Cours Génie Logiciel – M1 Informatique

Partie II – Test logiciel

Chapitre 2 – Tester pendant le cycle de vie du logiciel

Structure du chapitre 2

Niveau d’apprentissage

- Les niveaux de connaissance sont fournis pour chaque section et classés de la façon suivante :
  - **K1**: se souvenir
  - **K2**: comprendre
  - **K3**: utiliser
  - **K4**: analyser

2.1 Modèles de Développement Logiciel (K2)

- LO-2.1.1 Comprendre les relations entre le développement, les activités de test et les livrables dans le cycle de vie de développement, et donner des exemples basés sur des caractéristiques et contextes de produits et de projets (K2).

- LO-2.1.2 Reconnaître que les modèles de développement logiciel doivent être adaptés au contexte du projet et aux caractéristiques du produit (K1).

- LO-2.1.3 Rappeler que les bonnes pratiques de test sont applicables dans tous les modèles de cycle de vie (K1)

LO → Learning objective = objective d'apprentissage

Termes importants (définition à retenir)

- COTS – Logiciel commercial sur étagère
  - un produit logiciel qui est développé pour le marché général, p.ex. pour un nombre important de clients et qui est fourni pour de nombreux clients sous un format identique
- Validation
  - confirmation par l'examen et la fourniture des éléments objectifs que les exigences, pour un usage ou une application voulue, ont été remplies. [ISO 9000]
  - « *Are we producing the right product?* »
- Vérification
  - Confirmation par l'examen et la fourniture d'éléments objectifs que des exigences spécifiées ont été satisfaites. [ISO 9000]
  - « *Are we producing the product right?* »

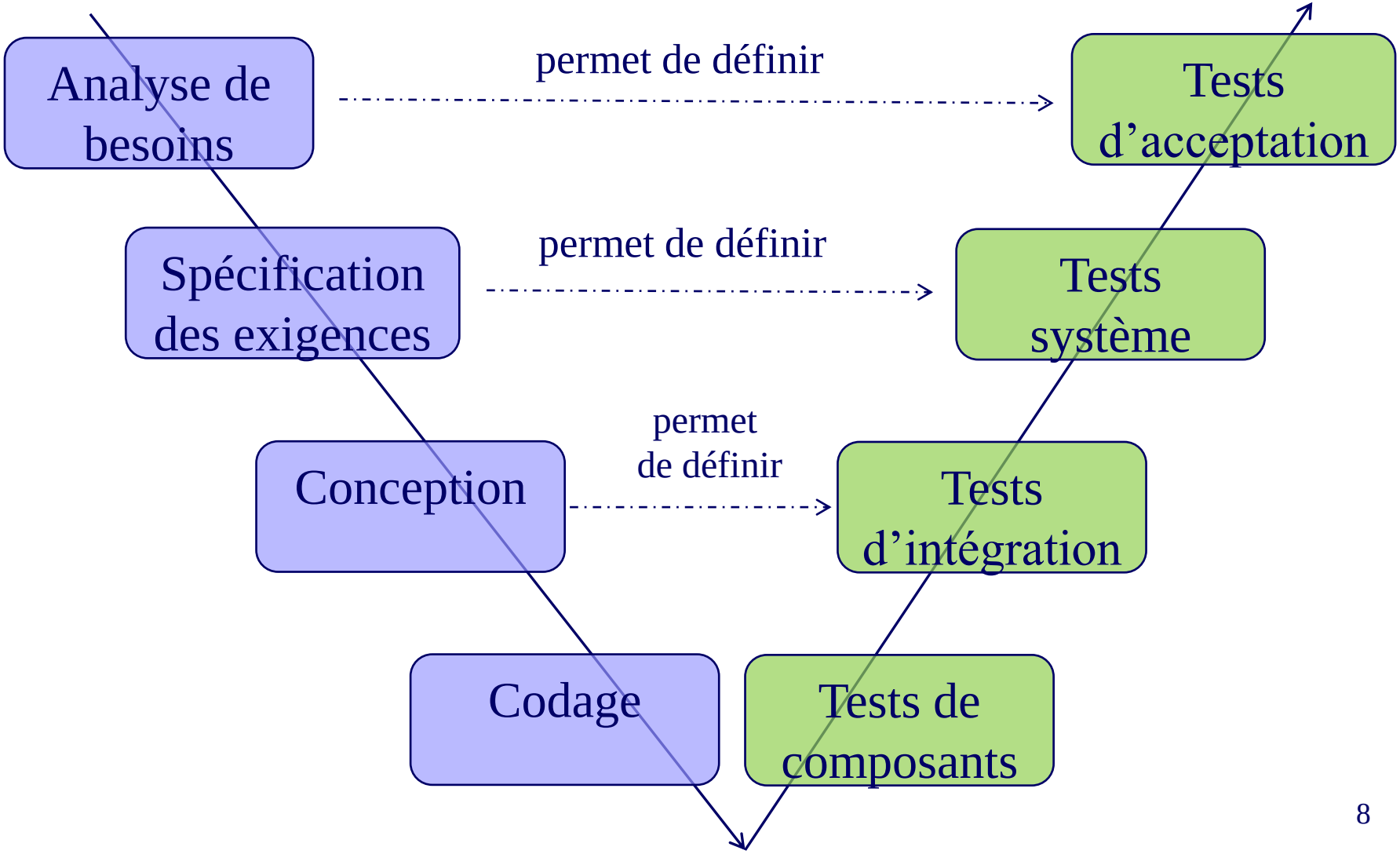
Termes importants (définition à retenir)

- Modèle en V
  - une structure décrivant les activités du cycle de développement logiciel, depuis la spécification des exigences jusqu'à la maintenance. Le modèle en V illustre comment les activités de tests peuvent être intégrées dans chaque phase du cycle de développement
- Modèle de développement incrémental
  - un modèle de cycle de développement où le projet est séparé en une série d'incréments, chacun d'entre eux fournissant une portion des fonctionnalités de l'ensemble des exigences du projet. Les exigences sont priorisées et fournies dans l'ordre des priorités lors de l'incrément approprié.
- Cycle de développement agile
  - Cycles de développement du logiciel fondé sur une approche itérative et incrémentale, et les pratiques issues du manisfeste agile

