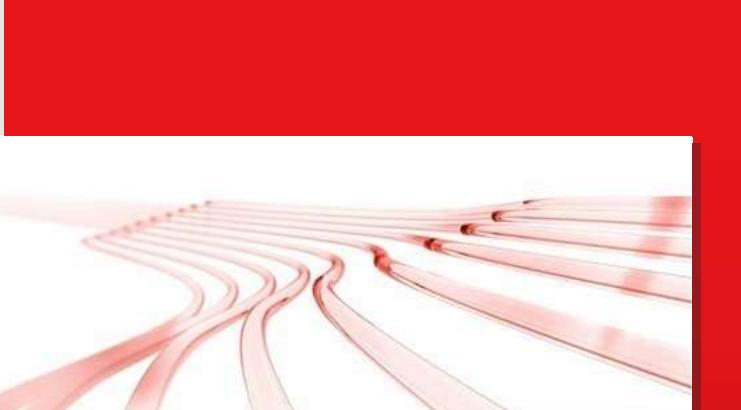


Sopra
group



Master AFD – Le Mans

Module tests

Date de modification : 10 sept. 2019

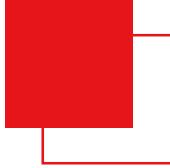
Version : 1.5

TALENTED TOGETHER

Rédacteurs : Fabrice Jumel – Rémy Vasseur

Unissons nos Talents

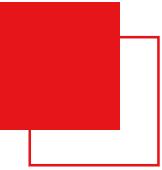
©Sopra Group 2013 / Formation ISTQB



QUI SUIS-JE ?
D'OU VIENS-JE ?
QUE FAIS-JE ?

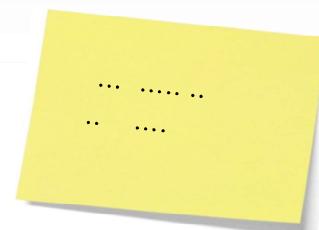


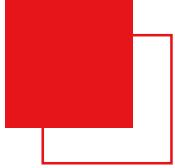
Votre animateur



On aimeraient ... / on n'aimeraient pas ...

- Dites ce que vous aimerez avoir vu, avoir appris, à la fin de cette formation
- Dites également ce que vous n'aimerez pas avoir vu
- Ecrivez chaque réponse sur un post-it





Qu'est-ce qu'un Projet ?

Qu'en connaissez vous ?

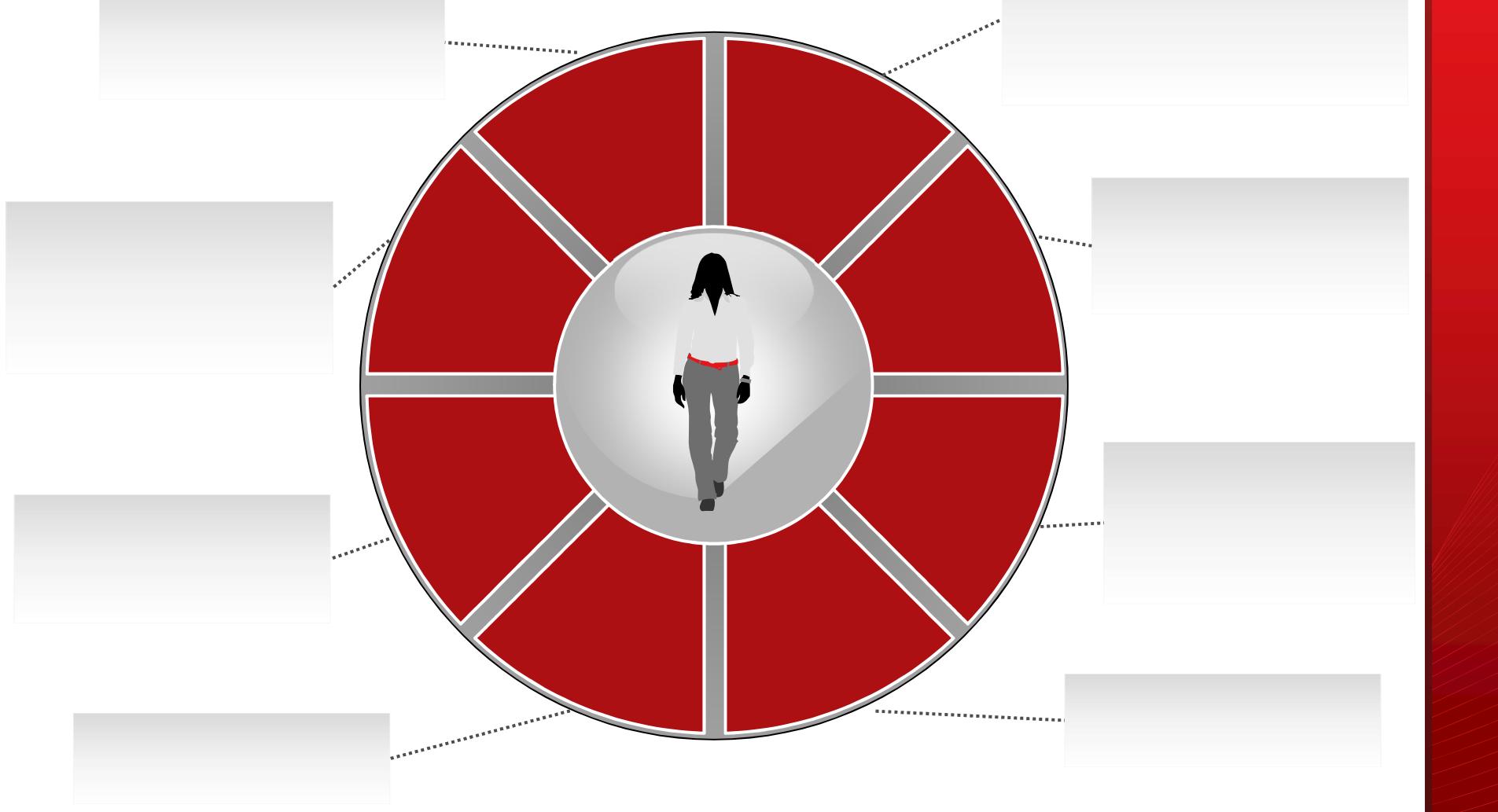
Liens avec le test ?

Activités, méthodes, phases ...



TP1 : Qu'est-ce qu'un Project Manager ?

La fonction globale du Project Manager



ISTQB c'est quoi ?

- **ISTQB (International Software Testing Qualification Board) / CFTL (Comité Français du Test Logiciel)**

- Organisme de certification, leader mondial sur la certification du test logiciel.
 - Il existe trois niveaux de certification

- Fondation
 - Avancé
 - Expert

- **Documents essentiels :**

- Un syllabus qui définit le cursus de formation des tests qui doit être appliqué pour construire la formation certifiante.
 - Un glossaire des termes du test logiciel

Contenu de la formation (1/3)

■ Objectifs

- Comprendre les principes de base des tests logiciels
- Savoir concevoir, planifier et réaliser des tests répondant à vos besoins spécifiques
- Être à même de mener les étapes fondamentales des processus de test

■ Connaissances nécessaires

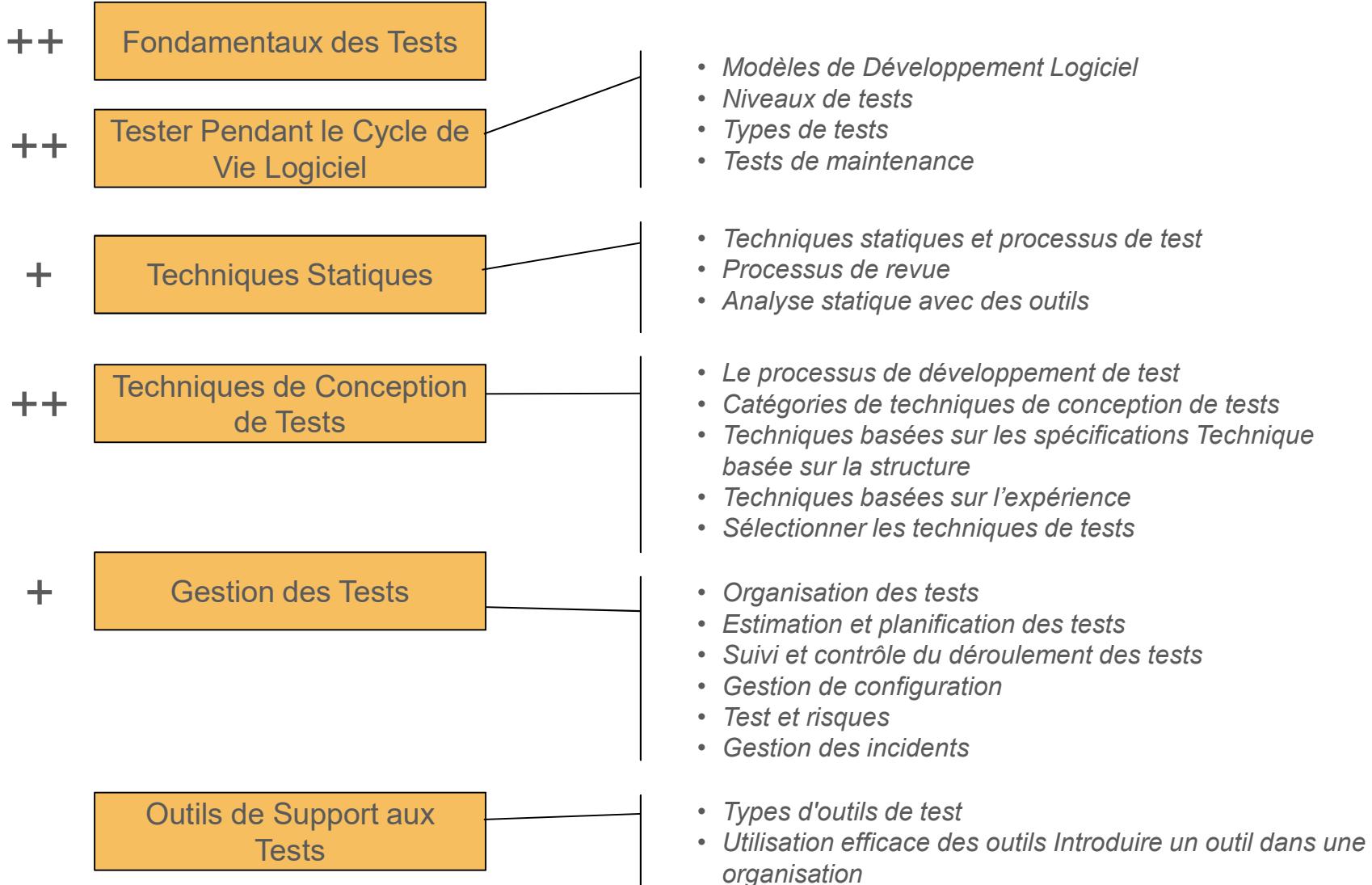
- Connaissance des cycles de développement logiciel
- Connaissance des bases du test logiciel
- Expérience des projets informatiques

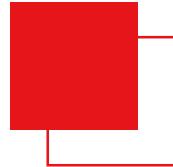
Contenu de la formation (2/3)

■ Evaluation

- Examen d'une heure
- Copies anonymes
- Sous forme de QCM (Questionnaire à choix multiples) contenant 40 questions
 - Une seule bonne réponse
 - Une mauvaise réponse correspond à 0 point et n'enlève pas de points
- Pendant l'examen il est interdit :
 - D'utiliser le portable
 - D'utiliser de l'aide
 - De tricher
 - De sortir

Contenu de la formation (3/3)





Atelier en équipe

Objectif - énoncé



Objectif

Définissez ce que représente a priori pour vous le Test



Énoncé

Par table, retenez 6 idées majeures ou éléments structurants

et venez les exposer

Donner aussi la charge de travail associée à l'activité de Test

Restitution de l'Atelier

- Idée 1 = _____
- Idée 2 = _____
- Idée 3 = _____
- Idée 4 = _____
- Idée 5 = _____
- Idée 6 = _____

- % de la charge d'un projet de développement = _____ %

Préambule

■ Process de test :

Processus consistant en toutes les activités du cycle de vie, statiques et dynamiques, concernant la planification et l'évaluation de produits logiciels et produits liés pour déterminer s'ils satisfont aux exigences spécifiées, pour démontrer qu'ils sont aptes aux objectifs et pour détecter des anomalies

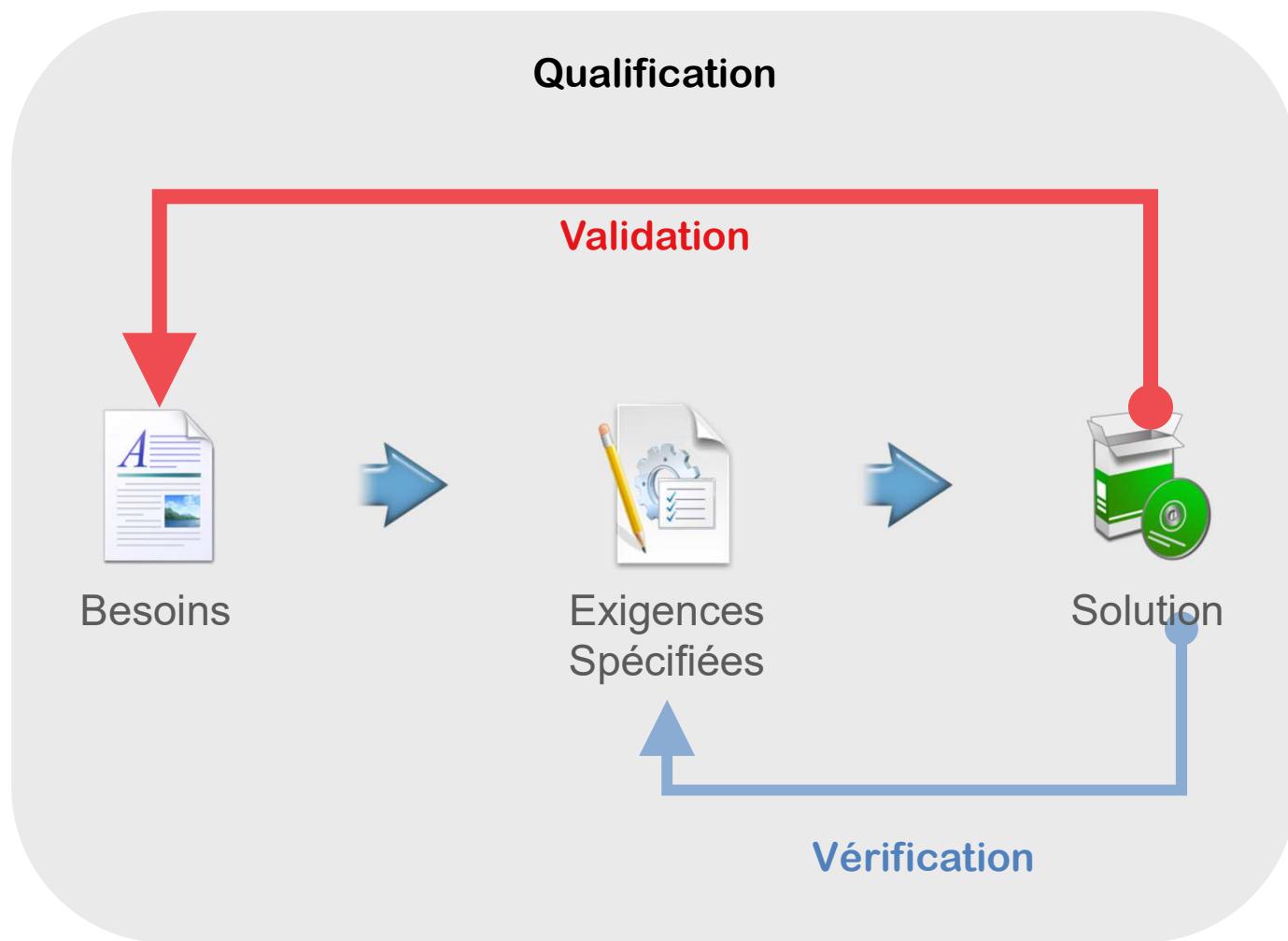
■ Glossaire

- **Vérification** : confirmation par l'examen et la fourniture de preuves objectives que des exigences spécifiées ont été remplies [ISO 9000]
- **Validation** : confirmation par l'examen et la fourniture de preuves objectives que les exigences, pour un usage ou une application voulue, ont été remplies [ISO 9000]
- **Qualification** : Vérification & Validation



Les Tests ne sont pas du « déboggage »

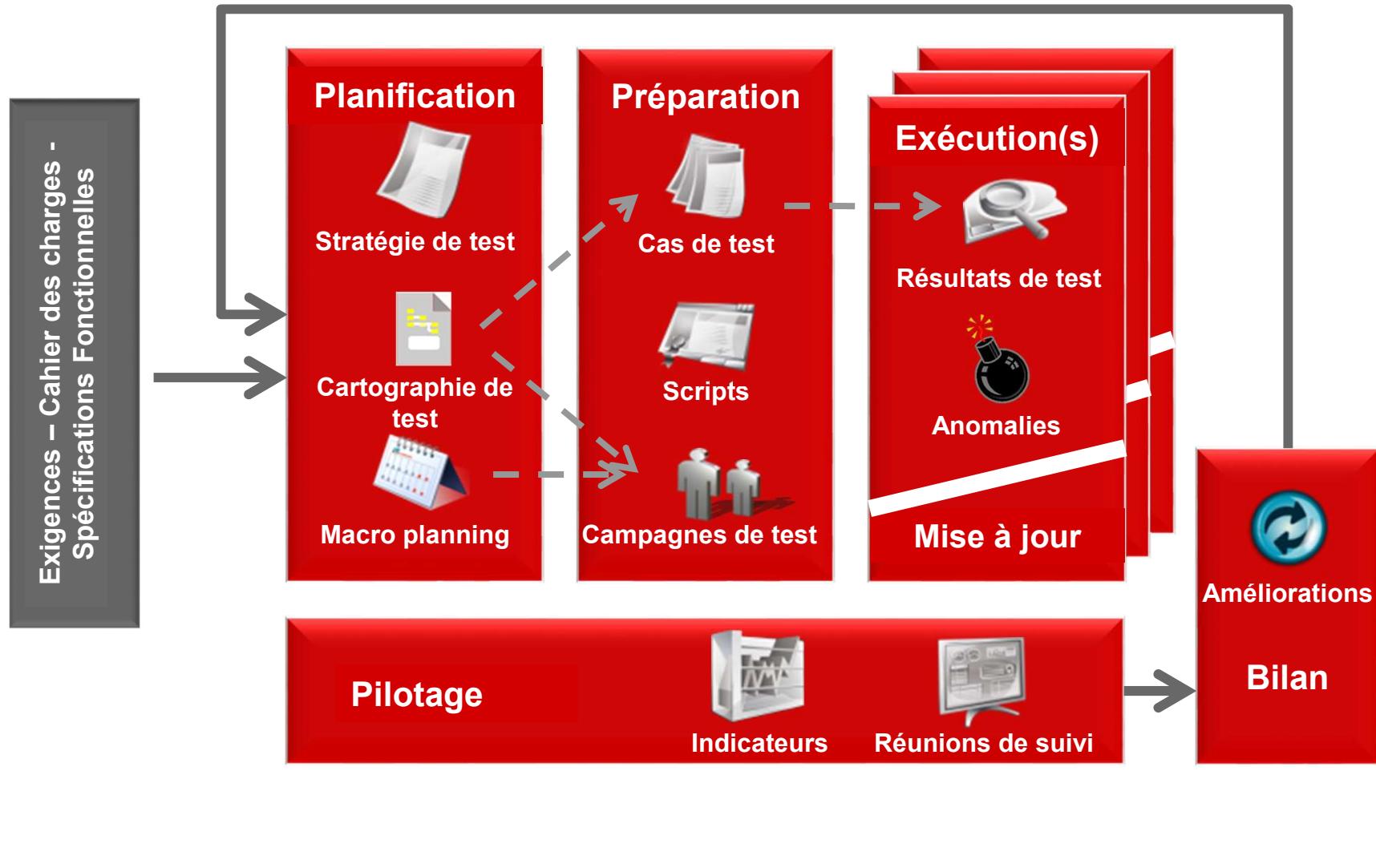
Qualification / Validation / Vérification



Standards/Normes

- Référentiels / standards :
 - IEEE : Institute of Electrical and Electronics Engineers
 - IEEE 610 : Standard définissant les termes de l'ingénierie logicielle
 - IEEE 829: Standard de l'IEEE pour la documentation de test logiciel. Ce standard donne des spécifications pour la forme d'un ensemble de documents pour l'usage du test logiciel.
 - ISO (International Organization for Standardization)
 - ISO 9126: La norme ISO/CEI 9126 définit un langage commun pour modéliser les qualités d'un logiciel.
 - La norme ISO/IEC 12207 a pour objectif de poser la référence pour les processus du cycle de vie logiciel pris dans sa généralité
- Évaluation de maturité du modèle :
 - CMMI : Capability Maturity Model Integration
 - TMMI : Test Maturity Model Integration

Rappel concernant le Modèle d'Implémentation des tests

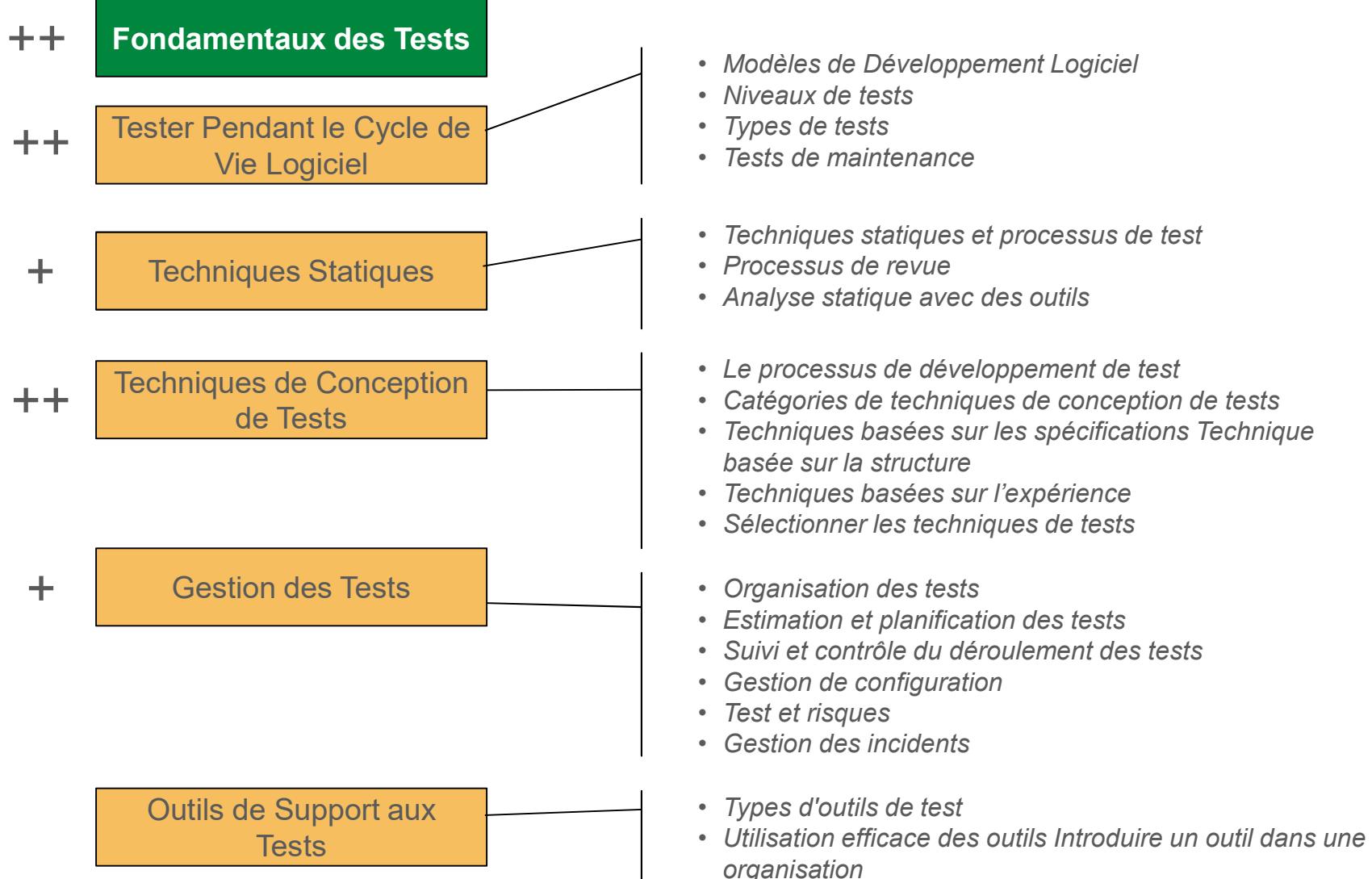


Un petit jeu pour évaluer nos connaissances

■ Tests Kahoot

- Rejoindre www.kahoot.it
- Et saisir le code ...

Contenu de la formation



1.1 Pourquoi les tests sont-ils nécessaires ? Contexte

- Les logiciels font partie intégrante de notre vie quotidienne (DAB, automobiles) malheureusement leur dysfonctionnement aussi
- Ces dysfonctionnements peuvent occasionner des :
 - Pertes financières
 - Pertes de temps
 - Baisse de réputation
 - Blessures ou la mort

1.1 Pourquoi les tests sont-ils nécessaires ? Origine des défauts logiciels

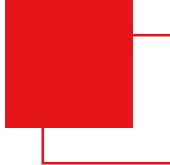
- Tout être humain peut faire une **erreur** (méprise) qui produit un défaut (bug) dans un logiciel ou un document
- Un **défaut** dans un logiciel peut générer une **défaillance**
- Les défauts générés par les humains peuvent être la conséquence de la complexité des systèmes d'information, leur interaction avec d'autres systèmes d'information, du délai toujours plus court pour produire le logiciel ...
- La cause des défaillances peut être environnementale : radiations, magnétisme, champs électronique ...

