests ou tests sur le terrain sont exécutés par des personnes sur leurs sites propres.

- **Tests fonctionnels** : Tests basés sur une analyse des spécifications d'une fonctionnalité d'un composant ou système

Que fait le système ?

- **Tests non fonctionnels** : Les tests non-fonctionnels incluent, mais pas uniquement, les tests de performances, tests de charge, tests de stress, tests d'utilisabilité, tests de maintenabilité, tests de fiabilité et les tests de portabilité.

Comment le système fonctionne ?

➤ Tests de maintenance

Les tests de maintenance sont effectués sur un système opérationnel existant et sont déclenchés par des modifications, migrations ou suppression de logiciels ou de systèmes. Ils incluent:

- Les tests des modifications
- Les tests de non régression

Tests liés au changement

- **Tests de confirmation (ou Retest)**

Tests qui exécutent des cas de test qui ont été en échec la dernière fois qu'ils furent exécutés, de façon à vérifier le succès des actions de correction.

- **Tests de non régression**

Tests d'un programme préalablement testé, après une modification, pour s'assurer que des défauts n'ont pas été introduits ou découverts dans des parties non modifiées du logiciel.



Test & validation (P3)

Conformément à la certification ISTQB
(International Software Testing Qualification Board)

3-Techniques de conception des tests

Techniques de conception des tests

- Tests statiques et tests dynamiques
- Différentes techniques de conception des tests
- Techniques basées sur les spécifications
- Techniques basées sur la structure
- Techniques basées sur l'expérience
- Sélection des techniques de test
- Tests boîte grise

Les tests statiques & tests dynamiques

➤ Tests statiques

Les produits d'activités qui peuvent être examinés par des tests statiques sont:

- Les spécifications, y compris les exigences métier, les exigences fonctionnelles et les exigences de sécurité
- Les épics, User Stories, et critères d'acceptation.
- Les spécifications d'architecture et de conception.
- Le code
- Le testware, y compris les plans de test, les cas de test, les procédures de test et les scripts de test automatisés
- Les contrats, les plans de projet, les calendriers et les budgets
- Les modèles, tels que les diagrammes d'activité, qui peuvent être utilisés pour les tests basés sur des modèles

Les tests statiques & tests dynamiques

- Peuvent avoir les mêmes objectifs ,tels que fournir une évaluation de la qualité des produits d'activités et identifier les défauts le plus tôt possible. Les tests statiques et dynamiques se complètent mutuellement en trouvant différents types de défauts.
- Les tests statiques détectent directement les défauts dans les produits d'activités plutôt que d'identifier les défaillances causées par des défauts lorsque le logiciel est exécuté.
- Les tests statiques peuvent permettre de trouver le défaut avec beaucoup moins d'efforts.
- Les tests statiques peuvent être utilisés pour améliorer la cohérence et la qualité interne des produits d'activités, tandis que les tests dynamiques se concentrent généralement sur les comportements visibles de l'extérieur.

Les tests statiques & tests dynamiques

Par rapport aux tests dynamiques, les défauts qui sont habituellement plus faciles et moins coûteux à trouver et à corriger grâce aux tests statiques sont les suivants :

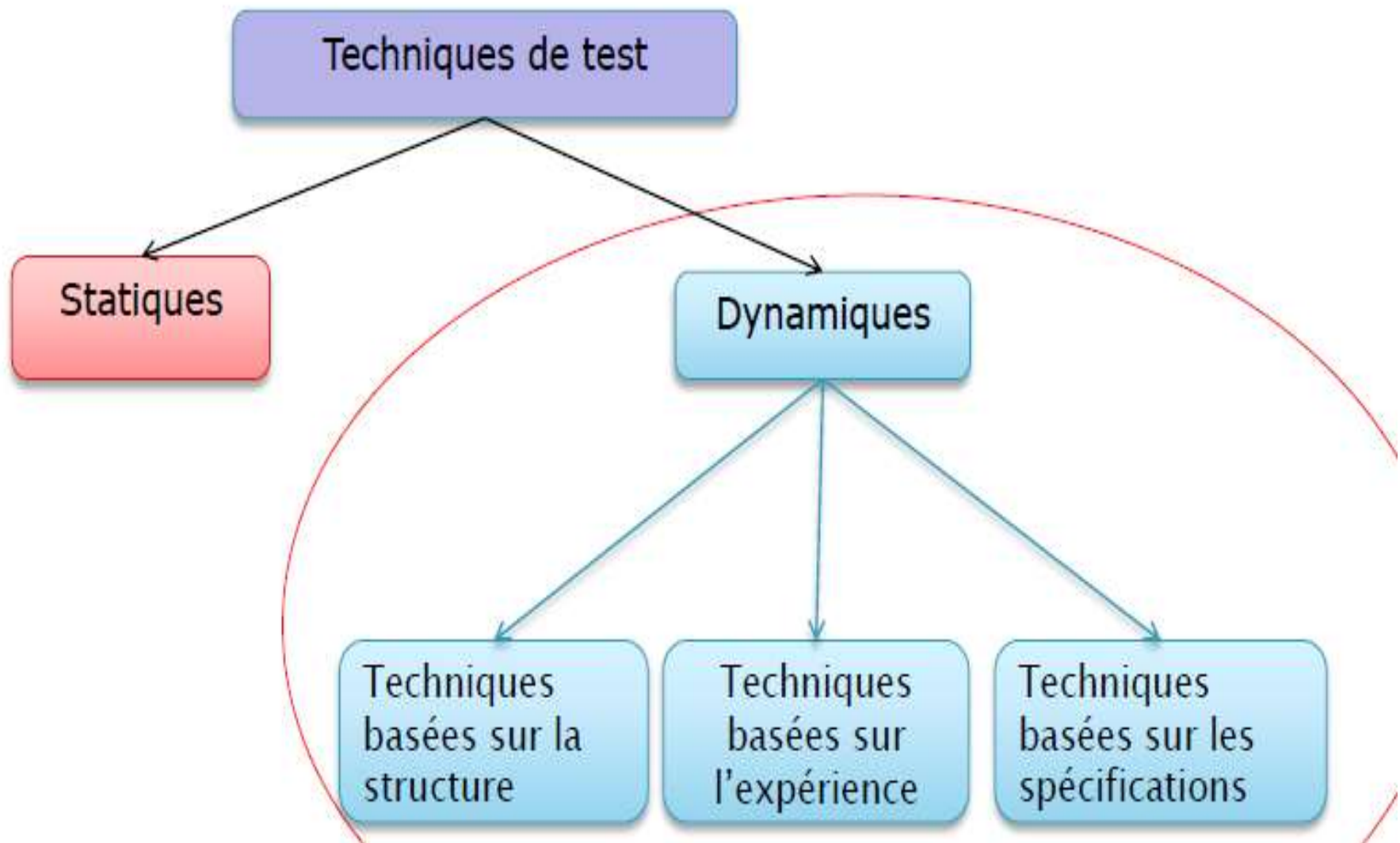
- Défauts dans les exigences (p. ex. incohérences, ambiguïtés, contradictions, omissions, inexactitudes et redondances)
- Défauts dans la conception (p. ex. algorithmes ou structures de base de données inefficaces, taux de dépendance élevé, faible cohésion)
- Défauts dans le code (p. ex. variables avec des valeurs non définies, variables déclarées mais jamais utilisées, code inatteignable (code mort), code dupliqué)
- Ecart par rapport aux normes (p. ex. non-respect des règles de codage)

