# Testeur certifié

# Syllabus niveau avancé
# Analyste de test

Version 2019

International Software Testing Qualifications Board

Sous-groupe de travail Syllabus niveau avancé – Analyste de test: Judy McKay (Responsable), Graham Bath

Pour le syllabus 2012: Judy McKay, Mike Smith, Erik van Veenendaal

Pour le syllabus 2019: Graham Bath, Judy McKay, Jan Sabak, Erik van Veenendaal

Traduction française : Olivier DENOO, Eric RIOU du COSQUER et Bruno LEGEARD

# Historique des modifications

| Version | Date | Remarques |
|---|---|---|
| V2012 | 19.10.2012 | Livraison ISTQB® |
| V2019 V1.0 | 18.10.2019 | Livraison ISTQB® |
| V2019 V1.1 | 19.12.2019 | Lancement ISTQB® |
| V2019 V1.1 FR | 22.06.2020 | Version française |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Table des matières

# Remerciements

Ce document a été produit par une équipe du groupe de travail sur le niveau avancé de l'International Software Testing Qualifications Board: Graham Bath, Judy McKay, Mike Smith

L'équipe principale remercie l'équipe de revue et les comités nationaux pour leurs suggestions et contributions.

Les personnes suivantes ont participé à la revue, aux commentaires et aux choix d'évolutions de ce syllabus:

Laura Albert
Chris Van Bael
Markus Beck
Henriett Braunné Bokor
Guo Chaonian
Wim Decoutere
Milena Donato
Klaudia Dussa-Zieger
Melinda Eckrich-Brajer
Péter Földházi Jr
David Frei
Chen Geng
Matthias Hamburg
Zsolt Hargitai
Zhai Hongbao
Tobias Horn
Ágota Horváth
Beata Karpinska
Ramit Manohar Kaul
Jan te Kock
József Kreisz
Dietrich Leimsner
Ren Liang
Claire Lohr
Attila Kovács
Rik Marselis
Marton Matyas
Don Mills
Blair Mo
Gary Mogyorodi
Ingvar Nordström
Francisca Cano Ortiz
Pálma Polyák
Meile Posthuma
Lloyd Roden
Adam Roman
Abhishek Sharma
Péter Sótér
Lucjan Stapp
Andrea Szabó
Benjamin Timmermans
Erik van Veenendaal
Jan Versmissen
Carsten Weise
Robert Werkhoven
Paul Weymouth

Ce document a été officiellement publié par l'assemblée générale de l'ISTQB® le 18 octobre 2019.

# 0. Introduction à ce syllabus

## 0.1 Objectif de ce syllabus

Ce syllabus constitue la base pour le niveau avancé Analyste de Test de l'International Software Testing Qualifications Board. L'ISTQB® fournit ce syllabus:

1. Aux comités nationaux, pour le traduire dans leur langue et pour accréditer des organismes de formation. Les comités nationaux peuvent adapter le syllabus aux besoins particuliers de leur langue et modifier les références pour prendre en compte des publications locales.
2. Aux fournisseurs d'examens pour créer des questions dans leur langue selon les objectifs d'apprentissage de ce syllabus.
3. Aux organismes de formation, pour produire le matériel de formation et déterminer les méthodes d'enseignement adaptées.
4. Aux candidats à la certification, pour se préparer à l'examen (lors d'une formation ou de façon indépendante).
5. A la communauté internationale de l'ingénierie des logiciels et des systèmes pour faire progresser les métiers du test de systèmes et logiciels et pour servir de base à la rédaction de livres et articles.

L'ISTQB® peut permettre à d'autres organismes d'utiliser ce syllabus pour d'autres raisons, à condition qu'ils en fassent préalablement la demande et obtiennent une réponse écrite.

## 0.2 Le niveau avancé testeur certifié en test logiciel

Le niveau avancé principal est composé de trois syllabus indépendants correspondant aux rôles suivants:

- Test manager
- Analyste de test
- Analyste technique de test

La vue générale sur le niveau avancé 2019 de l'ISTQB est un document séparé [ISTQB_AL_OVIEW] qui contient les informations suivantes:

- Bénéfices métier pour chaque syllabus
- Matrice montrant la traçabilité entre les bénéfices métier et les objectifs d'apprentissage
- Résumé de chaque syllabus
- Relations entre les syllabus

## 0.3 Objectifs d'apprentissage examinables et niveaux de connaissance

Les objectifs d'apprentissage couvrent les objectifs métier et sont utilisés pour créer l'examen de certification Analyste de Test niveau avancé.

Les niveaux de connaissance K2, K3 et K4 des objectifs d'apprentissage sont présentés au début de chaque chapitre et classés de la façon suivante::

- K2: Comprendre
- K3: Appliquer
- K4: Analyser

Les définitions de tous les termes listés comme mots-clés à la suite les titres de chapitres doivent être mémorisées (K1), même si cela n'est pas explicitement mentionné dans les objectifs d'apprentissage.

## 0.4 L'examen analyste de test niveau avancé

L'examen analyste de test niveau avancé est basé sur ce syllabus. Les réponses aux questions de l'examen peuvent nécessiter d'utiliser des éléments présents dans plusieurs sections de ce syllabus. Chaque section peut donner lieu à des questions, à l'exception de l'introduction et des annexes. Des standards, ouvrages et autres syllabus ISTQB sont inclus comme références, mais leur contenu ne peut pas donner lieu à des questions au-delà de ce qu'il en est résumé dans le syllabus..

Le format de l'examen est un questionnaire à choix multiples contenant 40 questions. Pour réussir l'examen il est nécessaire de répondre correctement à au moins 65% des questions.

Les examens peuvent être passés lors d'une formation accréditée ou de façon indépendante (par exemple dans un centre d'examen agréé ou lors d'un examen public).

## 0.5 Prérequis au passage de l'examen

Le certificat testeur certifié niveau fondation doit obligatoirement avoir été obtenu avant le passage de l'examen de certification analyste de test niveau avancé.

## 0.6 Expérience requise

Aucun des objectifs d'apprentissage pour l'analyste de test niveau avancé ne requiert une expérience particulière.

## 0.7 Accréditation des formations

Un comité national membre de l'ISTQB peut accréditer des organismes de formation dont le matériel de formation suit ce syllabus. Les organismes de formation peuvent obtenir les consignes d'accréditation auprès du Comité Français des Tests Logiciels ou d'un autre comité pouvant réaliser une accréditation. Un cours accrédité est officiellement reconnu conforme au syllabus et permet à l'organisme de formation d'organiser l'examen ISTQB au moment de la formation.

## 0.8 Niveau de détail du syllabus

Le niveau de détail de ce syllabus permet la création, au niveau international, de cours et examens cohérents. Pour cela, le syllabus est constitué des éléments suivants:

- Des objectifs généraux de formation décrivant l'intention de l'Analyste de Test Niveau Avancé
- Une liste d'éléments dont les étudiants doivent être capables de se souvenir.
- Des objectifs d'apprentissage pour chaque domaine de connaissance, décrivant les bénéfices à obtenir ou des standards.
- Une description des concepts clé, incluant des références vers des sources telles que des écrits reconnus ou des standards

Le contenu du syllabus n'est pas une description exhaustive du domaine de connaissance; il reflète le niveau de détail à couvrir dans les formations du Niveau Avancé. Il se concentre sur les éléments pouvant être appliqués à tout projet logiciel, y compris les projets en mode Agile. Le syllabus ne contient pas d'objectif d'apprentissage spécifique à un cycle de vie de développement logiciel

particulier mais il traite la question de la mise en œuvre de ces concepts dans les projets Agiles, dans les projets suivant un autre mode itératif et incrémental, et dans les cycles de vie séquentiel.

## 0.9 Organisation de ce syllabus

Il y a six chapitres sur lesquels peuvent porter des questions. Le titre principal de chaque chapitre indique le temps minimal nécessaire à la formation et aux exercices pour ce chapitre, les durées ne sont pas indiquées aux niveaux inférieurs. Pour une formation accréditée, le syllabus nécessite un minimum de 20 heures et 30 minutes de formation, réparties de la façon suivante:

- Chapitre 1: Les tâches de l'Analyste de Test dans le processus de test (150 minutes)
- Chapitre 2: Les tâches de l'Analyste de Test dans le test basé sur les risques (60 minutes)
- Chapitre 3: Techniques de test (630 minutes)
- Chapitre 4: Test des caractéristiques qualité du logiciel (180 minutes)
- Chapitre 5: Revues (120 minutes)
- Chapitre 6: Outils de test et automatisation (90 minutes)

# 1.Les tâches de l'Analyste de Test dans le processus de test - 150 mins.

**Mots clés**
critère de sortie, cas de test de haut niveau, cas de test de bas niveau, test, analyse des tests, base de test, condition de test, données de test , conception des tests, exécution des tests, planning d'exécution des tests, implémentation des tests, procédure de test,, suite de tests

**Objectifs d'apprentissage pour Les tâches de l'Analyste de Test dans le processus de test**

## 1.1 Introduction
Pas d'objectifs d'apprentissage

## 1.2 Le test dans le cycle de vie de développement logiciel
TA-1.2.1 (K2) Expliquer comment et pourquoi le moment et le niveau d'implication de l'Analyste de test varient en fonction des différents modèles de cycles de vie de développement logiciel en place

## 1.3 Analyse des tests
TA-1.3.1 (K2) Résumer les tâches adéquates pour l'Analyste de Test lors des activités d'analyse

## 1.4 Conception des tests
TA-1.4.1 (K2) Expliquer pourquoi les conditions de test doivent être comprises par les parties prenantes
TA-1.4.2 (K4) Pour un scénario de projet donné, sélectionner le niveau de conception adapté pour les cas de test (haut niveau ou bas niveau)
TA-1.4.3 (K2) Expliquer les problèmes à aborder lors de la conception des tests

## 1.5 Implémentation des tests
TA-1.5.1 (K2) Résumer les tâches adéquates pour l'Analyste de Test lors des activités d'implémentation des tests

## 1.6 Exécution des tests
TA-1.6.1 (K2) Résumer les tâches adéquates pour l'Analyste de Test lors des activités d'exécution des tests

## 1.1 Introduction

Dans le syllabus ISTQB® du Niveau Fondation, le processus de test est décrit avec les activités suivantes :

- Planification des tests
- Suivi et contrôle des tests
- Analyse de test
- Conception des tests
- Implémentation des tests
- Exécution des tests
- Clôture des tests

Dans de syllabus du Niveau Avancé, les activités les plus pertinentes pour l'Analyste de Test sont décrites avec plus de détail. Des précisions supplémentaires sur le processus de test sont apportées afin de mieux couvrir les différents modèles de cycle de vie de développement logiciel

Déterminer les tests appropriés, les concevoir et les implémenter, puis les exécuter, sont les principales activités ciblées par l'Analyste de Test. Bien qu'il soit important de comprendre les autres étapes dans le processus de test, la majeure partie du travail de l'Analyste de Test porte généralement sur les activités suivantes :

- Analyse de test
- Conception des tests
- Implémentation des tests
- Exécution des tests

Les autres activités du processus de test sont décrites de façon adéquate au Niveau Fondation et ne nécessitent pas plus de détail à ce niveau.

## 1.2 Le test dans le cycle de vie de développement logiciel

Le cycle de vie de développement logiciel global devrait être considéré lors de la définition d'une stratégie de test. Le moment auquel l'Analyste de Test est impliqué varie d'un cycle de vie à un autre ; la nature de l'implication, le temps nécessaire, l'information disponible et les attentes peuvent également être très différents. L'Analyste de Test doit être au courant des types d'informations à apporter aux autres rôles tels que:

- Ingénierie et gestion des exigences – revues d'exigences
- Gestion de projet – apport d'information pour le planning
- Gestion de la configuration et du changement - test de la vérification du build, contrôle de version
- Développement logiciel – notification des défauts trouvés
- Maintenance logicielle – gestion des défauts, temps consacré (par ex., temps pour détecter et documenter les anomalies, et ensuite le temps pour réaliser et documenter les tests de confirmation)
- Support technique – documentation précise des solutions de contournement et des problèmes connus
- Production de la documentation technique (par ex. : spécifications de conception de base de données, documentation de l'environnement de test) – entrées mais aussi revue technique pour ces documents)

Les activités de test doivent être alignées avec le cycle de vie de développement logiciel choisi qui peut être séquentiel, itératif, incrémental ou un mélange hybride des deux. Par exemple, dans le

modèle séquentiel en V, le processus de test appliqué au niveau test système peut se dérouler de la façon suivante :

- La planification des tests système se déroule en même temps que la planification du projet, et le suivi et le contrôle des tests continuent jusqu'à la fin des tests. Cela aura une influence sur les données de planification fournies par l'Analyste de Test pour la gestion de projet.
- L'analyse et la conception des tests système se basent sur des documents tels que la spécification des exigences système, la spécification de conception du système et de l'architecture (haut niveau) et la spécification de conception des composants (bas niveau).
- L'implémentation de l'environnement de test système peut démarrer lors de la conception du système, même si cela peut se faire en même temps que le codage et le test de composants, avec un travail sur les activités d'implémentation des tests système qui se déroule souvent jusqu'à quelques jours avant le début de leur exécution
- L'exécution des tests système commence lorsque tous les critères d'entrée des tests système sont satisfaits (ou que certains sont écartés), ce qui signifie en général qu'au moins les tests de composants et souvent aussi les tests d'intégration de composants ont satisfait leurs critères de sortie ou « definition of done » (définition du terminé). L'exécution des tests système continue jusqu'à ce que les critères de sortie des tests système soient satisfaits.
- Les activités de clôture des tests système se déroulent après que les critères de sortie des tests système aient été satisfaits.

Les modèles itératifs et les modèles incrémentaux peuvent ne pas suivre le même ordre d'activités et peuvent exclure certaines activités. Par exemple, un modèle itératif peut utiliser un ensemble réduit d'activités de test pour chaque itération. L'analyse, la conception, l'implémentation et l'exécution peuvent être effectuées pour chaque itération, alors que la planification de haut niveau n'est faite qu'au début du projet et les tâches de clôture à la fin.

Dans un projet Agile, il est commun d'utiliser un processus moins formalisé et une relation de travail beaucoup plus proche avec les parties prenantes du projet, ce qui permet aux changements de se produire plus facilement dans le projet. Il peut ne pas y avoir un rôle bien défini d'Analyste de Test. La documentation des tests est moins complète, et la communication est plus fréquente mais moins longue.

Les projets Agiles impliquent le test dès le départ. Cela commence au lancement du projet quand les développeurs réalisent leur travail initial d'architecture et de conception. Les revues peuvent ne pas être formalisées mais sont continues lors de l'évolution du logiciel. On s'attend à une implication tout au long du projet et les tâches de l'Analyste de Test devraient être faites par l'équipe.

Les modèles itératifs et incrémentaux vont de l'approche Agile, où l'on s'attend à des changements lors de l'évolution du logiciel, aux modèles de développement itératif/incrémental à l'intérieur d'un cycle en V (parfois appelés itératifs intégrés). Dans un modèle itératif intégré, l'Analyste de Test doit s'attendre à être impliqué dans certains aspects de la planification et de la conception, mais doit ensuite s'orienter vers un rôle plus interactif lorsque le logiciel est conçu, développé et testé.