1.1 Pourquoi les tests sont-ils nécessaires ? Quelques bugs significatifs

- **9 septembre 1945 à 15h45** : premier bug de l'histoire informatique. Il s'agissait d'une mite bloquée dans le calculateur Mark II de l'université de Harvard aux USA qui donna d'ailleurs le terme de bug (insecte en Anglais)
- **4 juin 1996** : vol inaugural de Ariane 5 soldé par un échec. La fusée explose 40 secondes après son décollage avec à son bord 4 sondes d'une valeur totale de 370 millions de dollars. L'incident est dû à un bug dans les appareils informatiques utilisés par le pilote automatique. C'est le bug le plus couteux de l'histoire. Pas de victime !

1.1 Pourquoi les tests sont-ils nécessaires ? Rôle des tests – Tests et qualité

- Les tests permettent de réduire les risques d'apparition d'une défaillance du système
- Ils améliorent, après correction des défauts, la qualité du logiciel
- Il est alors possible par leur intermédiaire de mesurer la qualité du logiciel et d'accroître le niveau de confiance de ce dernier
- Pour déterminer la quantité de test suffisant, il faut prendre en compte le niveau de risques incluant les risques techniques
- D'autre part, les tests doivent permettre de fournir aux décideurs de go / no go l'état le plus précis possible de la qualité du logiciel



Constat



■ La réalité des projets

■ Dérive dans le temps

- 51% des projets ne respectent pas leurs dates de livraison
- 16 % des projets livrent dans les temps et avec leur budget prévu

■ Surcoût du projet

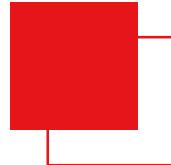
- 43% des projets dépassent leur budget initial
- dont 53% des projets coûtent 189% du budget initial estimé

■ Taux d'anomalies trop important

■ Turnover des équipes

■ Exigences oubliées, fausses ou mal-comprises

Source GARTNER

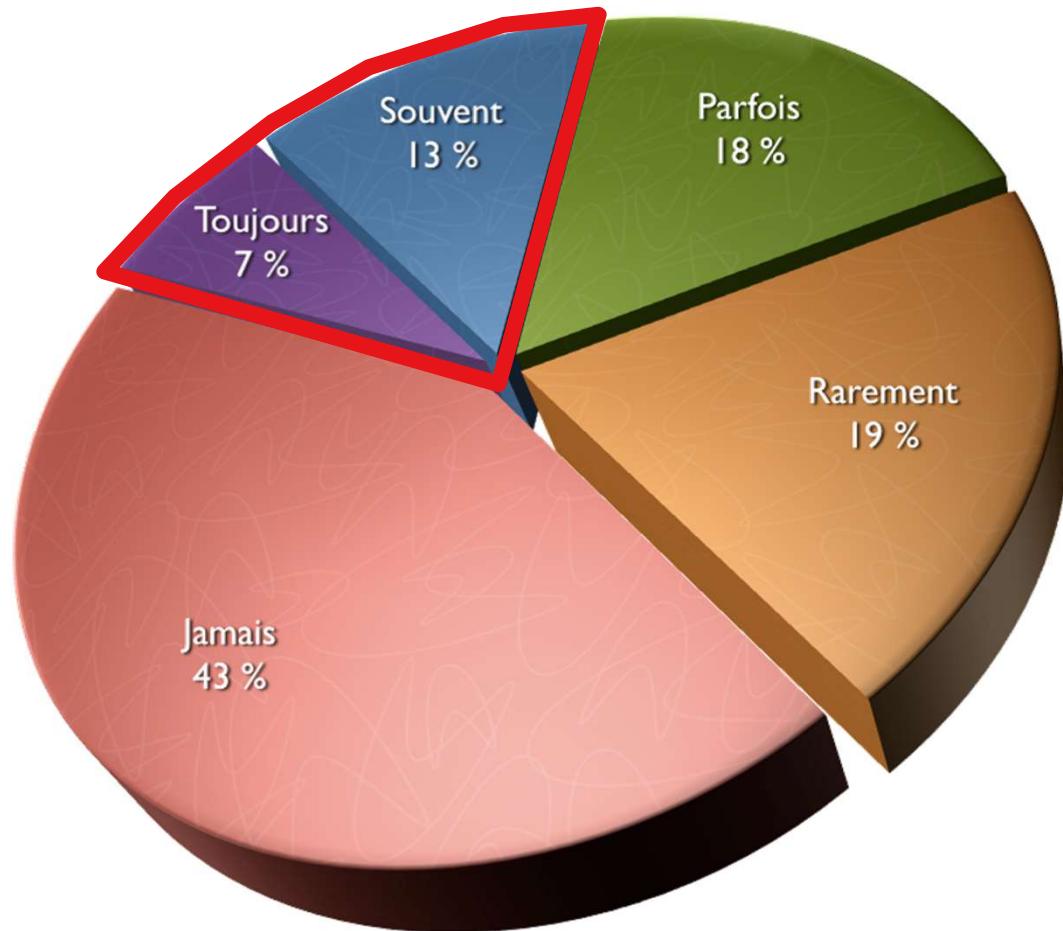


Constat

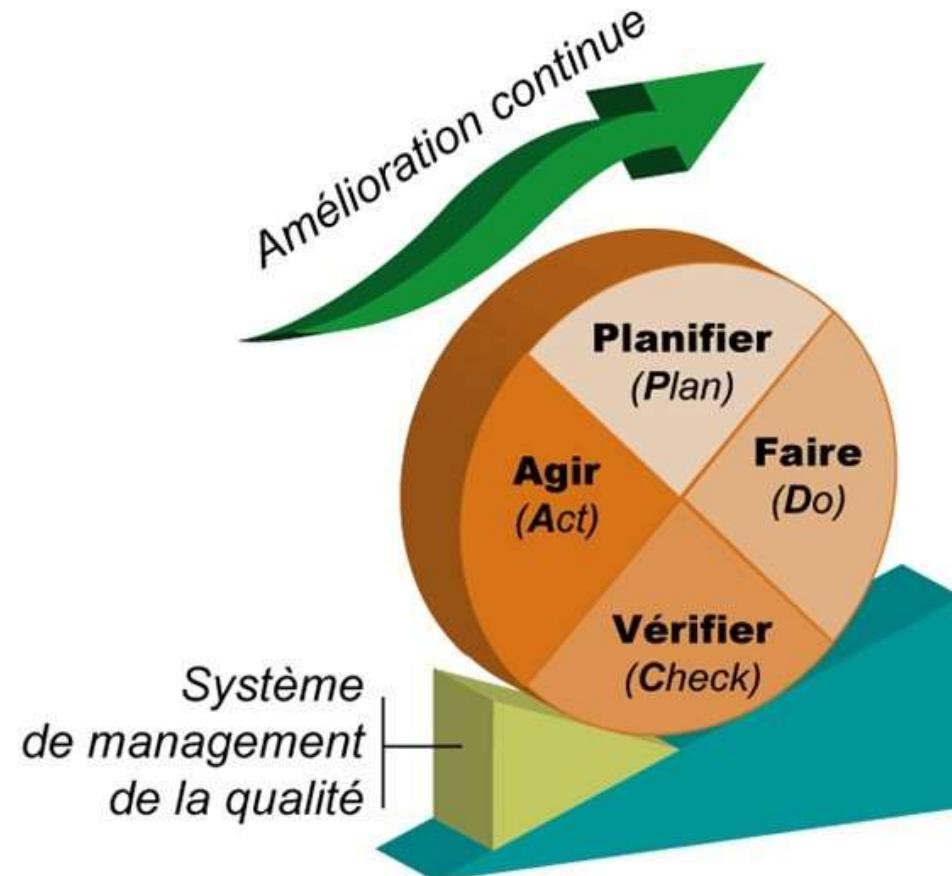


■ La difficulté de capturer les « bonnes exigences »

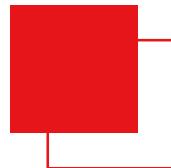
Utilisation effective
des fonctionnalités
spécifiées
dans un processus
séquentiel



1.1 Pourquoi les tests sont-ils nécessaires ? Rôle des tests – Tests et qualité

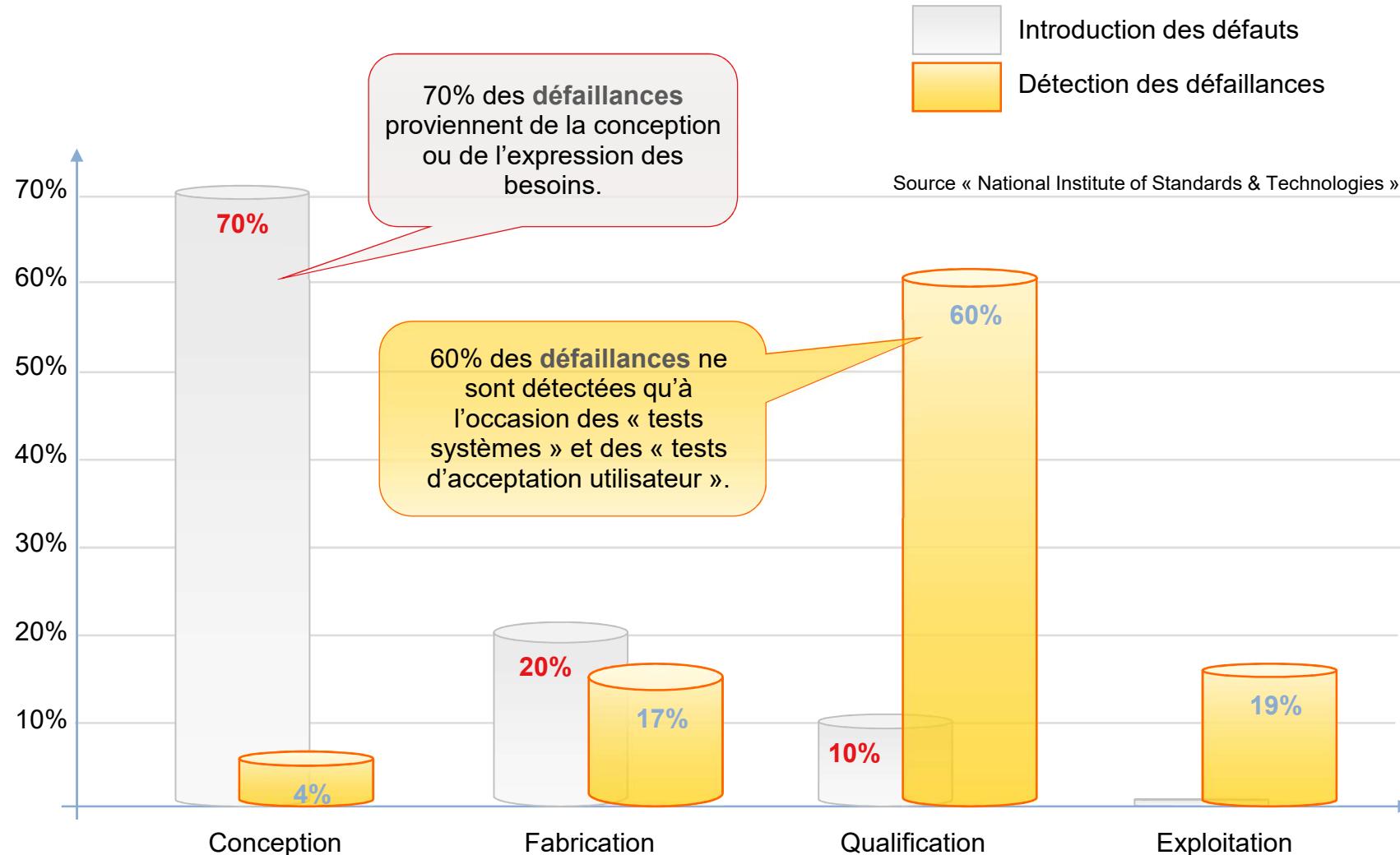


La roue de Deming (Source Internet Kaisen Skills)

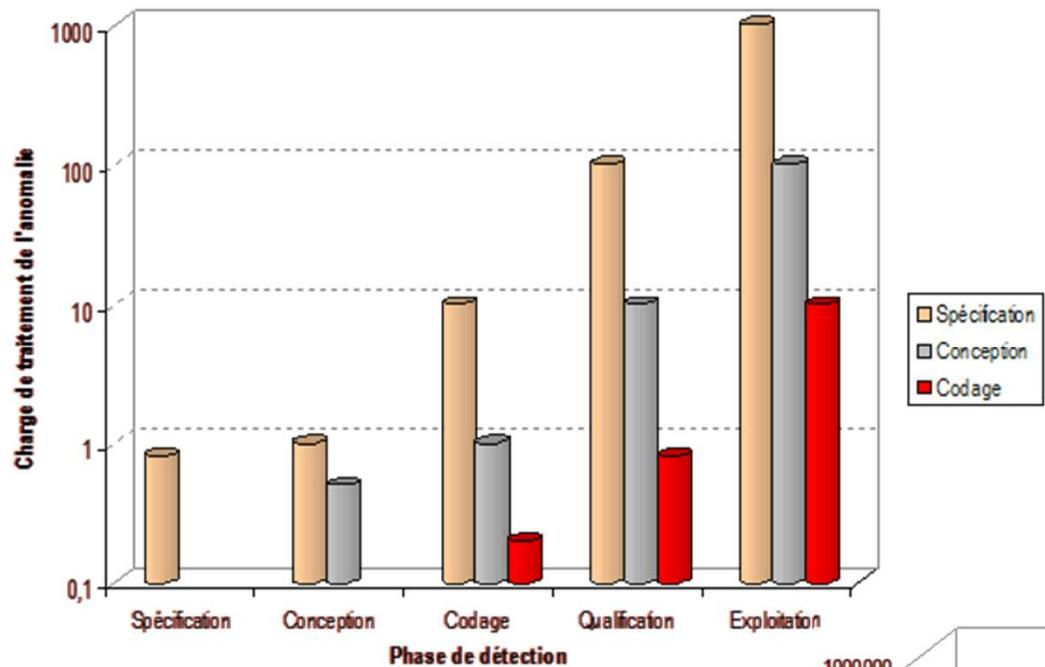


Constat

Décalage entre injection des défauts et détection des défaillances

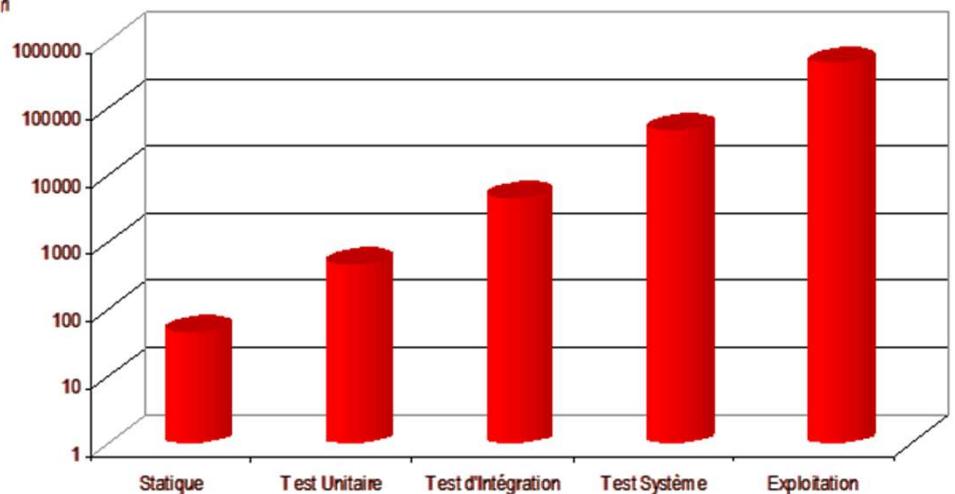


1.1 Pourquoi les tests sont-ils nécessaires ? Coût correction anomalie / survenance



Evolution de la charge
du traitement
(correction) d'une
anomalie détectée
pendant une activité

Evolution du coût (en €) du
diagnostic et de la correction
d'une anomalie par rapport aux
niveaux de test
(source : UK Orange services)



1.1 Pourquoi les tests sont-ils nécessaires ?

A retenir

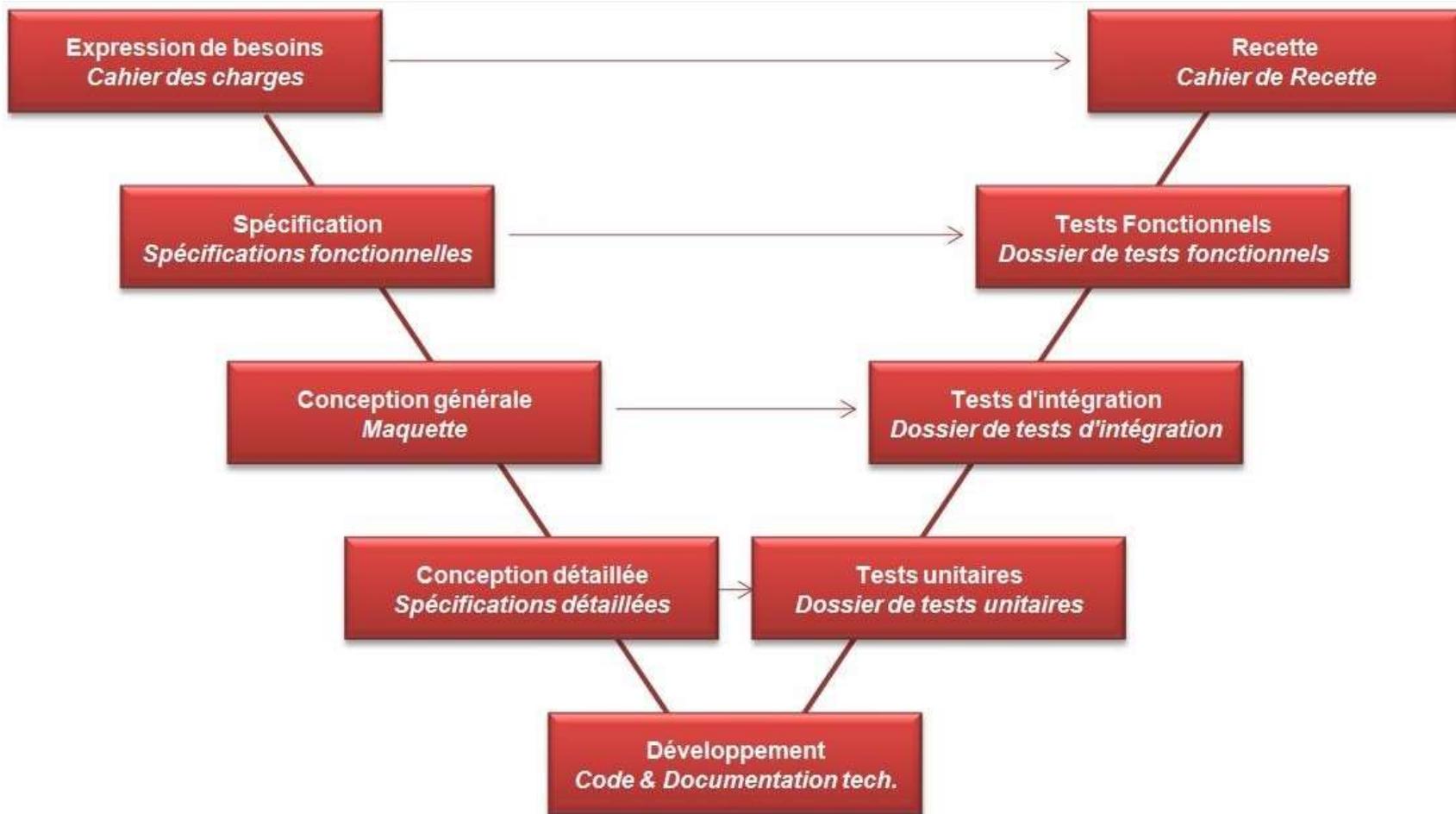


- **Défaut (ou bug)** : une imperfection dans un composant ou un système qui peut conduire à ce qu'un composant ou un système n'exécute pas les fonctions requises, par exemple une instruction ou une définition de données incorrecte. Un défaut, si rencontré lors de l'exécution, peut causer la défaillance d'un composant ou d'un système.
- **Erreur (Méprise)** : action humaine produisant un résultat incorrect [d'après IEEE 610]
- **Défaillance** :
 - Déviation constatée du composant ou système de la fourniture, du service ou du résultat attendu [d'après Fenton];
 - Incapacité d'un système ou d'un composant d'exécuter une fonction requise dans les limites spécifiées. Une défaillance peut être produite quand un défaut est rencontré [DO-178B]
- **Qualité** : degré par lequel un composant, système ou processus atteint des exigences spécifiées et/ou des besoins ou attentes des clients ou utilisateurs [d'après IEEE 610]
- **Risques** : un facteur qui pourrait résulter dans des conséquences négatives futures, généralement exprimé comme un impact et une probabilité.

1.2 Que sont les tests ? Objectif de test

- Les activités de test ne se réduisent pas à l'exécution des cas de test.
- On trouve parmi ces activités l'élaboration de la stratégie de test, la planification, le pilotage, le reporting, la revue de code ...
- Les principaux **objectifs de tests** sont :
 - Trouver des défauts
 - Acquérir de la confiance sur le niveau de qualité
 - Donner l'état de la qualité du logiciel le plus précis possible pour les prises de décision de go / no go
 - Prévenir des défauts
- Les objectifs de test peuvent différer en fonction du cycle de vie logicielle où ils interviennent

1.2 Que sont les tests ? Cycle de vie logiciel/cycle de tests



1.2 Que sont les tests ? Tester / déboguer

- Tester et déboguer ne représentent pas la même activité
- Les tests dynamiques mettent en évidence les défaillances
- Le débogage est une activité de développement qui permet d'analyser, trouver et supprimer les défauts à l'origine des défaillances

1.2 Que sont les tests ? A retenir



- **Débuguer** : le processus de trouver, analyser et éliminer les causes de défaillance dans les logiciels
- **Exigence** : une condition ou capacité requise par un utilisateur pour résoudre un problème ou atteindre un objectif
- **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 : revues, relecture technique, revue de code [d'après IEEE 1028]
- **Cas de test** : un ensemble de
 - valeurs d'entrée,
 - préconditions d'exécution,
 - résultats attendus
 - et de postconditions d'exécution,développées pour un objectif ou une condition de tests particulier, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique [d'après IEEE 610]
- **Objectif de test** : une raison ou but pour la conception et l'exécution d'un test.

1.3 Les 7 principes généraux des tests

Listes de principes 1/3

- **Principe 1 :** Les tests peuvent prouver la présence de défauts mais ne peuvent en aucun cas prouver leur absence
- **Principe 2 :** Tester de façon exhaustive en utilisant par exemple une table de décision ne doit pas être envisagé sauf pour des systèmes très simples ce qui n'est plus le cas aujourd'hui. Il faut privilégier l'analyse des risques et des priorités pour calibrer l'effort de test.
 - Exemple : Test de l'IHM suivante :
 - 20 écrans
 - 4 menus / 3 options
 - 10 champs
 - 2 types de données
 - 100 valeurs possibles
 - Total pour test « exhaustif »
 - $20 \times 4 \times 3 \times 10 \times 2 \times 100 = 480\,000$ cas de tests
 - Expert (10 s par test !) → 180 jours !

Plutôt que des tests exhaustifs, nous utilisons les risques et les priorités pour focaliser les efforts de tests !

1.3 Les 7 principes généraux des tests

Listes de principes 2/3

- **Principe 3 :** Tester au plus tôt permet de réduire considérablement le coût de correction d'un défaut. Les tests peuvent intervenir dès le début du projet.



- **Principe 4 :** Pour déterminer les efforts de tests en fonction des modules, on peut prendre en compte le niveau de probabilité de survenance de défaillances (par exemple des développements donnés à une personne junior)
- **Principe 5 :** Paradoxe du pesticide. Des mêmes tests exécutés de nombreuses fois ne décèlent plus d'anomalie. Il est important de réviser les tests pour couvrir d'autres chemins dans le système

1.3 Les 7 principes généraux des tests

Listes de principes 3/3

- **Principe 6 :** Les tests dépendent du contexte et du système testé.
- **Principe 7 :** Qualifier un système dont les exigences de test ont été mal définies avec les utilisateurs initialement n'a aucune plus value.

1.3 Les 7 principes généraux des tests

A retenir



- **Tests exhaustifs** : une approche des tests selon laquelle la suite de tests comprend toutes les combinaisons de valeurs d'entrée et de pré-conditions.