Les tests statiques & tests dynamiques

- Spécifications d'interface incorrectes (p. ex. unités de mesure utilisées par le système appelant différentes de celles utilisées par le système appelé)
- Vulnérabilités de sécurité (p. ex. sensibilité aux débordements de la mémoire tampon)
- Lacunes ou inexactitudes dans la traçabilité ou la couverture des bases de test (p. ex., des tests manquants pour un critère d'acceptation)
- De plus, la plupart des défauts de maintenabilité ne peuvent être trouvés que par des tests statiques (par exemple, une modularité inadéquate, une mauvaise réutilisation des composants, un code difficile à analyser et à modifier sans introduire de nouveaux défauts).

Techniques de conception des tests

Catégorie des techniques de conception des tests



1. Techniques basées sur les spécifications ou techniques « boîte noire »

- procédures documentées pour élaborer et sélectionner des cas de tests basés sur une analyse des spécifications, soit fonctionnelles soit non-fonctionnelles, d'un composant ou système sans faire référence à ses structures internes.
- Les outils :
 - Partitions d'équivalence ou classes d'équivalence
 - Analyse des valeurs limites
 - Tests par tables de décisions
 - Test de transition d'états
 - Tests de cas d'utilisation

- Partitions d'équivalence (ou classes d'équivalence)
 - les entrées d'un logiciel ou système sont divisées en groupes qui doivent montrer un comportement similaire, de ce fait elles auront un traitement identique .
 - Les partitions d'équivalence peuvent être trouvées pour des données valides, c.a.d des données qui devraient être acceptées et des données invalides c.à.d des valeurs qui devraient être rejetées.

➤ Partitions d'équivalence(Exemple)

Un compte d'épargne dans une banque a un taux d'intérêt différent selon le solde du compte. Afin de tester le logiciel qui calcule les intérêts dus, nous pouvons identifier les plages de valeurs d'équilibre que gagnent les différents taux d'intérêt. Par exemple, le taux d'intérêt de 3% est accordé si le solde du compte est de l'ordre de 0 \$ à 100 \$, le taux d'intérêt de 5% est accordé si le solde du compte est de l'ordre de plus de 100 \$ à moins de 1000 \$, le taux d'intérêt est donné à 7% si le solde du compte est de 1000 \$ et plus, ceci nous permet d'identifier trois partitions d'équivalence valides et une partition non valide, comme indiqué ci-dessous.

< 0.00 \$	Entre 0 et 100 \$	Entre 100.01 et 999.99 \$	A partir de 1000 \$
Valeur invalide	Valide pour 3%	Valide pour 5%	Valide pour 7%

Les valeurs limites sont :-0.01,0,100,100.01,999.99,1000

➤ Analyse des valeurs limites

- Une valeur d'entrée ou de sortie qui est au bord d'une partition, ou à la distance minimale d'un incrément de chaque côté de cette limite, par exemple le minimum ou le maximum d'une plage de valeurs.
- les cas de tests sont conçus sur la base des valeurs limites.
- Exemple précédent ?
- Pour un cas de test on doit choisir les valeurs limites des classes valides et invalides

➤ Analyse des valeurs limites (Exemple)

Dans un système conçu pour fonctionner sur l'impôt à payer:

- un employé qui touche un salaire annuel de 4000 et moins ne paie pas d'impôt
- un employé qui touche un salaire pouvant atteindre 5500 doit payer les impôts à un taux de 10%
- un employé dont le salaire peut atteindre 33500 doit payer les impôts à un taux de 22%
- tout employé dont le salaire dépasse les 33500 doit payer les impôts à un taux de 40 %

Q1 : déterminer les partitions d'équivalence et les valeurs limites

Q2 : proposer une suite de valeurs qui couvre 2 partitions d'équivalence ainsi que les valeurs limites correspondantes

1. Partitions d'équivalence :

1. Groupe 1 : Employés gagnant un salaire annuel de 4000 et moins (exemptés d'impôt)
2. Groupe 2 : Employés gagnant un salaire annuel entre 4001 et 5500 (taux d'impôt de 10%)
3. Groupe 3 : Employés gagnant un salaire annuel entre 5501 et 33500 (taux d'impôt de 22%)
4. Groupe 4 : Employés gagnant un salaire annuel supérieur à 33500 (taux d'impôt de 40%)

2. Valeurs limites :

1. Limite inférieure : 4000 (salaire minimum exempté d'impôt)
2. Limite supérieure du Groupe 2 : 5500 (salaire maximum du taux d'impôt de 10%)
3. Limite supérieure du Groupe 3 : 33500 (salaire maximum du taux d'impôt de 22%)
4. Aucune limite supérieure pour le Groupe 4 (salaire supérieur à 33500)