Quel que soit le cycle de vie de développement logiciel utilisé, il est important pour l'Analyste de Test de comprendre les attentes relatives à son implication ainsi que les moments auxquels il doit s'impliquer. De nombreux modèles hybrides sont utilisés, comme les modèle itératifs intégrés mentionnés plus haut. Les Analystes de Test doivent déterminer leur rôle le plus efficace. Ils doivent travailler de façon proactive sur le meilleur moment de leur implication plutôt qu'être dépendant d'un modèle prédéfini.

## 1.3 Analyse de test

Lors de la planification des tests, le périmètre du projet de test est défini. Les Analystes de Test utilisent cette définition du périmètre pour:

- Analyser la base de test
- Identifier des défauts de différents types dans la base de test
- Identifier et prioriser les conditions de test et les caractéristiques à tester
- Etablir la traçabilité entre chaque élément de la base de test et les conditions de test associées
- Effectuer les tâches en lien avec le test basé sur les risques (voir Chapitre 2)

Pour que les Analystes de Test puissent mener efficacement l'analyse de test, les critères d'entrée suivants doivent être satisfaits :

- Un ensemble de connaissances (par ex. des exigences ou des User Stories) décrivant l'objet de test peut servir de base de test (voir [ISTQB_FL_SYL] Section 1.4.2 pour une liste d'autres sources de base de test possibles).
- Cette base de test a été revue avec des résultats raisonnables et a été mise à jour en conséquence après la revue. Notez que si des cas de test de haut niveau doivent être définis (voir Section 1.4.1), il peut ne pas être nécessaire de définir entièrement la base de test à ce moment-là. Dans un projet Agile, ce cycle de revue sera itératif avec le raffinement des User Stories au début de chaque itération.
- Un budget et un planning approuvés sont disponibles pour l'accomplissement des tâches de test restantes pour cet objet de test.

Les conditions de test sont généralement identifiées par l'analyse de la base de test en conjonction avec les objectifs de test (tels que définis dans la planification des tests). Dans certaines situations où la documentation peut être ancienne ou inexistante, les conditions de test peuvent être identifiées par discussion avec les parties prenantes concernées (p. ex., dans les ateliers ou pendant la planification de l'itération). Dans un projet Agile, les critères d'acceptation qui sont définis dans des User Stories sont souvent utilisés comme base pour la conception des tests.

Bien que les conditions de test soient généralement spécifiques à l'élément testé, il existe certaines considérations standard pour l'Analyste de Test.

- Il est généralement conseillé de définir les conditions de test à différents niveaux de détail. Initialement, les conditions de haut niveau sont identifiées pour définir des cibles générales pour les tests, telles que «fonctionnalité de l'écran x». Par la suite, des conditions plus détaillées sont identifiées comme la base de cas de test spécifiques, tels que « l'écran x rejette un numéro de compte qui a un chiffre de moins que la longueur correcte ». L'utilisation de ce type d'approche hiérarchique pour définir les conditions de test peut aider à s'assurer que la couverture est suffisante pour les éléments de haut niveau. Cette approche permet également à un Analyste de Test de commencer à travailler sur la définition des conditions de test de haut niveau pour les User Stories qui n'ont pas encore été affinées.
- Si les risques produit ont été définis, les conditions de test qui seront nécessaires pour répondre à chaque risque produit doivent être identifiées et liées à cet élément de risque.

L'application de techniques de test (telles qu'elles sont identifiées dans la stratégie de test et/ou dans le plan de test) peut être utile dans le processus d'analyse de test et peut être utilisée pour contribuer aux objectifs suivants:

- Identifier les conditions de test
- Réduire la probabilité d'omettre des conditions de test importantes
- Définir des conditions de test plus précises et justes
- Après l'identification et le raffinement des conditions de test, une revue de ces conditions avec les parties prenantes peut être menée pour s'assurer que les exigences ont été comprises clairement et que le test est aligné avec les objectifs du projet.

À la fin des activités d'analyse de test pour un domaine donné (p. ex., une fonction spécifique), l'Analyste de Test doit savoir quels tests spécifiques doivent être conçus pour ce domaine.

## 1.4 Conception des tests

Toujours en phase avec le périmètre déterminé lors de la planification des tests, le processus de test se poursuit au fur et à mesure que l'Analyste de Test conçoit les tests qui seront implémentés et exécutés. La conception des tests comprend les activités suivantes:

- Déterminer dans quels domaines de test les cas de test de bas ou de haut niveau sont appropriés
- Déterminer la ou les technique(s) de test qui permettra(permettront) d'atteindre la couverture de test nécessaire. Les techniques qui peuvent être utilisées sont définies lors de la planification des tests.
- Utiliser des techniques de test pour concevoir des cas de test et des ensembles de cas de test qui couvrent les conditions de test identifiées
- Identifier les données de test nécessaires pour les conditions de test et les cas de test
- Concevoir l'environnement de test et identifier toute infrastructure requise, y compris les outils
- Etablir la traçabilité bidirectionnelle (p. ex. entre la base de test, les conditions de test et les cas de test)

Les critères de priorisation identifiés lors de l'analyse des risques et de la planification des tests devraient être appliqués tout au long du processus, de l'analyse et de la conception à l'implémentation et à l'exécution.

Selon les types de tests conçus, l'un des critères d'entrée pour la conception des tests peut être la disponibilité d'outils qui seront utilisés pendant les travaux de conception.

Lors de la conception des tests, l'Analyste de Test doit prendre en compte au moins les points suivants:

- Certains éléments de test sont mieux abordés en définissant uniquement les conditions de test plutôt que d'aller plus loin dans la définition des scripts de test qui donnent la séquence des instructions nécessaires pour exécuter un test. Dans ce cas, les conditions de test doivent être définies comme un guide pour les tests non scénarisés.
- Les critères de passage/échec doivent être clairement identifiés.
- Les tests doivent être conçus pour être compréhensibles par d'autres testeurs, et pas seulement par l'auteur. Si l'auteur n'est pas la personne qui exécute le test, d'autres testeurs devront lire et comprendre les tests précédemment spécifiés afin de comprendre les objectifs de test et l'importance relative du test.
- Les tests doivent également être compréhensibles pour d'autres parties prenantes telles que les développeurs, qui peuvent revoir les tests, et les auditeurs, qui peuvent avoir à approuver les tests.
- Les tests doivent couvrir tous les types d'interaction avec l'objet de test et ne doivent pas être limités aux interactions des personnes via l'interface utilisateur. Ils peuvent également inclure, par exemple, l'interaction avec d'autres systèmes et des événements techniques ou physiques. (voir [IREB_CPRE] pour plus de détails).
- Les tests doivent être conçus pour tester les interfaces entre les différents objets de test, ainsi que les comportements des objets eux-mêmes
- L'effort de conception des tests doit être priorisé et équilibré pour être en phase avec les niveaux de risque et la valeur métier.

### 1.4.1 Cas de test de bas niveau et de haut niveau

L'un des travaux de l'Analyste de Test est de déterminer le meilleur niveau de conception des cas de test pour une situation donnée. Les cas de test de bas niveau et de haut niveau sont couverts dans le syllabus du Niveau Fondation ISTQB [ISTQB_FL_SYL]. Certains des avantages et des inconvénients de l'utilisation de ces éléments sont décrits dans les listes suivantes:

Les cas de test de bas niveau offrent les avantages suivants:

- Un testeur inexpérimenté peut s'appuyer sur des renseignements détaillés fournis dans le cadre du projet. Les cas de test de bas niveau apportent toute l'information spécifique et les procédures nécessaires au testeur pour exécuter le cas de test (y compris les exigences relatives aux données) et pour vérifier les résultats
- Les tests peuvent être rejoués par différents testeurs et devraient aboutir aux mêmes résultats.
- Des défauts non évidents dans la base de test peuvent être détectés
- Le niveau de détail permet une vérification indépendante des tests, tels que les audits, si nécessaire
- Le temps consacré à l'automatisation des tests peut être réduit.

Les cas de test de bas niveau ont les inconvénients suivants:

- Ils peuvent nécessiter beaucoup d'efforts, tant pour la création que pour la maintenance.
- Ils ont tendance à limiter l'ingéniosité des testeurs pendant l'exécution.
- Ils exigent que la base de test soit bien définie
- La traçabilité aux conditions de test peut nécessiter plus d'effort qu'avec les cas de test de haut niveau.

Les cas de test de haut niveau offrent les avantages suivants:

- Ils donnent des lignes directrices pour ce qui devrait être testé, et permettent à l'Analyste de Test de varier les données réelles ou même la procédure qui est suivie lors de l'exécution du test.
- Ils peuvent offrir une meilleure couverture des risques que les cas de test de bas niveau parce qu'ils varieront quelque peu chaque fois qu'ils sont exécutés.
- Ils peuvent être définis tôt dans le processus d'ingénierie des exigences.
- Ils utilisent l'expérience de l'Analyste de Test avec les tests et l'objet de test lorsque le test est exécuté.
- Ils peuvent être définis lorsqu'aucune documentation détaillée et formelle n'est requise
- Ils sont mieux adaptés pour la réutilisation dans différents cycles de test lorsque des données de test différentes peuvent être utilisées.

Les cas de test de haut niveau ont les inconvénients suivants:

- Ils sont moins reproductibles, ce qui rend la vérification difficile. C'est parce qu'ils n'ont pas la description détaillée des cas de test de bas niveau.
- Un testeur plus expérimenté peut être nécessaire pour les exécuter
- Lors de l'automatisation sur la base de cas de test de haut niveau, le manque de détails peut entraîner la validation des mauvais résultats ou des éléments manquants qui devraient être validés.

Les cas de test de haut niveau peuvent être utilisés pour développer des cas de test de bas niveau lorsque les exigences deviennent plus précises et stables. Dans ce cas, la création de cas de test se fait de façon séquentielle, passant de haut niveau à bas niveau, seuls les cas de test de bas niveau étant utilisés pour l'exécution.

### 1.4.2 Conception des cas de tests

Les cas de test sont conçus par l'élaboration progressive et le raffinement des conditions de test identifiées à l'aide de techniques de test (voir le chapitre 3). Les cas de test devraient être reproductibles, vérifiables et traçables jusqu'à la base de test (p. ex., exigences).

La conception des tests comprend l'identification de ce qui suit:

- Objectif (c.-à-d. l'objectif observable et mesurable de l'exécution des tests)
- Préconditions, telles que les exigences relatives à l'environnement de test, projet ou localisé, les plans de leur livraison, l'état du système avant l'exécution des tests, etc.
- Exigences en matière de données de test (les données d'entrée pour le cas de test ainsi que les données qui doivent exister dans le système pour que le cas de test soit exécuté)
- Résultats attendus avec des critères explicites de succès/échec
- Postconditions, telles que les données affectées, l'état du système après l'exécution des tests, les déclencheurs pour le traitement ultérieur, etc.

Un défi particulier peut être la définition du résultat attendu d'un test. Faire cela manuellement est souvent fastidieux et sujet aux erreurs. Si possible, il est préférable de trouver ou de créer un oracle de test automatisé. En identifiant le résultat attendu, les testeurs sont concernés non seulement par les sorties à l'écran, mais aussi par les données et les postconditions de l'environnement. Si la base de test est clairement définie, l'identification du résultat correct, théoriquement, devrait être simple. Toutefois, la documentation de la base de test peut être vague, contradictoire, dépourvue de couverture des domaines clés ou entièrement manquante. Dans de tels cas, un Analyste de Test doit avoir une expertise du domaine ou y avoir accès. En outre, même lorsque la base de test est bien spécifiée, les interactions complexes de stimuli et de réponses complexes peuvent rendre la définition des résultats attendus difficile. Par conséquent, un oracle de test est essentiel. Dans un projet Agile, l'oracle de test peut être le Product Owner. L'exécution des cas de test sans aucun moyen de déterminer la justesse des résultats peut avoir une très faible valeur ajoutée ou un très faible bénéfice, générant souvent des rapports de défaillance invalides ou une fausse confiance dans le système.

Les activités décrites ci-dessus peuvent être appliquées à tous les niveaux de test, bien que la base de test varie. Lors de l'analyse et de la conception des tests, il est important de se rappeler le niveau cible pour le test ainsi que l'objectif du test. Cela permet de déterminer le niveau de détail requis ainsi que tous les outils qui peuvent être nécessaires (p. ex., des pilotes (drivers) et bouchons (stubs) au niveau composant).

Au cours de l'élaboration des conditions de test et des cas de test, une certaine quantité de documentation est généralement créée, ce qui entraîne des produits d'activités. Dans la pratique, le niveau de détail avec lequel les produits d'activités de test sont documentés varie considérablement. Cela peut être affecté par l'un des éléments suivants:

- Risques projet (ce qui doit/ne doit pas être documenté)
- La valeur ajoutée que la documentation apporte au projet
- Normes à suivre et/ou règlementations à respecter
- Cycle de vie de développement logiciel ou approche utilisée (p. ex., une approche Agile vise le « juste assez » pour la documentation)
- L'exigence de traçabilité depuis la base de test au cours de l'analyse et la conception des tests

Selon le périmètre des tests, l'analyse de test et la conception des tests répondent aux caractéristiques de qualité de l'objet de test. La norme ISO 25010 [ISO25010] fournit une référence utile. Lors du test de systèmes matériels/logiciels, d'autres caractéristiques peuvent s'appliquer.

Les processus d'analyse de test et de conception des tests peuvent être améliorés en les corrélant avec des revues et de l'analyse statique. En fait, la réalisation de l'analyse de test et de la conception des tests est souvent une forme de test statique parce que des problèmes peuvent être trouvés dans les documents de la base de test au cours de ce processus. L'analyse de test et la conception des tests en fonction de la spécification des exigences sont un excellent moyen de se préparer à une réunion de revue des exigences. La lecture des exigences afin de les utiliser pour créer des tests exige de comprendre l'exigence et de pouvoir déterminer un moyen d'évaluer la satisfaction de l'exigence. Cette activité révèle souvent des exigences qui ne sont pas claires, qui ne sont pas

testables, ou qui n'ont pas de critères d'acceptation définis. De même, des produits d'activité du test tels que les cas de test, les analyses de risque et les plans de test, peuvent faire l'objet de revues.

Pendant la conception des tests, les exigences détaillées requises en matière d'infrastructure de test peuvent être définies, bien que dans la pratique elles peuvent ne pas être finalisées avant l'implémentation des tests. Il ne faut pas oublier que l'infrastructure de test comprend plus que des objets de test et le testware. Par exemple, les exigences en infrastructure peuvent inclure les salles, l'équipement, le personnel, les logiciels, les outils, les périphériques, l'équipement de communication, les autorisations des utilisateurs et tous les autres éléments requis pour exécuter les tests.

Les critères de sortie pour l'analyse de test et la conception des tests varient en fonction des paramètres du projet, mais tous les éléments discutés dans ces deux sections devraient être pris en considération pour être inclus dans les critères de sortie définis. Il est important que les critères de sortie soient mesurables et que toutes les informations et préparations requises pour les étapes suivantes aient été fournies.