1.4 Processus de test fondamental

Activités de test

- **Planifier et contrôler les tests**
 - Elaboration de la stratégie de test
 - Planifier les activités de test
 - Suivi de l'avancement des tests
- **Analyser et concevoir**
 - Etablir des exigences de test
 - Concevoir les cas de test à partir des spécifications
- **Elaborer et exécuter les campagnes**
 - Elaboration des campagnes de test
 - Exécution des cas de tests
- **Evaluer les critères de sortie et informer**
 - Vérifier si les critères de sortie sont atteints
 - Sinon établir les actions à entreprendre pour atteindre les objectifs
 - Effectuer les rapports de synthèse des tests
- **Activités de clôture des tests**
 - Vérifier que les livrables à livrer sont livrés
 - Clôturer les anomalies encore ouvertes ou créer des demandes d'évolution
 - Faire le bilans des activités de tests en précisant les points forts et les améliorations à entreprendre pour les futures campagnes de test

1.4 Processus de test fondamental

A retenir

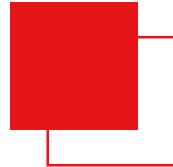


- **Couverture** : le degré, exprimé en pourcentage, selon lequel un élément de couverture spécifié a été exécuté lors d'une suite de test.
- **Test de régression** : tests d'un programme préalablement testé, après une modification, pour s'assurer que des défauts n'ont pas été introduits ou découverts dans des parties non modifiées du logiciel,
- **Critère de sortie** : l'ensemble des conditions génériques et spécifiques, convenues avec les responsables, pour permettre de terminer officiellement un processus.
L'objectif d'un critère de sortie est d'éviter qu'une tâche ne soit considérée comme achevée alors qu'il y a encore des parties de cette tâche qui n'ont pas été terminées.
- **Stratégie de test** : un document de haut niveau définissant, pour un programme, les niveaux de tests à exécuter et les tests dans chacun de ces niveaux (pour un ou plusieurs projets).

1.5 La psychologie des tests Influence sur les tests

- Le test est un métier à part entière
- Les testeurs et les développeurs travaillent ensemble dans un objectif commun celui d'améliorer la qualité du logiciel
- Les testeurs ne sont pas là pour juger le travail des développeurs
- Les tests sont importants à tous les niveaux :
 - Les tests unitaires des développeurs permettent de corriger bon nombre de défauts . Cependant, un développeur effectue des tests orientés (car il connaît le code)
 - Les tests d'intégration et de qualification des testeurs permettent de vérifier le bon fonctionnement de l'ensemble des développements par rapport aux spécifications. Leur indépendance assure une meilleure efficacité des tests

La meilleure façon d'établir la confiance dans la qualité d'une solution est d'essayer de la détruire !



Psychologie de la qualification

Développeur = créateur / Testeur = destructeur

- L'identification de défauts peut être perçue comme une critique contre la solution et contre ceux qui l'ont réalisée
- La qualification est souvent vue comme une activité destructrice, alors que c'est très constructif
- Amélioration de la communication et des relations entre les testeurs et leurs interlocuteurs
 - Commencer par une collaboration plutôt que par des conflits – rappeler à chacun l'objectif commun qui est de fournir des solutions de meilleure qualité
 - Communiquer les découvertes de façon neutre et factuelle sans critiquer la personne
 - Avoir de la diplomatie lors de la transmission des défauts
 - Rester aux faits et ne pas juger (les faits doivent être expliqués, l'individu ne peut cependant pas être évalué)
 - Confirmer que l'autre personne a compris ce que l'on a dit et vice versa
 - Essayer de comprendre ce que ressent une autre personne et pourquoi elle réagit comme elle le fait

1. Fondamentaux des Tests

QCM

1. Lequel des énoncés suivants n'est pas décidé durant la phase de planification des tests ..?

- A. Les dates et les livrables
- B. Les matériels et logiciels
- C. Les critères en entrée et en sortie
- D. Les types de cas de test

2. La sécurité relève ..?

- A. Des tests de conformité
- B. Des tests de catastrophe
- C. De la vérification du respect des règles
- D. Des tests fonctionnels

3. Qu'est ce qui n'est pas dans l'ordre au niveau du process de test fondamental ?

- A. Elaborer la stratégie de test
- B. Planifier les activités de test
- C. Concevoir les cas de test à partir des spécifications
- D. Etablir les exigences de test

4. Quand ce qui est visible pour les utilisateurs finaux est une déviation du comportement spécifique ou attendu, c'est ce qu'on appelle ..?

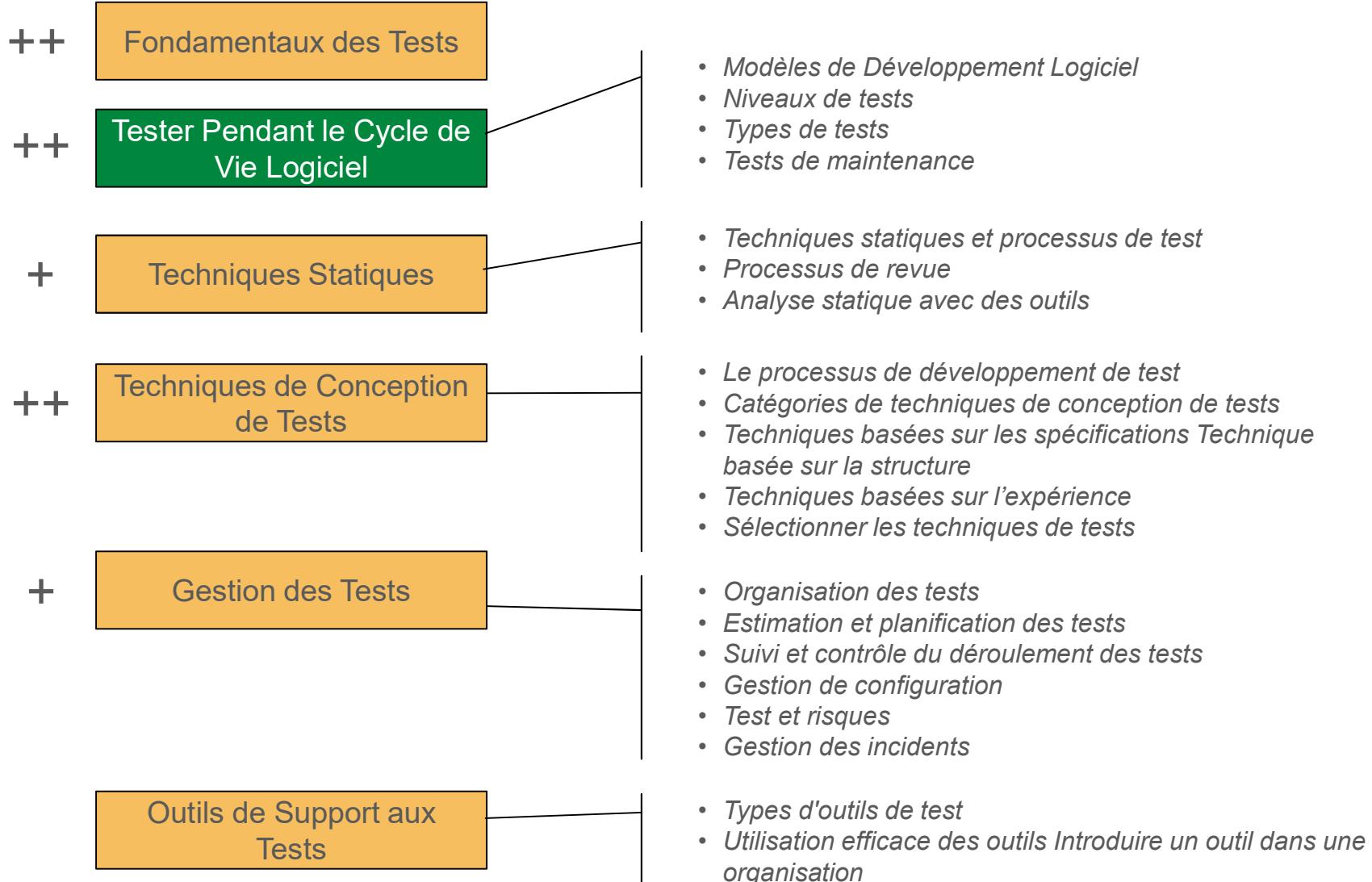
- A. Une erreur
- B. Une faute
- C. Une défaillance
- D. Un défaut
- E. Méprise

1. Fondamentaux des Tests

QCM

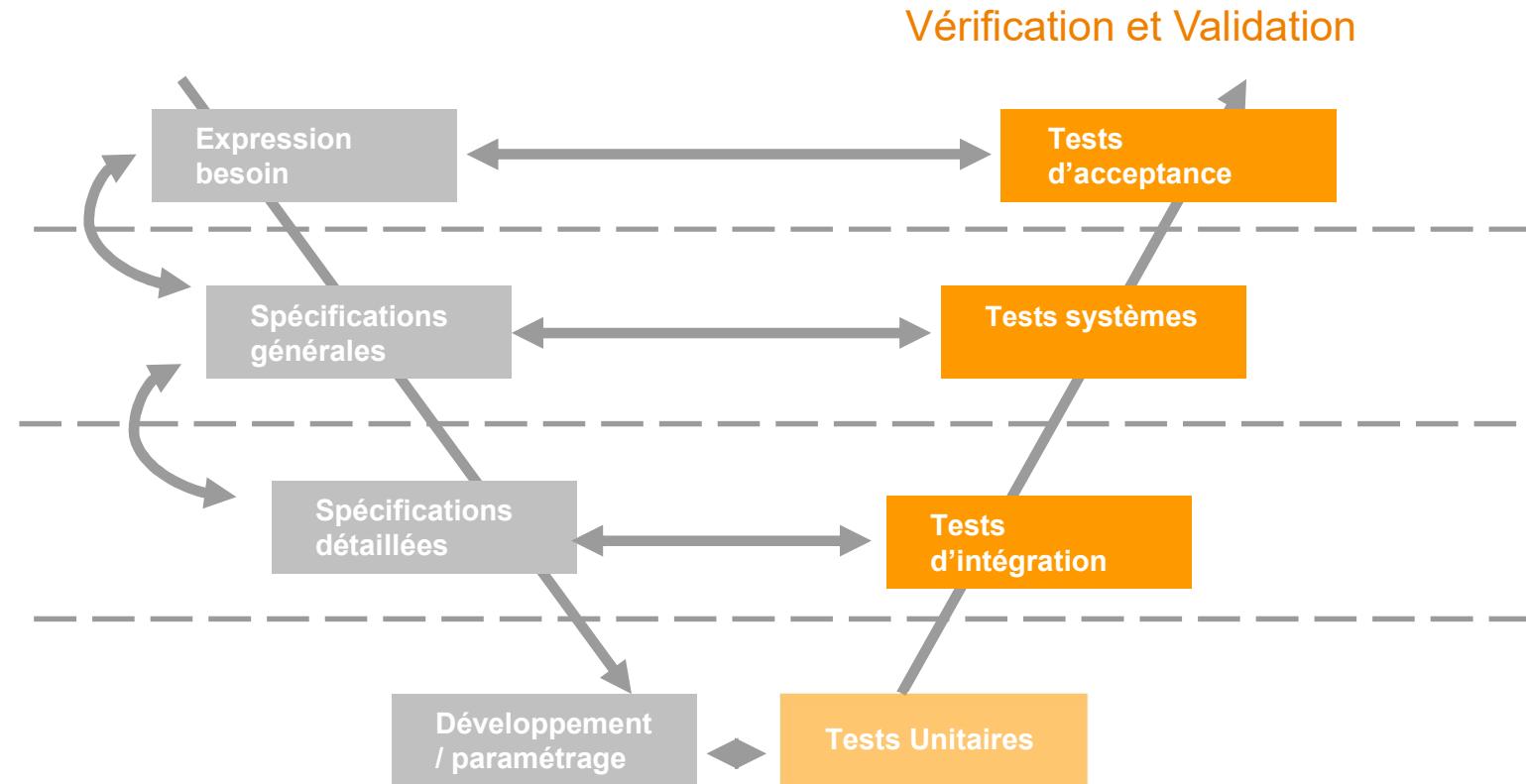
5. L'implémentation des Tests exhaustifs:
 - A. est difficile, mais possible
 - B. est possible et pratique
 - C. n'est pas pratique et est impossible
 - D. est toujours possible
6. Passer en revue les bases de test est une partie de quelle phase
 - A. Analyse et conception de test
 - B. Mise en œuvre et exécution de tests
 - C. Activités de clôture des tests
 - D. Evaluation des critères de sortie et reporting
7. Lequel des énoncés suivants est le plus important pour promouvoir et maintenir de bonnes relations entre les testeurs et les développeurs?
 - A. Comprendre ce que les gestionnaires apprécient sur le test
 - B. Expliquer les résultats des tests de façon neutre
 - C. Identifier les solutions de contournement potentielles des clients pour les bugs
 - D. Promouvoir une meilleure qualité des logiciels chaque fois que possible
8. Choisissez la meilleure définition de la qualité
 - A. La qualité est un travail
 - B. Zéro défaut
 - C. La conformité aux exigences
 - D. Travailler comme prévu

Contenu de la formation



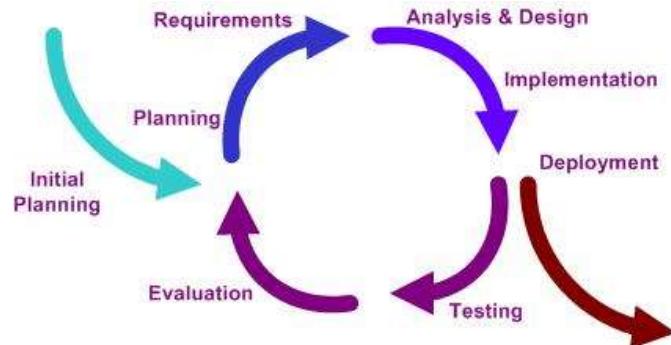
2.1. Les modèles de développement logiciel

Le modèle en V



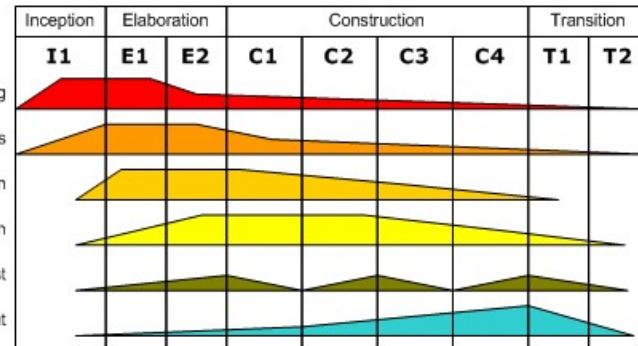
2.1. Modèles de développement logiciel

Modèle Itératif

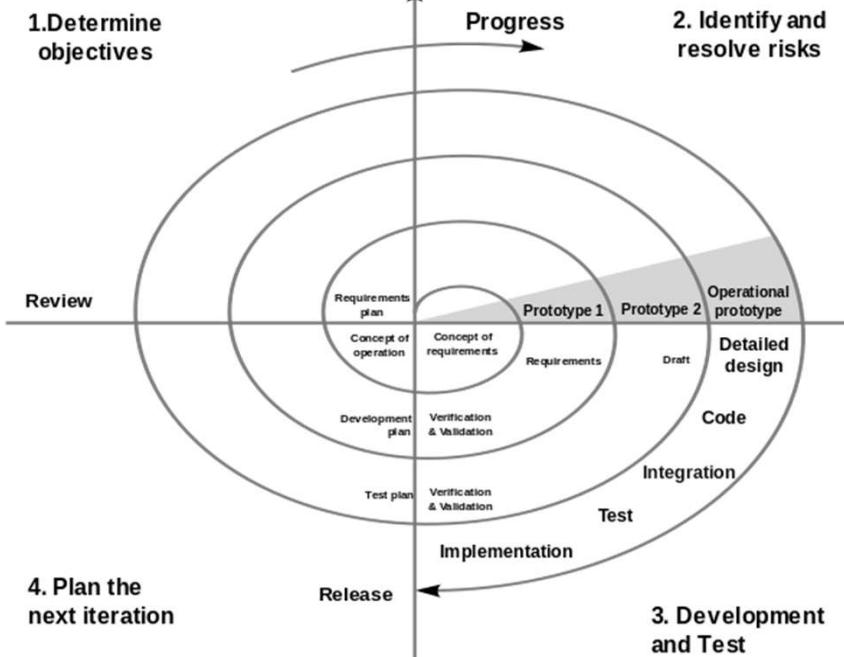
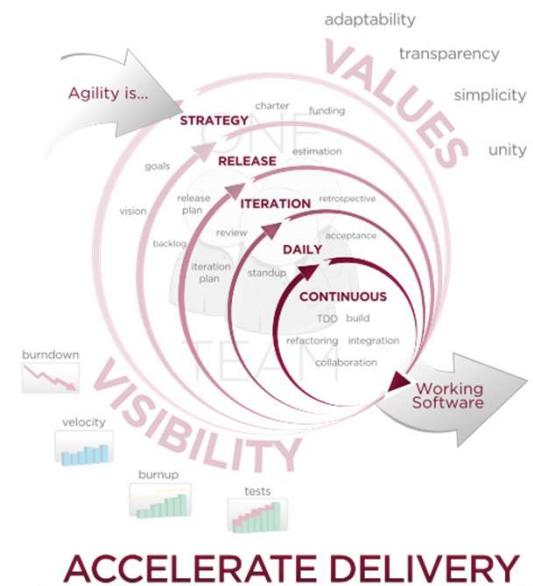


Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



AGILE DEVELOPMENT



2.1. Modèles de développement logiciel

Tests dans le cadre d'un modèle

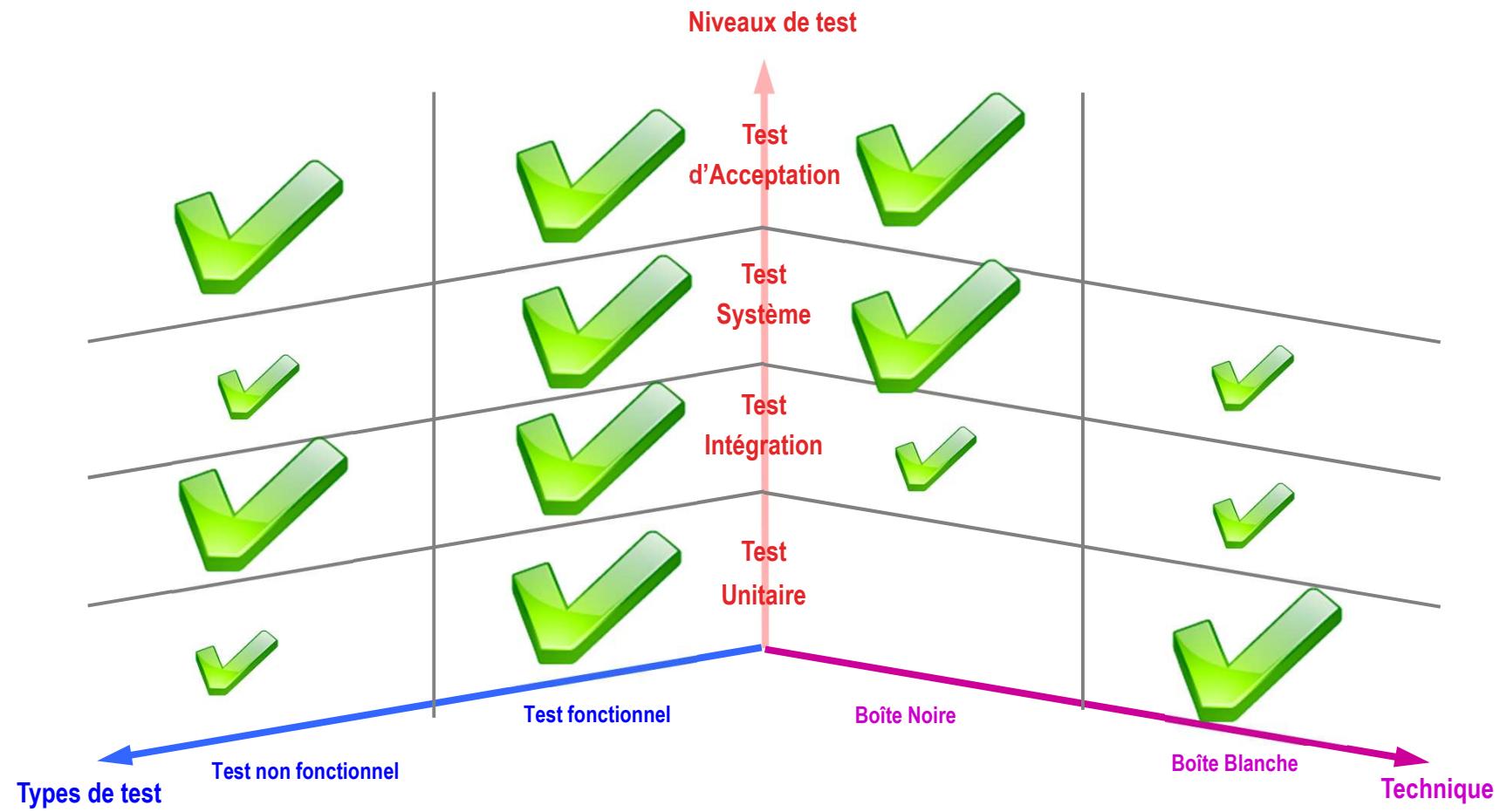
- Dans chaque modèle apparaissent des phases de développement qui génèrent des attendus et des activités de validation.

Les 4 niveaux de tests se retrouvent dans chacun des modèles, la répartition des activités et l'ordonnancement du processus de test pourra être différent.

- **Niveau de test** : groupe d'activités de tests organisées et gérées ensemble.
Un niveau est lié au responsabilité au sein d'un projet
- **Type de test** : groupe d'activités de tests dont l'objectif est de tester un composant ou un système sur un ou plusieurs attributs liés entre eux.
Un type est focalisé sur un objectif.

2.1. Modèles de développement logiciel

Typologie / niveaux de tests



2.2. Niveaux de test

Test de composants

Caractéristiques

■ Définition :

Tests de composants individuels

Il s'agit des tests effectués dans les logiciels (modules, programmes, objets, classes, ...) qui sont testables séparément.

■ Points importants

Généralement effectués sur des environnements de développement et impliquent le programmeur responsable du code

Les défauts sont généralement corrigés dès qu'il sont trouvés, les incidents ne sont pas systématiquement logués.

Les tests sont effectués avec accès au code

Bases de test

- Exigence de composants
- Conception détaillée
- Code

Objets habituels de test

- Composant
- Programmes
- Conversion de données, programmes de migration
- Modules de base de données

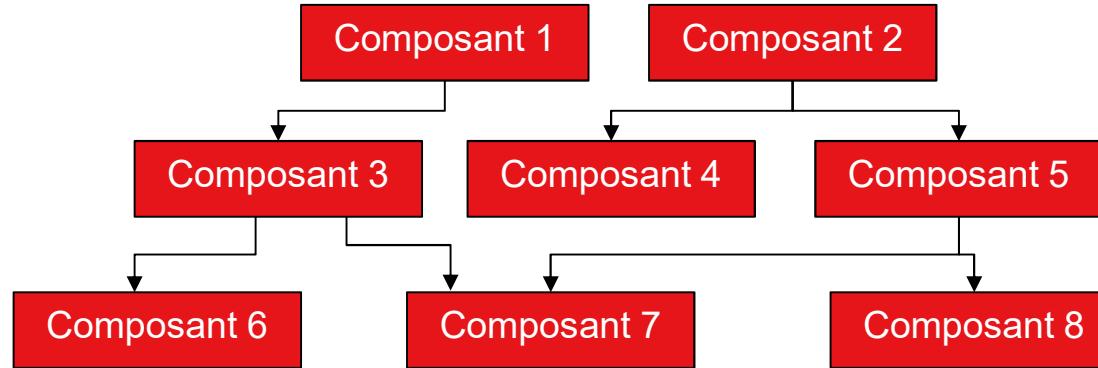
Vocabulaire

- Bouchon, simulateur, pilotes
- Tests statiques, dynamiques
- Débogage
- Approche de développement piloté par les tests

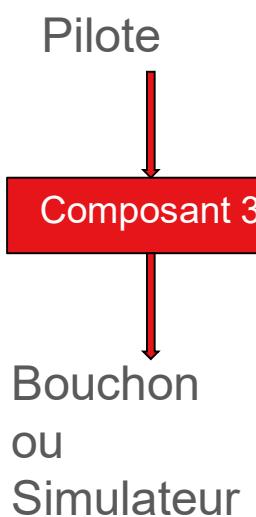
Exemples

- Régression
- Tests unitaires
- Revue de code

2.2. Niveaux de test Bouchons / Pilotes



La **simulation** va permettre de tester un composant en simulant le comportement des composants situés aux niveaux amont ou aval de ce dernier.



Pilote, Conducteur : Composant ou outil logiciel qui remplace un composant qui contrôle/appelle un composant ou système.

Bouchon : implémentation spéciale ou squelettique d'un composant logiciel qui remplace un composant appelé pour tester le composant qui l'appelle ou en est dépendant.

Simulateur : appareil ou programme ou système utilisé pendant les tests qui se comporte comme un système donné à la réception d'entrées contrôlées.

2.2. Niveaux de test

Test d'intégration

Caractéristiques

■ Définition :

Tests effectués pour démontrer des défauts dans les interfaces et interaction de composants ou systèmes intégrés.

