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

2.1 Les modèles de développement logiciel

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

Il existe un certain nombre de modèles de cycle de développement de logiciels différents, chacun d'entre eux nécessitant des approches de test différentes.

2.1.1 Développement de logiciel et tests logiciels

Une partie importante du rôle d'un testeur est de se familiariser avec les principaux modèles de cycle de vie du développement logiciel afin que les activités de test adaptées puissent être réalisées.

Quel que soit le modèle de cycle de vie de développement logiciel, il y a plusieurs caractéristiques de bonnes pratiques des tests :

- Pour chaque activité de développement, il y a une activité de test correspondante
- Chaque niveau de test a des objectifs de test spécifiques à ce niveau
- L'analyse et la conception des tests pour un niveau de test donné commencent au cours de l'activité de développement correspondante
- Les testeurs participent aux discussions pour définir et affiner les exigences et la conception, et sont impliqués dans la revue des produits d'activités (p. ex. les exigences, la conception, les User Stories, etc.) dès que des versions préliminaires sont disponibles

Les modèles de cycle de vie de développement de logiciels courants sont :

- Modèles de développement séquentiel
- Modèles de développement itératif et incrémental

Un modèle de développement séquentiel décrit le processus de développement logiciel comme un flux linéaire et séquentiel d'activités. Cela signifie que toute phase du processus de développement devrait commencer lorsque la phase précédente est terminée.

En théorie, il n'y a pas de chevauchement des phases, mais dans la pratique, il est bénéfique d'avoir un retour rapide de la phase suivante.

Le développement incrémental implique la définition des exigences, la conception, le développement et le test d'un système par morceaux, ce qui signifie que les fonctionnalités du logiciel augmentent de façon incrémentale. La taille de ces incréments de fonctionnalités varie, certaines méthodes ayant des étapes plus grandes et d'autres plus petites.

2.1.2 Modèles de cycle de vie du développement logiciel en contexte

Les modèles de cycle de vie du développement logiciel doivent être sélectionnés et adaptés au contexte du projet et aux caractéristiques du produit.

Par exemple, le développement et le test d'un système peu important de gestion interne devrait être différent du développement et du test d'un système critique pour la sécurité, comme un système de contrôle de freinage d'une automobile.

2.2 Niveaux de test

Les niveaux de test sont des groupes d'activités de test qui sont organisées et gérées ensemble. Chaque niveau de test est une instance du processus de test, constitué des activités décrites à la section 1.4, réalisées en relation avec le logiciel à un niveau de développement donné, depuis les unités ou composants individuels jusqu'aux systèmes complets ou, le cas échéant, aux systèmes de systèmes.

Les niveaux de test sont liés à d'autres activités du cycle de vie du développement logiciel. Les niveaux de test utilisés dans ce syllabus sont les suivants :

- Test de composants
- Test d’intégration
- Test système
- Test d’acceptation

	Test de composants	Test d'intégration	Test système	Test d'acceptation
Définition	Les tests de composants (tests unitaires) se concentrent sur des composants qui peuvent être testés séparément.	Les tests d'intégration se concentrent sur les interactions entre les composants ou les systèmes. Il y a deux niveaux différents de tests d'intégration : • Les tests d'intégration de composants se concentrent sur les interactions et les interfaces entre composants intégrés. • Les tests d'intégration de systèmes se concentrent sur les interactions et les interfaces entre systèmes, progiciels et micro-services. Les tests d'intégration de systèmes peuvent également couvrir les interactions avec des entités externes (p. ex. services Web)	Les tests système se concentrent sur le comportement et les capacités d'un système ou d'un produit entier, en considérant souvent les tâches de bout en bout que le système peut exécuter et les comportements non-fonctionnels qu'il présente pendant l'exécution de ces tâches.	Les tests d'acceptation, comme les tests système, se concentrent généralement sur le comportement et les capacités d'un système ou d'un produit entier.
Objectifs	• Réduire le risque • Vérifier si les comportements fonctionnels et non-fonctionnels du composant	• Réduire le risque • Vérifier si les comportements fonctionnels et non-	• Réduire les risques • Vérifier si les comportements fonctionnels et non-	• Établir la confiance dans la qualité du système dans son ensemble