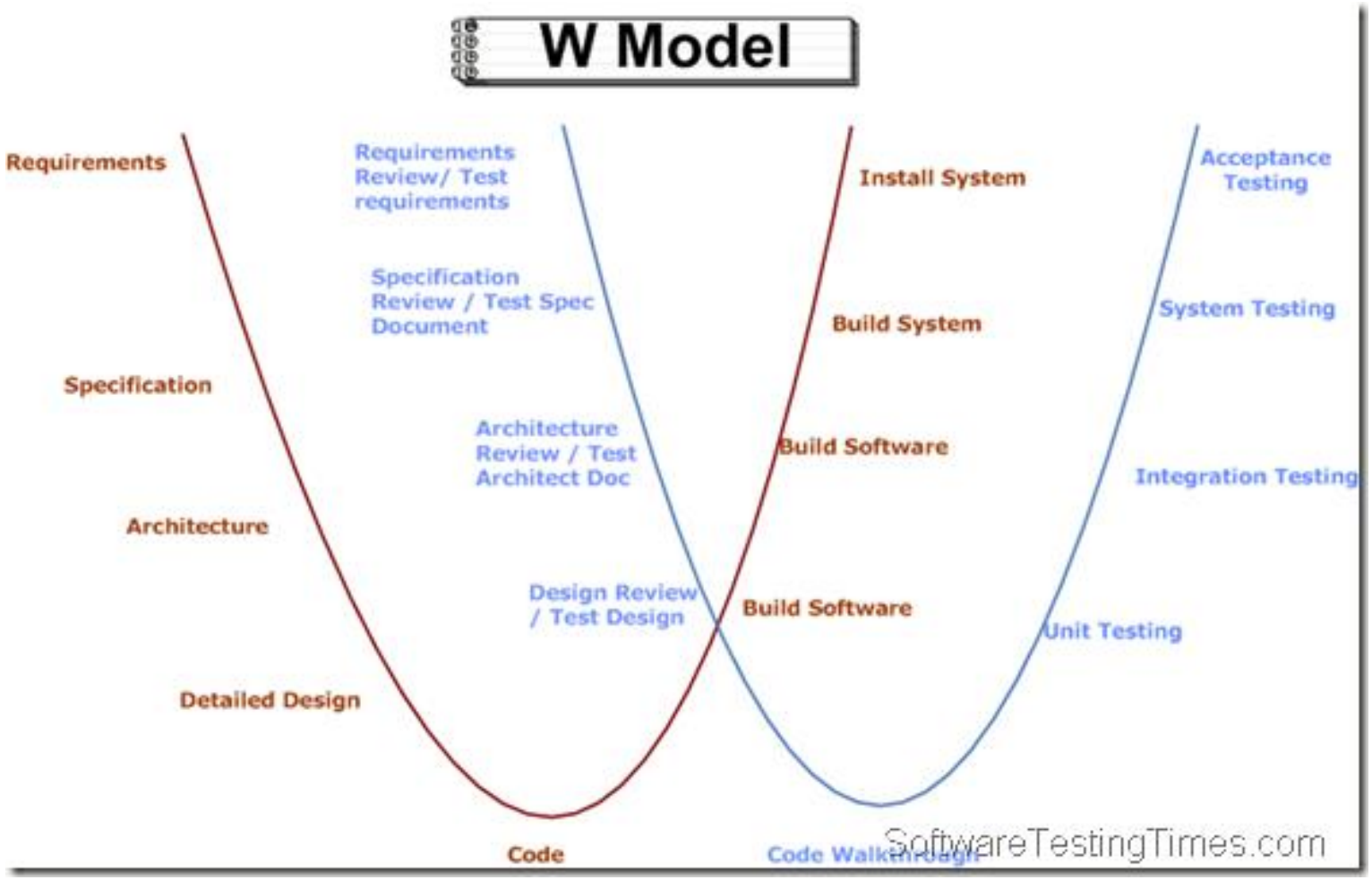
Termes importants (définition à retenir)

- Niveaux de test
  - Groupe d'activités de test organisées et gérées ensemble. Un niveau de test est lié aux responsabilités dans un projet.
  - Les exemples de niveaux de test sont :
    - les tests de composants (ou test unitaire),
    - les tests d'intégration,
    - les tests système
    - les tests d'acceptation.

2.1.1 Modèle en V - Exemple

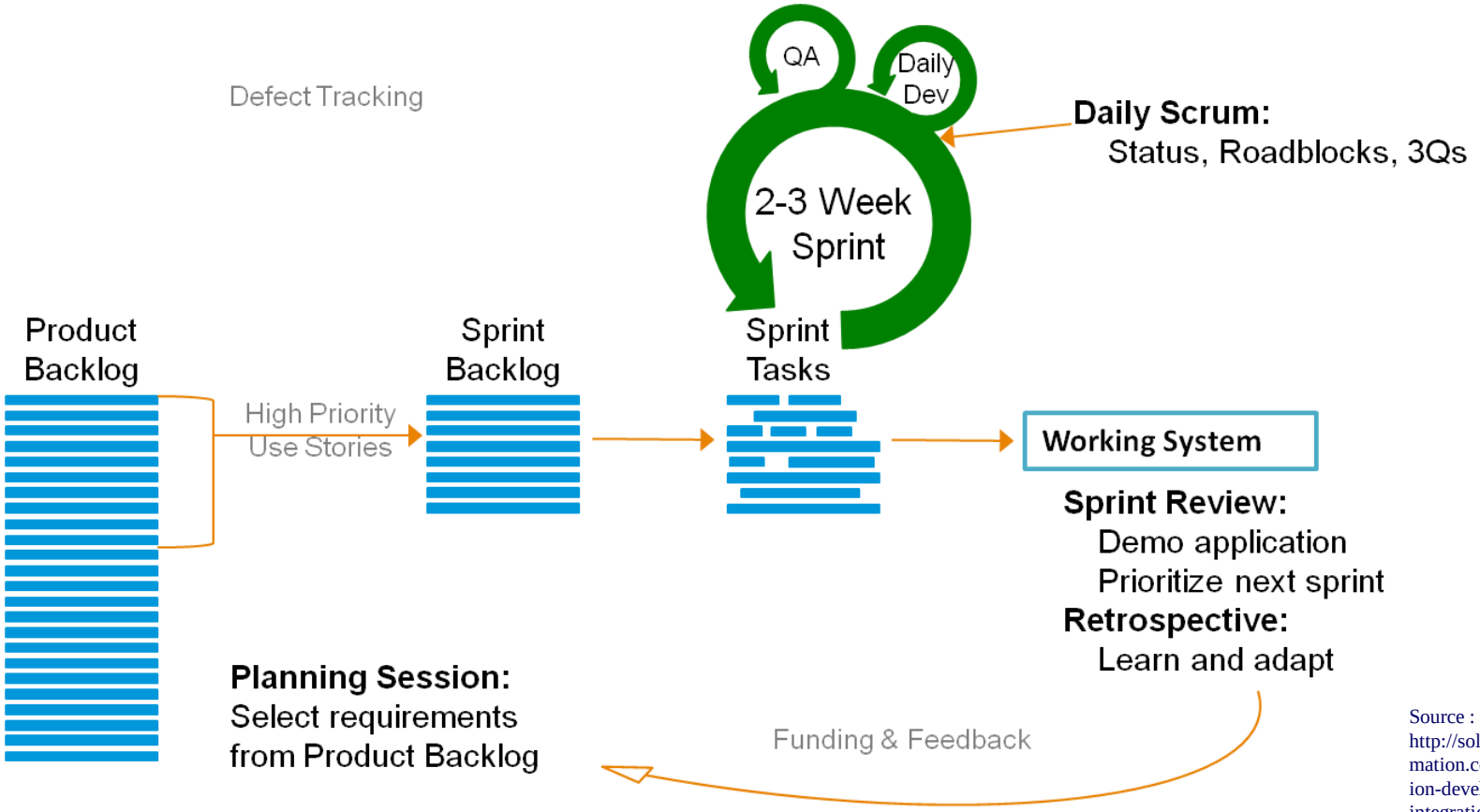


2.1.1 Modèle en W



Source : http://www.softwaretestingtimes.com/2010/04/v-model-to-w-model-w-model-in-sdlc.html

2.1.2 – Modèle de développement itératif et incrémental – Exemple SCRUM



Source : http://solutions.princetoninformation.com/solutions/application-development-system-integration/

- Les différents niveaux de test sont réalisés à l'intérieur des itérations.