■ Points importants :

- Tests fonctionnels ou non
- Tests essentiellement dynamiques, boite noire et boite blanche
- L'objectif principal est de contrôler les interactions du système

Bases de test

- Conception du logiciel et du système
- Architecture
- Workflows
- Cas d'utilisations

Vocabulaire

- Big Bang
- Incrémentale ascendante (Bottom up)
- Incrémentale descendante(Top Down)

Objets habituels de test

- Implémentation de bases de données du système
- Infrastructure
- Interfaces
- Configuration système et données de configuration

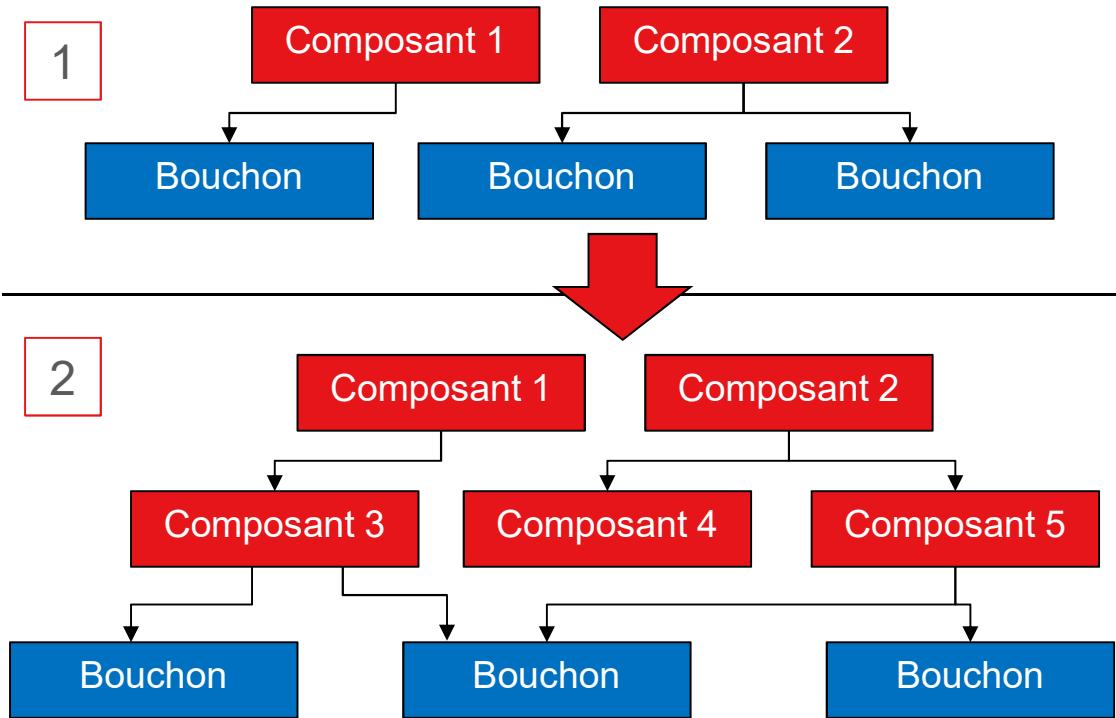
Exemples

- Régression
- Test IHM
- Test interaction de composants

2.2. Niveaux de test

Intégration incrémentale descendante

Test Top Down : Les composants testés en premier sont en haut de la hiérarchie. On commence donc par simuler les composants situés au niveau inférieur par des bouchons. On utilise les composants testés pour tester les composants du niveau juste en dessous.



Les plus

- Pas besoin de pilotes
- Permet de trouver facilement les erreurs

Les moins

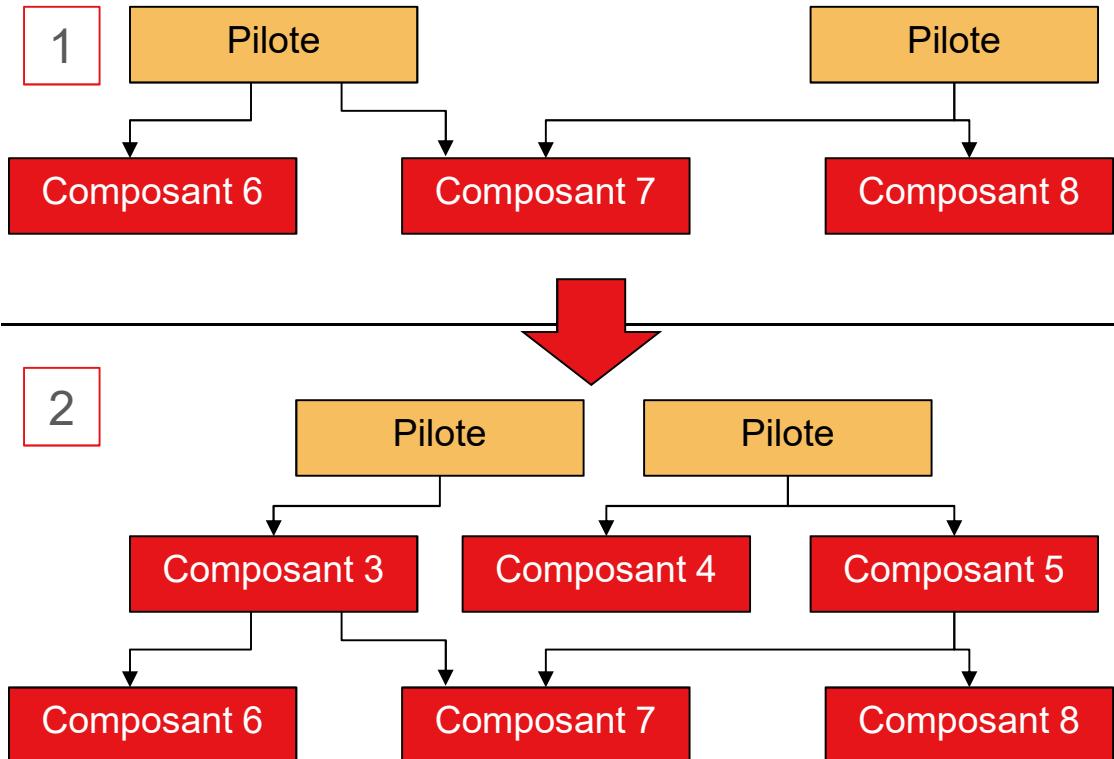
- Peu de possibilité de parallélisme .
- Les coûts de développement des bouchons sont difficiles à estimer

Les risques

Les composants validés servent à piloter les composants testés, il n'est pas possible de faire des cas de messages entrants erronés.

2.2. Niveaux de test Intégration incrémentale ascendante

Test Top Down : Les composants testés en premier sont en bas de la hiérarchie. On commence donc par simuler les composants situés au niveau supérieur par des pilotes. On utilise les composants testés pour tester les composants du niveau juste au dessus.



Les plus

- Pas besoin de bouchons
- Conditions de tests plus faciles à créer

Les moins

- Les pilotes sont difficiles à implémenter et concevoir
- Rien ne garantit que le pilote du composant validé se comporte comme le composant en question
- Le programme n'est autonome qu'en fin de process quand le dernier pilote est remplacé

2.2. Niveaux de test

Test système

Caractéristiques

- Définition

Processus de test d'un système intégré pour vérifier qu'il corresponde à des exigences spécifiques

- Points importants

Environnement doit être proche de la cible finale

Les tests systèmes peuvent cibler des objectifs fonctionnel ou non fonctionnel

Essentiellement du test en boîte noire

Bases de test

- Spécification d'exigences système et logiciel
- Cas d'utilisation
- Spécifications fonctionnelles
- Rapport d'analyse des risques

Vocabulaire

- Qualification

Objets habituels de test

- Manuels système, utilisateur et opérationnels
- Configuration système et données de configuration

Exemples

- Régression
- Tests de performance
- Tests systèmes fonctionnels

2.2. Niveaux de test

Test d'acceptation

Caractéristiques

■ Définition

Test formel en rapport avec les besoins , exigences et processus métier, conduit pour déterminer si un système satisfait ou non aux critères d'acceptation et permettre aux utilisateurs, clients ou autres entités autorisées de déterminer l'acceptation ou non du système.

■ Points importants

Responsabilité client/ utilisateurs

Prise de confiance sur le système (fonctionnelle et non fonctionnelle)

La recherche d'anomalies fonctionnelles n'est pas l'objectif principal, évaluation de la capacité à déployer

L'acceptation peut avoir lieu avant la fin de cycle de développement, on peut faire des tests d'acceptation de l'utilisabilité d'un composant avant les test d'intégration

Bases de test

- Exigences utilisateur
- Exigences du système
- Cas d'utilisation
- Processus métier
- Rapport d'analyse des risques

Vocabulaire

- COTS (Composant sur étagère)
- Alpha
- Beta, field testing (tests sur le terrain)

Objets habituels de test

- Processus métier sur l'intégralité du système
- Processus opérationnel de maintenance
- Procédures utilisateur
- Formulaires
- Rapports
- Données de configuration

Exemples

- Acceptation opérationnelle
- Acceptation utilisateur
- Acceptation réglementaire
- Régression

2.3. Types de Test

Introduction : ISO 9126

- **Capacité fonctionnelle :**

Est-ce que le logiciel répond aux besoins fonctionnels exprimés ?

- **Fiabilité :**

Est-ce que le logiciel maintient son niveau de service dans des conditions précises et pendant une période déterminée ?

- **Facilité d'utilisation :**

Est-ce que le logiciel requiert peu d'effort à l'utilisation ?

- **Rendement / Efficacité :**

Est-ce que le logiciel requiert un dimensionnement rentable et proportionné de la plate-forme d'hébergement en regard des autres exigences ?

- **Maintenabilité :**

Est-ce que le logiciel requiert peu d'effort à son évolution par rapport aux nouveaux besoins ?

- **Portabilité :**

Est-ce que le logiciel peut être transféré d'une plate-forme ou d'un environnement à un autre ?

2.3. Types de Test

Tests des caractéristiques fonctionnelles

Fonctionnalité : Capacité d'un produit logiciel à fournir des fonctions qui répondent à des besoins explicites ou implicites quand le logiciel est utilisé sous des conditions spécifiées

Capacité fonctionnelle :

- Pertinence
- Exactitude
- Interopérabilité
- Sécurité
- Conformité

2.3. Types de Test

Tests des caractéristiques non fonctionnelles

Fiabilité :

- Maturité
- Tolérance aux pannes
- Facilité de récupération

Facilité d'utilisation :

- Facilité de compréhension
- Facilité d'apprentissage
- Facilité d'exploitation
- Pouvoir d'attraction

Rendement / Efficacité :

- Comportement temporel
- Utilisation des ressources

Maintenabilité :

- Facilité d'analyse
- Facilité de modification
- Stabilité
- Testabilité

Portabilité :

- Facilité d'adaptation
- Facilité d'installation
- Coexistence
- Interchangeabilité

2.3. Types de Test

Tests de la structure, architecture logicielle

Objectifs : obtenir des éléments permettant de déterminer la qualité de la structure logicielle (Voir 4.4 Techniques de tests basées sur la structure)

2.3. Types de Test Tests liés au changement

Les tests liés au changement interviennent à la suite de modification sur le logiciel testé.

Deux cas de tests peuvent apparaître :

- les tests de confirmation
- les tests de régression

Les tests de confirmation permettent d'établir que des anomalies corrigées par une mise à jour sont effectivement résolues.

Les tests de régression contrôle que des fonctionnalités existantes ou des anomalies corrigées sont toujours dans le même état après une mise à jour effectuées.

2.3. Types de test A retenir (1/2)



- **Test de maintenabilité** : processus de tester la facilité avec laquelle un produit logiciel peut être modifié pour en corriger les défauts, modifié pour couvrir de nouvelles exigences, modifié pour rendre des maintenances ultérieures plus aisées, ou adapté à un changement d'environnement
- **Test de portabilité** : processus de tester la facilité avec laquelle un produit logiciel peut être transféré d'un environnement matériel ou logiciel vers un autre
- **Test de fiabilité** : processus de tester la capacité d'un produit logiciel à effectuer les fonctions requises dans les conditions spécifiées pour des périodes de temps spécifiées, ou pour un nombre spécifique d'opérations
- **Test d'utilisabilité** : processus de tester la capacité du logiciel à être compris, appris, utilisé et attrayant par/pour l'utilisateur quand il est utilisé dans des conditions spécifiées

2.4. Test de maintenance

Les tests de maintenance sont effectués sur un produit logiciel dont la mise en production est effective

Typologies de maintenance:

- **Modification** : Tests de maintenance corrective et évolutive (planifiées ou urgences)
- **Migration** : Tests de conversion, tests de migration de données
- **Suppression** : Test de migration de données ou d'archivages

Analyse d'impact : L'évaluation de modifications dans les niveaux de documentations de développement, de la documentation de tests et des composants de façon à implémenter la modification d'une exigence spécifique donnée.

L'analyse d'impact peut permettre de :

- Déterminer les exigences à tester pour couvrir les modifications
- Déterminer la charge de tests à modifier et à exécuter
- Anticiper les risques

2. Tester pendant le cycle de vie logiciel QCM (1/3)

1. Comment s'appelle le concept d'introduction d'un petit changement au niveau du programme dont les effets se présentent dans certains tests ..?
 - A. L'introduction de mutations
 - B. Les tests de performance
 - C. Une erreur de mutation
 - D. Le débogage d'un programme
2. La sélection des cas de test pour les tests de régression ..?
 - A. Nécessite des connaissances sur les corrections de bugs et comment ils affectent le système
 - B. Comprend la partie des défauts fréquents
 - C. Comprend la partie qui a subi de nombreuses / récentes modifications du code
 - D. Tous les éléments ci-dessus
3. L'activité de Tests qui est exécutée afin de mettre en évidence les défauts au niveau des interfaces et de l'interaction entre les composants intégrés est :
 - A. tests au niveau système
 - B. tests au niveau intégration
 - C. tests au niveau unitaire
 - D. tests de composants
4. Les défauts trouvés par les utilisateurs sont dus à ..?
 - A. Une mauvaise qualité du logiciel
 - B. Au mauvais logiciel et aux mauvais tests
 - C. La malchance
 - D. Le manque de temps pour tester

2. Tester pendant le cycle de vie logiciel QCM (2/3)

5. Selon les lectures, il existe plusieurs risques à la gestion de planning de votre projet avec un modèle de fiabilité statistique. Il s'agit notamment de:

- A. Les testeurs dépensent plus d'énergie au début du produit à essayer de trouver des bugs que de préparer efficacement le travail restant à faire sur le projet
- B. Les managers ne réalisent pas que l'effort de test est inefficace, en fin de projet, car ils s'attendent à trouver un faible taux de bug, de sorte que le faible taux atteint ne les inquiète pas.
- C. Cela peut augmenter la pression de fin de projet sur les testeurs pour ne pas trouver de bugs, ou pour ne pas reporter de bugs.
- D. Tous les éléments ci-dessus

~~6. Les méthodologies adoptées pendant les tests de maintenance:~~

- ~~A. Test d'étendue et test de profondeur~~
- ~~B. Le retesting~~
- ~~C. Les tests de confirmation~~
- ~~D. Les Tests de "bonne santé"~~

7. Un type de tests fonctionnels, qui étudie les fonctions relatives à la détection de menaces telles que les virus d'intrusions extérieures.

- A. des tests de sécurité
- B. des tests de reprise
- C. des tests de performance
- D. des tests de fonctionnalités

2. Tester pendant le cycle de vie logiciel QCM (3/3)

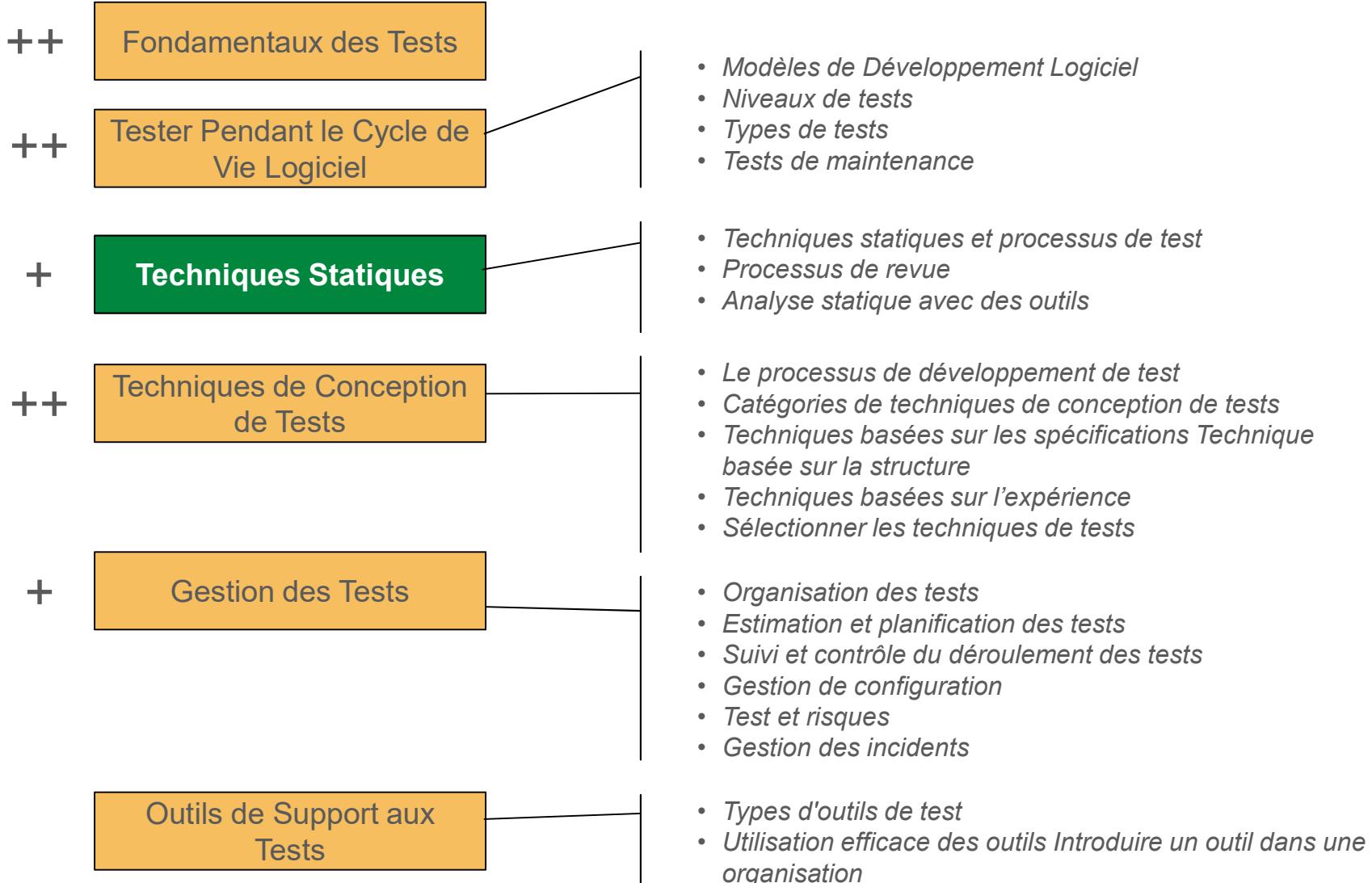
8. Les tests où l'objectif est de soumettre la cible à des degrés divers de charges de travail pour mesurer et évaluer les comportements de performance et sa capacité à continuer à fonctionner correctement pour les différentes charges de travail.

- A. des tests de charge
- B. des tests d'intégration
- C. des tests du système
- D. des tests d'utilisabilité

9. Quelles sont les principales fonctionnalités sur lesquelles il faut se concentrer lorsque vous faites des tests pour des sites web ..?

- A. L'Interaction entre les pages html
- B. La performance côté client
- C. Les aspects de sécurité
- D. Tous les éléments ci-dessus

Contenu de la formation



3.1 Techniques statiques et processus de test

Test dynamique / test statique (1/2)

- A la différence des tests dynamiques qui nécessitent l'exécution du logiciel, les tests statiques consistent en des examens manuels (revue) ou des analyses statiques du code ou de la documentation sans exécution du code
- Les tests statiques peuvent être exécutés bien avant les tests dynamiques et déceler des défauts très tôt dans le cycle de vie d'un projet
- L'objet des revues ne se limite pas au code mais peut être la documentation ou les artefacts de test :
 - les exigences,
 - la stratégie de test,
 - les cas de test,
 - les spécifications fonctionnelles et techniques,
 - les guides utilisateur ...

3.1 Techniques statiques et processus de test

Test dynamique / test statique (2/2)

- Les défauts typiques plus faciles à trouver lors des revues qu'en dynamique sont :
 - non respect des standards,
 - défauts d'exigences,
 - défauts de conception,
 - maintenabilité insuffisante
 - spécifications incorrectes
- Tous les défauts ne peuvent pas être vus en test statique.
Les tests statiques et dynamiques sont complémentaires

3.2 Processus de revue Récapitulatif des revues

	Revue informelle	Relecture technique	Revue technique	Inspection	
Formalisme					
Présence pairs					
Métriques					
Réunion de préparation					
Rapport					



Pas obligatoire mais pas interdit

3.2 Processus de revue

Facteurs de succès des revues

- Objectifs prédéfinis et clairs des revues
- Intervention des personnes adéquates pour la revue
- Les testeurs sont des réviseurs potentiels
- Les défauts trouvés doivent être partagés par tout le monde
- Les aspects personnels et psychologiques sont traités
- La revue ne vise pas les participants mais ce qui a été produit
- Les techniques de revue utilisées sont adaptées aux objectifs, aux types et au niveau de livrable logiciel et aux types et niveau des réviseurs
- Utilisation appropriée des check-list et des rôles
- Des formations sont données sur les techniques de revue
- L'encadrement prévoit du temps pour les revues dans le planning du projet
- L'accent est mis sur l'apprentissage et l'amélioration du processus

3.3 Analyse statique avec des outils

Intérêts des analyses statiques

- Détection très tôt de défauts avant l'exécution des tests
- Un compte rendu très tôt sur des métriques du code ou de la conception tel que la complexité, la non utilisation des normes ...
- Identification de défauts difficilement détectables par des tests dynamiques
- Détection de dépendances anormales et d'inconsistances dans les modèles logiciels
- L'amélioration de la maintenabilité du code et de la conception
- La prévention des défauts si les lacunes sont comblées dès le développement
- Donne un métrique sur la qualité du code à partir de critères pondérés : taux de commentaires, profondeur des appels de modules ...

3.3 Analyse statique avec des outils

Principaux défauts découverts

- Référencement d'une variable avec valeur indéfinie
- Interface inconsistante entre modules et composants
- Variables qui ne sont jamais utilisées ou déclarées de façon incorrecte
- Code non accessible
- Logique absente et erronée
- Constructions trop compliquées
- Violation des standards de programmation
- Vulnérabilité de sécurité
- Violation de syntaxe dans le code et les modèles logiciels

3. Techniques Statiques QCM (1/4)

1. Qui est responsable du pilotage pour la revue afin de s'assurer que les tests soient prêts ..?
 - A. Le Chef de projet
 - B. L'ingénieur des Tests
 - C. le Manager des tests
 - D. Aucune de ces réponses
2. Les défauts typiques qui sont plus faciles à trouver dans les revues de tests que dans les tests dynamiques sont:
 - A. Les écarts par rapport au standard,
 - B. Les défauts d'exigences,
 - C. Les défauts de conception
 - D. La Maintenabilité insuffisante et les spécifications d'interface incorrectes.
 - E. Toutes ces réponses
3. Les revues de tests, les analyses statiques et les tests dynamiques ont le même objectif -
 - A. Identifier les défauts.
 - B. Corriger les défauts.
 - C. 1 et 2
 - D. Aucun

3. Techniques Statiques QCM (2/4)

4. Lequel des énoncés suivants est vrai à propos de la révision formelle ou inspection:
- i. Dirigé par le modérateur formé (pas l'auteur).
 - ii. Aucune préparation pré-réunion
 - iii. processus de suivi formel
 - iv. L'objectif principal est de trouver des défauts
 - A. ii est vrai et i, iii, iv sont faux
 - B. i, iii, iv sont vrais et ii est faux
 - C. i, iii, iv sont faux et ii est vrai
 - D. iii est vrai et I, ii, iv sont faux
5. Les phases du processus de revue formelle sont mentionnées ci-dessous. Rangez-les dans le bon ordre.
- i. la planification
 - ii. La Réunion de revue
 - iii. Le Re-travail
 - iv. les préparations individuelles
 - v Le lancement
 - vi. Le suivi
 - a) i, ii, iii, iv, v, vi
 - b) vi, i, ii, iii, iv, v
 - c) i, v, iv, ii, iii, vi
 - d) i, ii, iii, v, iv, vi

3. Techniques Statiques QCM (3/4)

6. La phase de planification d'une revue formelle comprend les éléments suivants:

- A. Expliquer les objectifs
- B. Choisir les personnes, attribuer les rôles.
- C. Assurer le suivi
- D. les préparations des réunions individuelles

7. Une personne qui documente toutes les questions, les problèmes et les points ouverts qui ont été identifiés lors d'une revue formelle.

- A. le Modérateur
- B. le Scribe
- C. l'Auteur
- D. le Manager

8. Qui sont les personnes impliquées dans une revue formelle :

- i. le manager
- ii. le Modérateur
- iii. le Scribe / L'Enregistreur
- iv. L'Assistant du Manager
 - A. i, ii, iii, iv sont vrais
 - B. i, ii, iii sont vrais et iv est faux.
 - C. ii, iii, iv sont vrais et i est faux.
 - D. i, iv sont vrais et ii, iii sont faux.