Techniques de conception des tests

...Catégorie des techniques de conception des tests

- **Q2 : Suite de valeurs couvrant les partitions d'équivalence et les valeurs limites**
- Voici une suite de valeurs qui couvre deux partitions d'équivalence ainsi que les valeurs limites correspondantes :
- Valeurs pour le Groupe 1 (exempté d'impôt) :
 - 0 (limite inférieure)
 - 4000 (limite supérieure)
- Valeurs pour le Groupe 2 (taux d'impôt de 10%) :
 - 4001 (limite inférieure)
 - 5500 (limite supérieure)
- Valeurs pour le Groupe 3 (taux d'impôt de 22%) :
 - 5501 (limite inférieure)
 - 33500 (limite supérieure)
- Valeurs pour le Groupe 4 (taux d'impôt de 40%) :
 - 33501 (limite inférieure)
 - Une valeur supérieure à 33500 pour illustrer que tout salaire dépassant cette limite entre dans ce groupe, par exemple, 40000.
- Cette suite de valeurs couvre toutes les partitions d'équivalence ainsi que les valeurs limites correspondantes, garantissant une couverture adéquate des cas de test pour le système de calcul d'impôt

Tests par table de décision

- Une table de décision montre la combinaison des entrées et/ou stimuli (causes) et de leurs sorties et/ou actions (effets) associées, qui peut être utilisée pour concevoir des cas de tests.
- Une technique de conception des tests boîte noire dans laquelle les cas de tests sont conçus pour exécuter les combinaisons d'entrées et/ou de stimuli (causes) présentes dans une table de décision .

Tests par table de décision (Exemple)

```
if ((u==0) || (x>5)) &&((y<6) || (z==0))
{
    /*instruction*/
}
Else
{
    /*instruction*/
}
```

Une couverture de test intégral consiste à construire la table de vérité suivante et vérifier chaque cas :

Test N°	A(u==0)	B(x>5)	C(y<6)	D(z==0)	Décision	
1	F	F	F	F		
2	F	F	F	V		
3	F	V	F	F		
4	F	V	F	V		
5						
6						
7						
8						
9						

Tests de transition d'états

- Une transition entre deux états d'un composant ou système.
- Une technique de conception de tests boîte noire dans laquelle les cas de tests sont conçus pour exécuter les transitions d'états valides et invalides.

Tests de cas d'utilisation :

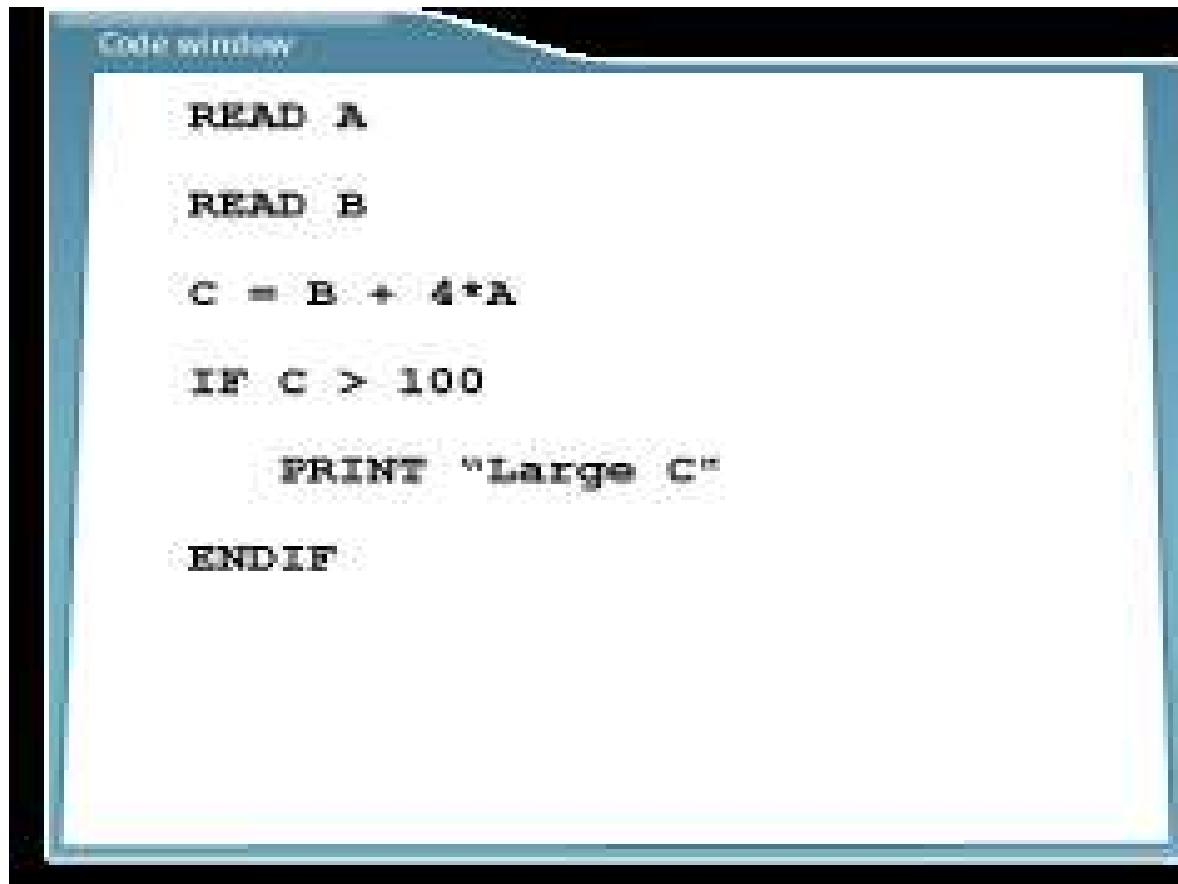
- Description, acteurs, pré-conditions, scénario nominal, scénario alternatif, extension, post-conditions
- Une technique de conception de tests boîte noire dans laquelle les cas de tests sont conçus pour exécuter des scénarios de cas d'utilisation.