## 1.5 Implémentation des tests

L'implémentation des tests prépare le testware nécessaire à l'exécution des tests à partir de l'analyse et la conception des tests. Elle comprend les activités suivantes:

- Créer un calendrier d'exécution des tests, y compris l'allocation des ressources, pour permettre le début de l'exécution des cas de test (voir [ISTQB_FL_SYL] Section 5.2.4)
- Organiser des tests (manuels et automatisés) dans des suites de test et définir l'ordre d'exécution des tests
- Créer des tests automatisés (ou identification des cas de test à automatiser par un développeur ou un ingénieur en automatisation)
- Développer des procédures de test
- Finaliser les données de test et les environnements de test
- Mettre à jour de la traçabilité entre la base de test et le testware comme les conditions de test, les cas de test et les suites de test.

Lors de l'implémentation des tests, les Analystes de Test identifient les cas de test qui peuvent être regroupés (par exemple, ceux qui correspondent au test d'un processus métier de haut niveau particulier), et les organisent en suites de test . Cela permet d'exécuter ensemble des cas de test connexes.

Des procédures de test sont créées et finalisent et confirment l'ordre dans lequel les tests manuels et automatisés et les suites de test doivent être exécutés. La définition des procédures de test nécessite d'identifier soigneusement les contraintes et les dépendances qui pourraient influencer la séquence d'exécution des tests. Les procédures de test documentent toutes les préconditions initiales (p. ex., le chargement des données de test à partir d'un référentiel de données) et toute activité suivant l'exécution (p. ex., la réinitialisation de l'état du système). Si une stratégie de test basée sur les risques est utilisée, le niveau de risque peut dicter l'ordre d'exécution pour les cas de test. D'autres facteurs peuvent déterminer l'ordre d'exécution des tests tels que la disponibilité des bonnes personnes, de l'équipement, des données et de la fonctionnalité à tester.

Il n'est pas rare que le code soit livré par morceaux et l'effort de test doit être coordonné avec la séquence selon laquelle le logiciel devient disponible pour les tests. En particulier dans les modèles de cycles de vie itératifs et incrémentaux, il est important que l'Analyste de Test se coordonne avec l'équipe de développement pour s'assurer que le logiciel sera livré pour les tests dans un état testable.

Les facteurs ci-dessus sont pris en compte lors de la création d'un calendrier d'exécution des tests.

Le niveau de détail et la complexité associée pour le travail effectué pendant l'implémentation des tests peuvent être influencés par le détail des cas de test et des conditions de test. Dans certains cas, les règles réglementaires s'appliquent, et les produits d'activité de test devraient fournir des preuves de conformité aux normes applicables telles que la norme des États-Unis DO-178C(en Europe, ED 12C). [RTCA DO-178C/ED-12C].

Comme indiqué ci-dessus, les données de test sont nécessaires pour la plupart des tests, et dans certains cas ces ensembles de données peuvent être assez importants. Pendant l'implémentation, les Analystes de Test créent des données d'entrée et d'environnement à charger dans des bases de données et d'autres référentiels de ce type. Ces données doivent être « adaptées à l'usage » pour permettre la détection des défauts. Les Analystes de Test peuvent également créer des données à utiliser avec des tests d'automatisation dirigée par les données et les mots clés (voir la section 6.2) ainsi que pour les tests manuels.

L'implémentation des tests concerne également le ou les environnement(s) de test. Au cours de cette activité, le ou les environnement(s) doi(ven)t être entièrement configuré(s) et vérifié(s) avant l'exécution des tests. Un environnement de test « adapté à l'usage » est essentiel, c'est-à-dire que l'environnement de test devrait permettre de mettre en évidence des défauts présents lors des tests contrôlés, fonctionner normalement en l'absence de défaillances, et reproduire adéquatement, si nécessaire, la production ou l'environnement de l'utilisateur final pour des niveaux de test plus élevés. Des changements d'environnement de test peuvent être nécessaires pendant l'exécution des tests en fonction de changements imprévus, de résultats de tests ou d'autres considérations. Si des changements d'environnement se produisent pendant l'exécution, il est important d'évaluer l'impact des changements sur les tests qui ont déjà été exécutés.

Pendant l'implémentation des tests, les Analystes de Test devraient vérifier que les responsables de la création et de la maintenance de l'environnement de test sont connus et disponibles, et que tout le testware , les outils de support au test et les processus associés sont prêts à être utilisés. Cela comprend la gestion de la configuration, la gestion des défauts et l'enregistrement et la gestion des tests. En outre, les Analystes de Test doivent vérifier les procédures qui recueillent des données pour évaluer l'état actuel par rapport aux critères de sortie et aux résultats des tests.

Il est sage d'utiliser une approche équilibrée pour l'implémentation des tests telle qu'elle est déterminée lors de la planification des tests. Par exemple, les stratégies de test analytiques basées sur les risques sont souvent couplées à des stratégies de test réactives. Dans ce cas, un certain pourcentage de l'effort de mise en œuvre des tests est alloué aux tests qui ne suivent pas des scripts prédéterminés (non scénarisés).

Le test non scénarisé ne doit pas être aléatoire ou sans but, car cela peut être imprévisible tant en durée qu'en couverture, et donner un faible rendement en défauts. Au contraire, il devrait être effectué lors de sessions de durée limitée, chacune donnant une orientation initiale dans une charte, mais avec la liberté de s'écarter des prescriptions de la charte si des possibilités de tests potentiellement plus productifs sont découvertes au cours de la session. Au fil des ans, les testeurs ont développé une variété de techniques de test basées sur l'expérience, telles que les attaques [Whittaker03], l'estimation d'erreurs [Myers11], et les tests exploratoires [Whittaker09]. L'analyse de test, la conception des tests et l'implémentation des tests se produisent toujours, mais elles ont lieu principalement pendant l'exécution des tests.

En suivant de telles stratégies de test réactif, les résultats de chaque test influencent l'analyse, la conception et l'implémentation des tests suivants. Bien que ces stratégies soient faciles à mettre en œuvre et souvent efficaces pour trouver des défauts, il y a certains inconvénients, comme:

- L'expertise de l'Analyste de Test est requise
- La durée peut être difficile à prévoir
- La couverture peut être difficile à suivre

- La répétabilité peut être perdue sans une bonne documentation ou le support d'outils

## 1.6 Exécution des tests

L'exécution des tests est menée selon le calendrier d'exécution des tests et inclut les tâches suivantes: (voir [ISTQB_FL_SYL])

- Exécuter les tests manuels, y compris des tests exploratoires
- Exécuter les tests automatisés
- Comparer les résultats obtenus avec les résultats attendus
- Analyser les anomalies pour établir leurs causes probables
- Signaler les défauts en fonction des défaillances observées
- Enregistrer les résultats de l'exécution des tests
- Mettre à jour la traçabilité entre la base de test et le testware pour prendre en compte les résultats de test
- Exécuter des tests de régression

Les taches de l'exécution des tests listées ci-dessus peuvent être réalisées par le testeur ou l' Analyste de Test.

Des tâches supplémentaires peuvent généralement être effectuées par l'Analyste de Test:

- Reconnaître des regroupements de défauts qui peuvent indiquer la nécessité d'effectuer plus de tests sur une partie particulière de l'objet de test
- Faire des suggestions pour de futures sessions de tests exploratoires sur la base des résultats des tests exploratoires
- Identifier de nouveaux risques à partir des informations obtenues lors de la réalisation des tâches d'exécution des tests
- Faire des suggestions pour améliorer l'un ou l'autre des produits d'activités de l'implémentation des tests (p. ex., améliorations des procédures de test)

# 2. Les tâches de l'Analyste de Test dans le test basé sur les risques - 60 mins.

## Mots clés

Risque produit, identification des risques, niveau de risque, atténuation des risques, test basé sur les risques

## Objectifs d'apprentissage pour Les tâches de l'Analyste de Test dans le test basé sur les risques

## Les tâches de l'Analyste de Test dans le test basé sur les risques

TA-2.1.1 (K3) Pour une situation donnée, participer à l'identification des risques, effectuer une évaluation des risques et proposer une atténuation des risques appropriée

## 2.1 Introduction

Les Test Managers ont souvent la responsabilité globale d'établir et de gérer une stratégie de test basée sur les risques. Ils demandent habituellement la participation d'un Analyste de Test pour s'assurer que l'approche basée sur les risques est mise en œuvre correctement.

Les Analystes de Test devraient participer activement aux tâches de test basé sur les risques suivantes:

- Identification des risques
- Evaluation des risques
- Atténuation des risques

Ces tâches sont effectuées de manière itérative tout au long du cycle de vie de développement logiciel pour faire face aux risques émergents, changer les priorités et évaluer et communiquer régulièrement l'état des risques (voir [vanVeenendaal12] et [Black02] pour plus de détail). Dans un projet Agile, les trois tâches sont souvent combinées dans une session dite de risque avec un accent sur une itération ou une release.

Les Analyste de Tests devraient travailler au sein du cadre mis en place par le Test Manager pour le test basé sur les risques. Ils devraient apporter leur connaissance des risques liés au domaine métier inhérents au projet, tels que les risques liés à la sécurité, aux préoccupations commerciales et économiques, et aux facteurs politiques, entre autres.

## 2.2 Identification des risques

C'est en faisant appel à l'échantillon le plus large possible de parties prenantes que le processus d'identification des risques sera le plus susceptible de détecter le plus grand nombre possible de risques importants.

Les Analystes de Test possèdent souvent des connaissances uniques sur le domaine métier particulier du système sous test. Cela signifie qu'ils sont particulièrement bien adaptés aux tâches suivantes:

- Mener des entretiens d'experts avec des experts métier et des utilisateurs
- Effectuer des évaluations indépendantes
- Utiliser des modèles de risques
- Participer à des ateliers de risques
- Participer à des sessions de brainstorming avec des utilisateurs potentiels et actuels
- Définir des checklists de test
- Faire appel à l'expérience passé sur des projets ou systèmes similaires

En particulier, les Analystes de Test devraient travailler en étroite collaboration avec les utilisateurs et d'autres experts métier (p. ex., ingénieurs d'exigences, analystes métier) pour déterminer les domaines de risques métier qui devraient être abordés pendant les tests. Dans les projets Agile, cette relation étroite avec les parties prenantes permet d'identifier régulièrement les risques, par exemple lors des réunions de planification des itérations.

Les exemples de risques qui pourraient être identifiés dans un projet incluent:

- Problèmes de correction fonctionnelle, p. ex., calculs incorrects
- Problèmes d'utilisabilité, p. ex., raccourcis clavier insuffisants
- Problèmes de portabilité, par exemple, incapacité d'installer une application sur des plates-formes particulières

## 2.3 Évaluation des risques

Alors que l'identification des risques vise à identifier autant de risques pertinents que possible, l'évaluation des risques est l'étude de ces risques identifiés. Plus précisément, catégoriser chaque risque et déterminer la probabilité et l'impact associés à chaque risque.

La détermination du niveau de risque implique généralement d'évaluer, pour chaque élément de risque, la probabilité d'occurrence et l'impact. La probabilité d'occurrence est habituellement interprétée comme la probabilité que le problème potentiel puisse exister dans le système sous test et sera observé lorsque le système sera en production. Les Analystes Techniques de Test devraient contribuer à trouver et à comprendre la probabilité potentielle pour chaque élément de risque, tandis que les Analystes de Test contribuent à comprendre l'impact métier potentiel du problème s'il se produit (dans les projets Agile, cette distinction basée sur les rôles peut être moins forte).

L'impact est souvent interprété comme la gravité de l'effet sur les utilisateurs, les clients ou d'autres parties prenantes. En d'autres termes, il découle du risque métier. Les Analystes de Tests devraient contribuer à identifier et à évaluer le domaine métier ou l'impact potentiel de chaque élément de risque. Les facteurs influençant le risque commercial sont les facteurs suivants ::

- Fréquence d'utilisation de la fonction affectée
- Perte commerciale
- Dommages financiers
- Pertes ou responsabilités écologiques ou sociales
- Sanctions juridiques civiles ou pénales
- Préoccupations en matière de sécurité
- Amendes, perte de licence
- Absence de solutions de contournement raisonnables si les gens ne peuvent plus travailler
- Visibilité de la fonctionnalité
- Visibilité de la défaillance entraînant une publicité négative et des dommages potentiels à l'image de marque
- Perte de clients

Compte tenu de l'information disponible sur les risques, les Analystes de Test doivent définir les niveaux de risque métier selon les lignes directrices fournies par un Test Manager. Ceux-ci pourraient être classés avec des termes (p. ex., bas, moyen, élevé) ou utiliser des nombres et/ou des couleurs.

À moins qu'il n'y ait un moyen de mesurer le risque sur une échelle définie, il ne peut pas être une véritable mesure quantitative. Des nombres peuvent être attribués à la valeur qualitative, mais cela n'en fait pas une véritable mesure quantitative. Par exemple, des Test Managers peuvent déterminer que le risque métier doit être mesuré sur une échelle numérique particulière (par ex., 1-5 pour la probabilité et pour l'impact). Une fois que la probabilité et l'impact ont été attribués, ces valeurs peuvent être utilisées pour déterminer le niveau global de risque pour chaque élément de risque. Ce niveau global sert ensuite à prioriser les activités d'atténuation des risques.[vanVeenendaal12].

## 2.4 Atténuation des risques

Lors du projet, les Analystes de Test devraient chercher à faire ce qui suit:

- Réduire le risque de produit en concevant des cas de test efficaces qui démontrent sans ambiguïté si le test réussit ou échoue, et en participant à des revues de produits d'activités logiciels tels que les exigences, les documents de conception et la documentation utilisateur
- Mettre en œuvre des activités appropriées d'atténuation des risques identifiés dans la stratégie de test et le plan de test (p. ex., tester un processus métier particulièrement risqué à l'aide de techniques de test particulières)

- Réévaluer les risques connus en fonction de l'information supplémentaire recueillie au fur et à mesure du déroulement du projet, en ajustant la probabilité, l'impact ou les deux, selon le cas
- Identifier de nouveaux risques liés à l'information obtenue lors des tests

Lorsqu'on parle d'un risque produit, le test contribue de façon essentielle à son atténuation. En trouvant des défauts, les testeurs réduisent les risques en sensibilisant les gens aux défauts et aux possibilités de les traiter avant la livraison. Si les testeurs ne trouvent aucun défaut, les tests réduisent alors le risque en fournissant la preuve que, dans certaines conditions (c.-à-d. les conditions testées), le système fonctionne correctement. Les Analystes de Test aident à déterminer les options d'atténuation des risques en étudiant les possibilités de recueillir des données de test précises, de créer et de tester des scénarios utilisateurs réalistes et de mener ou de superviser des études de facilité d'utilisation, entre autres.

### 2.4.1 Prioriser les tests

Le niveau de risque est également utilisé pour prioriser les tests. Un Analyste de Test pourrait déterminer qu'il existe un risque élevé dans le domaine de l'exactitude transactionnelle dans un système comptable. Par conséquent, pour atténuer le risque, le testeur peut collaborer avec d'autres experts du domaine métier pour recueillir un ensemble solide de données échantillonnées pouvant être traitées et vérifiées pour leur précision. De même, un Analyste de Test pourrait déterminer que les problèmes d'utilisabilité représentent un risque important pour un nouvel objet de test. Plutôt que d'attendre un test d'acceptation utilisateurs pour découvrir des problèmes, l'Analyste de Test pourrait prioriser un test d'utilisabilité précoce avec un prototype pour aider à identifier et résoudre les problèmes de conception de l'utilisabilité tôt avant le test d'acceptation utilisateurs. Cette priorisation doit être prise en considération le plus tôt possible aux étapes de la planification afin que le calendrier puisse tenir compte des tests nécessaires au moment nécessaire.