9. Que ne peut PAS trouver l'analyse statique ?

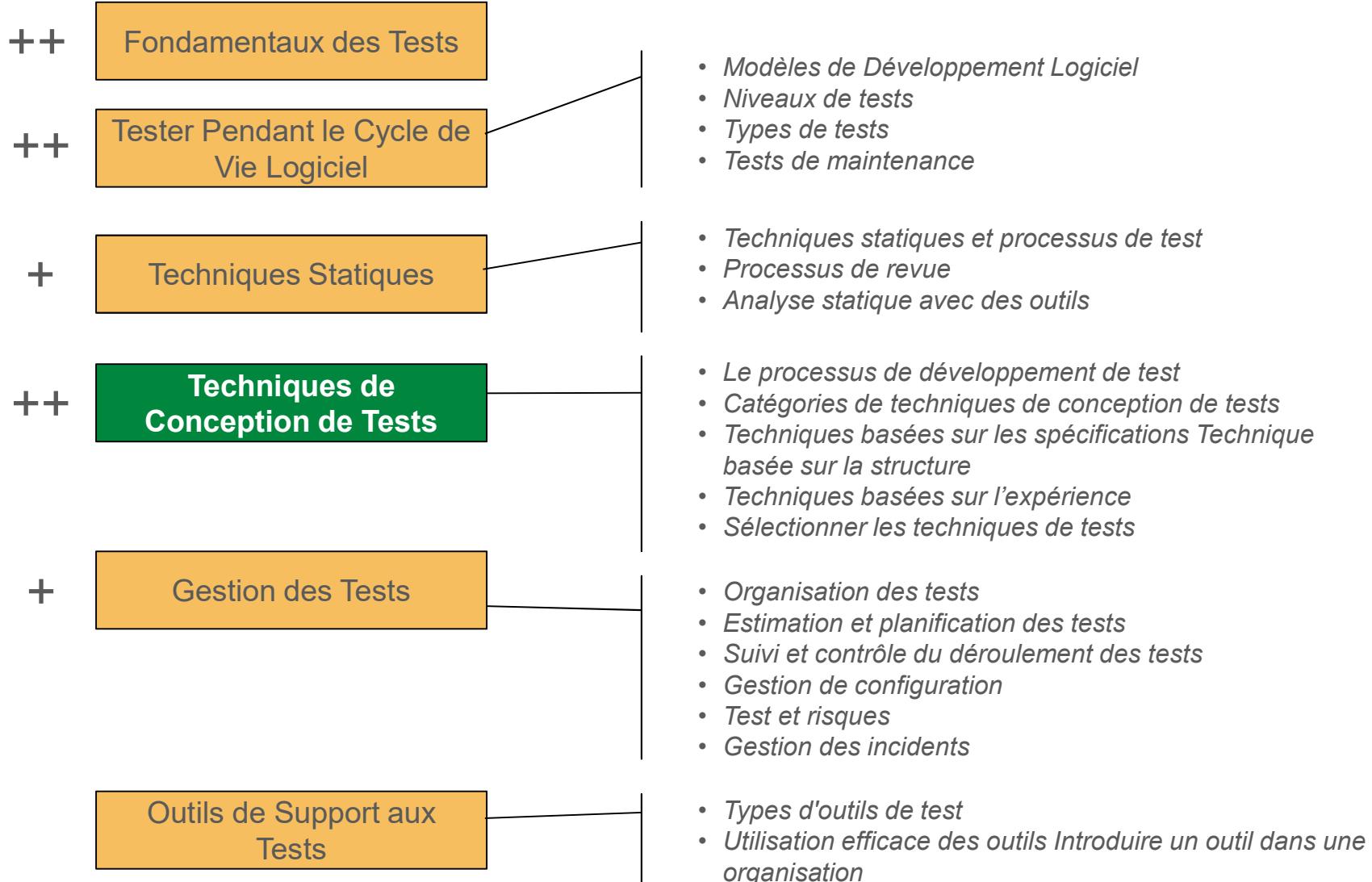
- A. l'utilisation d'une variable avant qu'elle n'ait été définie
- B. un code inaccessible («mort»)
- C. des fuites de mémoire
- D. des violations de limites d'intervalles

3. Techniques Statiques QCM (4/4)

10. L'analyse statique est décrite le mieux comme:

- A. L'analyse des programmes de traitement par lots.
- B. La revue des plans de test.
- C. L'analyse du code de programme.
- D. L'utilisation des tests de boîte noire.

Contenu de la formation



4.1 Le processus de développement de test

Contexte

- Il peut être effectué de manière informelle avec peu ou pas de documentation ou de manière très formelle comme décrit ci-dessous. Cela dépend du contexte des tests comme :
 - la maturité des tests et des processus de développement
 - des contraintes de temps,
 - des exigences de sûreté de fonctionnement ou réglementaires
 - des personnes impliquées
- Pendant la phase d'analyse de conception des tests, la documentation de base des bases des tests est analysée pour déterminer ce qui est à tester, c'est-à-dire à identifier les conditions de test.
- Etablir un lien entre les spécifications / exigences et les conditions de tests permet d'effectuer une analyse d'impact lorsque les exigences changent et de garantir la couverture des exigences.
- Pendant la phase d'analyse de conception des tests, des considérations sur les risques peuvent être prises en compte

4.1 Le processus de développement de test

Conception des tests (1/2)

- Pendant la conception des tests, les cas de tests et données de tests sont créées et spécifiées.
 - Un cas de test est composé
 - d'un ensemble de valeurs d'entrée,
 - de pré conditions d'exécution,
 - de résultats attendus
 - et de post conditions d'exécution,
 - développé pour couvrir certains objectifs de tests et conditions de tests.
 - Les résultats attendus doivent être définis pendant la conception des tests et avant l'exécution des tests

4.1 Le processus de développement de test

Conception des tests (2/2)

- Pendant l'implémentation des tests, les cas de tests sont
 - développés,
 - implémentés,
 - priorisés
 - et organisés dans les spécifications de procédures de test
- La procédure de test spécifie la séquence des actions pour l'exécution d'un test
- Si les tests sont exécutés avec un outil d'exécution des tests, la séquence des actions est spécifiée dans un script de test (qui est une procédure de test automatisée).

4.2 Catégorie de techniques de conception de tests

Contexte

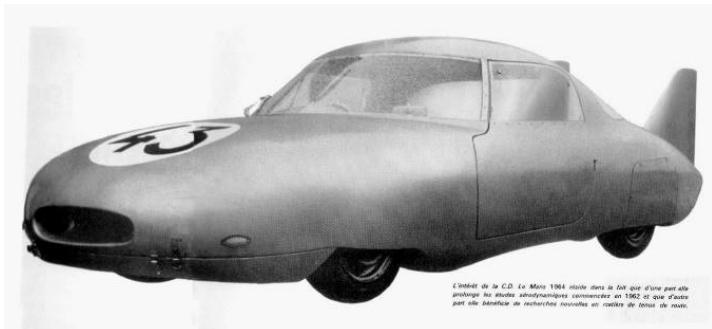
- L'objectif de la technique de conception des tests est d'identifier
 - les conditions de tests,
 - les cas de tests
 - les données de tests.
- Les techniques de conception **boîte noire** (aussi appelées techniques basées sur les spécifications) sont une façon de concevoir les tests en se basant sur **une analyse de la documentation de la base de tests**. Ceci inclut les tests fonctionnels et non fonctionnels.
- Les techniques de conception **boîte blanche** (aussi dites techniques structurelles ou basées sur les structures) sont basés sur **une analyse de la structure du composant ou du système**.
- Les tests boîtes noire et blanche peuvent aussi **s'inspirer de l'expérience des développeurs, des testeurs et des utilisateurs** pour déterminer ce qui doit être testé.

4.2 Catégorie de techniques de conception de tests

Illustration boîte noire / boîte blanche

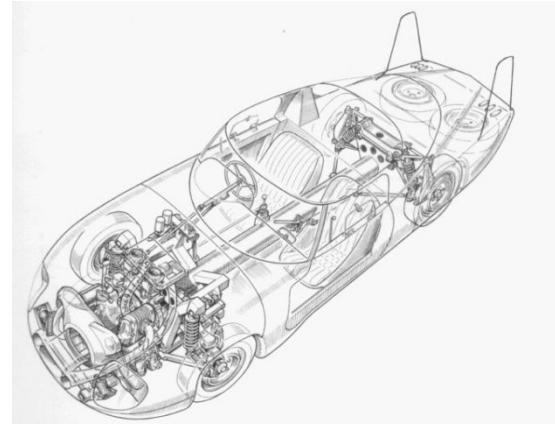
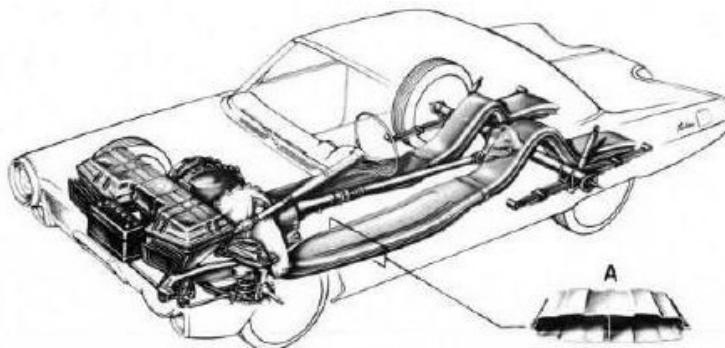
■ Boîte Noire

- Test bâti sans référence au contenu interne de l'objet testé



■ Boîte Blanche

- Test basé sur le contenu interne de l'objet testé



4.3 Techniques basées sur les spécifications

Partitions d'équivalence

- Pour les partitions d'équivalence, les entrées d'un logiciel ou d'un système sont divisées en **groupe qui ont un comportement similaire** et donc un traitement identique
- Les partitions (ou classes) d'équivalence s'appliquent pour :
 - Des données valides ou données acceptées par le logiciel ou le système
 - Des données invalides ou données rejetées
- Des tests peuvent être conçus pour couvrir toutes les partitions valides et invalides.
- Les partitions d'équivalence sont applicables à tous les niveaux de tests
- Elles peuvent être appliquées aux entrées humaines, aux entrées via l'interface du système ou aux paramètres d'interface des tests d'intégration

4.3 Techniques basées sur les spécifications

Partitions d'équivalence : exemple

- Une entreprise met à disposition des salariés un outil de saisie des frais de déplacement en voiture selon les règles suivantes :
 - Seuls les déplacements inférieurs à 50 kms sont remboursés
 - Si le nombre de kilomètres est inférieur à 0 alors affichage du message « Le nombre de kilomètres doit être supérieur à 0 »
 - Si le nombre de kilomètres est supérieur à 50 alors affichage du message « Le nombre de kilomètres doit être inférieur à 50 »



- Déterminer le nombre de partitions d'équivalence et en déduire le nombre de cas de test minimum pour couvrir toutes les partitions

4.3 Techniques basées sur les spécifications

Analyses des valeurs limites

- Les valeurs limites (minimum et maximum) de chaque partition d'équivalence sont propices pour déceler des défauts
- Une valeur limite pour une partition valide est une valeur limite valide.
Idem pour les valeurs limite invalide.
- L'analyse des valeurs limites peut être appliquée à tous les niveaux de tests.
- Cette technique est simple à implémenter et permet de détecter un grand nombre de défauts.
- Cette technique est souvent considérée comme une extension des partitions d'équivalence.

4.3 Techniques basées sur les spécifications Tests par tables de décision (1/2)

- Les tables de décisions sont une bonne façon de déterminer des exigences système contenant des conditions logiques et pour documenter la conception système interne.
- Lors de la création des tables de décision, la spécification est analysée
 - Les conditions d'entrée et les actions sont souvent décrites de façon booléenne : soit vrai soit faux.
 - Les tables de décision contiennent alors les conditions de déclenchement pour toutes les conditions d'entrée et sur les actions résultantes pour chaque combinaison de conditions
- Chaque colonne de la table correspond à une règle de gestion qui définit une combinaison unique de conditions
- La force des tests par les tables de décision est la considération de combinaisons qui autrement n'auraient pas été testées.

4.3 Techniques basées sur les spécifications Tests par tables de décision (1/2)

Rappel du Principe 2 des principes généraux des tests :

- Principe 2 : Tester de façon exhaustive en utilisant par exemple une table de décision ne doit pas être envisagé sauf pour des systèmes très simples ce qui n'est plus le cas aujourd'hui. Il faut privilégier l'analyse des risques et des priorités pour calibrer l'effort de test.

4.3 Techniques basées sur les spécifications

Tests par tables de décision : exemple

- On considère une société d'assurance qui propose un contrat prévoyance aux personnes de 18 à 70 ans et dont les revenus annuels sont compris entre 10 k€ et 100 k€
- Voici la table de décision associée :

Age	Revenus	< 10 k€	Entre 10 k€ et 100 k€ inclus	> 100 k€
< 18 ans		✗	✗	✗
Entre 18 et 70 ans inclus		✗	✓	✗
> 70 ans		✗	✗	✗

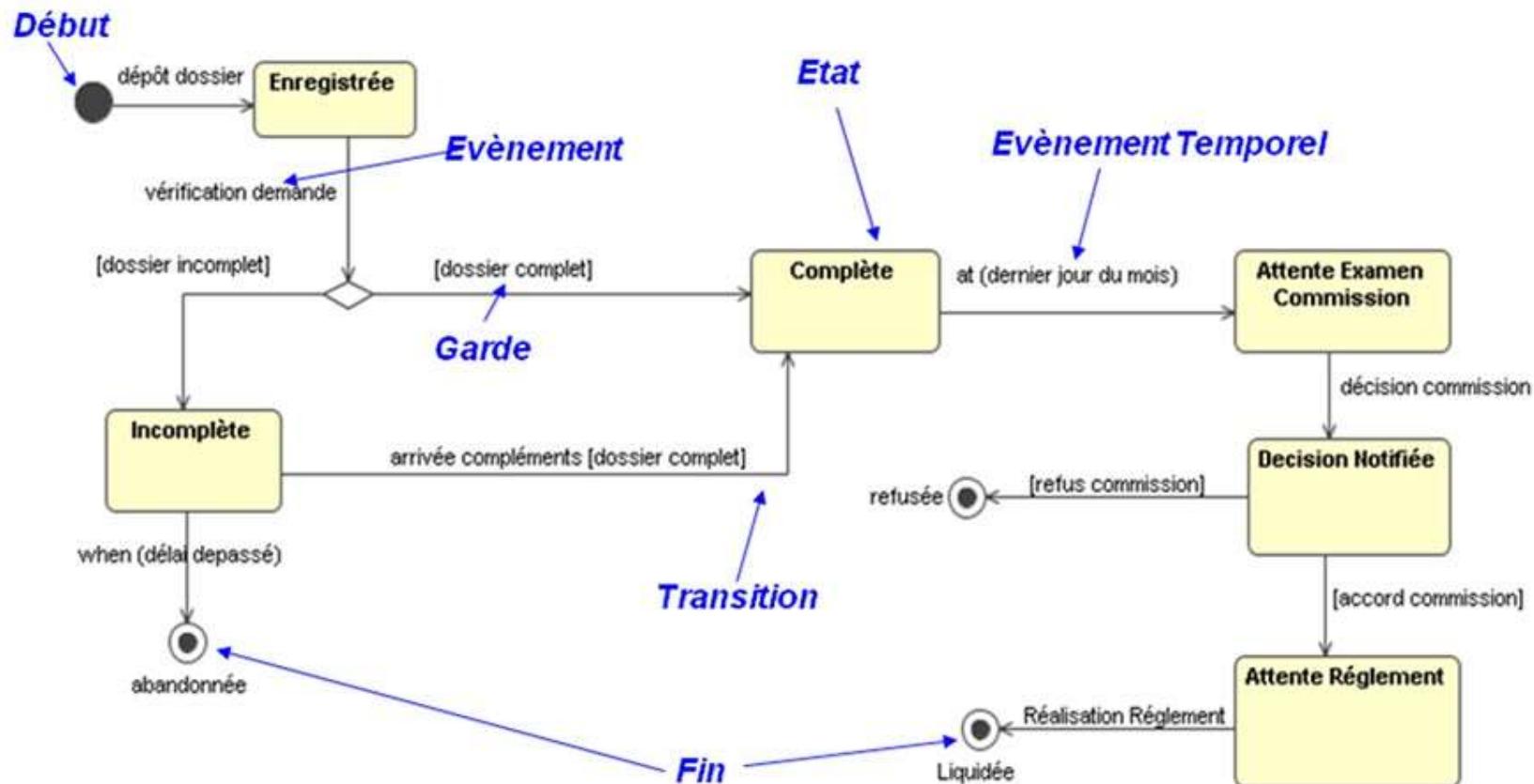
4.3 Techniques basées sur les spécifications

Test de transition d'états

- Un système peut se comporter différemment en fonction des conditions actuelles ou passées (son état).
Cela peut se représenter par un diagramme d'états et de transitions.
- Ce diagramme permet au testeur de visualiser le logiciel
 - en terme d'état,
 - de transitions entre les états,
 - de données d'entrées ou des évènements qui déclenchent des changements d'état (transitions)
 - et des actions qui peuvent résulter de ces transitions.
- Une table des états montre la relation entre les états et les entrées et peut mettre en lumière des transitions invalides
- Les tests de transitions d'états sont fréquemment utilisés dans l'industrie du logiciel embarqué et généralement dans l'automatisation technique.

4.3 Techniques basées sur les spécifications

Test de transition d'états : exemple

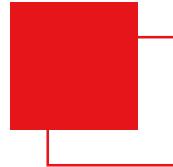


Source Internet Wikimémoires

4.3 Techniques basées sur les spécifications

Test de cas d'utilisation

- Les tests peuvent être élaborés à partir de cas d'utilisation qui décrit l'interaction entre des acteurs, incluant utilisateurs et système, qui produit un résultat ayant une valeur pour l'utilisateur du système ou le client
- Les cas d'utilisation peuvent être décrits à un niveau abstrait (cas d'utilisation métier, indépendant de la technologie, niveau processus métier)
- Chaque cas d'utilisation a des pré conditions qui doivent être remplies pour que le cas d'utilisation soit exécuté avec succès
- Chaque cas d'utilisation se termine par des post-conditions qui sont les résultats observables et l'état final du système après la fin d'exécution du cas d'utilisation. Un cas d'utilisation a généralement un scénario dominant (nominal) et parfois des branches alternatives.
- Les cas d'utilisation décrivent les flux du processus dans un système donc les cas de test qui en résultent permettent de découvrir les défauts dans le flux de traitement pendant l'utilisation réelle du système. Ils sont très utiles pour concevoir des tests d'acceptation avec la participation du client/utilisateur.
- La conception de cas de test à partir des cas d'utilisation peut être combinée avec d'autres techniques de tests basées sur les spécifications

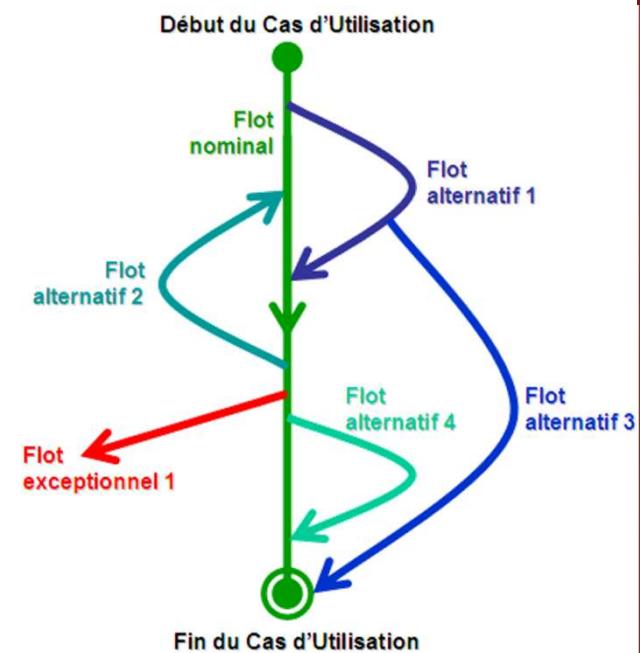


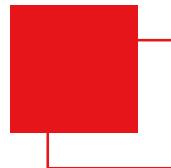
Use-Cases

Flows



- Un Use-Case raconte, sous forme d'histoires, la manière dont l'acteur utilise la Solution et le Système lui répond
- Les interactions entre l'acteur et le Système sont organisées selon des Flows. 3 types de Flows
 - Les **Flows nominaux** → tout se passe normalement, l'objectif est atteint
 - Les **Flows alternatifs** → variations par rapport au flot nominal permettant d'atteindre l'objectif (total ou partiel)
 - Les **Flows exceptionnels** → variations qui correspondent à une situation anormale qui ne permet pas d'atteindre l'objectif

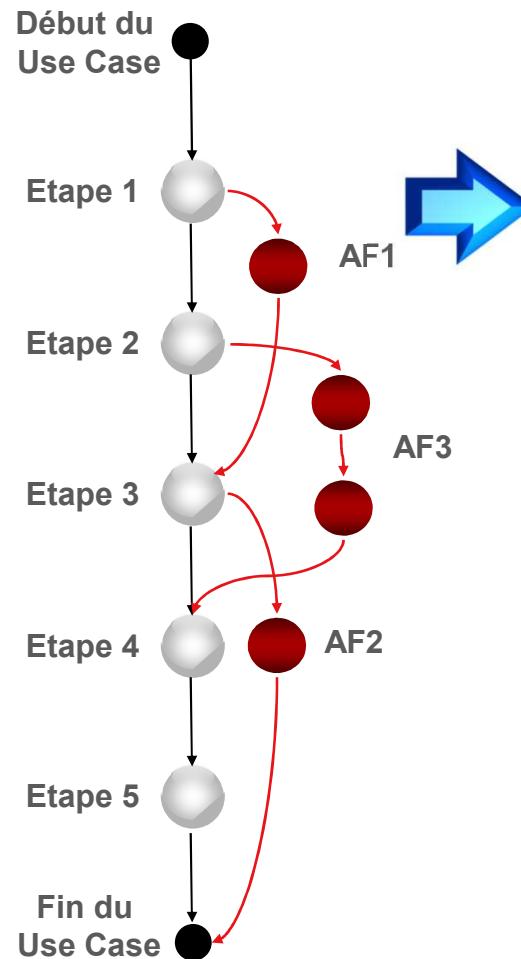




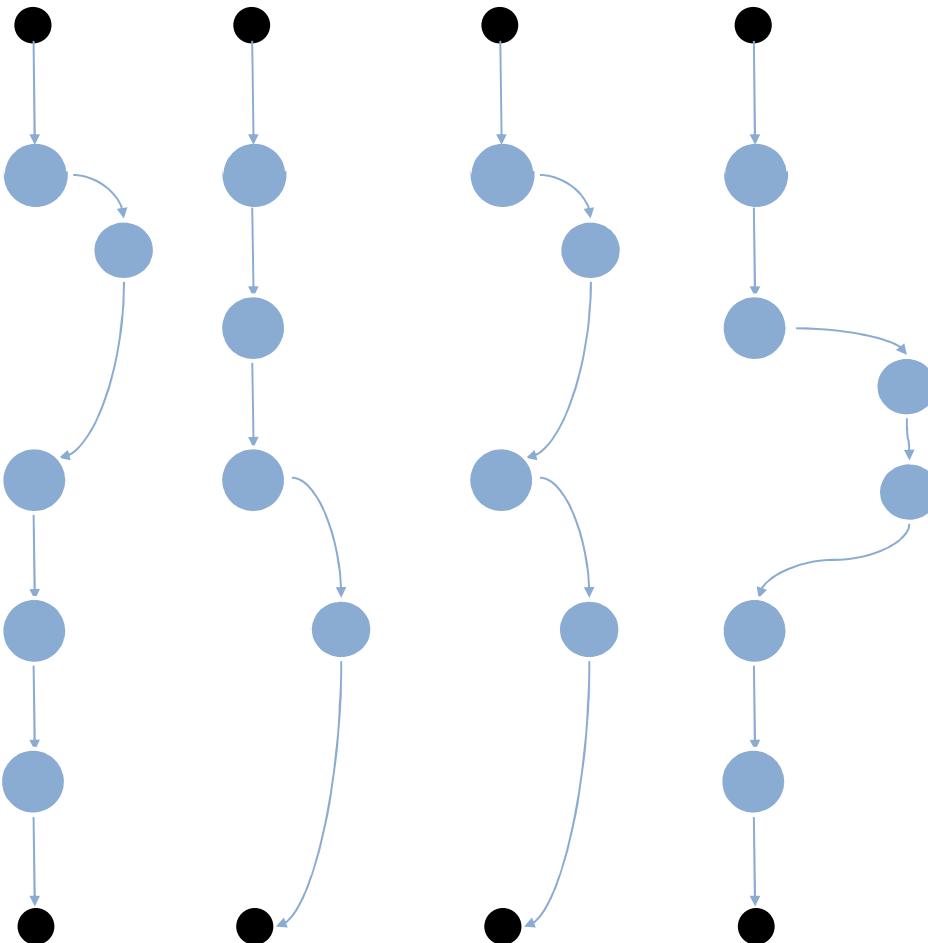
Use-Cases Scénarios

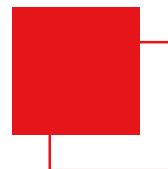


1 Use Case



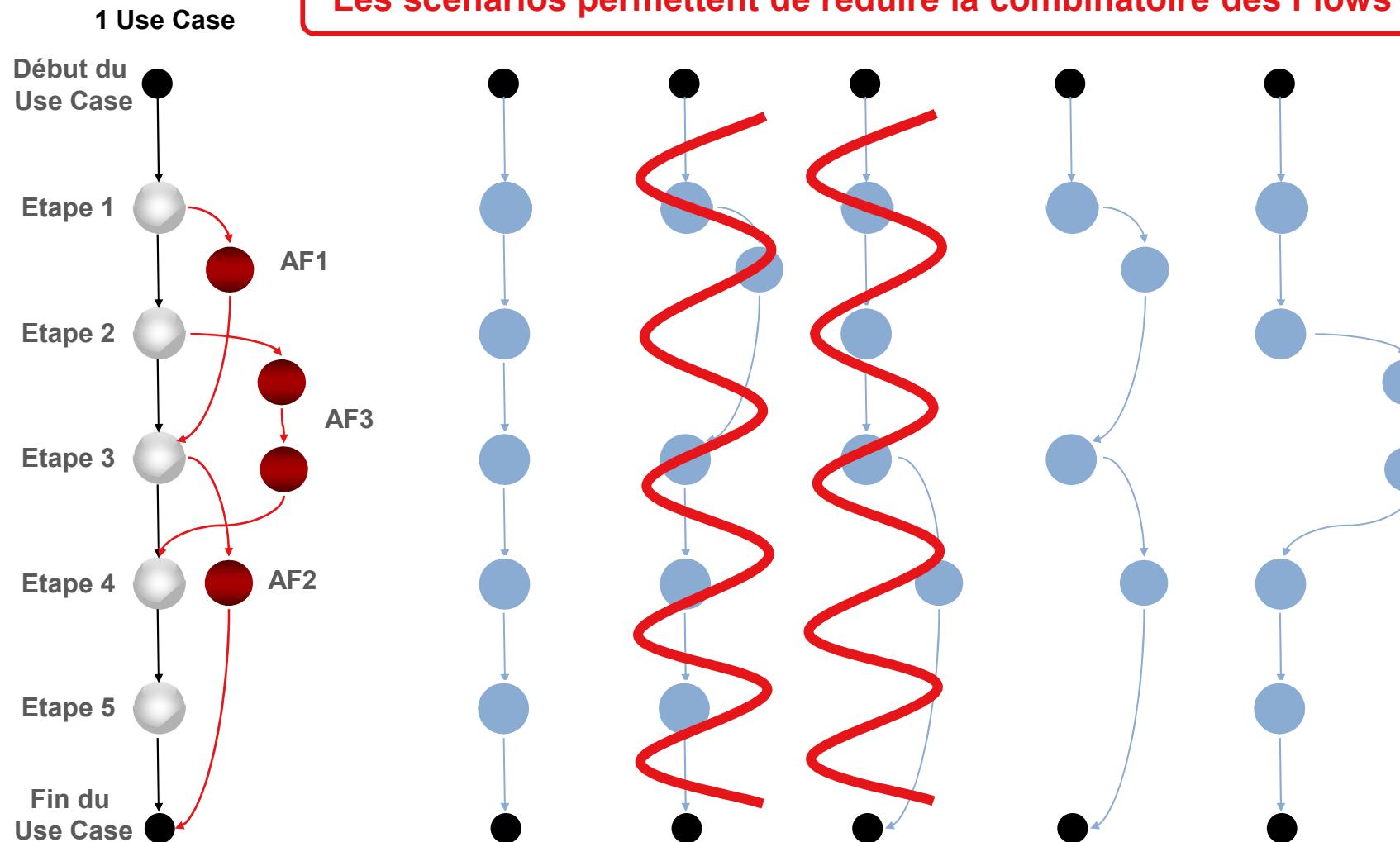
5 Combinaisons





Use-Cases

Scénarios



4.3 Techniques basées sur les spécifications

A retenir



- **Analyse des valeurs limites** : une technique de conception de tests boîte noire dans laquelle les cas de tests sont conçus sur la base des valeurs limites.
- **Test par tables de décisions** : 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 [Veenendaal]
- **Partition d'équivalence** : une portion d'un domaine d'entrée ou de sortie pour laquelle le comportement d'un composant ou système est supposé être le même, basé sur ces spécifications.
- **Test de transition d'état** : 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. Voir aussi *tests N-Switch*
- **Test des cas d'utilisation** : Une technique de conception de tests boîte noire dans laquelle les cas de tests sont conçus pour exécuter des scénarios de cas d'utilisation