2.1.3 Tester au sein d'un modèle de cycle de vie

- Quelques principes indépendant du modèle de cycle de vie :
  - A chaque activité de développement, correspond une activité de test
  - Chaque niveau de test a des objectifs de tests spécifiques pour ce niveau.
  - L'analyse et la conception des tests pour un niveau de test devraient commencer pendant l'activité correspondante de développement
  - Les testeurs doivent être impliqués dans la revue des documents aussi tôt que des brouillons sont disponibles dans le cycle de développement.
- Les niveaux de tests peuvent être combinés ou réorganisés selon la nature du projet ou de l'architecture du système.

Quiz – section 2.1

Q1 – D'après le glossaire CFTL / ISTQB, quelle est la phrase fausse ?

a) Un « COTS » est composant sur étagère

b) Validation est un synonyme de vérification

c) Le modèle en V est un cycle de vie par phase

d) Les approches agiles sont itératives et incrémentales

Quiz – section 2.1

Q2 – Mettez en correspondance certaines activités de développement et certaines activités de test

I) Analyse de besoins
II) Spécification des exigences
III) Conception et architecture du logiciel
IV) Codage

A) Développement des tests unitaires
B) Elaboration des tests d'acceptation
C) Définition des tests système
D) Définition des tests d'intégration

Réponse :

a) I-D, II-C, III-B, IV-A
b) I-A, II-B, III-C, IV-D
c) I-B, II-C, III-A, IV-D
d) I-B, II-C, III-D, IV-A

2.2 Niveaux de tests

- LO-2.2.1 Comparer les différents niveaux de tests: objectifs principaux, types d'objets habituels à tester, types de tests habituels (p.ex. fonctionnels ou structurels) et livrables associés, personnes en charge des tests, types de défauts et de défaillances à identifier. (K2)

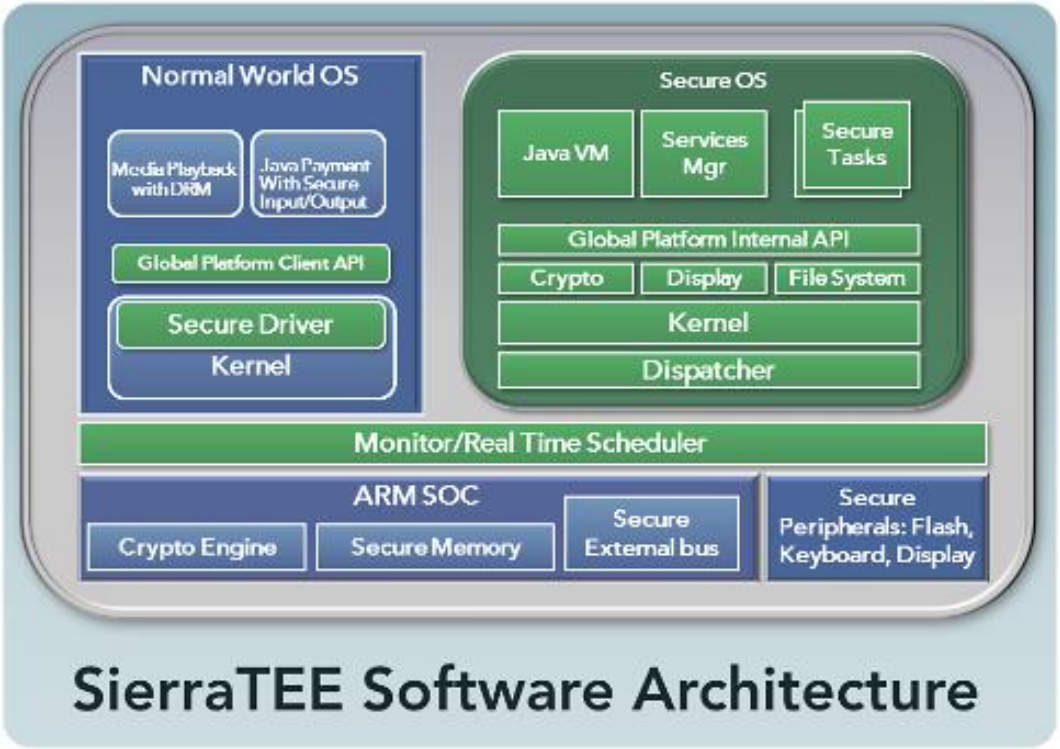
Sommaire

Synthèse sur les niveaux de test

- Tests de composants / Tests unitaires
  - Test des composants / Tests unitaires des méthodes et procédures
  - Réalisés par un développeur
- Tests d'intégration des composants
  - Test effectué pour révéler les défauts dans les interfaces (interactions)
  - Réalisés par un développeur ou un testeur
- Tests système
  - Tests de l'implémentation correcte des exigences au niveau du système
  - Réalisés par un testeur
- Tests d'acceptation
  - Appelé aussi « test de recette »
  - Vérifie si un système satisfait aux critères d'acceptation
  - Réalisé par l'utilisateur, le client ou une équipe de test

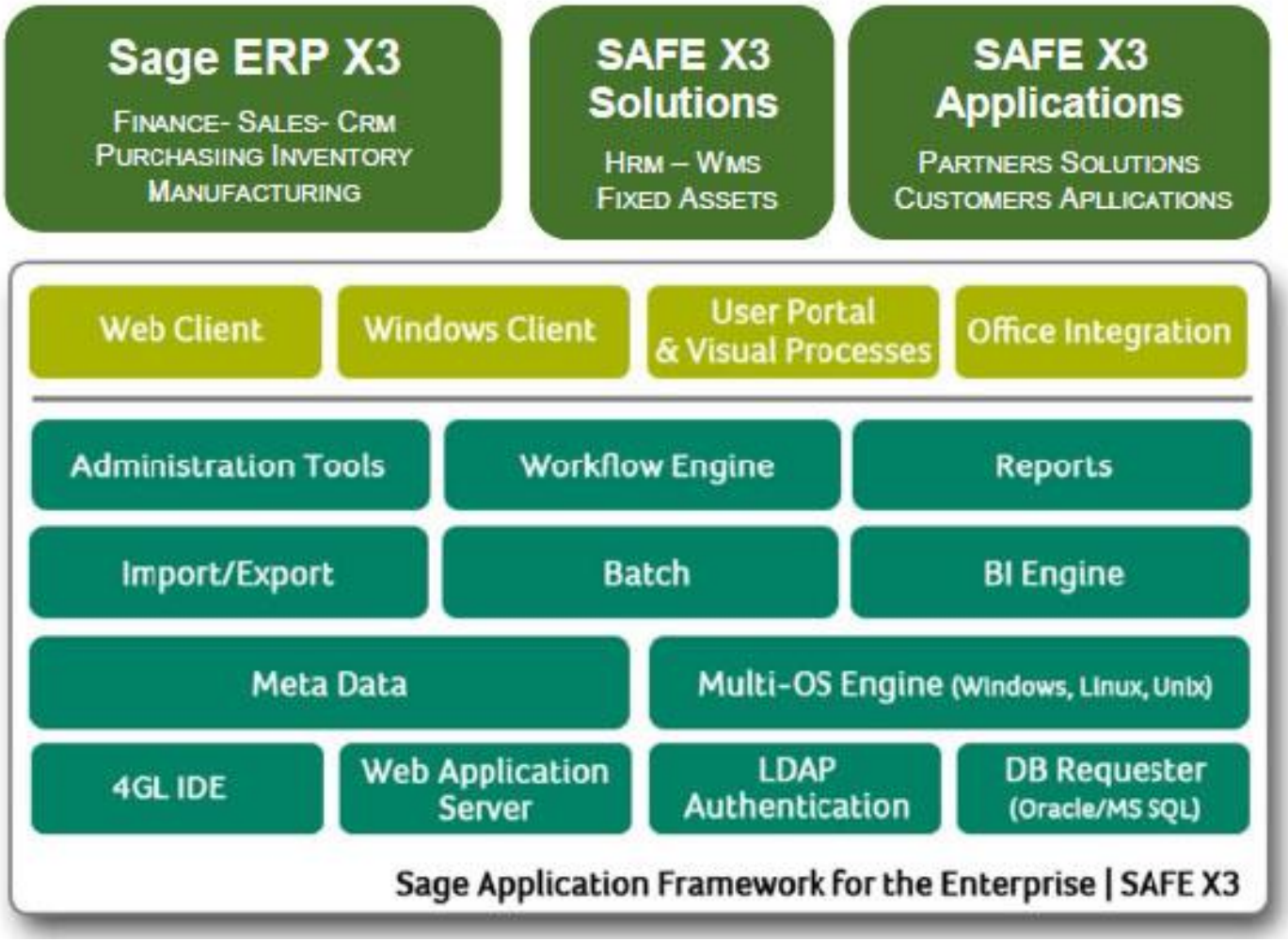
Niveaux de test - Exemple Composant de sécurité