Dans certains cas, tous les tests à risque le plus élevé sont effectués avant tout test à faible risque, et les tests sont exécutés dans un ordre de risque strict (appelé « profondeur d'abord »); dans d'autres cas, une approche d'échantillonnage est utilisée pour sélectionner un échantillon de tests pour tous les risques identifiés en utilisant le niveau de risque pour pondérer la sélection tout en assurant la couverture de chaque risque au moins une fois (appelé « largeur d'abord»).

Que les tests basés sur les risques soient effectués en profondeur d'abord ou en largeur d'abord, il est possible que le temps alloué aux tests soit consommé sans que tous les tests soient exécutés. Les tests basés sur les risques permettent aux testeurs de fournir des rapports à la direction en fonction du niveau de risque restant à ce stade, et permettent à la direction de décider s'il faut étendre les tests ou transférer le risque restant aux utilisateurs, aux clients, au service d'assistance/au support technique et/ou au personnel opérationnel.

### 2.4.2 Ajustement des tests pour les cycles de test futurs

L'évaluation des risques n'est pas une activité ponctuelle effectuée avant le début de la mise en œuvre des tests; c'est un processus continu. Chaque futur cycle de test planifié doit être soumis à une nouvelle analyse des risques pour prendre en compte des facteurs tels que:

- Tout risque produit nouveau ou considérablement modifié
- Zones instables ou sujettes à l'échec découvertes au cours des tests
- Risques liés aux défauts corrigés
- Défauts typiques trouvés lors des tests
- Domaines ayant été insuffisamment testés (faible couverture de test)

# 3. Techniques de test - 630 mins.

## Mots clés

technique de test boîte noire, analyse des valeurs limites, test basé sur des checklists, technique de classification arborescente, table de décision, taxonomie de défauts, technique de test basée sur les défauts, partitions d'équivalence, estimation d'erreurs, test basé sur l'expérience, technique de test basée sur l'expérience, test exploratoire, test par paires, test de transitions d'états, charte de test, test des cas d'utilisation

## Objectifs d'apprentissage pour Techniques de test

### 3.1 Introduction

Pas d'objectifs d'apprentissage

### 3.2 Techniques de test boîte noire

TA-3.2.1 (K4) Analyser un ou des éléments d'une spécification donnée et concevoir des cas de test en appliquant les partitions d'équivalence

TA-3.2.2 (K4) Analyser un ou des éléments d'une spécification donnée et concevoir des cas de test en appliquant l'analyse des valeurs limites

TA-3.2.3 (K4) Analyser un ou des éléments d'une spécification donnée et concevoir des cas de test en appliquant le test des tables de décision

TA-3.2.4 (K4) Analyser un ou des éléments d'une spécification donnée et concevoir des cas de test en appliquant le test des transitions d'état

TA-3.2.5 (K2) Expliquer comment les diagrammes de classification arborescente soutiennent les techniques de test

TA-3.2.6 (K4) Analyser un ou des éléments d'une spécification donnée et concevoir des cas de test en appliquant le test par paires

TA-3.2.7 (K4) Analyser un ou des éléments d'une spécification donnée et concevoir des cas de test en appliquant le test des cas d'utilisation

TA-3.2.8 (K4) Analyser un système, ou sa spécification des exigences, afin de déterminer les types de défauts les plus probables et sélectionner la ou les technique(s) de test boîte noire appropriée(s)

### 3.3 Techniques de test basées sur l'expérience

TA-3.3.1 (K2) Expliquer les principes des techniques de test basées sur l'expérience et les avantages et les inconvénients par rapport aux techniques de test boîte noire et basé sur les défauts

TA-3.3.2 (K3) Identifier des tests exploratoires à partir d'un scénario donné

TA-3.3.3 (K2) Décrire l'application de techniques de test basées sur les défauts et différencier leur utilisation des techniques de test boîte noire

### 3.4 Application les techniques de test les plus appropriées

TA-3.4.1 (K4) Pour une situation de projet donnée, déterminer quelles techniques de test boîte noire ou basées sur l'expérience doivent être appliquées pour atteindre des objectifs spécifiques

## 3.1 Introduction

Les techniques de test présentées dans le présent chapitre sont divisées en deux catégories:

- Test boîte noire
- Test basé sur l'expérience

Ces techniques sont complémentaires et peuvent être utilisées comme il convient pour toute activité de test donnée, quel que soit le niveau de test effectué.

Notez que les deux catégories de techniques peuvent être utilisées pour tester les caractéristiques de qualité fonctionnelle et non fonctionnelle. Le test des caractéristiques logicielles est discuté dans le chapitre suivant.

Les techniques de test discutées dans ces sections peuvent se concentrer principalement sur la détermination de données de test optimales (p. ex., à partir des partitions d'équivalence) ou sur la conception des procédures de test (p. ex., à partir de modèles d'état). Il est courant de combiner des techniques pour créer des cas de test complets.

## 3.2 Techniques de test boîte noire

Les techniques de test boîte noire sont introduites dans le syllabus ISTQB Niveau Fondation [ISTQB_FL_SYL].

Les caractéristiques courantes des techniques de test boîte noire incluent les pratiques suivantes:

- Des modèles, par exemple des diagrammes d'état et des tables de décision, sont créés pendant la conception des tests selon la technique de test
- Les conditions de test sont dérivées systématiquement de ces modèles

Les techniques de test fournissent généralement des critères de couverture, qui peuvent être utilisés pour mesurer les activités de conception et d'exécution des tests. Remplir complètement les critères de couverture ne signifie pas que l'ensemble des tests est complet, mais plutôt que le modèle ne suggère plus de tests supplémentaires pour augmenter la couverture basée sur cette technique.

Le test boîte noire est généralement basé sur une certaine forme de documentation de spécification, telle qu'une spécification d'exigences système ou des User Stories. Puisque la documentation de spécification devrait décrire le comportement du système, en particulier dans le domaine fonctionnel, dériver des tests des exigences fait souvent partie du test du comportement du système. Dans certains cas, il peut n'y avoir aucun document de spécification, mais il existe des exigences implicites, telles que le remplacement de la fonctionnalité d'un système hérité (« legacy system »).

Il existe un certain nombre de techniques de test boîte noire. Ces techniques ciblent différents types de logiciels et de scénarios. Les sections ci-dessous montrent l'applicabilité de chaque technique ; certaines limitations et difficultés que l'Analyste de Test peut rencontrer ; la méthode par laquelle la couverture est mesurée et les types de défauts qui sont ciblés.

Merci de vous référer à [ISO29119-4], [Bath14], [Beizer95], [Black07], [Black09], [Copeland04], [Craig02], [Koomen06], et [Myers11] pour plus de détails.

### 3.2.1 Partitions d'équivalence

La technique des partitions d'équivalence est utilisée pour réduire le nombre de cas de test nécessaires pour tester efficacement le traitement des entrées, des sorties, des valeurs internes et des valeurs liées au temps. Le partitionnement est utilisé pour créer des classes d'équivalence (souvent appelées partitions d'équivalence) qui sont créées à partir d'ensembles de valeurs qui

doivent être traitées de la même manière. En sélectionnant une valeur représentative par partition, la couverture de tous les éléments d'une même partition est supposée.

**Applicabilité**
Cette technique est applicable à n'importe quel niveau de test et est appropriée lorsque tous les éléments d'un ensemble de valeurs à tester sont censés être manipulés de la même manière et lorsque les ensembles de valeurs utilisées par l'application n'interagissent pas. La sélection des ensembles de valeurs s'applique aux partitions valides et non valides (c.-à-d. les partitions contenant des valeurs qui doivent être considérées comme non valides pour le logiciel testé).

Des valeurs doivent être considérées dans les catégories suivantes:

- Valeurs dans une plage continue. Par exemple, l'expression 0 < x < 5000 représente la plage continue de la variable «x» dont la partition valide se situe entre 1 et 4999. Une valeur représentative valide sur cette plage continue peut être, par exemple, 3249.
- Valeurs discrètes. Par exemple, les couleurs « rouge, bleu, vert, jaune » sont toutes des partitions valides et il n'y a qu'une seule valeur possible par partition.

Le test des partitions d'équivalence est plus fort lorsqu'il est utilisé en combinaison avec l'analyse des valeurs limites qui élargit les valeurs de test pour inclure celles sur les limites des partitions. La technique des partitions d'équivalence, en utilisant des valeurs pour les partitions valides, est une technique couramment utilisée pour les « smoke tests » d'un nouveau build ou d'une nouvelle release car elle détermine rapidement si la fonctionnalité de base fonctionne.

**Limitations/Difficultés**
Si l'hypothèse est incorrecte et que les valeurs de la partition ne sont pas traitées exactement de la même manière, cette technique peut manquer des défauts. Il est donc important de sélectionner soigneusement les partitions. Par exemple, un champ d'entrée qui accepte des nombres positifs et négatifs serait mieux testé sous la forme de deux partitions valides, l'une pour les nombres positifs et l'autre pour les nombres négatifs, en raison de la probabilité d'un traitement différent. Selon que le zéro est autorisé ou non, cela pourrait aboutir à une autre partition (puisque zéro n'est ni positif ni négatif). Il est important pour un Analyste de Test de comprendre le traitement sous-jacent afin de déterminer le meilleur partage des valeurs. Cela peut nécessiter un soutien dans la compréhension de la conception du code.

**Couverture**
La couverture est déterminée en prenant le nombre de partitions pour lesquelles une valeur a été testée et en divisant ce nombre par le nombre de partitions qui ont été identifiées. La couverture des partitions d'équivalence est alors indiquée en pourcentage. L'utilisation de plusieurs valeurs pour une seule partition n'augmente pas le pourcentage de couverture.

**Types de défauts**
Un Analyste de Test utilise cette technique pour trouver des défauts dans le traitement de diverses valeurs de données.

### 3.2.2 Analyse des valeurs limites

L'analyse des valeurs limites est utilisée pour tester la bonne gestion des valeurs qui existent sur les limites des partitions d'équivalence. Il y a deux approches des valeurs limites: tester deux valeurs ou tester trois valeurs pour chaque limite. Pour le test de deux valeurs, la valeur limite (sur la limite) et la valeur qui se trouve juste à l'extérieur de la limite (par le plus petit incrément possible) sont utilisées. Par exemple, si la partition incluait les valeurs 1 à 10 par incréments de 0,5, les deux valeurs de test de valeur pour la limite supérieure seraient 10 et 10,5.Les valeurs de test de la limite inférieure seraient 1 et 0,5. Les limites sont définies par les valeurs maximales et minimales de la partition d'équivalence définie.

Pour le test de trois valeurs, les valeurs en dessous, sur et au-dessus de la limite sont utilisées. Dans l'exemple précédent, les tests de la limite supérieure comprendraient 9,5, 10 et 10,5. Les tests de la limite inférieure comprendraient 1,5, 1 et 0,5. La décision d'utiliser deux ou trois valeurs limites devrait être fondée sur le risque associé à l'objet testé, l'approche avec trois valeurs étant utilisée pour les éléments à risque élevé.

**Applicabilité**
Cette technique est applicable à n'importe quel niveau de test et est appropriée lorsqu'il existe des partitions d'équivalence ordonnées. Pour cette raison, la technique d'analyse des valeurs limites est souvent menée en même temps que la technique des partitions d'équivalence. Des partitions d'équivalence ordonnées sont requises en raison du concept d'être sur et à côté de la limite. Par exemple, une plage de nombres est une partition ordonnée. Une partition composée de tous les objets rectangulaires n'est pas une partition ordonnée et n'a pas de valeurs limites. En plus des plages de nombres, l'analyse de la valeur limite peut être appliquée à ce qui suit:

- Attributs numériques de variables non numériques (p. ex., longueur)
- Boucles, y compris celles dans les diagrammes d'état, dans les cas d'utilisation, et dans les structures de données stockées telles que les tableaux
- Objets physiques (y compris la mémoire)
- Activités déterminées dans le temps

**Limitations/Difficultés**
Comme l'exactitude de cette technique dépend de l'identification précise des partitions d'équivalence afin d'identifier correctement les limites, elle est soumise aux mêmes limitations et difficultés. L'Analyste de Test doit également être au courant des incréments dans les valeurs valides et non valides pour être en mesure de déterminer avec précision les valeurs à tester. Seules les partitions ordonnées peuvent être utilisées pour l'analyse des valeurs limites, mais cela ne se limite pas à un ensemble d'entrées valides. Par exemple, lors du test du nombre de cellules prises en charge par une feuille de calcul, il existe une partition qui contient le nombre de cellules jusqu'au nombre maximal de cellules autorisées (la limite), ce nombre étant inclus, et une autre partition qui commence par le nombre correspondant à une cellule au-dessus du maximum (au-dessus de la limite).

**Couverture**
La couverture est déterminée en prenant le nombre de valeurs limites qui sont testées et en les divisant par le nombre de valeurs limites identifiées (à l'aide de la méthode de deux valeurs ou de trois valeurs). La couverture est indiquée en pourcentage.

**Types de défauts**
L'analyse des valeurs limites détecte de manière fiable le déplacement ou l'omission des limites, et peut trouver des cas de limites supplémentaires. Cette technique détecte des défauts concernant la manipulation des valeurs limites, en particulier les erreurs avec les signes inférieur et supérieur (c.-à-d. le déplacement). Elle peut également être utilisée pour trouver des défauts non fonctionnels, par exemple, un système prend en charge 10.000 utilisateurs simultanés, mais pas 10.001.

### 3.2.3 Test des tables de décision

Les tables de décision permettent de saisir les conditions traitées par le logiciel à tester et d'évaluer les résultats de l'application de valeurs vraies ou fausses différentes à ces conditions. L'Analyste de Tests peut utiliser des tables de décision pour saisir les règles de décision qui s'appliquent au logiciel sous test.

Pour utiliser cette technique, l'Analyste de Test définit une table de décision initiale complète où le nombre de colonnes est de 2 à la puissance du nombre de conditions ($2^n$ où n est le nombre de conditions). Par exemple, la table de décision initiale pour trois conditions aurait huit colonnes ($2^3$).

Chaque colonne dans la table représente une règle de décision. Une règle de décision (générique) typique pourrait être “Si la condition A est vraie et la condition B est vraie et la condition C est fausse, alors l’action X est attendue”.

Deux approches peuvent être appliquées au test des tables de décision, en fonction de ce qui doit être couvert:

- Couverture des combinaisons de condition qui représentent les règles
- Couverture des conditions individuelles

**Couverture des combinaisons de conditions**

L’utilisation “classique” des tables de décision est de tester les règles de décisions qui correspondent aux combinaisons de conditions.

