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

Séquentiel

Modèle en cascade: • Les activités de développement sont réalisées l'une après l'autre.

•Dans ce modèle, les activités de test ont seulement lieu une fois que toutes les autres activités de développement sont terminées.

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

Itérative et incrémental

Le développement incrémental: • Le développement incrémental implique la définition des exigences, la conception, le développement et le test d'un système par morceaux, ce qui signifie que les fonctionnalités du logiciel augmentent de façon incrémentale.

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

Rational Unified Process: chaque itération a tendance à être relativement longue (p. ex. deux à trois mois), et les incréments de caractéristiques sont proportionnellement importants, de l'ordre de deux ou trois groupes de caractéristiques connexes.

Scrum: Chaque itération a tendance à être relativement courte, et les incréments de caractéristiques sont proportionnellement petits, de l'ordre de quelques améliorations et/ou deux ou trois nouvelles caractéristiques.

Kanban: Mis en œuvre avec des itérations de durée fixe ou non, pouvant soit livrer à la fin une seule amélioration ou caractéristique, soit regrouper des groupes de caractéristiques pour les livrer en une fois.

Spiral (ou par prototypage): il s'agit de créer des incréments expérimentaux, dont certains peuvent être fortement remaniés ou même abandonnés lors de travaux de développement ultérieurs.

La livraison continue: les équipes utilisent la livraison continue ou le déploiement continu, les deux impliquant une automatisation significative de multiples niveaux de test dans le cadre de leurs flux de livraison.

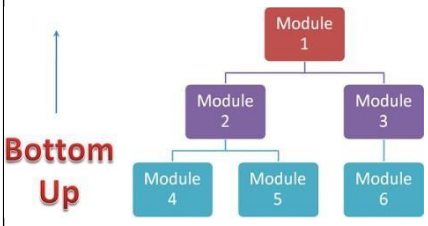
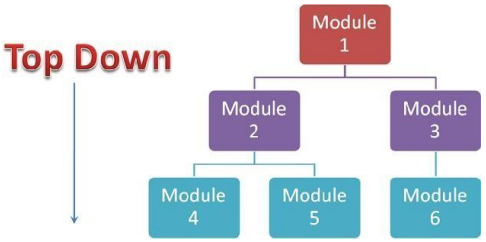
Niveau de test	Test de composant	Test d'intégration	Test de système	Test D'acceptance
Objectifs	•Réduire le risque •Vérifier si les comportements fonctionnels et non-fonctionnels • Renforcer la confiance • Trouver des défauts •Empêcher les défauts	Réduire le risque • Vérifier si les comportements fonctionnels et non-fonctionnels • Renforcer la confiance • Trouver des défauts • Empêcher les défauts	Réduire le risque • Vérifier si les comportements fonctionnels et non-fonctionnels • Renforcer la confiance • Trouver des défauts • Empêcher les défauts	. • Vérifier si les comportements fonctionnels et non-fonctionnels • Établir la confiance
Bases de test	•Conception détaillée • Code • Modèle de données •Spécifications des composants	•Conception du logiciel et du système • Diagrammes de séquence • Spécifications des protocoles d'interface et de communication • Cas d'utilisation• Workflows	•Spécifications des exigences système et logicielles • Rapports d'analyse des risques • Cas d'utilisation • Epics et User Stories • Modèles de comportement du système • Diagrammes d'états	•Processus métier • Exigences utilisateur ou métier • Réglementations, contrats légaux et normes • Cas d'utilisation • Exigences système • Documentation système ou utilisateur • Procédures d'installation • Rapports d'analyse des risques
Objet de test	•Composants, unités ou modules • Code et structures de données • Classes • Modules de bases de données	•Sous-systèmes • Bases de données • Infrastructure • Interfaces • APIs • Micro-services	•Applications • Systèmes Matériel/Logiciel • Systèmes d'exploitation• Configuration du système et données de configuration	•Systèmes sous test • Systèmes de récupération et sites sensibles •Processus d'exploitation et de maintenance • Formulaires • Rapports
Défauts & défaillances	Fonctionnalité incorrecte • Problèmes de flux de donnée s •Code et logique incorrects	•Données incorrectes• Décalages au niveau des interfaces • Défaillances dans la communication	•Calculs incorrects • Comportement fonctionnel ou non-fonctionnel incorrect ou inattendu	•Les workflows du système• les règles de Business • Contrat •Les défaillances non-fonctionnelles
Approches & responsabilités	. Fait par les développeurs / TDD	Fait par les testeurs / Buttom up-Top Down - big bang-stubs-drivers	. Fait par le testeurs	Fait par les testeurs/des clients/ des utilisateurs métier