- Test unitaire au sein de chaque composant
- Test d'intégration entre les composants
- Test système par bloc fonctionnel
- Test d'acceptation de la totalité



➔ Les niveaux de test sont complémentaires

Niveaux de test – Exemple ERP



ERP – Enterprise Resource Planning – Progiciel de gestion intégré

Termes (à retenir)

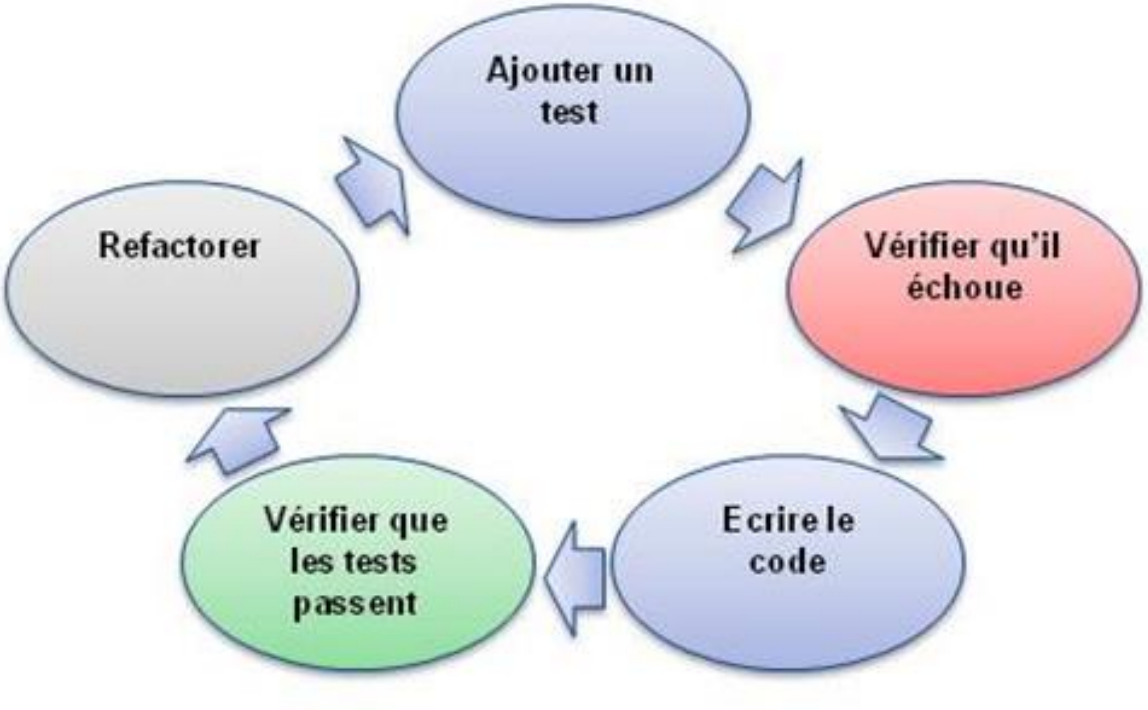
- Bouchon
  - une implémentation spéciale ou squelettique d'un composant logiciel, utilisé pour développer ou tester un composant qui l'appelle ou en est dépendant. Cela remplace un composant appelé.
- Pilote
  - un composant logiciel ou outil de tests qui remplace un composant qui contrôle et/ou appelle un composant ou système
- Environnement de test
  - Environnement contenant le matériel, les instruments, les simulateurs, les outils logiciels et les autres éléments de support nécessaires à l'exécution d'un test

2.2.1 Tests de composants / Tests unitaires (K2)

- Bases de tests:
  - Exigences des composants
  - Conception détaillée
  - Code
- Les tests unitaires visent
  - le test de fonctionnalités (*le module répond t'il a ce qu'il doit faire ?)*
  - le test de caractéristiques non fonctionnelles
    - » tests de performance (p.ex. fuites mémoire)
    - » tests de robustesse
  - la couverture structurelle du code (p.ex. couverture des branches)
- Avantages du test de composants / tests unitaires
  - Trouver les défauts le plus tôt possible
  - Sécuriser la maintenance
  - Documenter le code

Pratique du TDD – Test Driven-Development

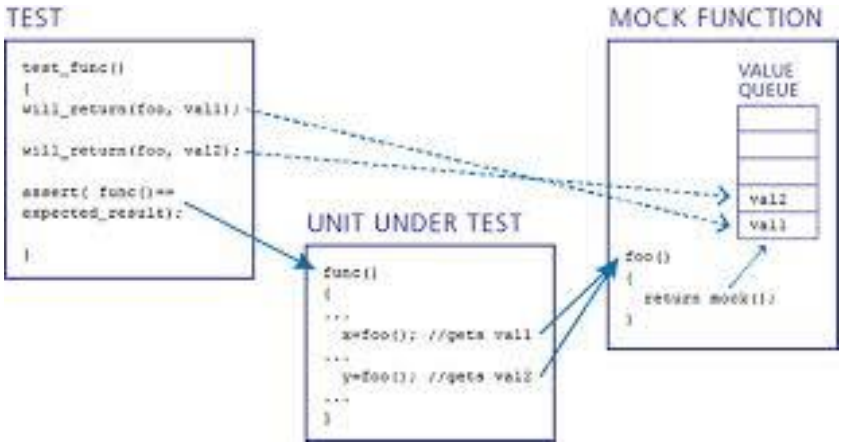
- Ecrire les tests unitaires d'une fonction, méthode ou procédure avant d'écrire son code
- Piloter le développement par les tests
- Permet de réfléchir au rôle du code testé avant de l'écrire