Lorsque l’on essaie de tester toutes les combinaisons possibles, les tables de décision peuvent devenir très grandes. Une méthode consistant à réduire le nombre de combinaisons, parmi toutes celles possibles, à celles qui se différencient des autres est le test des tables de décision réduites [Copeland04]. Quand cette technique est utilisée, les combinaisons sont réduites à celles qui produiront des résultats différents, en supprimant de la table de décision les combinaisons qui ne sont pas pertinentes pour les résultats. Ces règles sont considérées comme redondantes et sont notées “indifférent” dans la table de décision. De plus, on supprime les règles de décision qui sont impossibles. Par exemple, si une machine délivrant des tickets applique différentes règles en fonction des conditions “avant 9h” et “après 17h”, alors il n’est pas possible de définir une règle de décisions où les deux conditions sont vraies (le même jour). Les règles de décisions impossibles sont supprimées de la table. La suppression des règles impossibles ou des règles redondantes aboutit à une table partiellement réduite.. La suppression des deux aboutit à une table entièrement réduite.

En couvrant les combinaisons de conditions, un Analyste de Test essaie de créer une table de décision réduite. Si un Analyste de Test considère que le nombre de cas de test issus d’une table de décision réduite est encore trop grand, une sélection basée sur les risques est opérée.

**Couverture des résultats des conditions individuelles**

L’objectif de cette approche est de s’assurer que les résultats vrais et faux de chaque condition sont individuellement testés. La première règle de décision à considérer définit une même valeur pour toutes les conditions individuelles (p.ex. vrai). Cette règle de décision est placée dans la table de décision de même que l’action attendue associée. La définition de la seconde règle de décisions consiste à passer la valeur de la première condition à l’opposé (p.ex. faux). Cette seconde règle de décisions est aussi entrée dans la table de décisions avec l’action attendue. La définition de la troisième règle de décision consiste à remettre la valeur de la première condition à celle utilisée dans la première règle, et c’est la valeur de la deuxième condition qui est modifiée. De même, la règle de décision est entrée dans la table de décision et l’action attendue est définie. La procédure est répétée jusqu’à ce que chaque condition ait pris les valeur “vrai” et “faux” dans les règles de décisions définies. Cela aboutit à une table dont le nombre de règles de décision est égal au nombre de conditions + 1. De même que dans l’approche de “toutes les combinaisons de décisions” décrite ci-dessus, toutes les règles infaisables ou redondantes sont supprimées de la table.

Cette approche systématique aboutit généralement à moins de règles de décisions à tester qu’avec l’approche de “toutes les combinaisons de décisions”. Chaque condition est testée au moins une fois avec la valeur “vraie” et au moins une fois avec la valeur “faux”. Comme chaque règle de décision se concentre sur une condition spécifique, la localisation de défauts mis en évidence est simplifiée. Avec cette approche, les défauts ne se produisant qu’avec certaines combinaisons de conditions peuvent être manqués puisque seules les combinaisons simples sont prises en compte.

**Applicabilité**
Les deux approches de cette technique sont couramment appliquées pour les niveaux de test intégration, système et acceptation. Selon le code, elle peut également être applicable pour le test de composants lorsqu'un composant est responsable d'un ensemble de logique de décision. Cette technique est particulièrement utile lorsque les exigences sont présentées sous forme de diagrammes de flux ou de tableaux de règles métier.

Les tables de décision sont également une technique de définition des exigences et certaines spécifications d'exigences peuvent déjà être définies dans ce format. Même lorsque les exigences ne sont pas présentées sous forme de tableau ou d'organigramme, les conditions et les combinaisons possibles de conditions sont souvent trouvées dans le texte.

Lors de la conception des tables de décision, il est important de tenir compte des règles de décision définies ainsi que de celles qui ne sont pas expressément définies mais qui existeront. En général, un Analyste de Test doit être en mesure d'identifier les actions attendues pour toutes les règles de décision conformément à la spécification ou à l'oracle de test.

**Limitations/Difficultés**
Lorsque l'on considère les combinaisons de conditions, il peut être difficile de trouver toutes les conditions d'interaction, en particulier lorsque les exigences ne sont pas bien définies ou n'existent pas. Il faut faire attention lors de la sélection du nombre de conditions considérées dans une table de décisions à ce que le nombre de combinaisons de ces conditions demeure gérable. Lorsque le nombre de conditions entraîne un nombre de combinaisons ingérable, un Analyste de Test peut envisager de définir une hiérarchie des tables de décision, où les conditions d'une table de décision particulière peuvent résulter de l'invocation d'une autre table de décision.

**Couverture**
Trois niveaux de couverture sont généralement considérés:

- Couvrir tous les résultats de conditions: des règles de décision suffisantes sont appliquées pour exercer les résultats vrai/faux de chaque condition. Cela donne une couverture maximale basée sur le nombre de colonnes dans la table de décisions initiale (c.-à-d. 2 à la puissance du nombre de conditions). Il s'agit de la couverture la plus rentable, mais elle n'exerce pas nécessairement toutes les actions possibles.
- Couvrir toutes les actions: des règles de décision suffisantes sont appliquées pour générer toutes les actions attendues possibles.
- Couvrir toutes les règles de décision non redondantes et réalisables: toutes les règles de décision dans une table de décision complètement réduite sont couvertes. Il s'agit d'un niveau de couverture plus élevé, mais pouvant générer un grand nombre de cas de test.

Lors de la détermination des tests à partir d'une table de décision, il est aussi important de considérer toutes les conditions limites qui devraient être testées. Ces limites conditions peuvent entraîner une augmentation du nombre de cas de test nécessaires pour tester adéquatement le logiciel. L'analyse des valeurs limites et les partitions d'équivalence sont complémentaires à la technique de la table de décisions.

**Types de défauts**
Les défauts typiques incluent un traitement incorrect basé sur des combinaisons particulières de conditions entraînant des résultats inattendus. Lors de la création des tables de décisions, des défauts peuvent être trouvés dans le document de spécification. Il n'est pas rare de préparer un ensemble de conditions et de constater que le résultat attendu n'est pas précisé pour une ou plusieurs règles de décision. Les types de défauts les plus courants sont les omissions (il n'y a pas d'information sur ce qui devrait réellement se produire dans une certaine situation) et les contradictions. Les tests peuvent

également trouver des problèmes avec des combinaisons de condition qui ne sont pas gérées ou qui ne sont pas correctement gérées.

### 3.2.4 Test des transitions d'état

Le test des transitions d'état est utilisé pour tester la capacité de l'objet de test à entrer dans des états et à sortir d'états définis via des transitions valides, ainsi qu'à entrer dans des états non valides ou à exercer des transitions non valides. Des événements entraînent la transition de l'objet de test d'un état à un autre et la réalisation d'actions. Les événements peuvent être qualifiés par des conditions (parfois appelées conditions de garde ou gardes de transition) qui influencent le chemin de transition à prendre. Par exemple, un événement de connexion avec une combinaison nom d'utilisateur/mot de passe valide entraînera une transition différente d'un événement de connexion avec un mot de passe non valide. Ces informations sont représentées dans un diagramme de transitions d'état ou dans une table de transitions d'état (qui peut également inclure des transitions non valides potentielles entre les états) [Black09].

**Applicabilité**
Le test des transitions d'état s'applique à tout logiciel qui a des états définis et qui réagit à des événements qui provoqueront des transitions entre ces états (p. ex., changement d'écran). Le test des transitions d'état peut être utilisé à n'importe quel niveau de test. Les logiciels embarqués, les logiciels web et tout type de logiciel transactionnel sont de bons candidats pour ce type de test. Les systèmes de contrôle, par exemple les contrôleurs des feux de circulation, sont également de bons candidats pour ce type de test.

**Limitations/Difficultés**
La détermination des états est souvent la partie la plus difficile de la définition du diagramme de transitions d'état ou de la table de transitions d'état. Lorsque l'objet de test dispose d'une interface utilisateur, les différents écrans qui s'affichent pour l'utilisateur sont souvent utilisés pour définir les états. Pour les logiciels embarqués, les états peuvent dépendre des états du matériel

Outre les états eux-mêmes, l'unité de base des tests de transition d'état est la transition individuelle, également connue sous le nom de 0-switch. Tester uniquement chaque transition individuelle permettra de trouver certains types de défauts de transition d'état, mais plus de défauts pourront être trouvés en testant des séquences de transitions. Une séquence de deux transitions successives est appelée 1-switch; une séquence de trois transitions successives est appelée 2-switch, et ainsi de suite. (Ces séquences de transitions sont parfois alternativement désignées comme N-1 switch, où N représente le nombre de transitions qui seront traversées. Une transition unique, par exemple (un 0-switch), serait une séquence 1-1 switch. [Bath14])

**Couverture**
Comme pour d'autres types de techniques de test, il existe une hiérarchie des niveaux de couverture. Le degré minimum acceptable de couverture est d'avoir visité tous les états et traversé chaque transition au moins une fois. 100% de couverture des transitions (aussi désignée par 100% de couverture 0-switch ou 100% de couverture logique des branches) garantira que chaque état est visité et chaque transition est traversée, à moins que la conception du système ou que le modèle de transitions d'état (diagramme ou table) soit défectueux. Selon les relations entre les états et les transitions, il peut être nécessaire de traverser certaines transitions plus d'une fois afin d'exécuter d'autres transitions une seule fois.

Le terme « couverture n-switch» se rapporte au nombre de séquences de longueur N+1 couvertes, en pourcentage du nombre total de séquences de cette longueur. Par exemple, l'obtention de 100% de couverture 1-switch exige que chaque séquence valide de deux transitions successives ait été

testée au moins une fois. Ce test peut déclencher certains types de défaillances que la couverture 100% 0-switch manquerait.

La « couverture aller-retour » s'applique aux situations dans lesquelles des séquences de transitions forment des boucles. La couverture aller-retour à 100% est atteinte lorsque toutes les boucles de n'importe quel état vers le même état ont été testées pour tous les états où les boucles commencent et se terminent. Une boucle ne peut contenir plus d'une occurrence d'un état particulier (à l'exception de la première/dernière) [Offutt16].

Pour l'une ou l'autre de ces approches, un degré de couverture encore plus élevé tentera d'inclure toutes les transitions non valides identifiées dans une table d'états. Les exigences relatives à la couverture et la couverture de séquences de transitions à tester doivent préciser si les transitions invalides sont comprises.

La conception des cas de test pour atteindre la couverture souhaitée est soutenue par le diagramme de transitions d'état ou la table de transitions pour l'objet de test particulier. Cette formation peut également être représentée dans un tableau qui montre les transitions n-switch pour une valeur particulière de « n » [Black09].

Une procédure manuelle peut être appliquée pour l'identification des éléments à couvrir (p. ex., transitions, états ou n-switch). Une méthode suggérée consiste à imprimer le diagramme de transitions d'état et la table de transitions d'état et à utiliser un stylo ou un crayon pour marquer les éléments couverts jusqu'à ce que la couverture requise soit visible [Black09]. Cette approche deviendrait trop longue pour des diagrammes ou des tables de transitions d'état plus complexes. Un outil devrait donc être utilisé pour soutenir les tests de transitions d'état.

**Types de défauts**
Les défauts typiques [Beizer95] incluent:

- Types ou valeurs d'événements incorrects
- Types ou valeurs d'actions incorrects
- État initial incorrect
- Incapacité d'atteindre un ou des état(s) de sortie
- Incapacité d'atteindre les états requis
- États supplémentaires (inutiles)
- Incapacité d'exécuter correctement certaines transitions valides
- Possibilité d'exécuter des transitions non valides

Lors de la création du modèle de transitions d'état, des défauts peuvent être trouvés dans le document de spécification. Les types de défauts les plus courants sont les omissions (il n'y a pas d'information sur ce qui devrait réellement se produire dans une certaine situation) et les contradictions.

### 3.2.5 Technique de classification arborescente

Les classifications arborescentes visent à prendre en charge certaines techniques de test boîte noire en permettant la création d'une représentation graphique de l'espace de données qui s'applique à l'objet de test.

Les données sont organisées en classifications et classes comme suit:

- Classifications: Celles-ci représentent des paramètres dans l'espace de données de l'objet de test. Il peut s'agir de paramètres d'entrée, qui peuvent contenir davantage les états de l'environnement et les préconditions, ainsi que de paramètres de sortie. Par exemple, si une application peut être configurée de différentes façons, les classifications peuvent inclure le client, le navigateur, la langue et le système d'exploitation.

- Classes: Chaque classification peut avoir n'importe quel nombre de classes et de sous-classes décrivant l'occurrence du paramètre. Chaque classe, ou partition d'équivalence, est une valeur spécifique dans une classification. Dans l'exemple ci-dessus, la classification linguistique peut inclure des cours d'équivalence pour l'anglais, le français et l'espagnol.

Les classification arborescentes permettent à l'Analyste de Test d'entrer des combinaisons comme bon lui semble. Cela inclut, par exemple, les combinaisons de paires (NDT : pairwise, voir Section 3.2.6), les combinaisons 3-wise et 1-wise.

Des informations supplémentaires concernant l'utilisation de la technique de classification arborescente sont fournies dans [Bath14] et [Black09].

**Applicabilité**
La création d'une classification arborescente permet à un Analyste de Test d'identifier les partitions d'équivalence (classifications) et les valeurs (classes) dans une partition qui présentent un intérêt.

Une analyse plus poussée du diagramme de classification arborescente permet d'identifier les valeurs limites possibles et d'identifier certaines combinaisons d'intrants qui présentent un intérêt particulier ou peuvent être écartées (p. ex., parce qu'elles sont incompatibles). La classification arborescente qui en résulte peut ensuite être utilisée pour prendre en charge des tests de partitions d'équivalence, l'analyse des valeurs limites ou les tests par paires (voir la section 3.2.6).

**Limitations/Difficultés**
À mesure que la quantité de classifications et/ou de classes augmente, le diagramme devient plus grand et moins facile à utiliser.

**Couverture**
Les cas de test peuvent être conçus pour obtenir, par exemple, une couverture minimale des classes (c.-à-d. toutes les valeurs d'une classe testée au moins une fois).L'Analyste de Test, peut également décider de couvrir les combinaisons de test par paires ou même 3-wise.

**Types de défauts**
Les types de défauts trouvés dépendent de la ou des techniques que les classifications arborescentes soutiennent (c.-à-d. les partitions d'équivalence, l'analyse des valeurs limites ou le test par paires).

### 3.2.6 Test par paires

Le test par paires est utilisé lors du test d'un logiciel dans lequel plusieurs paramètres d'entrée, chacun avec plusieurs valeurs possibles, doivent être testés en combinaison, donnant lieu à plus de combinaisons que ce qu'il est possible de tester dans le temps imparti. Les paramètres d'entrée doivent être compatibles en ce sens que toute option pour n'importe quel facteur (c.-à-d. toute valeur sélectionnée pour n'importe quel paramètre d'entrée) peut être combinée avec n'importe quelle option pour n'importe quel autre facteur. La combinaison d'un paramètre spécifique (variable ou facteur) avec une valeur spécifique de ce paramètre est appelée valeur de paire de paramètre (par exemple, si « couleur » est un paramètre avec (disons) sept valeurs autorisées, alors « couleur = rouge » serait une valeur de paire de paramètre).