Approches, Stratégies, Méthodologies de tests d'intégration

Approche de Big Bang:

Approche	Avantages	Inconvénients
Ici, tous les composants sont intégrés ensemble à la fois, puis tester.	. Pratique pour les petits systèmes.	. La localisation des pannes est difficile. . Puisque les tests d'intégration ne peuvent commencer qu'après que "tous" les modules sont conçus, l'équipe de test aura moins de temps pour l'exécution dans la phase de test. . Les modules critiques à haut risque ne sont pas isolés et testés en priorité.

Approche Incrémental: les tests sont effectués en joignant deux ou plusieurs modules qui sont logiquement liés.

Approches	Avantages	Inconvénients
Intégration Bottom-up Dans la l'approche bottom-up, chaque module aux niveaux inférieurs est testé avec des modules supérieurs jusqu'à ce que tous les modules soient testés. Il faut l'aide de Driver pour les tests 	. La localisation des pannes est plus facile. . Pas de temps perdu à attendre que tous les modules soient développés contrairement à l'approche Big-bang	. Les modules critiques (au plus haut niveau de l'architecture logicielle) qui contrôlent le flux d'application sont testés en dernier et peuvent être sujets à des défauts. . Un premier prototype n'est pas possible
Intégration Top-down: Dans l'approche Top down, les tests ont lieu de haut en bas en suivant le flux de contrôle du système logiciel. Prend l'aide de stubs pour les tests. 	La localisation des pannes est plus facile. .Possibilité d'obtenir un prototype précoce. Les modules critiques sont testés en priorité; des défauts de conception majeurs ont pu être trouvés et corrigés en premier.	. Nécessite de nombreux stubs. . Les modules à un niveau inférieur sont testés de manière inadéquate.

Stub: Est appelé par le module sous test. **Driver:** Appelle le module à tester.

Types de test	Définition	Couverture
Test Fonctionnels	•Les tests fonctionnels d'un système impliquent des tests qui évaluent les fonctionnalités que le système devrait réaliser. •Les fonctionnalités sont "ce que" le système doit faire. • Les tests fonctionnels devraient être effectués à tous les niveaux de test.	•La couverture fonctionnelle est la mesure selon laquelle un certain type d'élément fonctionnel a été exercé par des tests. Elle est exprimée en pourcentage du ou des types d'éléments couverts.
Test Non-Fonctionnels	•Les tests non-fonctionnels d'un système évaluent les caractéristiques des systèmes et des logiciels comme l'utilisabilité, la performance ou la sécurité. •Le test nonfonctionnel est le test de "comment" le système se comporte. • les tests non-fonctionnels peuvent et devraient souvent être effectués à tous les niveaux de test.	•La couverture non-fonctionnelle est la mesure selon laquelle un certain type d'élément non-fonctionnel a été exercé par des tests et est exprimée en pourcentage du ou des types d'éléments couverts.
Tests boîte-blanche	• Les tests boîte-blanche sont des tests basés sur la structure ou l'implémentation interne du système.	•La couverture structurelle est la mesure dans laquelle un certain type d'élément structurel a été exercé par des tests et est exprimée en pourcentage du type d'élément couvert.

Test de Confirmation: Après la correction d'un défaut, le logiciel doit être testé

Le but d'un test de confirmation est de confirmer que le défaut d'origine a été réparé avec succès.

Test de Régression : • Les tests de régression sont des tests visant à détecter de tels effets de bord involontaires.

Test de Maintenance Testing: • Les tests de maintenance se concentrent sur le test des changements apportés au système, ainsi que sur le test des parties inchangées qui auraient pu être affectées par les changements.

Facteurs déclencheurs pour la maintenance: Modification / Migration / Suppression.

⇨**Analyse d'impact** évalue les changements qui ont été apportés pour une version de maintenance afin d'identifier les conséquences imprévues.