2. Techniques basées sur la structure ou technique de conception « boîte blanche »

- Test des instructions et couverture
- Test des décisions et couverture
- **Instructions** : une entité dans un langage de programmation, qui est typiquement la plus petite unité indivisible d'exécution.
- **Couverture des instructions** : le nombre d'instructions exécutables couvertes par des cas de test (conçus ou exécutés) divisé par le nombre de toutes les instructions exécutables dans le code testé.

Techniques de conception des tests

...Catégorie des techniques de conception des tests



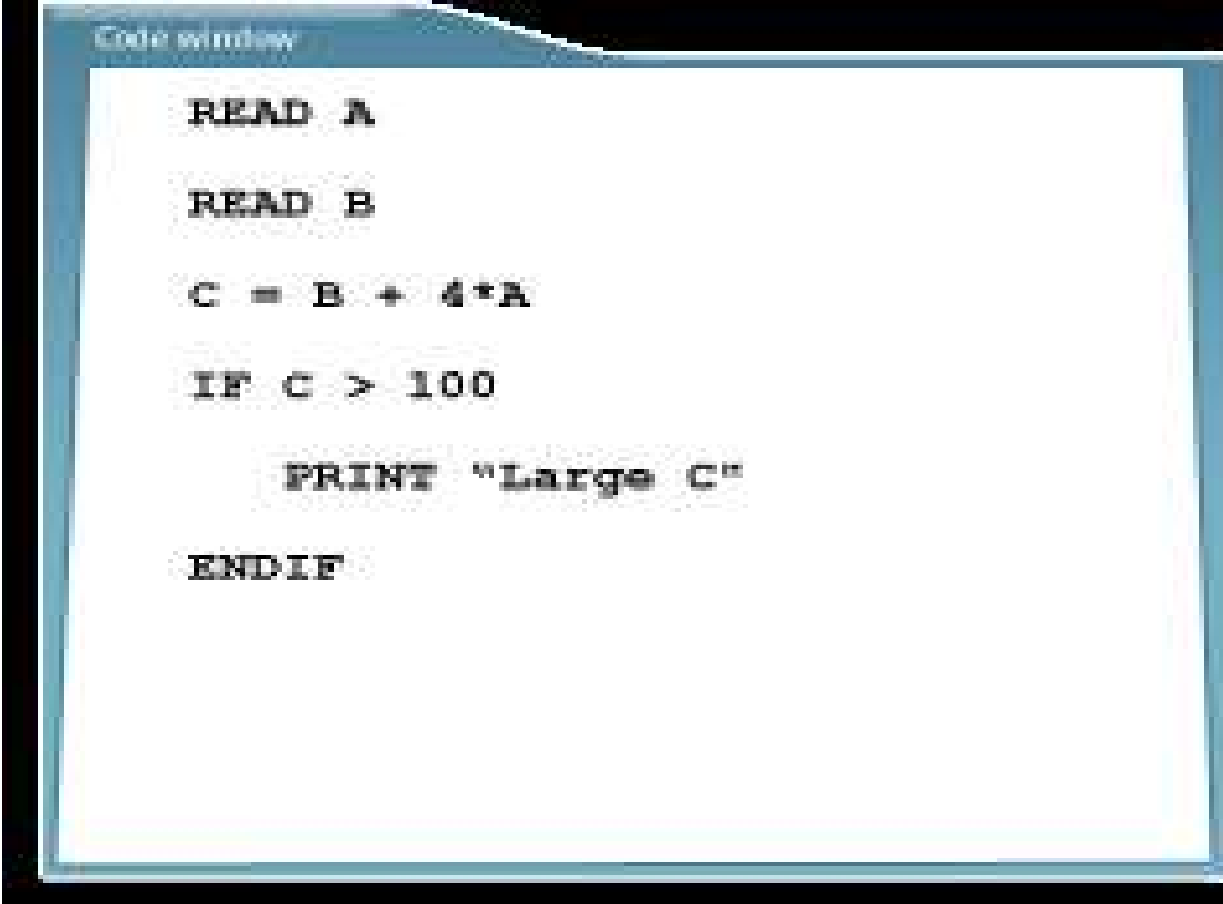
```
Code window  
  
READ A  
  
READ B  
  
C = B + 4*A  
  
IF C > 100  
    PRINT "Large C"  
  
ENDIF
```

Cas de test 1: A=50, B=10

Couverture des instructions= 100%

Techniques de conception des tests

...Catégorie des techniques de conception des tests



```
Code window  
  
READ A  
  
READ B  
  
C = B + 4*A  
  
IF C > 100  
    PRINT "Large C"  
ENDIF
```

Cas de test 1: A=10, B=50
Couverture des branches= 50%

- **Décision** : Un point dans un programme où le flot de contrôle a deux ou plus chemins possibles. Un nœud avec deux ou plus liens vers des branches séparées.
- **Couverture des décisions** : le nombre de tous les résultats des décisions couvertes par des cas de test (conçus ou exécutés) divisé par le nombre de tous les résultats possibles des décisions dans le code sous test.