Le test par paires utilise l'arithmétique combinatoire pour s'assurer que chaque valeur de paire de paramètre est testée une fois avec chaque valeur de paire de paramètre de l'autre paramètre (c.-à-d. « toutes les paires » de valeurs de paire de paramètre sont testées), tout en évitant de tester toutes les combinaisons de paires de valeur de paramètres. Si l'Analyste de Test utilisait une approche manuelle (non recommandée), un tableau serait établi avec une ligne pour chaque cas de test (p. ex., seize lignes) et une colonne pour chaque paramètre (p. ex., quatre colonnes). L'Analyste de Test

remplirait ensuite le tableau de valeurs de façon à ce que toutes les paires de valeurs puissent être identifiées dans le tableau (voir [Black09]). Toutes les cases non complétées dans le tableau peuvent être remplies de valeurs par l'Analyste de Test sur la base de ses propres connaissances du domaine.

Il existe un certain nombre d'outils disponibles pour aider un Analyste de Test dans cette tâche (voir www.pairwise.org pour des exemples). Ils ont besoin en entrée d'une liste de paramètres et de leurs valeurs et calculent un ensemble de test minimal couvrant toutes les paires de valeurs de paire de paramètre. La sortie de l'outil peut être utilisée comme entrée pour les cas de test. Notez que l'Analyste de Test doit fournir les résultats attendus pour chaque combinaison créée par les outils.

Les classifications arborescentes (voir la section 3.2.5) sont souvent utilisés conjointement avec le test par paires [Bath14]. La conception des classifications arborescentes est prise en charge par des outils et permet de visualiser des combinaisons de paramètres et de leurs valeurs (certains outils offrent une amélioration pour le test par paires). Cela permet d'identifier les informations suivantes:

- Entrées à utiliser pour la technique de test par paires.
- Combinaisons particulières d'intérêt (p. ex., fréquemment utilisées ou sources courantes de défauts)
- Combinaisons particulières incompatibles. Cela ne suppose pas que les facteurs combinés ne s'affecteront pas les uns les autres; ils le pourraient très bien, mais ne devraient s'affecter les uns les autres que de manière acceptable.
- Relations logiques entre les variables. Par exemple, "si variable1 = x, alors la variable 2 ne pourra pas être y". Les classifications arborescentes qui capturent ces relations sont appelées « modèle de caractéristiques ».

**Applicabilité**
Le problème d'avoir trop de combinaisons de valeurs de paramètres se manifeste dans au moins deux situations différentes liées aux tests. Certaines situations de test impliquent plusieurs paramètres chacun avec un certain nombre de valeurs possibles, par exemple un écran avec plusieurs champs d'entrée. Dans ce cas, les combinaisons de valeurs de paramètres constituent les données d'entrée pour les cas de test. En outre, certains systèmes peuvent être configurables selon un certain nombre de dimensions, ce qui entraîne une variété de configurations potentiellement grande. Dans ces deux situations, les tests par paires peuvent être utilisés pour identifier un sous-ensemble de combinaisons, de taille réalisable.

Pour les paramètres ayant de nombreuses valeurs, la technique des partitions d'équivalence ou un autre mécanisme de sélection peut d'abord être appliqué à chaque paramètre individuellement pour réduire le nombre de valeurs pour chaque paramètre, avant que des tests par paires ne soient appliqués pour réduire l'ensemble des combinaisons résultantes. La représentation des paramètres et de leurs valeurs dans une classification arborescente soutient cette activité.

Ces techniques sont généralement appliquées aux niveaux de test suivants : intégration, système et intégration de systèmes.

**Limitations/Difficultés**
La principale limite à ces techniques est l'hypothèse que les résultats de quelques tests sont représentatifs de tous les tests et que ces quelques tests représentent l'utilisation prévue. S'il y a une interaction inattendue entre certaines variables, elle peut passer inaperçue avec ce type de test si cette combinaison particulière n'est pas testée. Ces techniques peuvent être difficiles à expliquer à un public non technique car il peut ne pas comprendre la réduction logique des tests. Une telle explication devrait être modérée en mentionnant les résultats d'études empiriques [Kuhn16], qui ont montré que dans le domaine des dispositifs médicaux à l'étude, 66% des défaillances ont été déclenchées par une seule variable et 97% par une variable ou deux variables en interaction. Il existe

un risque résiduel que les tests par paires ne détectent pas les défaillances dues à l'interaction de 3 variables ou plus.

Il est parfois difficile d'identifier les paramètres et leurs valeurs respectives. Cette tâche doit donc être effectuée avec l'appui des classifications arborescentes dans la mesure du possible (voir la section 3.2.5). Il est difficile de trouver manuellement un ensemble minimal de combinaisons pour satisfaire un certain niveau de couverture. Les outils trouvent généralement le plus petit ensemble possible de combinaisons. Certains outils permettent de forcer certaines combinaisons à être incluses dans ou exclues de la sélection finale des combinaisons. Cette possibilité peut être utilisée par un Analyste de Test pour mettre plus ou moins l'accent sur certains facteurs sur la base de ses connaissances du domaine ou de l'utilisation du produit.

**Couverture**
La couverture à 100% des tests par paires exige que chaque paire de valeurs de n'importe quelle paire de paramètres soit incluse dans au moins une combinaison.

**Types de défauts**
Le type de défauts le plus courant présent avec ce type de test est celui des défauts liés aux valeurs combinées de deux paramètres.

### 3.2.7 Test des cas d'utilisation

Le test des cas d'utilisation fournit des tests transactionnels, basés sur des scénarios qui doivent émuler l'utilisation prévue du composant ou du système spécifié par le cas d'utilisation. Les cas d'utilisation sont définis en termes d'interactions entre des acteurs et un composant ou système qui accomplit certains objectifs. Les acteurs peuvent être des utilisateurs humains, du matériel externe ou d'autres composants ou systèmes.

**Applicabilité**
Le test des cas d'utilisation est généralement appliquée aux niveaux de test suivants : système et acceptation. Il peut être utilisé pour les tests d'intégration lorsque le comportement des composants ou des systèmes est spécifié par des cas d'utilisation et même pour les tests de composants lorsque le comportement des composants est spécifié par des cas d'utilisation. Les cas d'utilisation sont aussi souvent la base pour les tests de performance parce qu'ils décrivent l'utilisation réaliste du système. Les scénarios décrits dans les cas d'utilisation peuvent être attribués à des utilisateurs virtuels pour créer une charge réaliste sur le système ( à condition que des exigences de charge et de performance soient spécifiées dans ou pour les cas d'utilisation).

**Limitations/Difficultés**
Pour être valides, les cas d'utilisation doivent représenter des transactions utilisateur valides. Les spécifications de cas d'utilisation sont une forme de conception de système. Les exigences de ce que les utilisateurs doivent accomplir doivent provenir d'utilisateurs ou de représentants d'utilisateurs, et doivent être vérifiées par rapport aux exigences organisationnelles avant de concevoir les cas d'utilisation correspondants. La valeur d'un cas d'utilisation est réduite s'il ne reflète pas les besoins réels de l'utilisateur et de l'organisation, ou entrave plutôt que d'aider à l'accomplissement des tâches utilisateur.

Une définition précise de la gestion des comportements d'exception, des alternatives et des erreurs est importante pour que la couverture des tests soit rigoureuse. Les cas d'utilisation doivent être considérés comme une ligne directrice, mais pas comme une définition complète de ce qui devrait être testé, car ils peuvent ne pas fournir une définition claire de l'ensemble des exigences. Il peut également être utile de créer d'autres modèles, tels que des diagrammes de flux et/ou des tables de décision, à partir de la spécification des cas d'utilisation pour améliorer l'exactitude des tests et pour

vérifier le cas d'utilisation lui-même. Comme avec d'autres formes de spécification, cela est susceptible de révéler des anomalies logiques dans la spécification de cas d'utilisation, si elle existe.

**Couverture**
Le niveau minimum acceptable de couverture d'un cas d'utilisation est d'avoir un cas de test pour le comportement de base et suffisamment de cas de test supplémentaires pour couvrir chaque comportement alternatif ou de gestion d'erreur. S'il est nécessaire de réduire le nombre de tests, plusieurs comportements alternatifs peuvent être incorporés dans un cas de test à condition qu'ils soient mutuellement compatibles. Si une meilleure capacité de diagnostic est nécessaire (p. ex., pour aider à isoler les défauts),un cas de test supplémentaire par comportement alternatif peut être conçu, même si des comportements alternatifs imbriqués exigeront toujours que certains soient fusionnés en cas de test uniques (par exemple, des comportements alternatifs de fin par rapport à des comportements alternatifs non terminaux au sein d'un comportement d'exception « réessayer »).

**Types de défauts**
Les défauts incluent une mauvaise gestion des comportements définis, des comportements alternatifs manqués, un traitement incorrect des conditions présentées et des rapports d'erreur mal mis en œuvre ou incorrects.

### 3.2.8 Combiner les techniques

Parfois, les techniques sont combinées pour créer des cas de test. Par exemple, les conditions identifiées dans une table de décision peuvent être soumises aux partitions d'équivalence pour découvrir plusieurs manières de remplir une condition. Les cas de test couvriraient alors non seulement toutes les combinaisons de conditions, mais aussi, pour les conditions qui ont été partitionnées, des cas de test supplémentaires seraient générés pour couvrir les partitions d'équivalence. Lors de la sélection de la technique particulière à appliquer, l'Analyste de Test doit tenir compte de l'applicabilité de la technique, de ses limites et difficultés, ainsi que des objectifs de test en termes de couverture et de défauts à détecter. Ces aspects sont décrits pour les techniques individuelles couvertes dans le présent chapitre. Il se peut qu'il n'y ait pas une seule « meilleure » technique pour une situation donnée. Combiner les techniques fournira souvent la couverture la plus complète, à condition qu'il y ait suffisamment de temps et de compétences pour appliquer correctement les techniques.

## 3.3 Technique de test basée sur l'expérience

Le test basé sur l'expérience utilise les compétences et l'intuition des testeurs, ainsi que leur expérience avec des applications ou des technologies similaires pour cibler les tests afin d'augmenter la détection des défauts. Ces techniques de test vont des « tests rapides », dans lesquels le testeur n'a pas d'activités officiellement préplanifiées, à des sessions planifiées à l'avance et documentées. Elles sont presque toujours utiles, mais ont une valeur particulière lorsque les aspects inclus dans la liste suivante d'avantages peuvent être atteints.

Le test basé sur l'expérience présente les avantages suivants:
- Il est efficace pour trouver des défauts.
- Il peut être une bonne alternative à des approches plus structurées au cas où la documentation du système est pauvre.
- Il peut être appliqué lorsque le temps de test est sévèrement limité.
- Il permet d'appliquer au test l'expertise disponible dans le domaine et la technologie. Cela peut impliquer ceux qui ne participent pas aux tests, par exemple, des Analystes Métier, des utilisateurs ou des clients.
- Il permet de fournir tôt des commentaires aux développeurs.
- Il aide l'équipe à se familiariser avec le logiciel tel qu'il est produit.

- Il est efficace lorsque des défaillances opérationnelles sont analysées.
- Il permet d'appliquer une diversité de techniques de test.

Le test basé sur l'expérience présente les inconvénients suivants:

- Il peut être inapproprié pour les systèmes nécessitant une documentation détaillée des tests.
- Des niveaux élevés de répétabilité sont difficiles à atteindre.
- La capacité à évaluer avec précision la couverture des tests est limitée.
- Les tests sont moins adaptés à une automatisation ultérieure.

Lorsqu'ils utilisent des approches réactives et heuristiques, les testeurs utilisent normalement les tests basés sur l'expérience, qui sont plus réactifs aux événements que les approches de test préplanifiées. En outre, l'exécution et l'évaluation sont des tâches concurrentes. Certaines approches structurées des tests basés sur l'expérience ne sont pas entièrement dynamiques, c'est-à-dire que les tests ne sont pas entièrement créés au moment où le testeur exécute le test. Cela peut être le cas, par exemple, lorsque l'estimation d'erreur est utilisée pour cibler des aspects particuliers de l'objet de test avant l'exécution du test.

Notez que bien que certaines idées sur la couverture soient présentées pour les techniques discutées ici, les techniques de test basées sur l'expérience n'ont pas de critères formels de couverture.

### 3.3.1 Estimation d'erreur

Lors de l'utilisation de la technique de l'estimation d'erreur, un Analyste de Test utilise son expérience pour deviner les erreurs potentielles qui auraient pu être faites lors de la conception et du développement du code. Lorsque les erreurs attendues ont été identifiées, l'Analyste de Test détermine ensuite les meilleures méthodes à utiliser pour découvrir les défauts qui en résultent. Par exemple, si un Analyste de Test s'attend à ce que le logiciel présente des défaillances lorsqu'un mot de passe non valide est entré, des tests seront exécutés pour entrer une variété de valeurs différentes dans le champ de mot de passe pour vérifier si l'erreur a effectivement été commise et a entraîné un défaut qui peut être considéré comme une défaillance lorsque les tests sont exécutés.

En plus d'être utilisée comme technique de test, l'estimation d'erreurs est également utile lors de l'analyse des risques pour identifier les modes de défaillance potentiels. [Myers11]

**Applicabilité**
L'estimation d'erreur se fait principalement pendant les tests d'intégration et système, mais peut être employée à n'importe quel niveau de test . Cette technique est souvent utilisée avec d'autres techniques et contribue à élargir la portée des cas de test existants. L'estimation d'erreur peut également être utilisée efficacement lors du test d'une nouvelle version du logiciel pour tester les défauts courants avant de commencer des tests plus rigoureux et scénarisés.

**Limitations/Difficultés**
Les limitations et les difficultés suivantes s'appliquent à l'estimation d'erreur:

- La couverture est difficile à évaluer et varie considérablement en fonction de la capacité et de l'expérience de l'Analyste de Test.
- Elle est mieux utilisée par un testeur expérimenté qui est familier avec les types de défauts qui sont couramment introduits dans le type de code testé.
- Elle est couramment utilisée, mais n'est souvent pas documentée et peut donc être moins reproductible que d'autres formes de test.
- Les cas de test peuvent être documentés, mais d'une manière que seul l'auteur comprend et peut reproduire.

**Couverture**

Lorsqu'une taxonomie est utilisée, la couverture est déterminée en prenant le nombre d'éléments de la taxonomie testés divisé par le nombre total d'éléments de la taxonomie et en indiquant la couverture en pourcentage. Sans taxonomie, la couverture est limitée par l'expérience et la connaissance du testeur et le temps disponible. La quantité de défauts trouvés avec cette technique varie en fonction de la capacité du testeur à cibler les zones problématiques.

**Types de défauts**
Les défauts typiques sont généralement ceux définis dans la taxonomie particulière ou « devinés » par l'Analyste de Test, et qui pourraient ne pas avoir été trouvés avec des tests boîte noire.

### 3.3.2 Test basé sur une checklist

Lors de l'application de la technique de test basé sur une checklist, un Analyste de Test expérimenté utilise une checklist de haut niveau d'éléments à noter, à vérifier ou à retenir, ou un ensemble de règles ou de critères sur lesquels un objet de test doit être vérifié. Ces checklists sont établies en fonction d'un ensemble de normes, d'expériences et d'autres considérations. Une checklist standard d'interface utilisateur peut être utilisée comme base pour tester une application et est un exemple de test basé sur une checklist. Dans les projets Agile, les checklists peuvent être établies à partir des critères d'acceptation d'une User Story.