	sont tels qu'ils ont été conçus et spécifiés. • Renforcer la confiance dans la qualité du composant. • Trouver des défauts dans le composant. • Empêcher les défauts de passer à des niveaux de test plus élevés.	fonctionnels des interfaces sont tels qu'ils ont été conçus et spécifiés. • Renforcer la confiance dans la qualité des interfaces. • Trouver des défauts. • Empêcher les défauts de passer à des niveaux de test plus élevés.	fonctionnels du système sont tels qu'ils ont été conçus et spécifiés. • Valider que le système est complet et qu'il fonctionnera comme prévu • Renforcer la confiance dans la qualité du système dans son ensemble • Trouver des défauts • Empêcher les défauts de passer à des niveaux de test plus élevés ou en production	• Valider que le système est complet et qu'il fonctionnera comme attendu • Vérifier que les comportements fonctionnels et non-fonctionnels du système sont tels que spécifiés.
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 • Architecture au niveau du composant ou du système • Workflows • Définitions des interfaces externes	• Spécifications des exigences système et logicielles (fonctionnelles et non-fonctionnelles) • Rapports d'analyse des risques • Cas d'utilisation • Epics et User Stories • Modèles de comportement du système • Diagrammes d'états • Manuels système et manuels d'utilisation	• Processus métier • Exigences utilisateur ou métier • Réglementations, contrats légaux et normes • Cas d'utilisation • Procédures de sauvegarde et de restauration • Procédures de reprise après sinistre • Exigences non-fonctionnelles • Documentation d'exploitation……
Objets de test	• Composants, unités ou modules	• Sous-systèmes • Bases de données	• Applications	• Processus d'exploitation et de maintenance

	• Code et structures de données • Classes • Modules de bases de données	• Infrastructure • Interfaces • APIs • Micro-services	• Systèmes Matériel/Logiciel • Systèmes d'exploitation • Configuration du système et données de configuration	• Formulaires • Rapports • Données de production existantes et modifiées
Défauts et défaillances courants	• Fonctionnalité incorrecte (par exemple, non conforme aux spécifications de conception) • Problèmes de flux de données • Code et logique incorrects	• Décalages au niveau des interfaces • Défaillances dans la communication entre les composants ou systèmes. • …	• Calculs incorrects • Comportement fonctionnel ou non-fonctionnel du système incorrect ou inattendu • Flux de contrôle et/ou de données incorrects au sein du système • …..	• Les règles métier ne sont pas correctement implémentées • Le système ne satisfait pas aux exigences contractuelles ou réglementaires • Les défaillances non-fonctionnelles telles que les vulnérabilités de sécurité, le manque de performance en cas de charges élevées,
Approches spécifiques et responsabilités	• Les tests de composants sont généralement effectués par le développeur qui a écrit le code. • Les développeurs peuvent alterner le développement des composants avec la recherche et la correction de défauts.	• Les tests d'intégration de composants sont souvent la responsabilité des développeurs. • Les tests d'intégration de systèmes relèvent généralement de la responsabilité des testeurs.	• Les tests système devraient se concentrer sur le comportement global de bout en bout du système dans son ensemble, à la fois fonctionnel et non-fonctionnel. • Des testeurs indépendants procèdent en général aux tests système.	Les tests d'acceptation relèvent souvent de la responsabilité des clients, des utilisateurs métier, des Product Owners ou des exploitants d'un système, et d'autres parties prenantes pouvant également être impliquées.

2.3 Types de test

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

2.3.1 Tests fonctionnels

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

Les tests fonctionnels devraient être effectués à tous les niveaux de test (par exemple, les tests de composants peuvent être basés sur une spécification de composant), bien que la focalisation soit différente à chaque niveau.

La complétude des tests fonctionnels peut être mesurée par la couverture fonctionnelle. 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.

2.3.2 Tests non-fonctionnels

Les tests non-fonctionnels d'un système évaluent les caractéristiques des systèmes et des logiciels comme l'utilisabilité, la performance ou la sécurité. Il convient de se reporter à la norme ISO (ISO/CEI 25010) pour une classification des caractéristiques de qualité des produits logiciels. Le test non-fonctionnel est le test de "comment" le système se comporte.