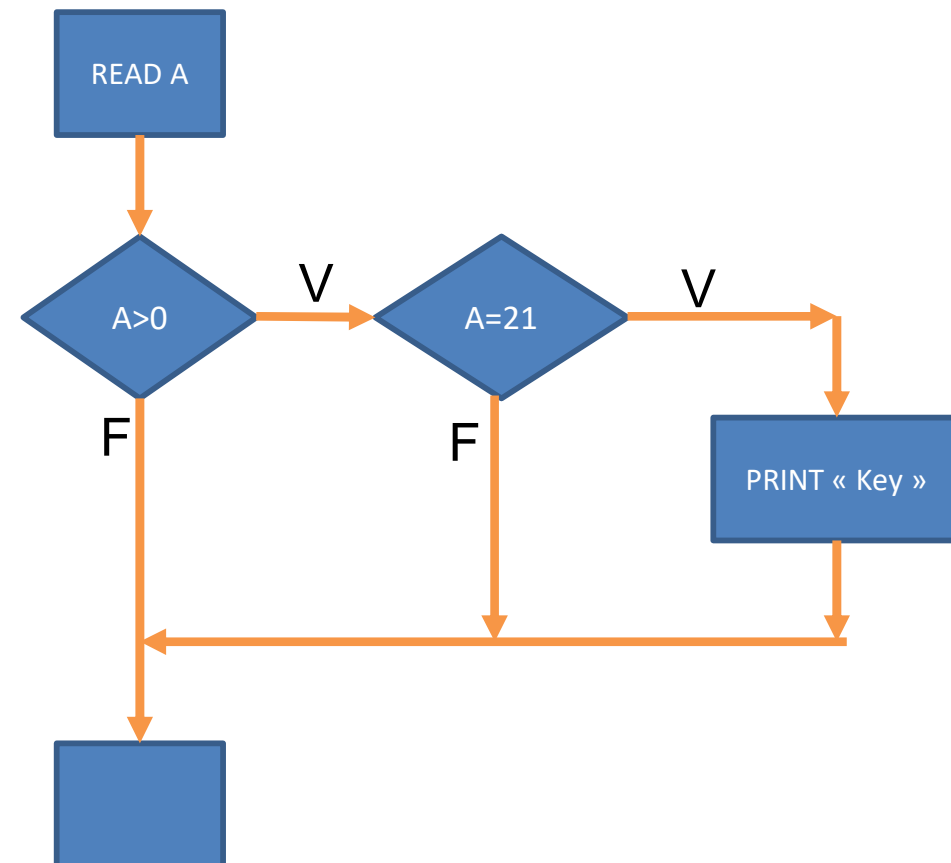
- **Couverture des branches:** le pourcentage des branches qui ont été exécutés dans une suite de tests. 100% de couverture des branches implique 100% de couverture des décisions et 100% de couverture des instructions.
- **Test des branches :** une technique de conception des tests boîte blanche dans laquelle les cas de tests sont conçus pour exécuter les branches.

➤ Couverture des instructions et couverture des branches (exemple)

```
READ A
IF A > 0 THEN
    IF A=21 THEN
        PRINT "Key"
    END IF
END IF
```

Nbre minimum de cas
de test des instructions = 1 (A=21)

Nbre minimum de cas de test
Des branches : 3 (ex : A=-5 ,A=5,A=21)



Conclusion : La couverture totale des branches implique la couverture totale des instructions

➤ Exemple couvertures des instructions et des branches :

.....

Si $x \geq 3$ alors

Afficher message x

Fin si

Si $x=2$ alors

afficher message y

Sinon

afficher message z

Fin si

''''

- Dessiner le graphe d'exécution
- Indiquer le nombre minimum de cas de test pour une couverture totale des instructions
- Indiquer le nombre minimum de cas de test pour une couverture totale des branches

Exemple couverture des conditions /décisions

On considère le code suivant :

```
If (A || B) && C)
{
    /*instruction*/
}
Else
{
    /*instruction*/
}
```

A,B et C représentent des expressions booliennes

- Pour que les critères de couverture des conditions soient vérifiés sur cet exemple il faut que A,B et C aient été vérifiées au moins une fois True et une fois False. Donc la couverture des conditions est :
 - A=True /B=True/C=True
 - A=False/B=False/C=false
- Il faut aussi que la décision (AouB)et C soit évaluée au moins une fois à True et une fois à False :
 - A=True /B=True/C=True => (AouB)et C est True
 - A=False/B=False/C=false => (AouB)et C est False
- Donc la couverture des décisions est aussi réalisée

...Exemple couverture des conditions /décisions

- Les deux tests précédents ne permettent pas une couverture des conditions / décisions modifiées qui exigent les combinaisons des valeurs booléennes de chaque variable pour démontrer concrètement l'influence de ces changements sur le résultat final.
- En pratique pour n opérandes booléennes on doit avoir au minimum n+1 tests pour pouvoir assurer la couverture des conditions/décisions modifiées
 - $A=False / B=False / C=True \Rightarrow (A \vee B) \text{ et } C \text{ est False}$
 - $A=False / B=True / C=True \Rightarrow (A \vee B) \text{ et } C \text{ est True}$
 - $A=False / B=True / C=False \Rightarrow (A \vee B) \text{ et } C \text{ est False}$
 - $A=True / B=False / C=True \Rightarrow (A \vee B) \text{ et } C \text{ est True}$

...Exemple couverture des conditions /décisions

- Constats sur les cas de test :
 - Le 1^{er} et 4^{ème} scénario :seul A change de valeur et a fait changer la valeur de la décision
 - Du 1^{er} au 2^{ème} :seul B change et fait changer la valeur de la décision
 - Du 2^{ème} au 3^{ème} :seul C change et fait changer la valeur de la décision
 - De plus les critères de couverture des décisions et conditions sont toujours respectés (chaque variable prend au moins une fois les valeurs True et False.
- La couverture des conditions /décisions est alors assurée

3. Techniques basées sur l'expérience :

- Estimation d'erreur :
 - Liée aux compétences du testeur
 - Pas de règles, chercher des situations qui peuvent être omises par le développeur
- Tests exploratoires :
 - Conception et exécution des tests en même temps
 - Périmètre de test non défini d'avance

- **Tests exploratoires :**

- comprennent la conception et l'exécution des tests, l'écriture des résultats de tests et l'apprentissage (de l'application), basés sur une charte de tests contenant les objectifs de tests, et effectués dans des périodes de temps délimitées

- **Avantages:**

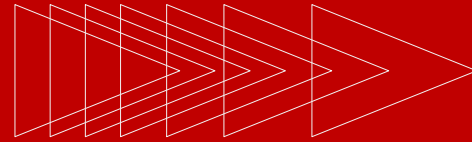
- Permettent d'augmenter ou compléter d'autres méthodes de tests plus formelles lorsque les spécifications sont rares ou non adéquates, et que le test est soumis à de sévères contraintes de temps
- Conception et exécution des tests en même temps