**Applicabilité**
Le test basé sur une checklist est le plus efficace dans des projets disposant d'une équipe de test expérimentée qui connaît bien le logiciel à tester ou qui connaît bien le domaine couvert par la checklist (p. ex., pour appliquer avec succès checklist de vérification de l'interface utilisateur, l'Analyste de Test peut être familier avec les tests d'interface utilisateur, mais pas avec le système spécifique en cours de test). Étant donné que les checklists sont de haut niveau et qu'elles ont tendance à ne pas avoir les étapes détaillées que l'on trouve couramment dans les cas de test et les procédures de test, la connaissance du testeur est utilisée pour combler les lacunes. En supprimant les étapes détaillées, les checklist nécessitent un effort de maintenance réduit et peuvent être appliquées à plusieurs versions similaires.

Les checklists conviennent parfaitement aux projets où le logiciel est livré et modifié rapidement. Cela permet de réduire à la fois le temps de préparation et le temps de maintenance pour la documentation des tests. Elles peuvent être utilisées pour n'importe quel niveau de test et sont également utilisées pour les tests de régression et les « smoke tests ».

**Limitations/Difficultés**
La nature haut niveau des checklists peut affecter la reproductibilité des résultats des tests. Il est possible que plusieurs testeurs interprètent les checklists différemment et suivent différentes approches pour remplir les éléments de la checklist. Cela peut entraîner des résultats différents, même si la même checklist est utilisée. Cela peut entraîner une couverture plus large, mais la reproductibilité est parfois sacrifiée. Les checklists peuvent également entraîner une confiance excessive quant au niveau de couverture qui est atteint puisque le test réel dépend du jugement du testeur. Les checklists peuvent être obtenues à partir de cas de test ou de listes de tests plus détaillés et ont tendance à croître au fil du temps. Une maintenance est requise pour s'assurer que les checklists couvrent les aspects importants du logiciel testé.

**Couverture**
La couverture peut être déterminée en prenant le nombre d'éléments de la checklist testés divisé par le nombre total d'éléments de la checklist et en indiquant la couverture en pourcentage. La couverture est aussi bonne que la checklist, mais, en raison de la nature haut niveau de la checklist, les résultats varient en fonction de l'Analyste de Test qui utilise la checklist.

**Types de défauts**

Les défauts typiques trouvés avec cette technique provoquent des défaillances résultant de la variation des données, de la séquence des étapes ou du workflow général pendant les tests.

### 3.3.3 Tests exploratoires

Les tests exploratoires sont caractérisés par la réalisation simultanée par le testeur de l'apprentissage de l'objet de test et de ses défauts, la planification du travail de test à effectuer, la conception et l'exécution des tests, et la communication des résultats. Le testeur ajuste dynamiquement les objectifs de test pendant l'exécution et ne prépare qu'une documentation légère. [Whittaker09]

**Applicabilité**
De bons tests exploratoires sont planifiés, interactifs, et créatifs. Ils nécessitent peu de documentation sur le système à tester et sont souvent utilisés dans les situations où la documentation n'est pas disponible ou n'est pas adaptée à d'autres techniques de test. Les tests exploratoires sont souvent utilisés en complément d'autres techniques de test et servent de base à l'élaboration de cas de test supplémentaires. Les tests exploratoires sont fréquemment utilisés sur les projets Agile pour effectuer des tests de User Stories de manière flexible et rapide avec seulement une documentation minimale. Toutefois, la technique peut également être appliquée à des projets utilisant un cycle de vie séquentiel de développement logiciel.

**Limitations/Difficultés**
La couverture des tests exploratoires peut être sporadique et la reproductibilité des tests effectués peut être difficile. L'utilisation de chartes de test pour désigner les zones à couvrir dans une session de test et des sessions de durée limitée pour déterminer le temps autorisé pour les tests sont des méthodes utilisées pour gérer les tests exploratoires. À la fin d'une session de test ou d'une série de sessions, le Test Manager peut tenir une séance de débriefing pour recueillir les résultats des tests et déterminer les chartes pour les prochaines sessions.

Une autre difficulté avec les séances exploratoires est de les suivre avec précision dans un système de gestion des tests. Cela se fait parfois en créant des cas de test qui sont en fait des sessions exploratoires. Cela permet de suivre le temps alloué aux tests exploratoires et la couverture prévue, au même titre que les autres efforts de test.

Étant donné que la reproductibilité peut être difficile avec les tests exploratoires, cela peut également causer des problèmes lorsque vous avez besoin de retrouver les étapes pour reproduire une défaillance. Certaines organisations utilisent la capacité de capture/rejeux d'un outil d'automatisation des tests pour enregistrer les étapes suivies par un testeur exploratoire. Ceci fournit un enregistrement complet de toutes les activités au cours de la session exploratoire (ou de toute session de test basé sur l'expérience). Analyser les détails pour trouver la cause réelle d'une défaillance peut être fastidieux, mais au moins il y a un enregistrement de toutes les étapes qui ont été effectuées.

D'autres outils peuvent être utilisés pour sauvegarder des séances de test exploratoires, mais ceux-ci n'enregistrent pas les résultats escomptés parce qu'ils ne capturent pas les interactions avec l'interface graphique. Dans ce cas, les résultats attendus doivent être notés de manière à ce qu'une analyse adéquate des défauts puisse être effectuée si nécessaire. En général, il est recommandé de prendre des notes lors des tests exploratoires pour permettre la reproductibilité en cas de besoin.

**Couverture**
Des chartes peuvent être conçues pour des tâches, des objectifs et des livrables. Des séances exploratoires sont ensuite prévues pour atteindre ces critères. La charte peut également indiquer où concentrer l'effort de test, ce qui est dans et hors de la portée de la session de test, et quelles ressources devraient être engagées pour effectuer les tests prévus. Une session peut être utilisée

pour se concentrer sur des types de défauts particuliers et d'autres domaines potentiellement problématiques qui peuvent être abordés, sans la formalité des tests scénarisés.

**Types de défauts**
Les défauts typiques trouvés lors des tests exploratoires sont les problèmes basés sur des scénarios qui ont été manqués lors des tests d'aptitude fonctionnelle scénarisés, les problèmes qui se situent entre les limites fonctionnelles et les problèmes liés aux workflows. Des problèmes de performance et de sécurité sont parfois découverts lors des tests exploratoires..

### 3.3.4 Techniques de test basées sur les défauts

Une technique de test basée sur les défauts est une technique dans laquelle le type de défaut recherché est utilisé comme base pour la conception des tests, avec des tests dérivés systématiquement de ce qui est connu sur le type de défaut. Contrairement aux tests boîte noire qui dérivent leurs tests de la base de test, les tests basés sur les défauts dérivent des tests de listes qui se concentrent sur les défauts. En général, les listes peuvent être organisées en types de défauts, causes racines, symptômes de défaillance et autres données liées aux défauts. Des listes standard s'appliquent à plusieurs types de logiciels et ne sont pas spécifiques à un produit. L'utilisation de ces listes permet de tirer parti des connaissances standard de l'industrie pour dériver des tests particuliers. En adhérant à des listes spécifiques à l'industrie, des métriques relatives à l'occurrence des défauts peuvent être suivies entre les projets et même entre les organisations. Les listes de défauts les plus courantes sont celles qui sont spécifiques à l'organisation ou au projet et qui font usage d'une expertise et d'une expérience spécifiques.

Les tests basés sur les défauts peuvent également utiliser des listes de risques et de scénarios de risque identifiés comme bases pour cibler les tests. Cette technique de test permet à un Analyste de Test de cibler un type spécifique de défaut ou de travailler systématiquement à travers une liste de défauts connus et communs d'un type particulier. À partir de ces informations, l'Analyste de Test crée les cas de test et les conditions de test qui feront en sorte que le défaut se manifestera, s'il existe.

**Applicabilité**
Les tests basés sur les défauts peuvent être appliqués à n'importe quel niveau de test, mais ils sont le plus souvent appliqués lors des tests système.

**Limitations/Difficultés**
De multiples taxonomies de défauts existent et peuvent être axées sur des types particuliers de tests, tels que la facilité d'utilisation. Il est important de choisir une taxonomie qui s'applique au logiciel testé, le cas échéant. Par exemple, il se peut qu'il n'y ait pas de taxonomies disponibles pour les logiciels innovants. Certaines organisations ont compilé leurs propres taxonomies de défauts probables ou fréquemment vus. Quelle que soit la taxonomie utilisée, il est important de définir la couverture prévue avant de commencer le test.

**Couverture**
La technique fournit des critères de couverture qui sont utilisés pour déterminer quand tous les cas de test utiles ont été identifiés. Les éléments de couverture peuvent être des éléments structurels, des éléments de spécification, des scénarios d'utilisation ou toute autre combinaison de ces éléments, selon la liste des défauts. En pratique, les critères de couverture des techniques de test basées sur les défauts ont tendance à être moins systématiques que pour les techniques de test boîtes noires dans la mesure où seules les règles générales de couverture sont données et que la décision spécifique sur ce qui constitue la limite de la couverture utile est discrétionnaire. Comme pour d'autres techniques, les critères de couverture ne signifient pas que l'ensemble des tests est complet, mais plutôt que les défauts considérés ne suggèrent plus de tests utiles basés sur cette technique..

**Types de défauts**

Les types de défauts découverts dépendent généralement de la taxonomie utilisée. Par exemple, si une liste de défauts d'interface utilisateur est utilisée, la majorité des défauts découverts seraient probablement liés à l'interface utilisateur, mais d'autres défauts peuvent être découverts comme sous-produits des tests spécifiques.

## 3.4 Appliquer la technique la plus appropriée

Les techniques de test boîte noire et basées sur l'expérience sont plus efficaces lorsqu'elles sont utilisées ensemble. Les techniques basées sur l'expérience comblent les lacunes dans la couverture des tests qui résultent de toute faiblesse systématique des techniques de test boîte noire.

Il n'y a pas une technique parfaite pour toutes les situations. Il est important pour l'Analyste de Test de comprendre les avantages et les inconvénients de chaque technique et de pouvoir choisir la meilleure technique ou le meilleur ensemble de techniques pour la situation, compte tenu du type de projet, du calendrier, de l'accès à l'information, des compétences du testeur et d'autres facteurs qui peuvent influencer la sélection.

Dans le traitement de chaque technique de test boîte noire ou basée sur l'expérience (voir les sections 3.2 et 3.3 respectivement), les informations fournies dans « applicabilité », « limitations/difficultés » et « couverture » guident un Analyste de Test dans la sélection des techniques de test les plus appropriées.

# 4. Test des caractéristiques qualité du logiciel - 180 mins.

## Mots clés

accessibilité, compatibilité, adéquation fonctionnelle, complétude fonctionnelle, exactitude fonctionnelle, aptitude fonctionnelle, interopérabilité, facilité d'apprentissage, opérabilité, Software Usability Measurement Inventory (SUMI), utilisabilité, protection contre les erreurs de l'utilisateur, expérience utilisateur, esthétique de l'interface utilisateur, Website Analysis and MeasureMent Inventory (WAMMI)

## Objectifs d'apprentissage pour Test des caractéristiques qualité du logiciel

### 4.1 Introduction

Pas d'objectifs d'apprentissage

### 4.2 Caractéristiques de qualité pour les tests de domaine métier

TA-4.2.1 (K2) Expliquer quelles techniques de test sont appropriées pour tester la complétude, l'exactitude et l'adéquation fonctionnelles

TA-4.2.2 (K2) Définir les défauts typiques à cibler pour les caractéristiques de complétude, exactitude et adéquation fonctionnelles

TA-4.2.3 (K2) Définir quand la complétude, l'exactitude et l'adéquation fonctionnelles doivent être testées dans le cycle de vie du développement logiciel

TA-4.2.4 (K2) Expliquer les approches qui seraient appropriées pour vérifier et valider à la fois la mise en œuvre des exigences d'utilisabilité et le respect des attentes de l'utilisateur

TA-4.2.5 (K2) Expliquer le rôle de l'Analyste de Test dans les tests d'interopérabilité, y compris l'identification des défauts à cibler

TA-4.2.6 (K2) Expliquer le rôle de l'Analyste de Test dans les tests de portabilité, y compris l'identification des défauts à cibler

TA-4.2.7 (K4) Pour un ensemble donné d'exigences, déterminer les conditions de test requises pour vérifier les caractéristiques de qualité fonctionnelles et/ou non fonctionnelles dans le champ d'application de l'Analyste de Test

## 4.1 Introduction

Bien que le chapitre précédent ait décrit des techniques spécifiques dont dispose le testeur, le présent chapitre examine l'application de ces techniques pour évaluer les caractéristiques utilisées pour décrire la qualité des applications ou des systèmes logiciels.

Ce syllabus traite des caractéristiques de qualité qui peuvent être évaluées par un Analyste de Test. Les attributs à évaluer par l'Analyste Technique de Test sont pris en considération dans le syllabus Analyste Technique de Test Avancé [CTAL-TTA].

La description des caractéristiques de qualité du produit fournies dans ISO25010 [ISO25010] est utilisée comme guide pour décrire les caractéristiques. Le modèle ISO de qualité du logiciel divise la qualité du produit en différentes caractéristiques de qualité du produit, chacune d'entre elles pouvant avoir des sous-caractéristiques. Celles-ci sont indiquées dans le tableau ci-dessous, ainsi qu'une indication des caractéristiques/sous-caractéristiques couvertes par les syllabus Analyste de Test et Analyste Technique de Test:

| Caractéristiques | Sous-caractéristiques | Analyste de Test | Analyste Technique de Test |
|---|---|---|---|
| Aptitude fonctionnelle | Exactitude fonctionnelle, adéquation fonctionnelle, complétude fonctionnelle | X | |
| Fiabilité | Maturité, tolérance aux défauts, récupération, disponibilité | | X |
| Utilisabilité | Reconnaissance de la pertinence, apprentissage, opérabilité, esthétique de l'interface utilisateur, protection contre les erreurs de l'utilisateur, accessibilité | X | |
| Efficacité de la performance | Comportement dans le temps, utilisation des ressources, capacité | | X |
| Maintenabilité | Facilité d'analyse, facilité de modification, testabilité, modularité, réutilisabilité | | X |
| Portabilité | Adaptabilité, facilité d'installation, facilité de remplacement | X | X |
| Sécurité | Confidentialité, intégrité, non-répudiation, responsabilité, authenticité | | X |
| Compatibilité | Coexistence | | X |
| | Interopérabilité | X | |

Les domaines de responsabilité de l'Analyste de Test et de l'Analyste Technique de Test sont indiqués dans le tableau ci-dessus. Bien que cette répartition du travail puisse varier selon les organisations, c'est celle qui est suivie dans les syllabi ISTQB associés.

Pour toutes les caractéristiques et les sous-caractéristiques de qualité abordées dans cette section, les risques typiques doivent être reconnus afin qu'une stratégie de test appropriée puisse être formée et documentée. Les tests des caractéristiques de qualité nécessitent une attention particulière au cycle de vie de développement logiciel, au calendrier, aux outils requis, à la disponibilité des logiciels et de la documentation, et à l'expertise technique. En l'absence d'une stratégie pour répondre à chaque caractéristique et à ses besoins uniques en matière de tests, le testeur peut ne pas avoir suffisamment de temps dans son planning pour la planification, la montée en compétence et le temps d'exécution des tests. [Bath14]. Certains de ces tests, par exemple les tests d'utilisabilité, peuvent nécessiter l'affectation de ressources humaines spéciales, une planification approfondie, des laboratoires dédiés, des outils spécifiques, des compétences spécialisées en test et, dans la plupart

des cas, une quantité importante de temps. Dans certains cas, les tests d'utilisabilité peuvent être effectués par un groupe distinct d'experts en utilisabilité ou en expérience utilisateur..