Contrairement aux idées fausses courantes, les tests non-fonctionnels peuvent et devraient souvent être effectués à tous les niveaux de test, et ce, le plus tôt possible.

La complétude des tests non-fonctionnels peut être mesurée par une couverture non-fonctionnelle. 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.

2.3.3 Tests boîte-blanche

Les tests boîte-blanche sont des tests basés sur **la structure ou l'implémentation interne du système.** La structure interne peut comprendre le code, l'architecture, les flux de travail et/ou les flux de données au sein du système (voir section 4.3).
La complétude des tests boîte-blanche peut être mesurée par la couverture structurelle. 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.
La conception et l'exécution de tests boîte-blanche peuvent nécessiter des compétences ou des connaissances particulières, comme la façon dont le code est construit (p. ex. utiliser des outils de couverture de code), la façon dont les données sont stockées (p. ex. évaluer les requêtes

possibles dans les bases de données) et la façon d'utiliser les outils de couverture et d'interpréter correctement leurs résultats.

2.3.4 Tests liés aux changements

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

- **Test de confirmation** : Après la correction d'un défaut, le logiciel peut être testé avec tous les cas de test qui ont échoué en raison du défaut, qui doivent être ré-exécutés sur la nouvelle version du logiciel. Le logiciel peut également être testé avec de nouveaux. Le but d'un test de confirmation est de confirmer que le défaut d'origine a été réparé avec succès.

- **Tests de régression** : Il est possible qu'une modification apportée à une partie du code, qu'il s'agisse d'une correction ou d'un autre type de modification, puisse accidentellement affecter le comportement d'autres parties du code, que ce soit au sein du même composant, dans d'autres composants du même système ou même dans d'autres systèmes. Les changements peuvent inclure des changements de l'environnement, comme une nouvelle version d'un système d'exploitation ou d'un système de gestion de base de données. **De tels effets de bord involontaires sont appelés régressions**. Les tests de régression sont des tests visant à détecter de tels effets de bord involontaires.

2.3.5 Types de test et niveaux de test

Il est possible d'effectuer n'importe quel type de test mentionné ci-dessus à n'importe quel niveau de test. Pour illustrer, des exemples de tests fonctionnels, non-fonctionnels, boîte blanche et liés au changement seront donnés pour tous les niveaux de test. (Exemples voir P43, P44).

2.4 Tests de maintenance

Une fois déployés dans les environnements de production, les logiciels et les systèmes doivent être maintenus. Des changements de diverses sortes sont presque inévitables dans les logiciels et les systèmes livrés :

- soit pour corriger des défauts découverts lors de l'utilisation opérationnelle,
- soit pour ajouter de nouvelles fonctionnalités,
- soit pour supprimer ou modifier des fonctionnalités déjà livrées.
- préserver ou améliorer les caractéristiques de qualité non-fonctionnelles du composant ou du système (la performance, la compatibilité, la fiabilité, la sécurité et la portabilité).

Lorsque des modifications sont apportées dans le cadre de la maintenance, des tests de maintenance devraient être effectués, à la fois pour évaluer le succès avec lequel les modifications ont été apportées et pour vérifier les effets secondaires possibles (p. ex. régressions) dans les parties du système qui demeurent inchangées (ce qui est habituellement la plus grande partie du système). 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.

La maintenance peut impliquer des versions planifiées et des versions non planifiées (corrections à chaud).

2.4.1 Facteurs déclencheurs pour la maintenance

Les facteurs déclencheurs de la maintenance :

- Modification, comme des améliorations planifiées (p. ex., basées sur les versions), des changements correctifs et d'urgence, …
- Migration, par exemple d'une plate-forme à une autre.
- Déclassement, par exemple lorsqu'une application arrive en fin de vie.

Lorsqu'une application ou un système est mis hors service, il peut être nécessaire de tester la migration ou l'archivage des données si de longues périodes de conservation des données sont nécessaires.

Il peut également être nécessaire de tester les procédures de restauration/récupération après l'archivage pour de longues périodes de conservation.

2.4.2 Analyse d'impact pour la maintenance

L'analyse d'impact évalue les changements qui ont été apportés pour une version de maintenance afin d'identifier les conséquences prévues ainsi que des effets secondaires attendus et possibles d'un changement, et d'identifier les zones du système qui seront affectées par le changement.