4. Facteurs de sélection des techniques de tests:

- Facteurs internes
 - Modèles utilisés (cas d'utilisation, états de transitions,...)
 - Documentation
 - Expérience/Compétence du testeur
- Facteurs externes
 - Risques
 - Type de système (système financier, ultra-transactionnel ,etc...)
 - Temps/Budget

5. Techniques basées sur les fonctionnalités et les processus « boîte grise »

- Eprouvent à la fois les fonctionnalités et le fonctionnement du système
- Un testeur va donner des données en entrée, vérifier les résultats obtenus par rapport à ceux attendus puis vérifier par le processus comment les résultats sont obtenus.
- Dans ce cas le testeur est sensé connaître le rôle du système et de ses fonctionnalités ainsi que ses mécanismes internes au niveau des données et des processus de traitement



www.esprit.ens.tn

Test & validation

Conformément à la certification ISTQB
(International Software Testing Qualification Board)
Partie N°4

5-Gestion des tests

5-Gestion des tests : Objectifs

- Organiser les tests
- Estimer et planifier les tests
- Suivre et contrôler le déroulement des test
- Assister les tests par la gestion de configuration
- Tester en tenant compte des risques
- Gérer les incidents

5-Gestion des tests : Organisation des tests et indépendance

- la découverte d'anomalies par le test et les revues peut être améliorée par l'emploi de testeurs indépendants.
- Les options d'indépendance:
 - développeurs testant leur propre code
 - Testeurs indépendants incorporés à l'équipe de développement.
 - Équipe ou groupe de test indépendant au sein de l'organisation
 - Testeurs indépendants de l'organisation, de la communauté des utilisateurs et de l'Informatique.
 - Spécialistes de test indépendants pour des objectifs spécifiques de test (sécurité , utilisabilité ou normalisé selon une certification)

5-Gestion des tests : Organisation des tests et indépendance

- **Avantages :**
 - Détection des défauts différents , impartiaux et d'une autre nature.
 - Pouvoir vérifier les hypothèses faites pendant la spécification et l'implémentation du système.
- **Inconvénients :**
 - Déconnexion vis-à-vis de l'équipe de développement
 - perte du sens de la responsabilité par les développeurs pour la qualité.
 - Les testeurs indépendants peuvent constituer un goulet d'étranglement aux développeurs comme dernier point de vérification et être accusés des retards.

- La planification du test est une activité continue et est effectuée tout au long des processus et activités du cycle de développement.
- Le retour issu des activités de test est employé pour constater l'évolution des risques et modifier alors la planification.

5-Gestion des tests : estimation et planification des tests

- Les activités de la planification des tests pour un système ou pour une partie de celui-ci peuvent être:
 - Définir le périmètre du test, les risques et identifier les objectifs du test
 - Définir l'approche générale du test (stratégie de test), y compris la définition des niveaux de test ainsi que celle des critères d'entrée et de sortie.
 - Intégrer et coordonner des activités de test dans les activités du cycle de développement.
 - Prendre des décisions quant à ce qui doit être testé, quels rôles vont exercer quelles activités, quand et comment ces activités doivent être exercées, comment évaluer les résultats des tests et quand arrêter les tests (critères de sortie)
 - Planifier les activités d'analyse , de conception ,d'execution et d'évaluation des tests.
 - Assigner les ressources aux différentes taches de tests
 - Définir le volume et la structure du modèle pour la documentation de tests.
 - Déterminer le niveau de détail pour les procédures de test

5-Gestion des tests : les critères d'entrée

- définissent quand démarrent les tests et couvrent :
 - La disponibilité et préparation de l'environnement de test
 - La préparation des outils de tests dans l'environnement de test
 - La disponibilité de code testable
 - La disponibilité des jeux de données

5-Gestion des tests : les critères de sortie

- définissent quand arrêter le test, à la fin d'un niveau de test ou lorsqu'une série de tests a atteint un objectif donné. Ils couvrent :
 - Des mesures d'exhaustivité, comme la couverture de code, de fonctionnalités ou de risques.
 - L'estimation de la densité des anomalies ou des mesures de fiabilité.
 - Le coût.
 - Les risques résiduels, comme les anomalies non corrigées ou le manque de couverture du test dans certaines parties.
 - Un calendrier, par exemple, basé sur la date de mise sur le marché.

5-Gestion des tests : Estimation des tests

- 2 approches :
 - Estimation de l'effort de test basée sur des mesures d'autres projets ou sur des valeurs typiques
 - L'approche experte
- L'effort de tests peut dépendre des facteurs suivants :
 - Les caractéristiques du produit :(exigences, taille, complexité ,sécurité, etc..)
 - Les caractéristiques du processus de développement (outils utilisés ,savoir faire des personnes impliquées ,contrainte de temps)
 - Les résultats des tests (nbre de défauts et reprise)

5-Gestion des tests : Stratégie de test

- L'approche de test est la mise en œuvre d'une stratégie de test pour un projet spécifique
- Elle est définie et affinée dans les plans et scénarii de test.
- Elle comprend typiquement les décisions basées sur le projet (de test), ses buts ainsi qu'une analyse des risques

5-Gestion des tests : Stratégie de test

- L'approche sélectionnée dépend du contexte et peut prendre en considération les risques, la sécurité et les dangers, les ressources disponibles et leur niveau d'expertise, la technologie et la nature du système.
- Les approches typiques peuvent comprendre:
 - Une approche analytique, comme le test basé sur les risques où le test est focalisé sur les parties à plus haut risque.
 - Les approches basées sur les modèles, comme le test stochastique qui utilise des informations statistiques sur les taux d'erreurs (comme les modèles de croissance de fiabilité) ou sur l'utilisation (comme les profils opérationnels).
 - Les approches méthodiques, comme le test basé sur les erreurs (y compris l'estimation d'erreurs et l'attaque par faute), basées sur des listes de vérification et sur des caractéristiques de la qualité.