4.4 Technique de conception basée sur la structure Test des instructions et couverture

- Dans les tests de composants, **la couverture des instructions est l'évaluation du pourcentage d'instructions exécutables qui ont été exercées par une suite de cas de test.** L'élaboration de cas de test pour exécuter des instructions spécifiques permet d'accroître la couverture des instructions

$$\text{Couverture instructions} = \frac{\text{Nb d'instructions couvertes par des tests}}{\text{Nb instructions exécutables dans le code testé}}$$

4.4 Technique de conception basée sur la structure Test des décisions et couverture

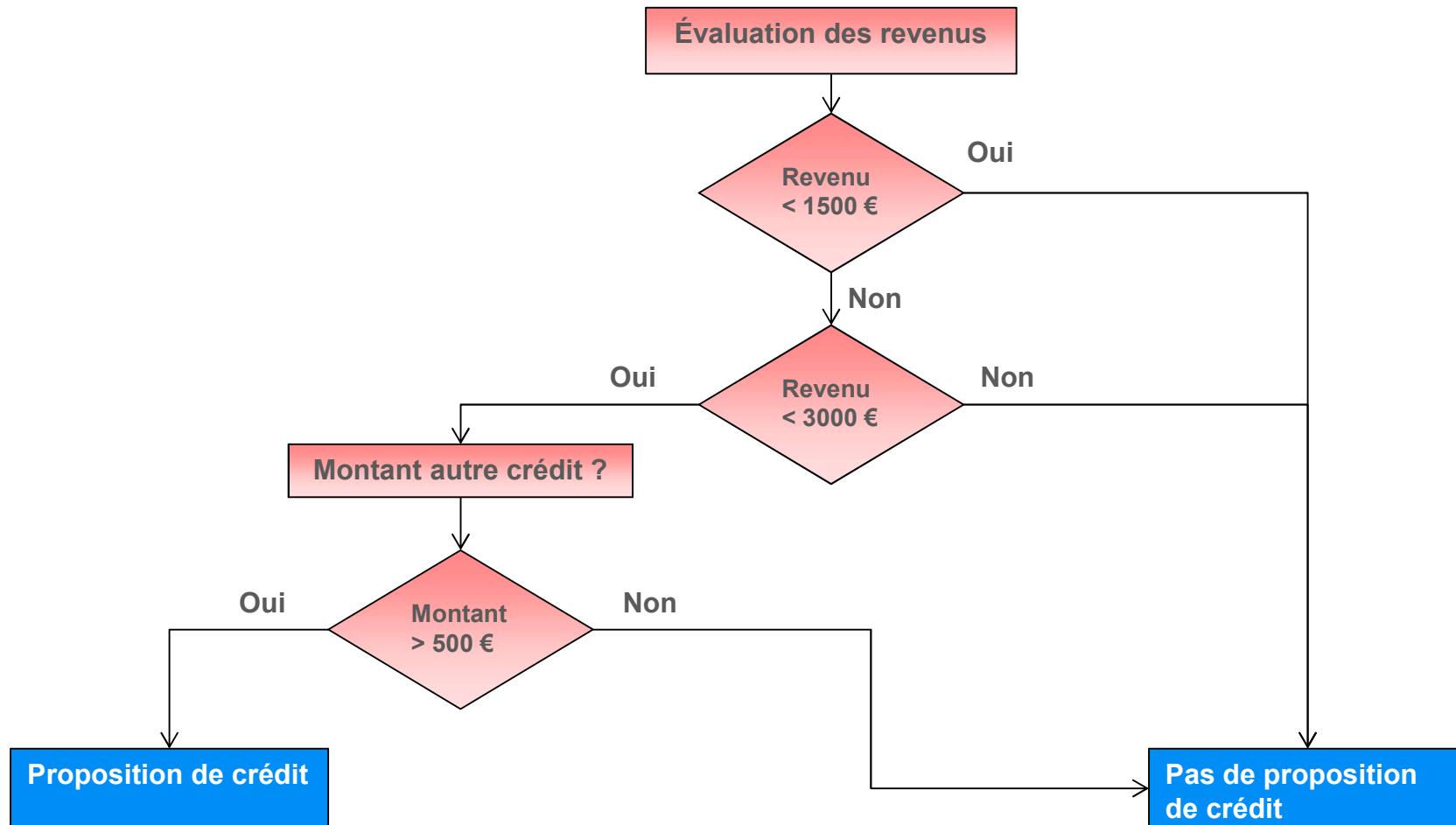
- La couverture des **décisions**, liées aux tests de branches, est l'évaluation des pourcentages de résultats de décisions (par exemple les options vrai ou faux d'une instruction IF)

Les branches forment des points de décision dans le code

$$\text{Couverture décisions} = \frac{\text{Nb décisions couvertes par des tests}}{\text{Nb de résultats de décisions dans le code testé}}$$

- La couverture des décisions est supérieure à la couverture des instructions : une couverture de 100 % des décisions garantit une couverture à 100 % des instructions, mais l'inverse n'est pas vrai

4.4 Technique de conception basée sur la structure Test des décisions et couverture : exemple



Déterminer le nombre de décisions et de branches

4.4 Technique de conception basée sur la structure Décision et conditions

- **Décision** : un point dans un programme où le flot de contrôle a deux ou plus chemins possibles. Un noeud avec deux ou plus liens vers des branches séparées.
- Dans l'expression $(x>0 \mid y >0)$ on trouve deux décisions :
 - Cas passant : $(x>0 \mid y>0)$ vrai
 - Cas non passant : $(x>0 \mid y>0)$ faux
- **Condition** : expression logique qui peut être évaluée à Vrai ou Faux, p.ex. $A>B$.
- Dans l'expression $(x>0 \mid y >0)$ on trouve deux conditions :
 - $x > 0$
 - $y > 0$

$x>0 \mid y >0$

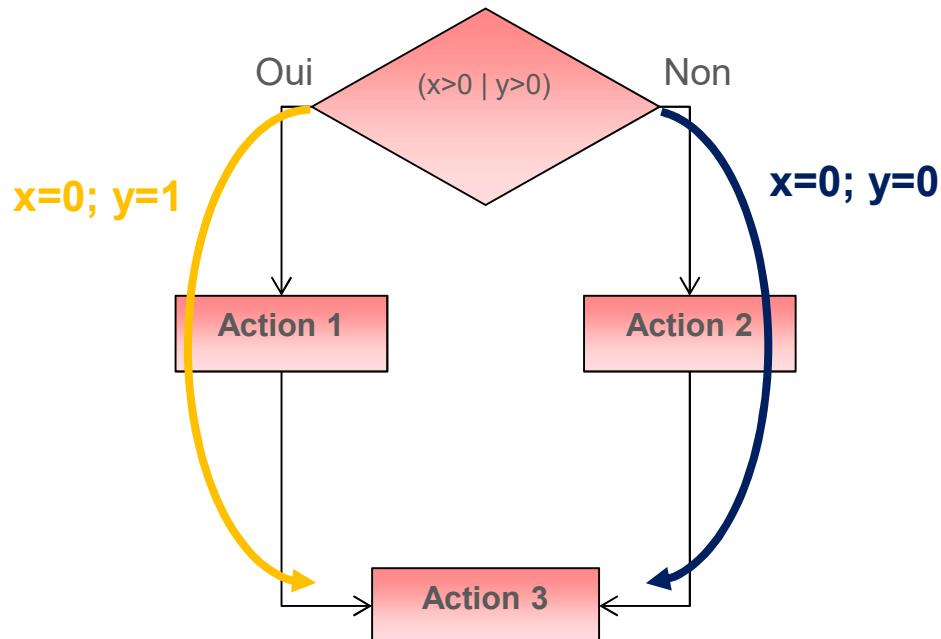
Condition Décision

4.4 Technique de conception basée sur la structure Couverture des décisions

$$\text{Couverture décisions} = \frac{\text{Nb décisions couvertes par des tests}}{\text{Nb décisions dans le code testé}}$$

100% de couverture des décisions implique 100% de couverture des branches et 100% de couvertures des instructions.

- Dans l'expression $(x>0 \mid y >0)$ pour couvrir toutes les décisions, il faut trouver une combinaison pour le cas passant et non passant

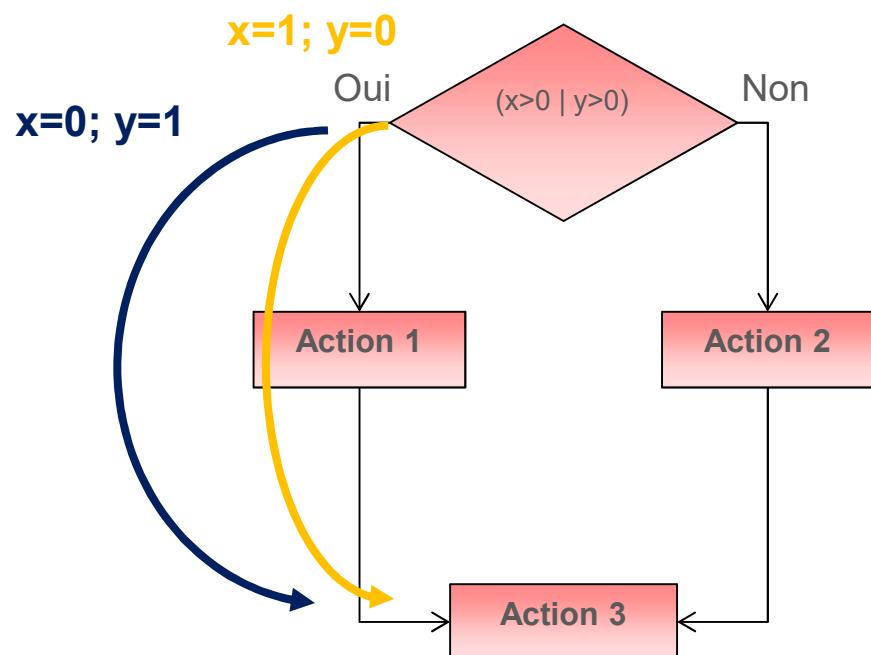


4.4 Technique de conception basée sur la structure Couverture des conditions

$$\text{Couverture conditions} = \frac{\text{Nb conditions couvertes par des tests}}{\text{Nb conditions possibles dans le code testé}}.$$

100% de couverture des conditions nécessite que chaque condition simple dans chaque instruction conditionnelle soit testée en Vrai et en Faux.

- Dans l'expression $(x>0 \mid y >0)$ pour couvrir toutes les conditions, il faut trouver des combinaisons faisant apparaître $x>0$ vrai et faux et $y>0$ vrai et faux



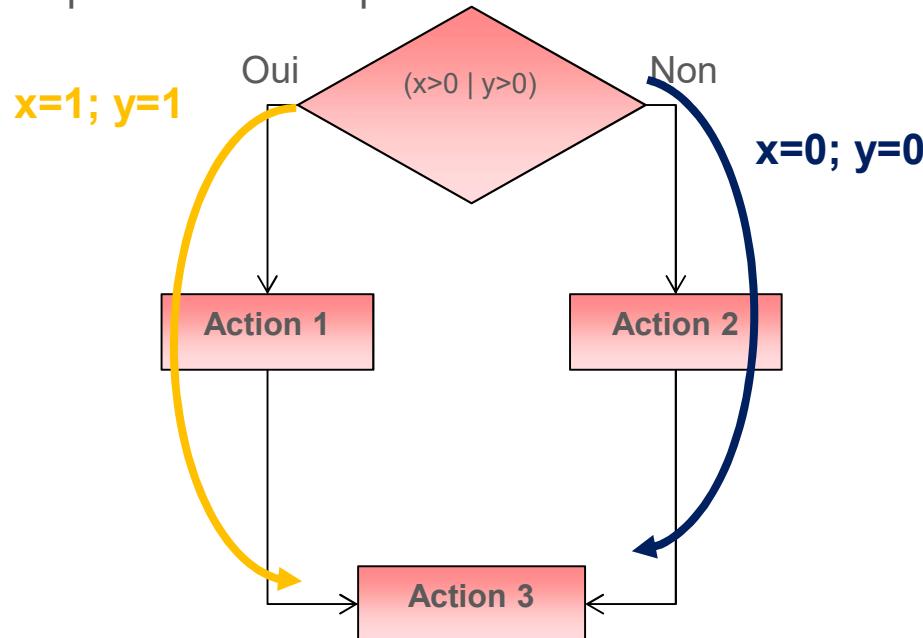
C'est une interprétation possible : on ne teste pas toutes les décisions

4.4 Technique de conception basée sur la structure Couverture des conditions / décisions

Couverture des conditions ET décisions : le pourcentage de tous les résultats de conditions simples qui affectent de façon indépendante les résultats des conditions qui ont été exercés par une suite de cas de tests.

100% de couverture des déterminations des conditions implique 100% de couvertures de conditions et décisions.

- Dans l'expression $(x>0 \mid y >0)$ pour couvrir toutes les conditions et décisions, il faut trouver les combinaisons faisant apparaître $x>0$ vrai et faux et $y>0$ vrai et faux et couvrant le cas passant et non passant

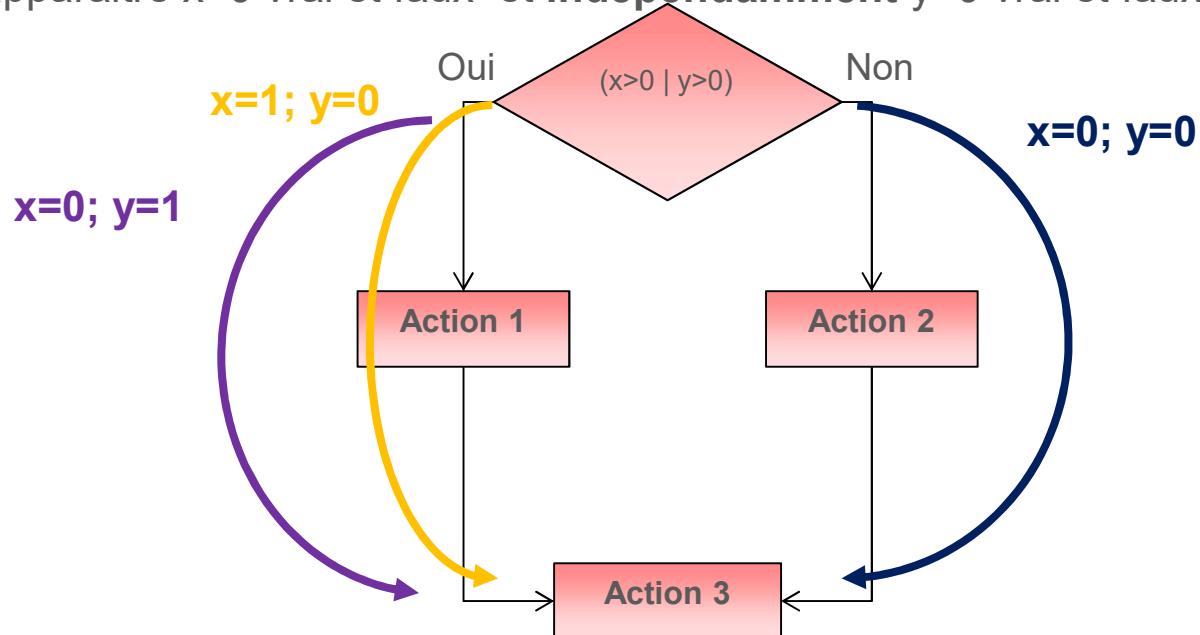


4.4 Technique de conception basée sur la structure Couverture des conditions / décisions modifiées

Couverture des **conditions ET décisions modifiées** : le pourcentage des résultats de toutes les conditions et résultats des décisions qui ont été exercées par une suite de tests.

100% de couvertures des décisions-conditions implique à la fois 100% de couverture des conditions et 100% de couverture des décisions.

- Dans l'expression $(x>0 \mid y >0)$ pour couvrir toutes les décisions et conditions, il faut trouver les combinaisons couvrant le cas passant et non passant puis faire apparaître $x>0$ vrai et faux et **indépendamment** $y>0$ vrai et faux

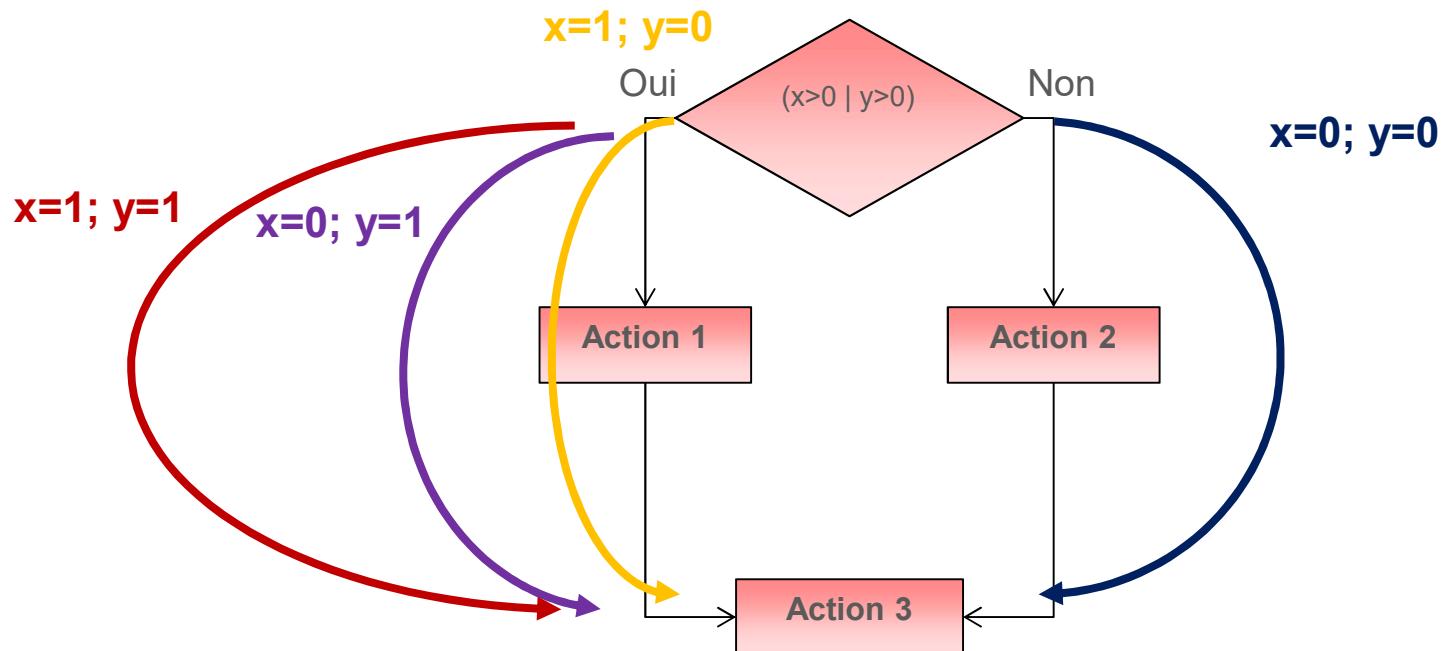


4.4 Technique de conception basée sur la structure Couverture des conditions multiples

Couverture des **conditions multiples (ou composées)** : le pourcentage de combinaison de tous les résultats de combinaisons simples dans une instruction exercées par une suite de tests.

Une couverture des conditions multiples à 100% implique la couverture à 100% des conditions.

- Dans l'expression $(x>0 \mid y >0)$ pour couvrir toutes les conditions multiples, il faut trouver **toutes** les combinaisons faisant apparaître $x>0$ vrai et faux et $y>0$ vrai et faux



4.5 Techniques basées sur l'expérience

Tests exploratoires

- Les 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.
- C'est l'approche la plus utile pour 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. Cela peut servir comme vérification de processus de tests pour s'assurer que les défauts les plus sérieux sont trouvés

4.6 Sélectionner les techniques de tests

Choix d'une technique de tests 1/2

- Le choix d'une technique de tests à utiliser dépend de différents facteurs incluant
 - le type de système
 - les standards réglementaires
 - les exigences client ou contractuelles
 - le niveau de risque, le type de risque
 - les objectifs de test
 - la documentation disponible
 - la connaissance des testeurs
 - Le temps disponible et le budget
 - Le cycle de vie de développement utilisé
 - Les modèles de cas d'utilisation
 - L'expérience sur les défauts découverts précédemment

4.6 Sélectionner les techniques de tests

Choix d'une technique de tests 2/2

- Quelques techniques sont plus applicables que d'autres à certaines situations et niveaux de tests, d'autres sont applicables à tous les niveaux de tests
- Lors de la création de cas de test, les testeurs utilisent généralement une combinaison de techniques de tests incluant les techniques orientés processus, règles et données pour assurer une couverture adéquate de l'objet testé

4. Techniques de Conception de Tests

QCM (1/4)

- 1 La couverture complète des instructions et des branches signifie ..?
 - A. Que vous avez testé chaque instruction dans le programme
 - B. Que vous avez testé chaque instruction et toute branche dans le programme
 - C. Que vous avez testé chaque instruction IF dans le programme
 - D. Que vous avez testé chaque combinaison des valeurs des instructions IF contenues dans le programme
2. Lequel des éléments suivants est l'intrus? ..
 - A. La boîte blanche
 - B. La boîte en verre
 - C. Structurel
 - D. Fonctionnel
3. Les techniques Boîte Blanche sont également appelées :
 - A. Tests structurels
 - B. Tests basés sur la conception
 - C. Technique de devinette d'erreur
 - D. Technique basée sur l'expérience
4. Qu'est-ce qu'une partition d'équivalence (aussi connue comme une classe d'équivalence)?
 - A. Un ensemble de cas de test pour tester les classes d'objets
 - B. Une plage de valeurs d'entrée ou de sortie de sorte qu'un cas de test corresponde à une seule valeur de la plage
 - C. Une plage de valeurs d'entrée ou de sortie de sorte qu'un cas de test corresponde à chaque valeur de la plage
 - D. Une plage de valeurs d'entrée ou de sortie telles que chaque dixième valeur de la plage devient un cas de test.