Source : http://blog.octo.com/le-test-driven-development-au-secours-de-javascript/

Notion de bouchonnage - Mock

- Composants simulés qui reproduisent le comportement d'objets réels de manière contrôlée pour permettre le test



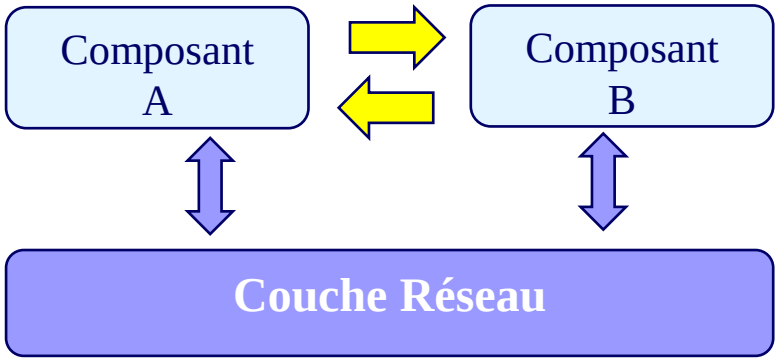
Source : http://www.grammatech.com/blog/unit-testing-c-programs-with-mock-functions

Objectifs du bouchonnage :

- Si l'objet auquel il faut accéder pour le test n'existe pas encore.
- Lorsque la méthode testée a des états difficiles à reproduire (une erreur de réseau par exemple).
- Pour remplacer un comportement non déterministe (l'heure ou la température ambiante).
- Pour accélérer l'exécution du test – Par exemple si l'initialisation de l'objet est longue (ex : création d'une base de données).

2.2.2 Test d'intégration (K2)

- Les tests d'intégration ont pour objectif le test :
  - des interfaces entre les composants
  - des interactions entre différentes parties d'un système tel que:
    - » le système d'exploitation
    - » le réseau
    - » la gestion de données
    - » le matériel
    - » les interfaces entre les systèmes

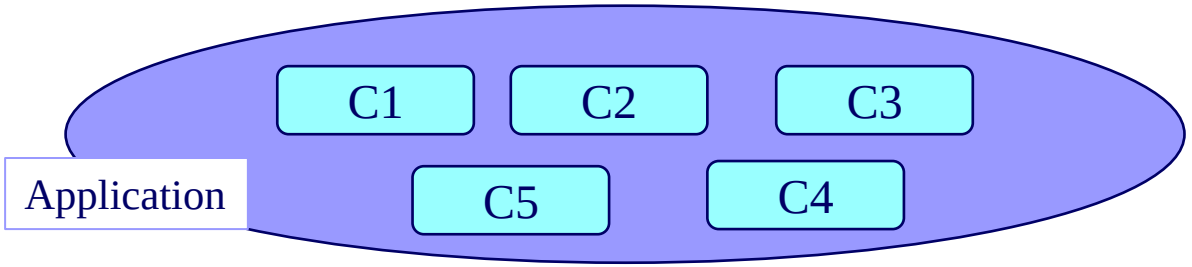


Tests d’intégration

- Bases de Tests:
  - Conception du logiciel et du système
  - Architecture
  - Workflows
  - Cas d‟utilisation
- Objets habituels de test:
  - Sous-systèmes
  - Implémentation de bases de données
  - Infrastructure
  - Interfaces
  - Configuration système et données de configuration

Le test d'intégration peut être réalisé à plusieurs niveaux → composant élémentaire

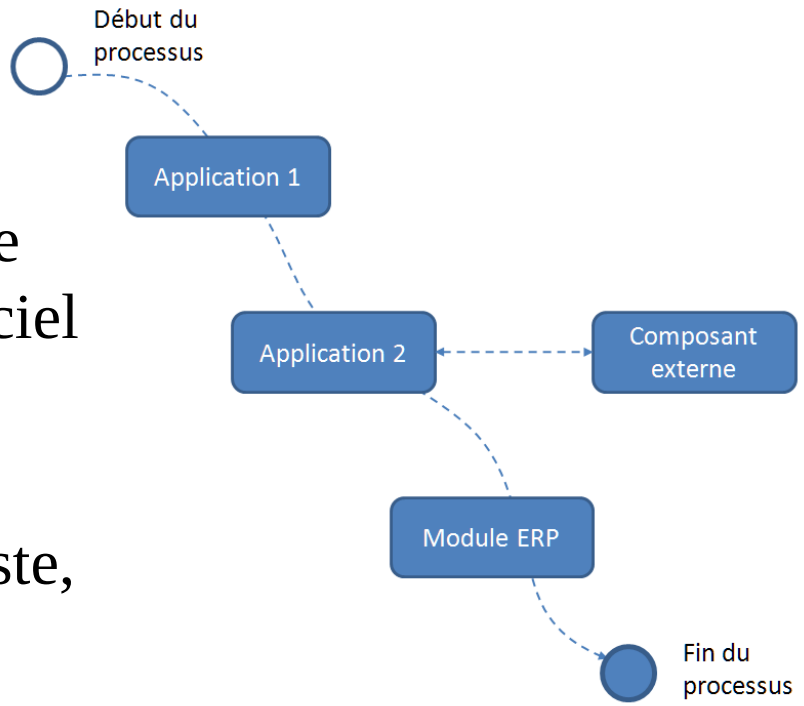
- Test d'intégration de composants : test de l'intégration des composants au niveau d'une application



- Dans l'intégration de composants : les tests d'intégration doivent couvrir les composants deux à deux, et les interactions entre les composants logiciels.
- Ces tests sont effectués après les tests de composants;