5-Gestion des tests : Stratégie de test

- Des approches basées sur la conformité aux processus et normes, tels que ceux spécifiés par des normes industrielles spécifiques ou les diverses méthodes agiles.
- Les approches dynamiques et heuristiques, comme le test exploratoire où le test est plutôt réactif aux événements que planifié, et où l'exécution et l'évaluation sont des tâches parallèles.
- Les approches consultatives, comme celles où la couverture de test est principalement définie par les avis et le guidage d'experts métier ou en technologie étrangers à l'équipe de test.
- Les approches basées régression, comme celles qui comportent le réemploi de matériel de test existant, une automatisation extensive du test de régression fonctionnel et des suites de tests standard.

L'objectif du suivi du test est de fournir un retour et une visibilité sur les activités de test. Les informations à suivre peuvent être recueillies manuellement ou automatiquement et peuvent être utilisées pour évaluer les critères de sortie, comme la couverture. Des mesures peuvent aussi être utilisées pour évaluer l'avancement par rapport au calendrier et au budget planifiés.

5-Gestion des tests : Suivi de l'avancement des tests

Les mesures de test habituelles sont:

- Le pourcentage du travail consacré à la préparation des cas de test (ou pourcentage des cas de test planifiés et préparés).
- Pourcentage du travail consacré à la préparation de l'environnement de test.
- L'exécution de cas de test (par exemple, nombre de cas de test exécutés ou non et nombre de cas de test réussis ou échoués).
- Les informations sur les défauts (par exemple, densité des défauts, défauts trouvés et corrigés, taux des défaillances et résultats du re-test).
- Couverture par le test des exigences, des risques ou du code.
- Confiance subjective des testeurs dans le produit.
- Dates des jalons du test.
- Coût du test, y compris le coût de l'avantage de trouver le prochain défaut comparé à celui de l'exécution du test suivant.

5-Gestion des tests : Suivi de l'avancement des tests

Les mesures de test habituelles sont:

- Le pourcentage du travail consacré à la préparation des cas de test (ou pourcentage des cas de test planifiés et préparés).
- Pourcentage du travail consacré à la préparation de l'environnement de test.
- L'exécution de cas de test (par exemple, nombre de cas de test exécutés ou non et nombre de cas de test réussis ou échoués).
- Les informations sur les défauts (par exemple, densité des défauts, défauts trouvés et corrigés, taux des défaillances et résultats du re-test).
- Couverture par le test des exigences, des risques ou du code.
- Confiance subjective des testeurs dans le produit.
- Dates des jalons du test.
- Coût du test, y compris le coût de l'avantage de trouver le prochain défaut comparé à celui de l'exécution du test suivant.

6-Les outils de support aux tests

6-Outils de support de tests : type d'outils de test

- Outils qui sont directement utilisés dans le test tel que les outils d'exécution de test, outils de génération de données de test et outils de comparaison de résultats.
- Outils qui aide au contrôle le processus de test comme ceux employés pour gérer les tests, des résultats de test, des données, des conditions, des incidents, des défauts, reporter et contrôler l'exécution des tests.
- Outils d'exploration et de contrôle des ressources pour une application
- Outils divers d'aide à l'organisation des tests.

6-Outils de support de tests : buts des outils de test

- Améliorer l'efficacité des activités de test en automatisant des tâches répétitives ou en supportant des activités de test manuelles comme la planification des tests, la conception de tests, le compte-rendu et le contrôle des tests
- Automatiser les activités qui exigent les ressources importantes une fois faites manuellement
- Automatiser les activités qui ne peuvent pas être exécutées manuellement (par exemple, test de performance exhaustif des applications client-serveur)
- Augmenter la fiabilité du test (par exemple, en automatisant la comparaison de beaucoup de données ou en simulant le comportement)

6-Outils de support de tests : Framework de test

- Bibliothèques réutilisables et extensibles de test qui peuvent être employées pour construire des outils de test (appelées aussi harnais de tests)
- Un type de conception d'automatisation des tests (par exemple, pilotés par les données, piloté par mots-clés)
- Processus globaux d'exécution de test

6-Outils de support de tests : Classification

Types d'outils de test

- Ces outils sont classés selon les activités du test qu'ils assistent.
- Certains outils supportent clairement une seule activité; d'autres supportent plus d'une activité .Ils sont classés dans la rubrique relative à l'activité à laquelle ils sont plus étroitement associés.
- Quelques types d'outil de test peuvent être intrusifs, ce qui signifie qu'ils peuvent affecter le résultat obtenu par le test. La conséquence des outils intrusifs s'appelle l'effet de sonde.
- Quelques outils offrent une assistance davantage tournée vers les développeurs (c.à.d.. pendant les tests de composants et d'intégration des composants).

6-Outils de support de tests : Outils d'aide à la gestion des tests

- Les outils de gestion s'appliquent à toutes les activités de tests pendant toute la durée du cycle de vie du logiciel.
 - **Outils de gestion des tests** :fournissent des interfaces pour exécuter des tests, dépister des défauts et gérer les exigences. Ils possèdent des fonctionnalités de gestion des versions et peuvent s'interfacer avec d'autres outils
 - **Outils de gestion des exigences** :enregistrent les énoncés des exigences, enregistrent les attributs des exigences et peuvent identifier des exigences contradictoires.
 - **Outils de gestion d'incidents (outils de suivi de défauts)** :enregistrent et gèrent les rapports d'incidents, c.-à-d. les défauts, les défaillances, les demandes de modification ou les problèmes et anomalies rencontrés .
 - **Outils de gestion de configuration** :nécessaires pour le stockage et la gestion de version du testware et du logiciel associé particulièrement lorsque sont configurés plusieurs environnements matériel/logiciel en termes de versions du système d'exploitation, compilateurs, navigateurs web, etc.

6-Outils de support de tests : Outils d'aide aux tests statiques

- fournissent une manière rentable de trouver plus de défauts à une étape amont dans le processus de développement.
- **Outils de revue** : aident aux revues des processus, des check-lists, des directives de revue et sont utilisés pour enregistrer et communiquer les commentaires des revues, et les rapports sur les défauts et les essais.
- **Outils d'analyse statique (D)** : aident les développeurs et les testeurs à trouver des défauts avant le test dynamique en fournissant une aide pour introduire des normes de codage (codage sécurisé y compris), l'analyse des structures et des dépendances.
- **Outils de modélisation (D)** : employés pour valider les modèles de logiciel (par exemple, modèle de données physiques (MDP) pour une base de données relationnelle), en énumérant des incohérences et en trouvant des défauts.