4. Techniques de Conception de Tests

QCM (2/4)

5. Les cas de test dérivés des cas d'utilisation

- A. sont les plus utiles pour détecter des défauts dans les flux des processus en cours d'utilisation dans le monde réel du système
- B. sont les plus utiles pour détecter des défauts dans les flux du processus durant l'utilisation du système pendant les tests
- C. sont les plus utiles pour masquer des défauts dans le flux des processus en cours d'utilisation dans le monde réel du système
- D. sont les plus utiles pour masquer les défauts au niveau de l'intégration

6. Lequel des éléments suivants ne fait pas partie de la mise en œuvre du test et de la phase d'exécution

- A. Créer des campagnes de tests à partir des cas de test
- B. Exécuter des cas de test manuellement ou en utilisant des outils d'exécution de tests
- C. Comparer aux résultats réels
- D. Concevoir les tests

7. Laquelle des techniques suivante n'est PAS une technique boîte blanche?

- A. Test et couverture des déclarations
- B. Tests et couverture des décisions
- C. La couverture des conditions
- D. L'analyse de la valeur limite

8. Un champ de saisie prend l'année de naissance entre 1900 et 2004.

Les valeurs limites pour tester ce champ sont les suivants:

- A. 0,1900,2004,2005
- B. 1900, 2004
- C. 1899,1900,2004,2005
- D. 1899, 1900, 1901,2003,2004,2005

4. Techniques de Conception de Tests

QCM (3/4)

9. Les Tests de la valeur limite :

- A. correspondent à la même chose que les tests de partitionnement d'équivalence
- B. teste les conditions aux limites, au au-dessous et au-dessus des bords d'entrée et de sortie des classes d'équivalence
- C. Aux combinaisons des Tests des circonstances d'entrée
- D. sont utilisés dans la stratégie de tests de boîte blanche

10. Laquelle des techniques suivantes n'est PAS une technique de boîte noire?

- A. Les tests de transition d'état
- B. LCSAJ (Linear Code Sequence and Jump = Séquence de code linéaire et saut)
- C. Les tests de syntaxe
- D. L'analyse de la valeur limite

11. Les tests minimaux requis pour la couverture d'instructions et la couverture de branches :

Lire P

Lire Q

Si p + q > 100 alors

Imprimer P

fin si

Si p > 50 alors

Imprimer Q

fin si

A. La couverture d'instructions est 2, la couverture de branches est 2

B. La couverture d'instructions est 3 et la couverture de branches est de 2

C. La Couverture d'instructions est 1 et la couverture de branches est de 2

D. La couverture des instructions est 4 et la couverture de branches est 2

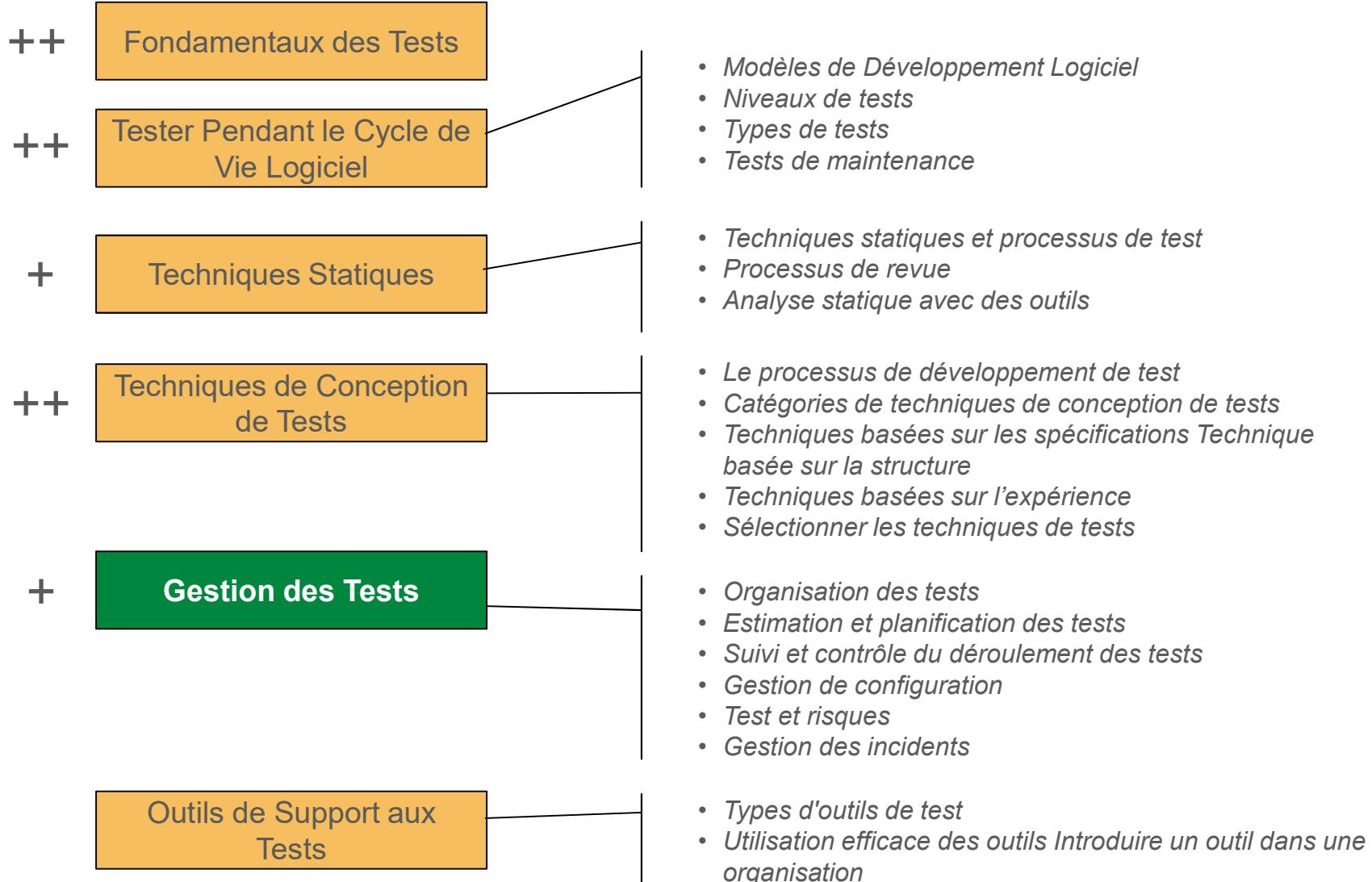
4. Techniques de Conception de Tests

QCM (4/4)

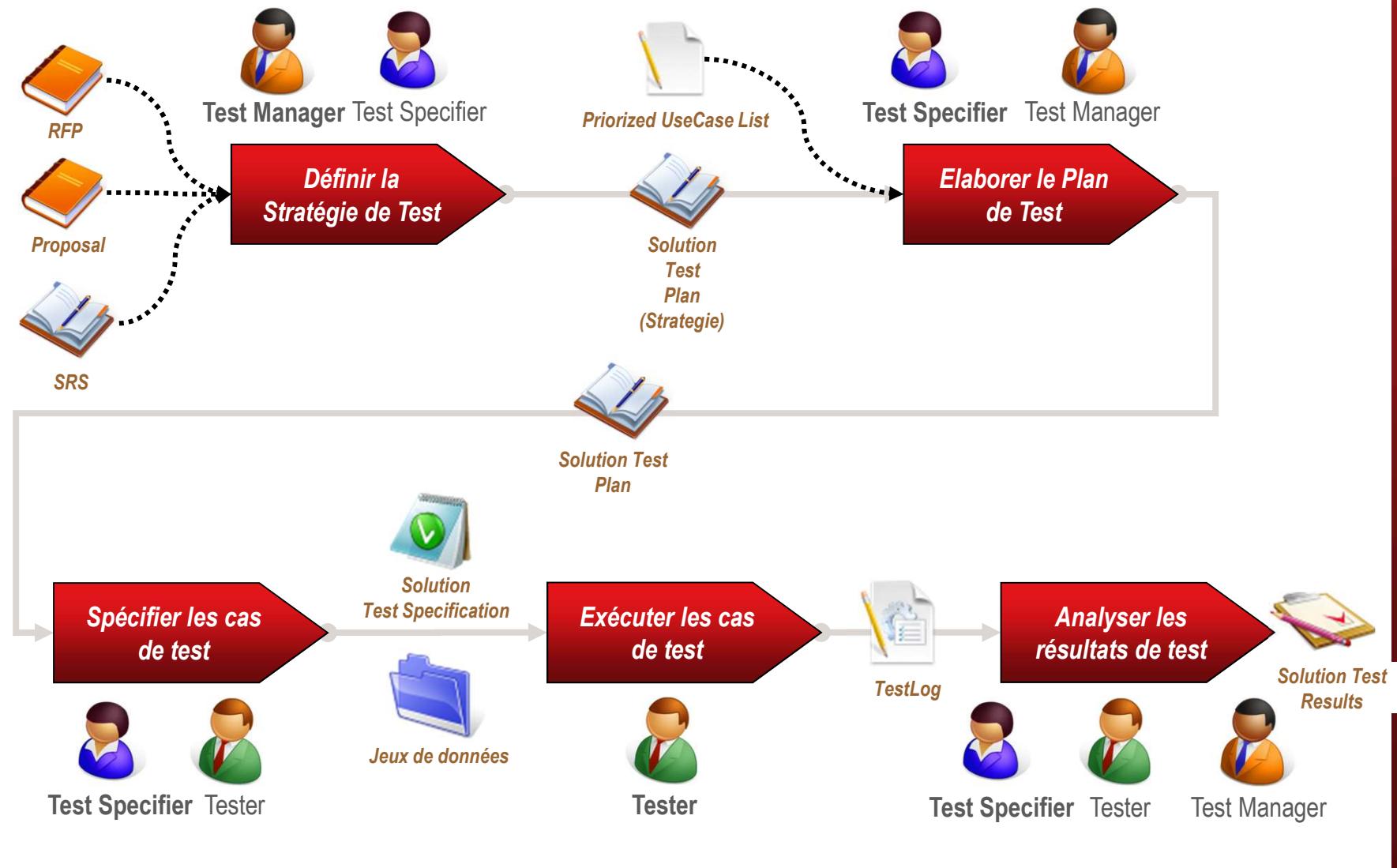
12. Les cas de tests sont conçus pendant:

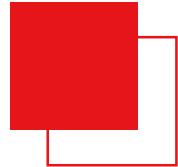
- A. L'enregistrement de test.
- B. La planification de test.
- C. La configuration de test.
- D. La spécification de test.

Contenu de la formation



Vue d'ensemble de la discipline Test (Rappel)





Les rôles



Test Manager

- Assure le pilotage et le suivi de la Qualification
- Définit la Stratégie de Test et veille à son application
- Valide les WorkProducts produits par son équipe



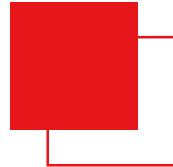
Test Specifier

- Elabore le Plan de Test
- Spécifie en détail les Cas de Test
- Analyse les résultats de test à l'issue des campagnes de test

■ Tester



- Exécute les Tests sur les différentes plates-formes
- Assure un soutien au TestSpecifier au cours de la spécification des Cas de Test et lors de l'analyse des Résultats de Test



Les principaux WorkProducts



Solution Test Plan organisé en deux parties

Livrable

- La Stratégie de Test : description des orientations de la Qualification, de l'organisation mise en œuvre et des procédures nécessaires pour le déroulement de la Qualification
- Le Plan de Test : identification des Scénarios et Cas de Test à dérouler et planification en fonction des itérations (Les Flows de Use Case qui sont traités (Requirements , Analysis, Design, Implementation, Test) sont décidés au niveau global du projet)



Solution Test Specification

- Contient l'ensemble des Cas de test nécessaires pour la Qualification de la Solution. Les Cas de Test sont décrits sous forme d'étapes (action(s), résultats attendus)



Solution Test Results compile les résultats des tests à l'issue de l'analyse

- Cas de Test déroulés
- Issues et Defects identifiés
- Exigences / Use-Case Modules vérifiés

5.1. Organisation de test

Options d'indépendance des testeurs

- Pas de testeurs indépendants, développeurs testant leur propre code
- Testeurs indépendants incorporés à l'équipe de développement.
- Équipe ou groupe de test indépendant au sein de l'organisation, qui réfère au gestionnaire du projet ou aux responsables décisionnaires.
- Testeurs indépendants de l'organisation, de la communauté des utilisateurs et de l'Informatique.
- Spécialistes de test indépendants pour des objectifs spécifiques de test tels que des tests d'utilisabilité, de sécurité informatique ou de certification (qui certifient un produit logiciel par rapport à des normes ou réglementations).
- Testeurs indépendants externes à l'organisation

5.1. Organisation de test

Avantages/ inconvénients de l'indépendance

- **Les avantages de l'indépendance sont principalement les suivants :**

- Des testeurs indépendants voient des défauts différents et d'une autre nature et sont impartiaux.
- Un testeur indépendant peut vérifier les hypothèses faites pendant la spécification et l'implémentation du système.

- **Les inconvénients sont principalement les suivants :**

- Déconnexion vis-à-vis de l'équipe de développement (en cas de totale indépendance).
- Les développeurs perdent le sens de la responsabilité pour la qualité.
- Des testeurs indépendants peuvent constituer un goulet d'étranglement comme dernier point de vérification et être accusés des retards.

5.1. Organisation de test

Tâches des testeurs

- Passer en revue les plans du test et y contribuer.
- Analyser, passer en revue et évaluer, quant à leur testabilité, les exigences utilisateurs, les spécifications et les modèles.
- Créer des spécifications de test.
- Mettre en place l'environnement de test (souvent en coordination avec l'administration système et la gestion de réseau).
- Préparer et obtenir les données de test.
- Implémenter des tests à tous les niveaux, exécuter et consigner les tests, évaluer les résultats et documenter les écarts vis-à-vis des résultats attendus.
- Employer les outils d'administration ou de gestion des tests et les outils de surveillance des tests en fonction du besoin.
- Automatiser les tests (éventuellement avec l'aide d'un développeur ou d'un expert en automatisation de test).
- Mesurer les performances des composants et systèmes (si pertinent).
- Passer en revue les tests développés par d'autres.

5.1. Organisation de test

Tâches des responsables de tests

- Coodonner la stratégie et le plan du test avec le chef de projet et d'autres acteurs.
- Établir ou réviser une stratégie de test pour le projet ainsi qu'une politique de test pour l'organisation.
- Apporter le point de vue du test aux autres activités du projet, comme la planification de l'intégration.
- Planifier les tests – en considérant le contexte et en comprenant les objectifs et les risques – y compris la sélection des approches de test, l'estimation du temps, de l'effort et des coûts du test, l'acquisition des ressources, la définition des niveaux de test, les cycles, l'approche et les objectifs ainsi que la planification de la gestion d'incidents.
- Démarrer la spécification, la préparation, l'implémentation et l'exécution des tests ainsi que surveiller et contrôler l'exécution.
- Adapter le planning en fonction des résultats et de l'avancement du test (parfois documenté dans un rapport d'état d'avancement) et entreprendre les actions nécessaires pour résoudre les problèmes
- Mettre en place une gestion de configuration adéquate du logiciel de test à des fins de traçabilité.
- Introduire des mesures appropriées pour mesurer l'avancement du test et évaluer la qualité du test et du produit
- Décider ce qui devrait être automatisé, à quel degré et comment.
- Sélectionner les outils pour aider le test et organiser la formation des testeurs à l'usage des outils.
- Décider de la mise en œuvre de l'environnement de test.
- Établir des rapports de synthèse de test à partir des informations recueillies pendant le test.

5.2. Estimation et planification des tests

Stratégie de test, Approche de test

■ Stratégie de test

Un document de haut niveau définissant pour un programme, les niveaux de tests à exécuter et les tests dans chacun des niveaux (pour un ou plusieurs projets)

■ Approche de test

Implémentation de la stratégie pour un projet spécifique. Cela inclut typiquement :

- les décisions prises qui sont basées sur les objectifs du projet (de test) et les évaluations de risques effectuées
- les points de départ des processus de test
- les techniques de conception à utiliser
- les types de tests à exécuter

5.2. Estimation et planification des tests

Critères d'entrée

Les critères d'entrée définissent les éléments nécessaires au commencement de la phase de test.

Voici quelques critères qui peuvent entrer en compte :

- Disponibilité et préparation de l'environnement de test
- Préparation des outils de tests dans l'environnement de test
- Disponibilité de code testable
- Disponibilité des jeux de données

5.2. Estimation et planification des tests

Critères de sortie

Les critères de sortie définissent les conditions nécessaires à la sortie de la phase de test. Il peut s'agir de critères de succès ou d'échec.

Voici quelques critères qui peuvent entrer en compte :

- 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.

5.3 Suivi et contrôle du déroulement des tests

Suivi de l'avancement des tests

Les mesures habituelles de test 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.

5.3 Suivi et contrôle du déroulement des tests

Reporting des tests

Il comprend les activités suivantes :

- Ce qui s'est passé pendant une phase de test, comme les dates où les critères de sortie ont été atteints.
- Les informations et mesures analysées pour étayer les recommandations et décisions pour de futures actions, comme une évaluation des défauts restants, les avantages économiques de tests prolongés, les risques non couverts et le niveau de confiance dans le logiciel testé.

Le descriptif d'un rapport de synthèse de test se trouve dans le guide « Standard for Software Test Documentation » (IEEE Std 829-1998).

Des mesures devraient être recueillies pendant et à la fin d'un niveau de test dans le but d'évaluer :

- L'adéquation des objectifs du test avec ce niveau de test.
- L'adéquation des approches du test empruntées
- L'efficacité du test par rapport à ses objectifs

5.3 Suivi et contrôle du déroulement des tests

Exemple de reporting et d'indicateurs

Voici un listing de reporting qui ont un sens sur la majorité des projets :

Suivi de l'avancement de la conception des tests :

- Nombre de cas de test conçus / Nombre de cas de tests à concevoir sur une période
- Suivi de couverture des exigences attendue par les tests

Suivi de l'avancement de l'exécution des cas de test :

- Nombre de cas de test exécutés / Nombre de cas de tests prévus sur une période
- Taux de couverture des exigences : pourcentage de cas de test passant / cas de tests total par exigence
- Répartition de cas de test par statut d'exécution sur une période
- Répartition de cas de test par statut d'exécution et par priorité (ou priorité de l'exigence associée) sur une période
- Répartition des anomalies par exigences
- Taux de régression sur une version

Suivi des bonnes pratiques projets :

- Pourcentage d'exigences non couvertes
- Pourcentage d'anomalies non liées à des tests
- Pourcentage de test non exécutés

5.4. Gestion de configuration

Définition et lien avec le test

- L'objectif de la gestion de configuration est d'établir et de maintenir l'intégrité des produits (composants, données et documentation) du logiciel ou du système durant le cycle de vie du projet et du produit.
- Dans le cadre d'un projet de test, la gestion de la configuration se traduira par un ensemble de méthodes et procédure visant à
 - s'assurer que l'ensemble des éléments d'un produit logiciel ont été identifiés et qu'il sont soumis à un contrôle de version
 - assurer la traçabilité entre les différents éléments du logiciel et faire le lien avec les documents
- Cette gestion de configuration va déterminer un ensemble de procédures, d'acteurs, de rôles, de livrables et d'indicateurs de suivi permettant
 - D'établir les documents liés à une version
 - De mettre à jour l'ensemble du référentiel suite à des changements
 - De permettre le retour arrière sur des versions précédentes

5.4. Gestion de configuration

A retenir



■ Gestion de configuration

Une discipline appliquant une direction et surveillance technique et administrative pour : identifier et documenter les caractéristiques fonctionnelles et physiques d'un élément de configuration, contrôler les modifications de ces caractéristiques, enregistrer et informer des modifications et états d'implémentation, et vérifier la conformité avec des exigences spécifiées [IEEE 610]

■ Contrôle de version

Un élément de la gestion de configuration, consistant en l'évaluation, la coordination, l'approbation ou la désapprobation, et l'implantation de modifications des éléments de configuration après l'établissement de leur identification de configuration [IEEE 610]

5.5. Tests et Risques

Définitions

- Le risque est un facteur qui pourrait résulter dans des conséquences futures négatives. Il est généralement exprimé comme un impact et une conséquence.
- Principe de précaution
- 4 actions possibles face aux risques :
 - Prévention (agir sur la probabilité)
 - Protection (agir sur l'impact)
 - Délégation (assurance)
 - Acceptation du risque (aucune mesure)

Activités d'une approche basée sur les risques :

- estimer (et ré-estimer régulièrement) ce qui peut faillir (les risques)=> **Analyser**
- déterminer quels sont les risques importants à couvrir=>**Prioriser**
- entreprendre des actions pour couvrir ces risques=>**Agir**

Lignes d'action possibles :

- déterminer les techniques de test à employer,
- déterminer le volume du test à effectuer.
- définir les priorités à affecter aux tests dans le but de trouver les défauts critiques le plus tôt possible.
- déterminer si des activités ne faisant pas partie du test pourraient aider à réduire les risques (par exemple, une formation fournie aux développeurs inexpérimentés).

5.5. Tests et Risques

Risques Liés au projet

- Facteurs organisationnels :
 - Manque de savoir-faire et de personnel ;
 - Problèmes de personnel;
 - Problèmes politiques, tels que
 - problèmes dus au fait que des testeurs ne communiquent pas leurs besoins et les résultats du test ;
 - incapacité à suivre les informations recueillies pendant le test et les revues (par exemple, ne pas améliorer les pratiques de développement et de test).
 - Attitude ou attentes inappropriées vis-à-vis du test (par exemple, ne pas apprécier la valeur de la découverte de défauts durant le test).
- Problèmes techniques :
 - Problèmes pour définir des exigences correctes
 - La mesure selon laquelle les exigences seront satisfaites en fonction de contraintes existantes
 - Environnement de test indisponible
 - Conversion des données tardives ; planning de migration et de développement, conversion des données de test / outils de migration
 - Qualité de la conception, du code et des tests
- Problèmes d'acquisition :
 - Défaillance d'une tierce partie
 - Problèmes contractuels.

5.5. Tests et Risques

Risques liés au produit

- Livraison d'un logiciel défectueux.
- Possibilité qu'un logiciel ou système entraîne des dommages à des personnes ou à des entreprises.
- Caractéristiques logicielles de moindre qualité (par exemple, fonctionnalité, sécurité, fiabilité, utilisabilité et performances).
- Intégrité et qualité des données de faible qualité (par exemple en raison de problèmes liés à leur migration, à leur conversion, à leur transport, à un non respect des standards)
- Logiciel n'offrant pas les fonctionnalités voulues.

5.6. Gestion des incidents

Objectifs

Les tests permettent d'établir des écarts entre l'attendu et le constaté. Ces écarts permettent d'établir des incidents et ces incidents seront ensuite analysés pour déterminer s'il s'agit de défauts ou non.

Pour pouvoir gérer efficacement l'analyse et la résolution des incidents, un processus doit être mis en place. Ce processus peut comprendre :

- des acteurs aux responsabilités différentes
- Un rapport d'incidents indiquant les informations obligatoires pour qualifier l'incident
- Un workflow et des tâches de résolution de l'incident (par correction ou annulation par exemple)

5.6. Gestion des incidents

Rapport d'incidents

Les rapports d'incidents peuvent avoir les objectifs suivants :

- Fournir aux développeurs et aux autres parties un retour sur le problème concerné pour en permettre l'identification, la localisation et la correction nécessaire.
- Fournir aux responsables du test le moyen de suivre la qualité d'un système sous test et l'avancement du test.
- Fournir des idées pour l'amélioration du processus de test.