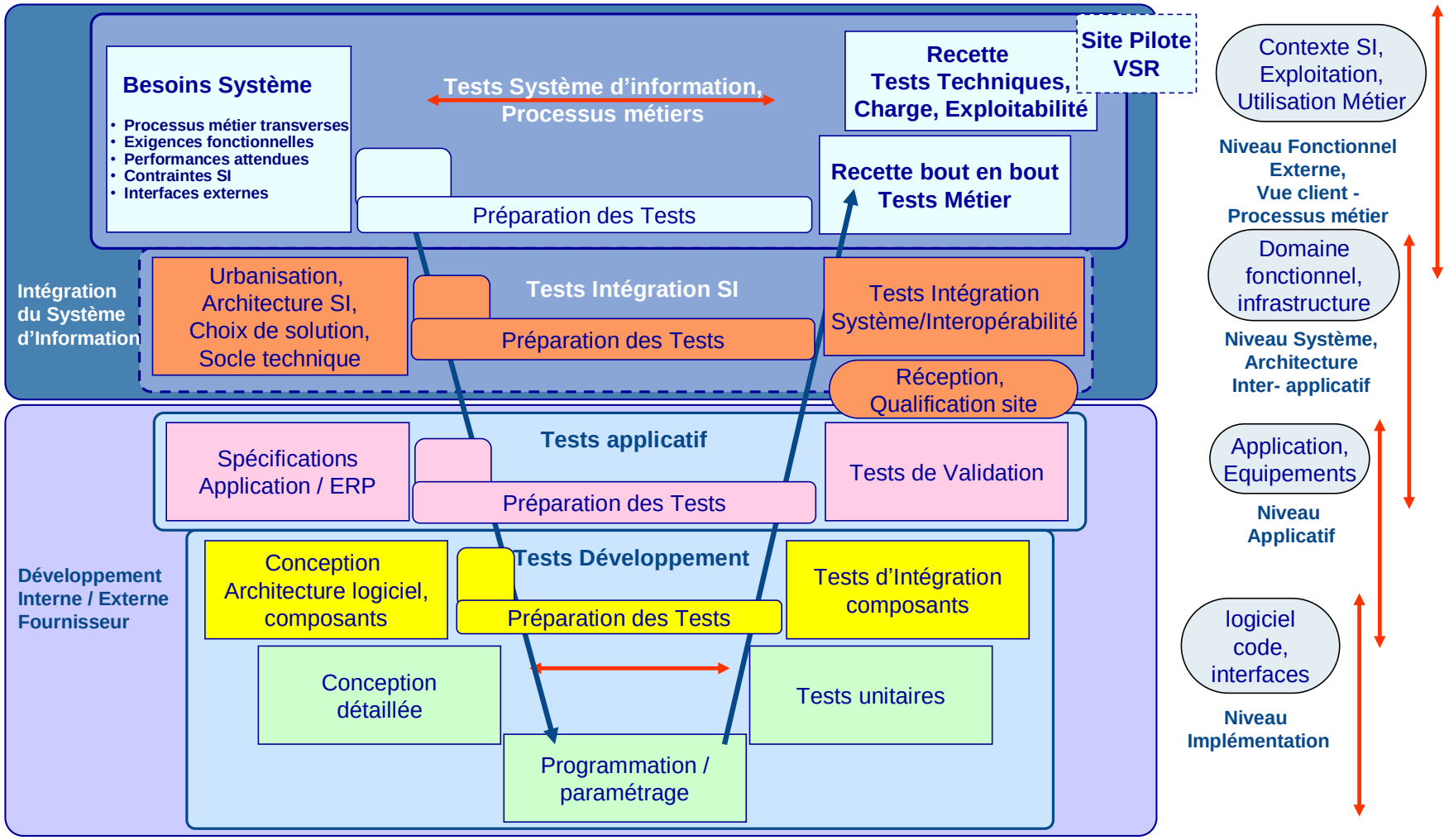
Le test d'intégration peut être réalisé à plusieurs niveaux → multi-applicatifs / système de systèmes

- Tests d'intégration système: test de l'intégration multi-applicatifs (par exemple dans un grand systèmes d'information)
- Ils visent le test de l'intégration entre les différents systèmes ou entre logiciel et matériel et pouvant être effectués après les tests système.
- Plus la portée de l'intégration est vaste, plus il devient difficile d'isoler les défauts liés à un composant ou un système particulier.



Exemple- Test de bout-en-bout

Test d'intégration système – exemple de processus pour les grands systèmes d'information



Techniques de test d'intégration – Bottom-up

- Test d'intégration de bas en haut : une approche incrémentale des tests d'intégration ou les composants du niveau le plus bas sont testés en premier, et ensuite utilisés pour faciliter les tests des composants de plus haut niveau. Ce processus est répété jusqu'au test du composant le plus haut de la hiérarchie.

Techniques de test d'intégration – Top-down

- Test d'intégration de haut en bas : Approche incrémentale de test d'intégration dont les composants de haut hiérarchique sont testés d'abord, et dont les composants de niveau inférieur sont simulés par des bouchons. Les composants testés sont ensuite utilisés pour tester des composants de niveaux inférieurs. Le processus est ainsi répété jusqu'à ce que les composants de plus bas niveau ont été testés.

Différents types d'intégration des composants

- Intégration big-bang
  - Le test d'intégration est réalisé lorsque l'ensemble des composants sont terminés
  - Pratique peu sûr → les anomalies sont détectées très tard

- Intégration continue
  - Les tests d'intégration sont réalisés à chaque commit de code dans le référentiel
  - Pratique de base dans les méthodes agiles

Bonnes pratiques du test d'intégration

- A chaque étape de l'intégration, les testeurs se concentrent uniquement sur l'intégration elle même.
  - Par exemple, s'ils sont en train d'intégrer le module A avec le module B, les testeurs s'appliquent à tester la communication entre les modules et non pas leurs fonctionnalités individuelles.
- Les tests de caractéristiques non-fonctionnelles particulières (p.ex. performances) peuvent être inclus dans les tests d'intégration.
- Les approches fonctionnelles et structurelles peuvent toutes deux être utilisées.

Termes (à retenir)

- Intégration
  - le processus de combiner des composants ou systèmes en assemblages plus grands
- Tests d'intégration
  - tests effectués pour montrer des défauts dans les interfaces et interactions de composants ou systèmes intégrés.
- Harnais de test
  - Environnement comprenant les bouchons et les pilotes nécessaires pour exécuter un test.

2.2.3 Test système (K2)

- Objectifs du test système :
  - Les tests systèmes traitent le comportement d'un système/produit complet.
  - Ils sont basés sur les exigences en ciblant à la fois les exigences fonctionnelles et les non-fonctionnelles.
- Les tests système peuvent inclure des tests basés sur les risques et/ou sur les spécifications et les exigences, les processus métier, les cas d'utilisation, ou autres descriptions de haut niveau du comportement du système.
- Les tests systèmes sont souvent des tests en mode boîte noire

Termes (à retenir)

- Technique de conception de tests boîte noire
  - procédure documentée pour élaborer et sélectionner des cas de tests basés sur une analyse des spécifications, soit fonctionnelles soit non-fonctionnelles, d'un composant ou système sans faire référence à ses structures internes.
- Technique de conception de test boîte blanche
  - Procédure documentée utilisée pour dériver et sélectionner des cas de tests basés sur une analyse de la structure interne d'un composant ou système

Test système

- Bases de Test:
  - Spécifications d‟exigences Système et logiciel
  - Cas d’utilisation
  - Spécifications fonctionnelles
  - Rapports d’analyse des risques

- Objets habituels du test:
  - Manuels système, utilisateur et opérationnels
  - Configuration système et données de configuration

Test système – Bonnes pratiques

- Le périmètre des tests doit être clairement défini dans le plan de test (cf chapitre 1)
- L'environnement de test devrait correspondre à la cible finale ou à un environnement de production de façon à minimiser les risques de défaillances lors de la mise en opération du système.
- Une équipe de tests indépendante exécute souvent des tests système.

2.2.4 Test d'acceptation (K2)

- Objectifs : test 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
- Exemples possibles de tests d'acceptation :
  - Acceptation utilisateurs
  - Acceptation opérationnelle
  - Acceptation contractuelle et réglementaire
  - Tests alpha et beta

Types de test d'acceptation – 1/2

- Tests d'acceptation utilisateur
  - Ces tests vérifient l'‟aptitude et l'‟utilisabilité du système par des utilisateurs.
- Tests (d'acceptation) opérationnelle
  - L'acceptation du système par les administrateurs système, dont:
    - » Tests des backups et restaurations;
    - » Reprise après sinistre;
    - » Gestion des utilisateurs;
    - » Tâches de maintenance;
    - » Chargements de données et tâches de migration
    - » Vérification périodique des vulnérabilités de sécurité.