6-Outils de support de tests : Outils d'aide à la spécification de tests

- **Outils de conception de tests** :utilisés pour générer des entrées de tests ou des tests exécutables et/ou des oracles de tests à partir des exigences, des interfaces utilisateur graphiques, des modèles de conception (état, données ou objet)
- **Outils de préparations de données de tests** :Les outils de préparation des données agissent sur des bases de données, fichiers ou transferts de données afin d'élaborer des données de tests utilisables lors de l'exécution des tests

6-Outils de support de tests : Outils d'aide à l'exécution et à l'enregistrement des tests

- **Outils d'exécution des tests** : permettent à des tests d'être exécutés automatiquement, ou semi-automatiquement, utilisant les entrées enregistrées et les résultats prévus et fournissent habituellement un registre de test pour chaque exécution de test .Ils disposent habituellement d'un langage de script ou une configuration via une interface graphique utilisateur pour le paramétrage des données et toute autre personnalisation dans les tests.
- **Harnais de tests/Outils framework de tests unitaires (D)** :facilite le test des composants ou des parties d'un système en simulant l'environnement dans lequel cet objet de tests s'exécutera par la fourniture d'objets factices comme des bouchons ou des pilotes.
- **Comparateurs de tests** :déterminent des différences entre les fichiers, bases de données ou résultats de tests. Un comparateur de tests peut utiliser un oracle de tests, en particulier s'il est automatisé.

6-Outils de support de tests : Outils d'aide à l'exécution et à l'enregistrement des tests

- **Outils de mesure de couverture (D)** : mesurent le pourcentage de certains types de structures de code qui ont été exercés (par exemple, des déclarations, des branches ou des décisions, et appels de module ou fonction) par un ensemble de tests.
- **Outils de test de sécurité** : utilisés pour évaluer les caractéristiques de sécurité du logiciel. Ceci inclut l'évaluation de la capacité du logiciel à protéger la confidentialité, l'intégrité, l'authentification, l'autorisation, la disponibilité, et la non-répudiation (rejet) des données.

6-Outils de support de tests : Outils de support de performance et de surveillance

- **Outils d'analyse dynamique (D)** :détectent des défauts qui ne se manifestent que lors de l'exécution du logiciel, telles que les dépendances temporelles ou les fuites de mémoire.
- **Outils de test de performance/test de charge/test de stress** :surveillent et rapportent sur la façon dont se comporte un système selon une grande variété de conditions d'usage simulées en termes de nombre d'utilisateurs simultanés, de leur modèle de montée en charge, de fréquence et de pourcentage relatif de transactions.
- **Outils de surveillance** :analysent continuellement, vérifient et rendent compte de l'utilisation de ressources systèmes spécifiques, et donnent des alertes sur de possibles problèmes de service.

6-Outils de support de tests : Outils de support pour des besoins de tests spécifiques

- **Evaluation de la qualité des données** :doivent être utilisés pour revoir et vérifier les règles de conversion et de migration des données (cas de projet de transposition des données) et pour s'assurer que les données traitées sont correctes, complètes et se conforment à une norme prédéfinie spécifique au contexte.

6-Outils de support de tests : Bénéfices potentiels et risques liés aux outils de test

- Chaque type d'outil nécessite des efforts supplémentaires pour atteindre des bénéfices réels et durables.
- Bénéfices potentiels à l'utilisation d'outils :
 - Réduction du travail répétitif (p.ex. exécution de tests de régression, réintroduction des mêmes données de tests, et vérification du respect de standards de codage).
 - Répétabilité et cohérence accrues (p.ex. tests exécutés par un outil suivant un ordre et une fréquence précis et tests déduits des exigences).
 - Evaluation objective (p.ex. mesure statiques, couverture).
 - Facilité d'accès aux informations concernant les tests ou leur exécution (p.ex. statistiques et graphiques sur l'avancement des tests, le taux d'incidents et les performances).

5-Outils de support de tests : Risques liés à l'utilisation d'outils

- Attentes irréalistes placées dans l'outil (dont la facilité d'utilisation et la fonctionnalité).
- Sous-estimation du temps, du coût et de l'effort pour l'introduction initiale d'un outil
- Sous-estimation du temps et de l'effort nécessaires pour obtenir de l'outil des bénéfices
- Sous-estimation de l'effort requis pour maintenir les acquis générés par l'outil.
- Confiance excessive dans l'outil
- Négligence du contrôle de version des éléments de test dans l'outil
- Négligence des problèmes de relation et interopérabilité entre les outils critiques, tels que des outils de gestion d'exigences, des outils de contrôle de version, des outils de gestion d'incidents, des outils de suivi de défauts et des outils de différents éditeurs

6-Outils de support de tests : Risques liés à l'utilisation d'outils

- Risque de faillite de l'éditeur d'outil, de retirer l'outil, ou de vendre l'outil à un autre éditeur
- Faible réactivité du vendeur pour le support, les mises à jour, et corrections des défauts
- Risque de suspension d'un logiciel ou projet open-source ou libre
- Imprévu, comme l'incapacité de support d'une nouvelle plate-forme

6-Outils de support de tests : Considérations spéciales pour quelques types d'outil

- **Outils d'exécution des tests** :exécutent des objets de tests utilisant des scripts de test automatisés. Ce type d'outil nécessite souvent un effort important pour obtenir des bénéfices significatifs.
- **Outils d'analyse statique** :Les outils d'analyse statique appliqués au code source peuvent imposer des standards de codage, mais s'ils sont appliqués à du code existant, peuvent engendrer de nombreux messages
- **Outils de gestion des tests** :Les outils de gestion des tests doivent s'interfacer avec d'autres outils ou des tableurs de façon à produire les informations dans le format le mieux adapté aux besoins présents de l'organisation.