Les détails du rapport d'incident peuvent inclure :

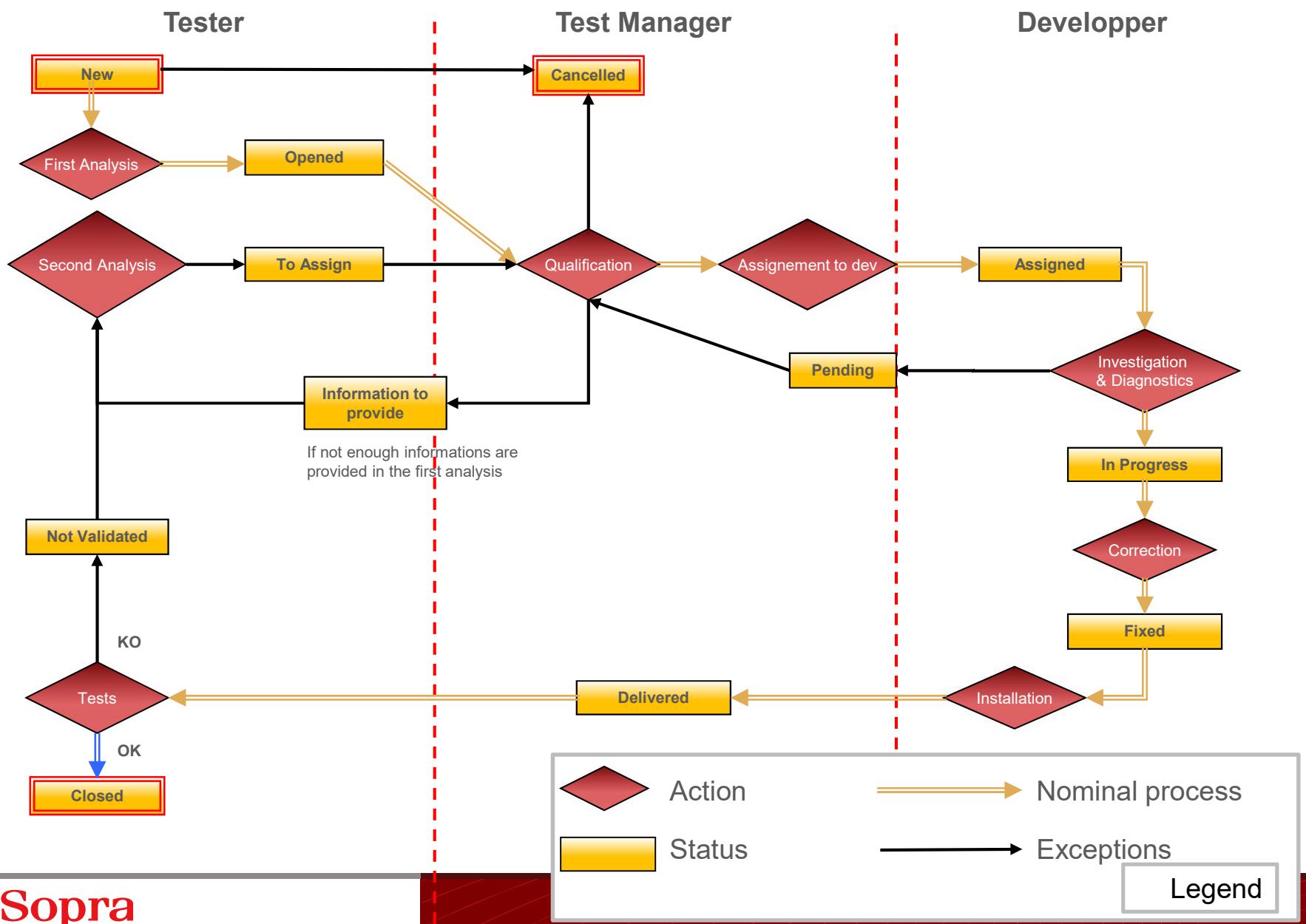
- Date de l'incident, organisation faisant part de l'incident, auteur
- Résultats attendus et effectifs.
- Identification ou élément de configuration du logiciel ou système.
- Processus du cycle de vie du logiciel ou du système au cours duquel l'incident a été observé
- Description de l'anomalie pour permettre son élimination, y compris les traces, extraits de la base de données ou captures d'écrans
- Degré de l'impact sur les intérêts des détenteurs d'enjeux.
- Sévérité de l'impact sur le système.
- Urgence ou priorité de la correction.
- État de l'incident (par exemple, ouvert, soumis, doublon, en attente de correction, corrigé en attente de test de confirmation, clôturé).
- Conclusions et recommandations.
- Problèmes globaux, comme d'autres parties pouvant être impactées par une modification résultant de l'incident.
- Historique des modifications, telles que la séquence des actions entreprises par des membres de l'équipe du projet afin de localiser l'incident, de corriger et d'en confirmer la correction.
- Références, incluant l'identité du cas de test ou la spécification qui a permis d'identifier le problème

5.6. Gestion des incidents

Bilan d'incidents

- Un bilan des incidents peut être créé, il fournit un regard sur un groupe d'incident en mettant l'accent sur des caractéristiques communes permettant de mettre en évidence des informations et de prendre des décisions.
- Voici quelques exemples de bilan d'incidents :
 - Liste des anomalies et des exigences attachées
 - Liste des incidents et de leur version
 - Liste des anomalies et critères de sévérité, complexité
 - Taux d'anomalies de régression

5.6. Gestion des incidents Cycle de vie / Workflow



5.6. Gestion des incidents

A retenir



- **Gestion des incidents** : le processus de reconnaissance, d'investigation, d'action et de traitement des incidents. Il implique l'enregistrement des incidents, leur classification et l'analyse de leur impact.
- **Rapport d'incident** : un document rendant compte de tout événement apparaissant pendant les tests et qui requiert une vérification.
- **Rapport d'anomalie** : un document fournissant une information sur un défaut dans un composant ou système qui peut conduire le composant ou le système à ne pas exécuter les fonctions requises.

5. Gestion des tests QCM (1/4)

1. Quelle est la meilleure définition de la complétude des tests :
 - A. Vous avez découvert chaque bug du programme.
 - B. Vous avez testé chaque déclaration, branche, et combinaison de branches au niveau du programme.
 - C. Vous avez terminé tous les tests du plan de tests.
 - D. Vous avez atteint la date de livraison prévue.
2. Que faire si le projet n'est pas assez grand pour justifier des tests approfondis ..?
 - A. Utiliser l'outil d'automatisation de test
 - B. Utiliser l'analyse fondée sur les risques pour savoir quelles parties doivent être testées
 - C. Les deux (A et B)
 - D. Aucune de ces réponses
3. Les conséquences importantes de l'impossibilité de faire des tests complets sont ..?
 - A. Nous ne pouvons jamais être certain que le programme est sans bug
 - B. Nous n'avons pas de point d'arrêt définitif des tests, ce qui rend plus facile chez certains managers le fait de demander à faire que peu de tests
 - C. Nous n'avons pas de réponse précise à apporter quant au temps que les tâches de test doivent prendre, parce que chaque tâche prend du temps et ce temps pourrait être consacré à d'autres tâches de grande importance
 - D. Tous les éléments ci-dessus
4. Qu'entend-on par «Avoir à dire NON» ..?
 - A. Non, le problème ne vient pas des testeurs
 - B. Non, le logiciel n'est pas prêt pour la production
 - C. Les réponses A et B
 - D. Aucune de ces réponses

5. Gestion des tests QCM (2/4)

5. Lequel des éléments suivants ne fait pas partie de la gestion de configuration ..?
 - A. Le dénombrement des statuts des éléments de configuration
 - B. L'audit de conformité à la norme ISO9001
 - C. L'identification des versions de test
 - D. L'enregistrement des modifications de la documentation au fil du temps
 - E. L'accès contrôlé à la bibliothèque
6. Les livrables de la phase de conception des tests incluent ce qui suit, sauf:
 - A. Les données de test
 - B. Le plan de données de test
 - C. Le rapport de synthèse des tests
 - D. le plan de la procédure de test
7. Lequel des éléments suivants n'est pas déterminé dans la phase de planification des tests ?
 - A. Les jalons et les livrables
 - B. Le Matériel et les logiciels
 - C. Les critères d'entrée et de sortie
 - D. Les types de cas de test
8. Quels sont les 2 principaux éléments pris en compte avec l'analyse des risques?
 - A. La probabilité que l'événement négatif se produise
 - B. La perte potentielle ou l'impact associé à l'événement
 - C. Les deux (A et B)
 - D. ni A ni B

5. Gestion des tests QCM (3/4)

9. Que faire si les exigences changent constamment ?

- A. Collaborer avec les intervenants du projet dès le début pour comprendre comment les exigences pourraient changer afin que les plans de tests alternatifs et les stratégies de tests soient élaborés à l'avance, si possible.
- B. Négocier pour permettre uniquement les nouveaux besoins faciles à mettre en œuvre dans le cadre du projet , tout en passant à de nouvelles exigences plus difficiles lors des futures versions de l'application.
- C. A et B
- D. Aucune de ces réponses

10. Les risques de projet incluent lequel des éléments suivants

- A. des facteurs organisationnels
- B. des caractéristiques logicielles faibles
- C. un logiciel livré sujet à l'erreur
- D. des logiciels qui ne remplissent pas les fonctions prévues

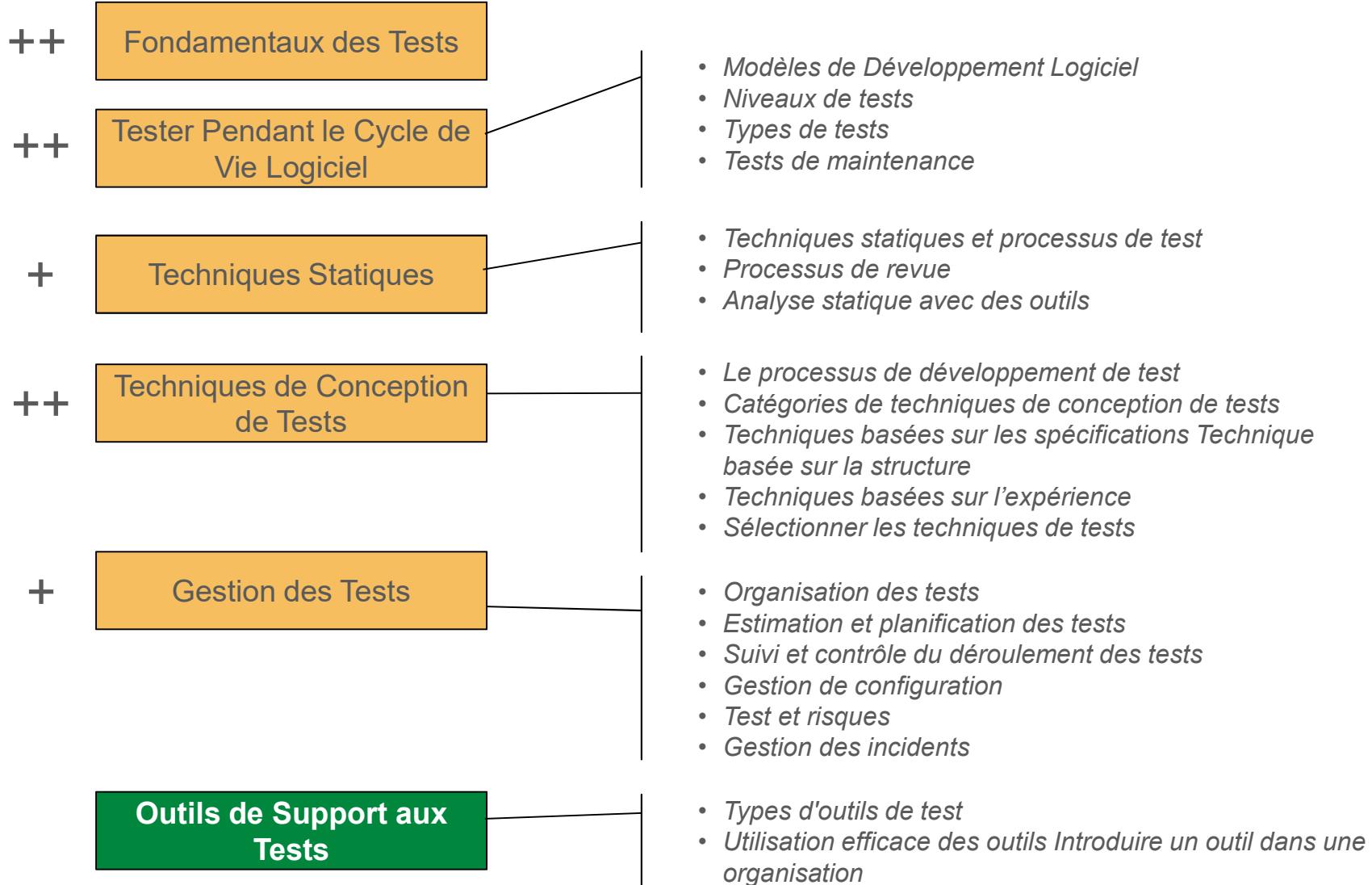
11. Dans une approche basée sur les risques, les risques identifiés peuvent être utilisés pour:

- i. Déterminer la technique de test à employer
 - ii. Déterminer l'étendue des tests à effectuer
 - iii. Prioriser les tests dans une tentative pour trouver des défauts critiques le plus tôt possible.
 - iv. Déterminer le coût du projet
- A. ii est vrai, I, III, IV et V sont faux
 - B. i, ii, iii sont vrais et iv est faux
 - C. ii et iii sont vrais; i, iv sont faux
 - D. ii, iii et iv sont vrais; i est faux

5. Gestion des tests QCM (4/4)

12. Lequel des énoncés suivants est la tâche d'un testeur?
- i. L'interaction avec le fournisseur de l'outil de tests pour identifier les meilleurs moyens d'utiliser l'outil de test sur le projet.
 - ii. Préparer et acquérir les données de test
 - iii. Mettre en œuvre des tests à tous les niveaux de test, exécuter et enregistrer les tests.
 - iv. Créer les spécifications de test
- A. i, ii, iii sont vrais et iv est faux
 - B. ii, iii, iv sont vrais et i est faux
 - C. i est vrai et ii, iii, iv sont faux
 - D. iii et iv sont corrects et I et II sont incorrects
13. On ne déclare pas d'incidents pour :
- A. les exigences
 - B. la documentation
 - C. les cas de test
 - D. les améliorations suggérées par les utilisateurs
14. Lequel des énoncés suivants n'est pas une tâche majeure des critères de sortie?
- A. Vérification des logs de test par rapport aux critères de sortie spécifiés dans la planification des tests.
 - B. Enregistrement des résultats de l'exécution des tests.
 - C. Evaluer si d'autres tests sont nécessaires.
 - D. Rédaction d'un rapport de synthèse de test pour les parties prenantes.

Contenu de la formation



6. Outils de support aux Tests

6.1 Types d'outils de test

6.2 Utilisation efficace des outils : Bénéfices potentiels et Risques

6.3 Introduire un outil dans une organisation

6.1. Types d'outils de test

Comprendre la signification et le but des outils

Un outil est un produit logiciel qui aide les utilisateurs dans leur travail.

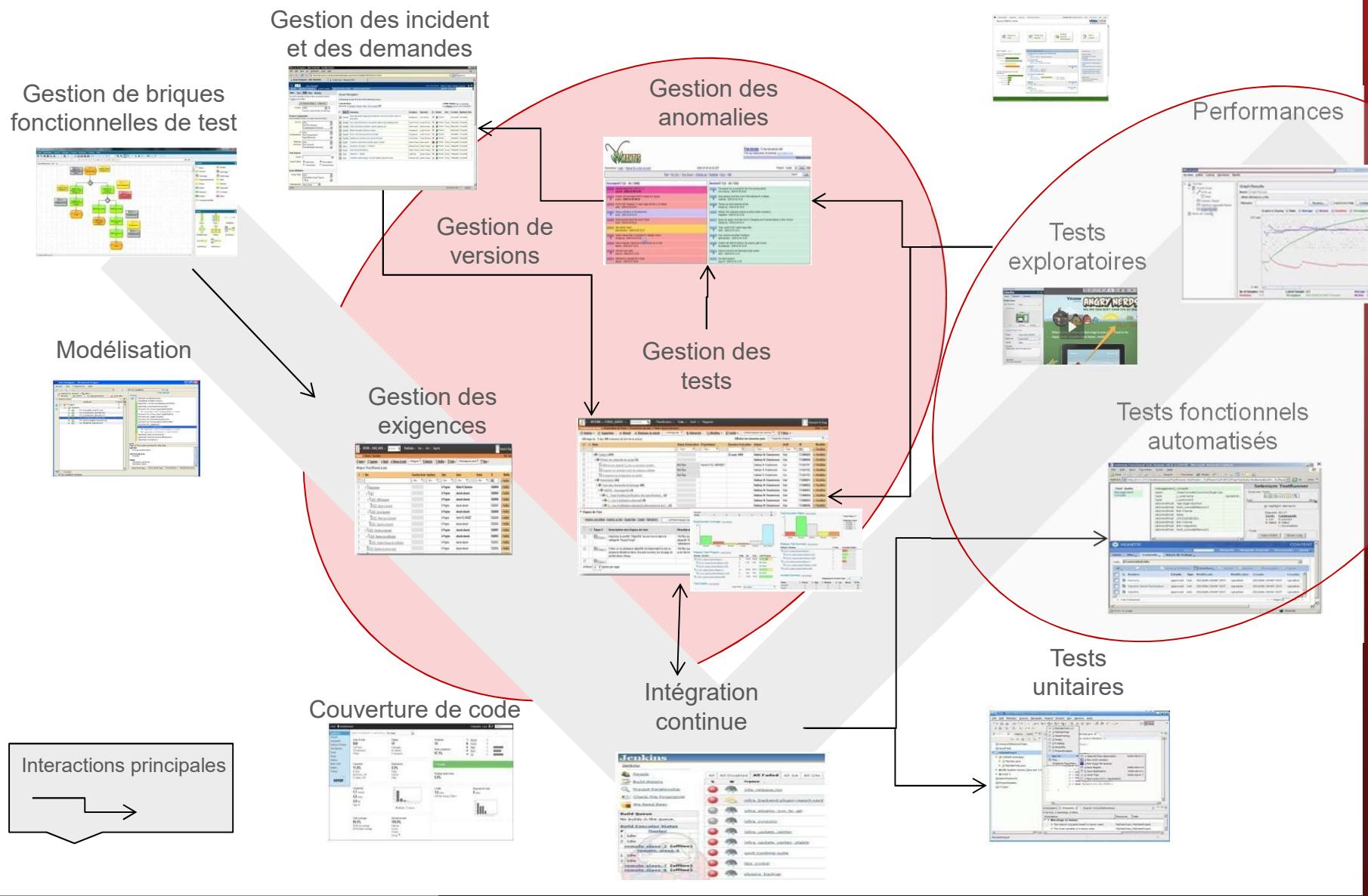
L'outil va permettre de favoriser le travail en :

- Accélérant la production (délai)
- Sécurisant les démarches (qualité)
- Réduisant les charges de travail (coût)

- Les outils sont une base sur laquelle s'appuient les procédures et méthodes, leur choix est important car il peut limiter ou étendre les possibilités offertes aux utilisateurs.

6.1. Types d'outils de test

Classification des outils



6.1. Types d'outils de test

Outils d'aide à la gestion du test et des tests

Objectif

- Outil assistant les utilisateurs (testeurs, responsable de test) dans leur activité de test et de suivi de l'activité.
- Ils permettent le stockage d'informations relatives au process de test (exigences, tests, campagnes, exécutions, anomalies, versions,...)
- Ces outils offre généralement la possibilité de générer un suivi par l'intermédiaire de rapports restituant des métriques.

Type d'outils

- Outil de gestion des tests
- Outil de gestion des exigences
- Outil de gestion des incidents
- Outil de gestion de configuration

Exemple d'outils

- Quality center, Spiratest, Testlink, Squash, Salome, Xstudio,
- Mantis, Jira, Clear Quest
- Requisite pro
- Doors, QARespository
- ... Excel

6.1. Types d'outils de test

Outils d'aide aux tests statiques

Objectif

- Outil permettant d'assister les testeurs dans les tests statiques.
- Il peut s'agir de
 - Checklist, normes servant de références au revues
 - Outils automatiques permettent de générer des métriques sur la qualité des livrables (code) et de détecter des erreurs potentielles (sans exécution du code)
 - Outils permettant de s'assurer de la cohérence d'un modèle de données

Type d'outils

- Outil de revue
- Outil d'analyse statique
- Outil de modélisation

Exemple d'outils

- Cast
- Checklists, normes

6.1. Types d'outils de test

Outils d'aide à la spécification des tests

Objectif

- Il s'agit d'outil permettant de fiabiliser ou d'accélérer la phase de spécification des tests.
- Les outils peuvent partir d'un modèle de fonctionnement pour générer des tests.
- Certains outils permettent de construire facilement des cas de test à partir de briques fonctionnelle préalablement définies sur le projet (BPT)

Type d'outils

- Outil de conception des tests
- Outil de préparation de données de tests

Exemple d'outils

- SmartTesting
- Générateurs de test à partir d'actions (BPT)
- Générateur de données (requêtes spécifiques, appels de services)

6.1. Types d'outils de test

Outils d'aide à l'exécution et à l'enregistrement des tests

Objectif

- Ces outils permettent de reproduire des actions sur un système à la place d'un testeur.
- L'automatisation de ces tests peut permettre d'accélérer les phases de test et éventuellement d'en réduire les coûts.

Type d'outils

- Outils d'exécution des tests
- Harnais de tests, outil de framework de tests unitaires
- Outils de comparaison
- Outils de mesure de couverture
- Outil de tests de sécurité

Exemple d'outils

- QTP, RFT, Testcomplete, Selenium, Testpartner
- Junit
- Appscan, Security center

6.1. Types d'outils de test

Outils de support de performance et de surveillance

Objectif

- Le but de ces outils est de mesurer la réactivité et la stabilité d'un système sous une charge donnée.
- La performance utilise généralement des outils intrusifs qui permettent d'envoyer des requêtes sur un système simulant ainsi l'utilisation par des utilisateurs virtuels dans le cadre de scénari à tester
- La performance peut avoir plusieurs objectifs : déterminer si les critères de performance sont conformes, déterminer le point de rupture, déterminer quelles parties dans l'architecture du système sont les causes de problèmes (fuites mémoires, dimensionnement, proxy...)

Type d'outils

- Outil d'analyse dynamique
- Outil de test de performance
- Outil de surveillance

Exemple d'outils

- LoadRunner, Jmeter, Neoload
- Sondes BAC

6.1. Types d'outils de test

Outils de support pour des besoins spécifiques

Objectif

- Il s'agit de tous les outils qui peuvent être utilisés au cours du projet.

Ils peuvent être utilisés lors des phases d'exécution et de conception. Il s'agit généralement d'outil développés spécifiquement pour le projet

Type d'outils

- Outil d'évaluation de la conformité des données (projet de migration)
- Outils de simulation

Exemple d'outils

- Outils de contrôles sur les données
- Simulateur, bouchons
- Outil d'émulation (mobile)

6.2. Utilisation pertinente des outils : avantages et risques potentiels

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).
- Évaluation 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).

6.2. Utilisation pertinente des outils : avantages et risques potentiels

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 (dont la formation et l'expertise externe).
- Sous-estimation du temps et de l'effort nécessaires pour obtenir de l'outil des bénéfices significatifs et continus de l'outil (incluant le besoin de modification du processus de tests et l'amélioration continue dans la manière d'utiliser l'outil).
- Sous-estimation de l'effort requis pour maintenir les acquis générés par l'outil.
- Confiance excessive dans l'outil (comme substitut à la conception des tests ou alors que des tests manuels seraient plus appropriés).
- 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
- 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

6.2. Introduire un outil dans une organisation

Les principes principaux liés à l'introduction d'un outil dans une organisation incluent :

- Évaluation de la maturité de l'organisation, de ses forces et de ses faiblesses et identification des possibilités d'amélioration du processus de test par le support d'outils.
- Évaluation au regard d'exigences claires et de critères objectifs.
- Une preuve de concept, à l'aide d'un outil de test pendant la phase d'évaluation pour vérifier qu'il fonctionne efficacement avec le logiciel en cours de test et dans l'infrastructure courante ou pour identifier des modifications requises de cette infrastructure pour utiliser l'outil efficacement
- Évaluation du vendeur (aspects de formation, de support et de commerce y compris) ou des fournisseurs de service de support en cas d'outils non commerciaux
- Identification des exigences internes pour le soutien et la tutelle dans l'utilisation de l'outil
- Évaluation de besoin de formation par rapport aux compétences en automatisation de tests de l'équipe de test courante
- Évaluation du rapport coût/bénéfice basée sur un cas métier concret

6.2. Introduire un outil dans une organisation

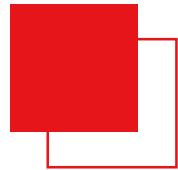
Les facteurs de succès du déploiement d'un outil dans une organisation incluent :

- Étendre l'outil au reste de l'organisation de façon incrémentale.
- Adapter et améliorer les processus de façon à les adapter à l'utilisation de l'outil.
- Fournir de la formation et une assistance aux nouveaux utilisateurs.
- Établir des guides d'utilisation.
- Implémenter une manière de tirer des enseignements de l'utilisation de l'outil.
- Surveiller l'utilisation de l'outil et les bénéfices recueillis
- Fournir le support pour l'équipe de test pour un outil donné
- Recueillir l'expérience acquise de toutes les équipes

6. Outils de support aux tests

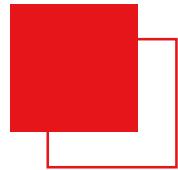
QCM

1. Des outils tels que Change Man, Clear case sont utilisés comme ..?
 - A. Outils d'automatisation fonctionnelle
 - B. Outils de test de performance
 - C. Outils de gestion de configuration
 - D. Aucune de ces réponses
2. Un outil qui prend en charge la traçabilité, l'enregistrement des incidents ou la planification des tests est appelé ..?
 - A. Un outil d'analyse dynamique
 - B. Un outil d'exécution de test
 - C. Un outil de débogage
 - D. Un outil de gestion des tests
 - E. Un outil de gestion de configuration
3. Les Outils des tests de charge
 - A. Réduisent le temps passé par les testeurs
 - B. Réduisent les ressources mobilisées (matériel)
 - C. Principalement utilisés dans les tests web
 - D. Toutes ces réponses
4. Quels outils apporte du support aux tests statiques ?
 - A. des outils d'analyse statique et d'exécution des tests
 - B. les outils de support pour les processus de revue, d'analyse statique et de mesure de couverture
 - C. les outils d'aide à l'analyse dynamiques et de modélisation
 - D. les outils de support pour les processus de revue, d'analyse statique et de modélisation



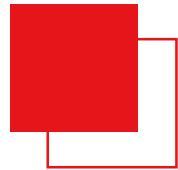
Bonnes Pratiques (1/3)





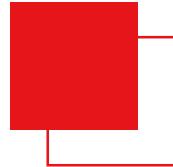
Bonnes Pratiques (2/3)



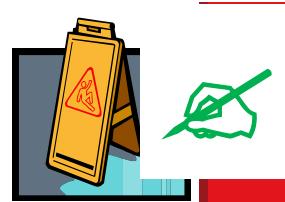


Bonnes Pratiques (3/3)

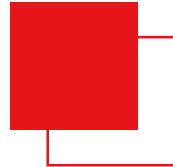




Certains préjugés sur le test : à éviter...



- **Le test introduit des Defects dans le logiciel**
- **Moins de Tests implique moins de « debugging »**
- **Nous devrions dès le départ produire un code de qualité, ne pas chercher les Defects**
- **Recoder est à imputer au budget de test pas au budget de développement (10% du budget total projet)**
- **Le test doit être “uniforme” (80% de fautes dans 20% de composants)**
- **Les bons tests utilisent des données de production**
- **Le test ralentit un projet en cours (non, c'est en réalité les défauts et le recodage qui ralentissent le projet)**



Points clés – synthèse



- **Le test est très important : partie très importante du budget global d'un projet**
- **La qualité du test se mesure dans la confiance qu'il permet de donner au système**
- **Le paradoxe du test : pour établir la confiance, il faut d'abord la « saboter »**
- **Le test est utilisé dans trois situations : évaluer, construire et préserver la qualité**
- **Résultat de la psychologie cognitive : les erreurs sont inévitables**
- **Le nombre de Defects détectés ne traduit en rien la qualité du test ni la qualité du logiciel en fin de Qualification**