Types de test d’acceptation – 2/2

- Tests d’acceptation contractuelle et réglementaire
  - Les tests d’acceptation contractuelle sont exécutés sur base des critères dacceptation contractuels définis lors de la rédaction du contrat.
  - Les tests d’acceptation réglementaires sont exécutés par rapport aux règlements et législations qui doivent être respectés, telles que les obligations légales, gouvernementales ou de sécurité.

- Tests alpha et bêta

Termes (à retenir)

- Tests Alpha
  - Tests opérationnels réels ou simulés par des utilisateurs/clients potentiels ou par une équipe de test indépendante sur le site de développement, mais en dehors de l'organisation de développement. Les tests Alpha sont souvent utilisés comme une forme de tests d'acceptation interne.
- Teests Bêta
  - Tests opérationnels par des utilisateurs/clients potentiels et/ou réels sur un site externe non associé aux développeurs, pour déterminer si un composant ou système satisfait ou non les besoins des utilisateurs/clients et s'adaptent aux processus d'entreprise. Le béta-test est souvent utilisé comme une forme de tests externe d'acceptation de façon à obtenir des informations de retour du marché.

Test d'acceptation

- Bases de test:
  - Exigences utilisateur
  - Exigences du système
  - Cas d'utilisation
  - Processus métier
  - Rapports d'analyse des risques

- Objets habituels de test:
  - Processus métier sur l'intégralité du système
  - Processus opérationnels de maintenance
  - Procédures utilisateur
  - Formulaires
  - Rapports

Quiz – section 2.2

Q1 – Quelle est la phrase fausse ?

a) Les tests d'acceptation sont réalisés par les développeurs du système

b) Les tests système visent à s'assurer que le système réponds aux exigences définies

c) Le test d'intégration système peut être réalisé par des tests de bout-en-bout

d) Les tests de composants peuvent être réalisés avec une approche fonctionnelle ou structurelle

Quiz – section 2.2

Q2 – Quelle est la phrase juste ?

a) Les méthodes de TDD (test-driven development) sont utilisées pour les tests d'acceptation

b) L'intégration big-bang est préférable aux méthodes d'intégration plus itératives pour détecter rapidement des bugs

c) Si les tests système sont bien réalisés, les autres types de test sont inutiles

d) Les tests d'intégration doivent se polariser sur les problèmes liés à l'interfaçage des composants plutôt que sur les fonctionalités de chaque composant

Quiz – section 2.2

Q3 – Quelle option est une bonne pratique pour tester pendant le cycle de vie logiciel ?

a) Effectuer tôt l'analyse et la conception des tests.

b) Définir plusieurs niveaux de test avec des objectifs spécifiques.

c) Impliquer les testeurs lorsque le code est disponible.

d) a et b ci-dessus

2.3 Types de test – Les cibles de test

- LO-2.3.1 Comparer les quatre types de tests (fonctionnels, non-fonctionnels, structurels et liés aux changements) avec des exemples. (K2)
- LO-2.3.2 Reconnaître que les tests fonctionnels et structurels peuvent apparaître à n'importe quel niveau. (K1)
- LO-2.3.3 Identifier et décrire des types de tests non-fonctionnels basés sur des exigences non-fonctionnelles. (K2)
- LO-2.3.4 Identifier et décrire des types de tests basés sur l'analyse de la structure ou de l‟architecture d‟un système logiciel. (K2)
- LO-2.3.5 Décrire l'utilité des tests de confirmation et des tests de régression. (K2)

2.3 Types de test

2.3.1 Tests des fonctions (tests fonctionnels) (K2)

2.3.2 Tests des caractéristiques non fonctionnelles des produits logiciels (tests non-fonctionnels) (K2)

2.3.3 Tests de la structure / architecture logicielle (tests structurels) (K2)

2.3.4 Tests liés au changement (tests de confirmation et de régression) (K2)

Types de test

- Un type de tests est focalisé sur un objectif de tests particulier, par exemples :
  - le test d'une fonction devant être effectuée par le logiciel;
  - une caractéristique non-fonctionnelle, telle que la fiabilité, la performance, la sécurité ou l'utilisabilité,
  - la structure ou l'architecture du logiciel ou du système;
  - lié aux changements, p.ex. confirmer que des anomalies ont été corrigées (tests de confirmation) et pour rechercher la présence de modifications non voulues (tests de régression).

2.3.1 – Tests fonctionnels (K2)

- Il s'agit des tests visant la bonne implémentation des fonctionnalités attendues du logiciel
- Les tests fonctionnels sont généralement conçus à partir de exigences et spécifications fonctionnelles
- Les tests fonctionnels peuvent être réalisé à tous les niveaux de test
  - Par exemple, le test de composant peut être vu d'un point de vue fonctionnel, comme au niveau système ou d'acceptation.

- Test fonctionnel
  - test basé sur une analyse des spécifications d'une fonctionnalité d'un composant ou système.

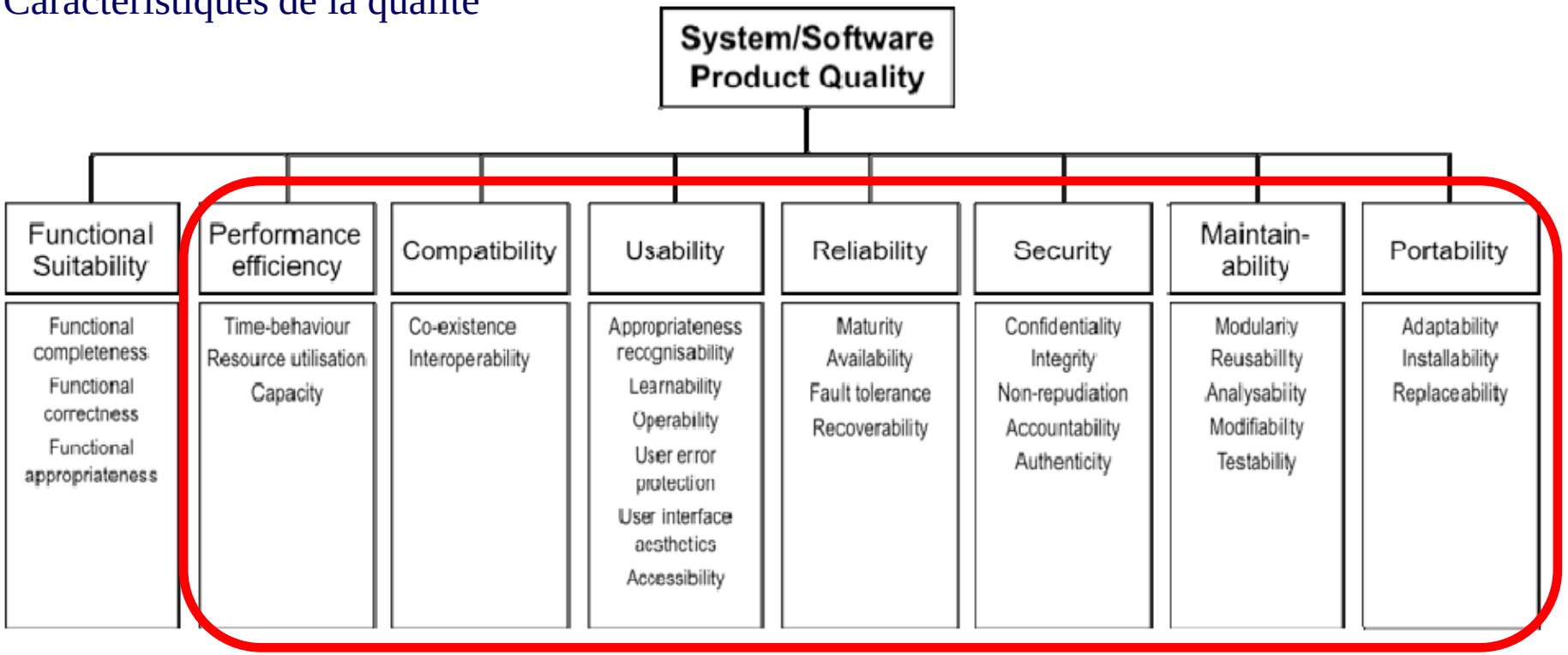
Cas particuliers de tests fonctionnels

- Test fonctionnel de sécurité
  - Lorsque l'objet du test est un composant de sécurité (par exemple Composant cryptographique), les tests fonctionnels portent sur la bonne implémentation des propriétés de sécurité (confidentialité, intégrité, disponibilité)
- Test d'interopérabilité / de conformité
  - Les tests d'interopérabilité et de confirmité à un standard sont des formes de test fonctionnels qui visent à s'assurer que les systèmes vont être compatibles (par exemple le logiciel sur une carte SIM avec le logiciel du téléphone mobile).

2.3.2 Tests non-fonctionnels (K2)

- Test des caractéristiques non fonctionnelles d'un logiciel

ISO 25010-2011
Caractéristiques de la qualité



Tests non fonctionnels

- Des tests non-fonctionnels peuvent être réalisés sur tous les niveaux de test
  - Par exemple, des tests de performances peuvent être réalisés au niveau test de composant comme au niveau système
- Ces tests évaluent “comment” le système fonctionne.

Termes (à retenir)

- Performance
  - Degré en fonction duquel le système accomplit ses fonctions dans le respect de contraintes relatives au temps d'exécution et taux de débit
- Test de charge
  - Un type de test concerné par la mesure du comportement d'un composant ou système avec une charge croissante, p.ex. nombre d'utilisateurs et/ou nombre de transactions en parallèle pour déterminer quelle charge peut être gérée par le composant ou système
- Test de stress
  - Un type de test de performance mené pour évaluer un système ou composant à ou audelà des limites de sescharges de travail anticipées ou spécifiées, ou avec une disponibilité réduites de ressources telles que l'accès mémoire ou serveurs

Termes (à retenir)

- Maintenabilité
  - 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
- Portabilité
  - Facilité avec laquelle un produit logiciel peut être transféré d'un environnement matériel ou logiciel vers un autre
- Fiabilité
  - 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

Termes (à retenir)

- Robustesse
  - le degré pour lequel un composant ou système peut fonctionner correctement en présence de données d'entrée invalides ou de conditions environnementales stressantes

2.3.3 Test de la structure / Tests structurels (K2)

- Ces tests sont basés sur une analyse de la structure interne du composant ou du système
- Les tests structurels sont appliqués au niveau architectural et code:
  - Analyse des flots de contrôle, de données, des condition logiques et des itérations.
  - En général mis en œuvre en mode boite blanche (avec accès au code).

2.3.4 Tests liés au changement (tests de confirmation et de régression) (K2)

- Quand un défaut est détecté et corrigé, le logiciel doit être re-testé pour confirmer que le défaut original a été correctement ôté : tests de confirmation (re-test)
- Après que des modifications du programme aient eu lieu, pour identifier tout nouveau défaut dû à ce(s) changement(s) : tests de (non) régression
  - Les tests de régression sont exécutés lors d'une modification du logiciel de son environnement
  - Une analyse de risque doit être effectuée pour déterminer le périmètre des tests de régression.

Quiz – section 2.3

Q1 – Quelle est la phrase fausse

a) Un test de sécurité peut aussi être un test fonctionel (de sécurité)

b) Le test fonctionnel vise le test de ce que « fait » le système

c) Les tests de régression doivent couvrir la totalité des fonctionnalités de façon systèmatique

d) L’ensemble des caractéristiques non fonctionnelles peuvent faire l’objet de tests

2.4 Tests de maintenance (K2)

- LO-2.4.1 : Comparer les tests de maintenance (tester un système existant) et les tests d'une nouvelle application en ce qui concerne les types de tests, les déclencheurs des tests et la quantité de tests (K2)
- LO-2.4.2 : Reconnaitre les indicateurs pour les tests de maintenance (modification, migration et abandon) (K1)
- LO-2.4.3 : Décrire le rôle des tests de régression et de l'analyse d'impact en maintenance (K2)

Tests de maintenance

- Les tests de maintenance sont effectués sur un système opérationnel existant et sont déclenchés par des modifications, migrations ou suppression de logiciels ou de systèmes.
- Les modifications incluent les changements dûs aux évolutions planifiées, aux modifications correctives et d'urgence, ainsi qu'aux changements d'environnements (montées en version planifiées des systèmes d'exploitation, des bases de données ou des COTS).
- Les tests de maintenance s'appuient sur une analyse d'impact

Termes (à retenir)

- Analyse d'impact
  - Evaluation de modifications dans la documentation de développement, dans la documentation de test et dans des composants, de façon à implémenter la modification d'une exigence
- Test de maintenance
  - Test des modifications d'un système opérationnel ou de l'impact d'une modification d'environnement sur un système opérationnel

Tests de maintenance – Bonnes pratiques

- Les tests de maintenance incluent :
  - Des tests des nouvelles fonctionnalités
  - Des tests de (non) régression pour les parties du système qui n'ont pas été modifiées
  - Les tests opérationnels du nouvel environnement (dans le nouveau contexte de la production)
- La planification des tests de maintenance doivent s'appuyer sur une analyse de risque

Quiz – section 2.4

Q1 – Quelle est la phrase juste ?

a) L’analyse d’impact permet de prévoir quel temps il va faire demain

b) Toute évolution du logiciel, mais aussi de son environnement, donne lieu à des tests de maintenance

c) Les tests de régression ne font pas partie des tests de maintenance

d) Après la mise en production, les tests sont finis (ouf !)

Ce qui a été vu dans le chapitre 2

- 2.1 Modèles de Développement Logiciel (K2)
  - 2.1.1 Modèle en V (K2)
  - 2.1.2 Modèle de développement itératif (K2)
  - 2.1.3 Tester au sein d‟un modèle de cycle de vie (K2)
- 2.2 Niveaux de tests (K2)
  - 2.2.1 Tests de composants (K2)
  - 2.2.2 Tests d‟intégration (K2)
  - 2.2.3 Tests système (K2)
  - 2.2.4 Tests d‟acceptation (K2)
- 2.3 Types de tests (K2)
  - 2.3.1 Tests des fonctions (tests fonctionnels) (K2)
  - 2.3.2 Tests des caractéristiques non fonctionnelles des produits logiciels (tests non-fonctionnels) (K2)
  - 2.3.3 Tests de la structure / architecture logicielle (tests structurels) (K2)
  - 2.3.4 Tests liés au changement (tests de confirmation et de régression) (K2)
- 2.4 Tests de maintenance (K2)