Bien que l'Analyste de Test ne soit pas responsable des caractéristiques de qualité qui nécessitent une approche plus technique, il est important que l'Analyste de Test soit conscient des autres caractéristiques et comprenne les zones qui se chevauchent pour les tests. Par exemple, un objet de test qui échoue aux tests de performances échouera probablement dans les tests d'utilisabilité s'il est trop lent pour que l'utilisateur l'utilise efficacement. De même, un objet de test ayant des problèmes d'interopérabilité avec certains composants n'est probablement pas prêt pour les tests de portabilité, car cela aura tendance à masquer les problèmes les plus fondamentaux lorsque l'environnement est modifié.

## 4.2 Caractéristiques de qualité pour les tests de domaine métier

Les tests d'aptitude fonctionnelle sont au centre des préoccupations de l'Analyste de Test. Les tests d'aptitude fonctionnelle sont axés sur « ce que » l'objet de test fait. La base de test pour les tests d'aptitude fonctionnelle est généralement constituée d'éléments tels que les exigences, une spécification, une expertise métier spécifique ou le besoin implicite. Les tests fonctionnels varient selon le niveau de test dans lequel ils sont effectués et peuvent également être influencés par le cycle de vie de développement logiciel. Par exemple, un test fonctionnel effectué lors des tests d'intégration testera la fonctionnalité d'interfaçage de modules qui correspond à l'implémentation d'une fonction définie unique. Au niveau des tests système, les tests fonctionnels incluent le test de la fonctionnalité du système dans son ensemble. Pour les systèmes de systèmes, les tests d'adéquation fonctionnelle se concentreront principalement sur les tests de bout en bout dans les systèmes intégrés. Une grande variété de techniques de test sont utilisées lors des tests d'aptitude fonctionnelle (voir le chapitre 3).

Dans un environnement Agile, les tests d'aptitude fonctionnelle comprennent généralement ce qui suit:

- Test d'une fonctionnalité spécifique (p. ex., User story) devant être implémentée dans une itération particulière
- Test de régression pour toutes les fonctionnalités inchangées

En plus des tests d'aptitude fonctionnelle couverts dans cette section, il existe également certaines caractéristiques de qualité faisant partie du domaine de responsabilité de l'Analyste de Test qui sont considérés comme des domaines de test non fonctionnels (axés sur « comment » l'objet de test fournit la fonctionnalité).

### 4.2.1 Test d'exactitude fonctionnelle

L'exactitude fonctionnelle implique la vérification de l'adhésion de l'application aux exigences spécifiées ou implicites et peut également inclure la précision de calcul. Les tests d'exactitude utilisent bon nombre des techniques de test expliquées au chapitre 3 et utilisent souvent la spécification ou un système hérité (legacy) comme oracle de test. Les tests d'exactitude peuvent être effectués à n'importe quel niveau de test et ciblent une mauvaise gestion des données ou des situations.

### 4.2.2 Test d'adéquation fonctionnelle

Les tests d'adéquation fonctionnelle impliquent d'évaluer et de valider la pertinence d'un ensemble de fonctions pour les tâches spécifiées prévues. Ces tests peuvent être basés sur la conception fonctionnelle (p. ex., cas d'utilisation et/ou User Stories). Les tests d'adéquation fonctionnelle sont habituellement effectués pendant les tests système, mais peuvent également être effectués au cours des dernières étapes des tests d'intégration. Les défauts découverts dans ce test sont des indications

que le système ne sera pas en mesure de répondre aux besoins de l'utilisateur d'une manière qui sera considérée comme acceptable.

### 4.2.3 Test de complétude fonctionnelle

Les tests de complétude fonctionnelle sont effectués pour déterminer la couverture des tâches spécifiées et des objectifs de l'utilisateur par la fonctionnalité implémentée. La traçabilité entre les éléments de spécification (p. ex., les exigences, les User Stories, les cas d'utilisation) et la fonctionnalité implémentée (p. ex., fonction, unité, workflow) est essentielle pour permettre de déterminer la complétude requise. La mesure de la complétude fonctionnelle peut varier en fonction du niveau de test particulier et/ou du cycle de vie de développement logiciel utilisé. Par exemple, l'exhaustivité fonctionnelle d'une itération Agile peut être basée sur des User Stories et des Features implémentées. La complétude fonctionnelle des tests d'intégration du système peut se concentrer sur la couverture des cas métier de haut niveau.

La détermination de la complétude fonctionnelle est généralement prise en charge par les outils de gestion des tests si l'Analyste de Test maintient la traçabilité entre les cas de test et les spécifications fonctionnelles. Des niveaux plus faibles que prévu de complétude fonctionnelle sont des indications que le système n'a pas été entièrement mis en œuvre.

### 4.2.4 Test d'interopérabilité

Les tests d'interopérabilité vérifient l'échange d'informations entre deux ou plusieurs systèmes ou composants. Les tests mettent l'accent sur la capacité d'échanger des informations et d'utiliser par la suite les informations qui ont été échangées. Les tests doivent couvrir tous les environnements cibles prévus (y compris les variations dans le matériel, le logiciel, le middleware, le système d'exploitation, etc.) pour s'assurer que l'échange de données fonctionnera correctement. En réalité, cela peut n'être possible que pour un nombre relativement restreint d'environnements. Dans ce cas, les tests d'interopérabilité peuvent être limités à la sélection d'un groupe représentatif d'environnements. La spécification des tests d'interopérabilité exige que les combinaisons des environnements cibles prévus soient identifiées, configurées et accessibles à l'équipe de test. Ces environnements sont ensuite testés à l'aide d'une sélection de cas de test fonctionnels qui exercent les différents points d'échange de données présents dans l'environnement.

L'interopérabilité se rapporte à la façon dont les différents composants et systèmes logiciels interagissent les uns avec les autres. Les logiciels ayant de bonnes caractéristiques d'interopérabilité peuvent être intégrés à un certain nombre d'autres systèmes sans nécessiter de changements majeurs. Le nombre de changements et les efforts nécessaires pour mettre en œuvre et tester ces changements peuvent être utilisés comme mesure de l'interopérabilité.

Les tests d'interopérabilité des logiciels peuvent, par exemple, se concentrer sur les caractéristiques de conception suivantes:

- Utilisation de normes de communication de l'industrie, telles que XML
- Capacité à détecter automatiquement les besoins de communication des systèmes avec lesquels le logiciel interagit et de s'ajuster en conséquence

Les tests d'interopérabilité peuvent être particulièrement importants pour:

- Les produits et outils logiciels disponibles sur étagère
- Les applications basées sur un système de systèmes
- Les systèmes basés sur l'Internet des objets
- Les Web services avec connectivité à d'autres systèmes

Ce type de test est effectué lors des tests d'intégration de composants et les tests système et se concentre sur l'interaction du système avec son environnement. Au niveau de l'intégration du système, ce type de test est effectué pour déterminer dans quelle mesure le système entièrement développé interagit avec d'autres systèmes. Étant donné que les systèmes peuvent interagir à plusieurs niveaux, l'Analyste de Test doit comprendre ces interactions et être en mesure de créer les conditions qui exerceront les différentes interactions. Par exemple, si deux systèmes échangent des données, l'Analyste de Test doit être en mesure de créer les données nécessaires et les transactions nécessaires pour effectuer l'échange de données. Il est important de se rappeler que toutes les interactions peuvent ne pas être clairement spécifiées dans les documents d'exigences. Au lieu de cela, bon nombre de ces interactions ne seront définies que dans l'architecture du système et les documents de conception. L'Analyste de Test doit être préparé et en mesure d'examiner ces documents afin de déterminer les points d'échange d'informations entre les systèmes et entre le système et son environnement afin de s'assurer que tous sont testés. Des techniques telles que les partitions d'équivalence, l'analyse des valeurs limites, les tables de décision, les diagrammes d'état, les cas d'utilisation et le test par paires sont toutes applicables aux tests d'interopérabilité. Les défauts typiques trouvés incluent l'échange de données incorrect entre les composants en interaction.

## 4.2.5 Test d'utilisabilité

Les Analystes de Test sont souvent amenés à coordonner et soutenir l'évaluation de l'utilisabilité. Il peut s'agir de spécifier des tests d'utilisabilité ou d'agir en tant que modérateur travaillant avec des utilisateurs pour effectuer des tests. Pour faire cela efficacement, un Analyste de Test doit comprendre les principaux aspects, objectifs et approches impliqués dans ces types de test. Veuillez consulter le Syllabus ISTQB de Spécialiste pour les tests d'utilisabilité [ISTQB_FL_UT] pour plus de détails que ceux fournis dans cette section.

Il est important de comprendre pourquoi les utilisateurs peuvent avoir de la difficulté à utiliser le système ou n'ont pas une expérience utilisateur positive (UX, User Experience) (par exemple, avec l'utilisation d'un logiciel pour le divertissement). Pour obtenir cette compréhension, il est d'abord nécessaire de comprendre que le terme « utilisateur » peut s'appliquer à un large éventail de types de personnes, allant des experts informatiques aux enfants en passant par les personnes handicapées.

### 4.2.5.1 Caractéristiques de l'utilisabilité

Voici les trois caractéristiques examinées dans la présente section:

- Utilisabilité
- Expérience Utilisateur (UX)
- Accessibilité

**Utilisabilité**

Les tests d'utilisabilité ciblent les défauts logiciels qui ont un impact sur la capacité d'un utilisateur à effectuer des tâches via l'interface utilisateur. Ces défauts peuvent affecter la capacité de l'utilisateur à atteindre ses objectifs de manière efficace, ou efficiente, ou avec satisfaction. Les problèmes d'utilisabilité peuvent entraîner de la confusion, une erreur, un retard ou une défaillance pure et simple de la part de l'utilisateur.

Voici les sous-caractéristiques individuelles [ISO 25010] de d'utilisabilité:

- Reconnaissance de la pertinence(c'est-à-dire, la facilité de compréhension) - attributs du logiciel qui affectent l'effort requis par l'utilisateur pour reconnaître le concept logique et son applicabilité
- Facilité d'apprentissage - attributs des logiciels qui affectent l'effort requis par l'utilisateur pour apprendre l'application
- Opérabilité - attributs du logiciel qui affectent l'effort requis par l'utilisateur pour effectuer des tâches efficacement et avec efficience
- Esthétique de l'interface utilisateur (c.-à-d. attractivité)) - attributs visuels du logiciel qui sont appréciés par l'utilisateur

- Protection contre les erreurs de l'utilisateur - degré auquel un système protège les utilisateurs contre les erreurs
- Accessibilité (voir ci-dessous)

**Expérience Utilisateur (UX)**
L'évaluation de l'expérience utilisateur répond à l'expérience utilisateur entière avec l'objet de test, et pas seulement à l'interaction directe. Ceci est d'une importance particulière pour les objets de test où des facteurs tels que le plaisir et la satisfaction des utilisateurs sont essentiels à la réussite commerciale de l'entreprise.

Les facteurs typiques qui influencent l'expérience utilisateur comprennent:
- Image de marque (c.-à-d. la confiance des utilisateurs dans le fabricant)
- Comportement interactif
- L'utilité de l'objet de test, y compris le système d'aide, le support et la formation

**Accessibilité**
Il est important de tenir compte de l'accessibilité aux logiciels pour ceux qui ont des besoins particuliers ou des restrictions pour son utilisation. Cela inclut les personnes handicapées. Les tests d'accessibilité devraient tenir compte des normes pertinentes, telles que les « Lignes directrices sur l'accessibilité du contenu Web » (WCAG, Web Content Accessibility Guidelines), et la législation, comme les Lois sur la discrimination liée au handicap (Disability Discrimination Acts) (Northern Ireland, Australia), Loi de 2010 sur l'égalité (Equality Act 2010) (England, Scotland, Wales) et la "Section 508" (US). L'accessibilité, de même que la facilité d'utilisation, doit être prise en considération lors de la conduite des activités de conception. Les tests se produisent souvent au niveau intégration et se poursuivent par des tests au niveau système et des tests au niveau acceptation. Les défauts sont généralement déterminés lorsque le logiciel ne respecte pas les règlements ou les normes à appliquer pour le logiciel.

Les mesures typiques visant à améliorer l'accessibilité mettent l'accent sur les possibilités offertes aux utilisateurs handicapés d'interagir avec l'application. Il s'agit notamment des éléments suivants :
- Reconnaissance vocale pour les entrées
- S'assurer que le contenu non textuel présenté à l'utilisateur dispose d'une alternative textuelle équivalente
- Permettre le redimensionnement de texte sans perte de contenu ou de fonctionnalité

Les lignes directrices sur l'accessibilité aident l'Analyste de Test en fournissant une source d'information et des checklists qui peuvent être utilisées pour les tests (des exemples de lignes directrices sur l'accessibilité sont données dans [ISTQB_FL_UT]). En outre, des outils et des plugins de navigateur sont disponibles pour aider les testeurs à identifier les problèmes d'accessibilité, tels que le mauvais choix de couleurs dans les pages Web qui violent les recommandations relatives au daltonisme.

#### 4.2.5.2 Approches d'évaluation de l'utilisabilité
La facilité d'utilisation, l'expérience utilisateur et l'accessibilité peuvent être testées par une ou plusieurs des approches suivantes:
- Tests d'utilisabilité
- Revues d'utilisabilité
- Enquêtes et questionnaires d'utilisabilité

**Test d'utilisabilité**
Les tests d'utilisabilité évaluent la facilité avec laquelle les utilisateurs peuvent utiliser ou apprendre à utiliser le système pour atteindre un objectif spécifié dans un contexte spécifique. Les tests d'utilisabilité visent à mesurer les caractéristiques suivantes:

- Efficacité - capacité de l'objet de test à permettre aux utilisateurs d'atteindre des objectifs spécifiés avec précision et exhaustivité dans un contexte d'utilisation spécifié
- Efficience - capacité de l'objet de test à permettre aux utilisateurs de dépenser des quantités appropriées de ressources par rapport à l'efficacité atteinte dans un contexte d'utilisation déterminé
- Satisfaction - capacité de l'objet de test à satisfaire les utilisateurs dans un contexte d'utilisation spécifié

Il est important de noter que la conception et la spécification des tests d'utilisabilité sont souvent effectuées par l'Analyste de Test en coopération avec des testeurs qui ont des compétences spéciales en test d'utilisabilité, et des ingénieurs de conception d'utilisabilité qui comprennent le processus de conception centrée sur l'humain (voir [ISTQB_FL_UT] pour plus de détails).

**Revues d'utilisabilité**
Les inspections et les revues d'utilisabilité sont un type de test effectués du point de vue de la facilité d'utilisation qui contribuent à accroître le niveau de participation de l'utilisateur. Cela peut être rentable en trouvant des problèmes d'utilisabilité dans les spécifications des exigences et les documents de conceptions au début du cycle de vie de développement logiciel. L'évaluation heuristique (inspection systématique de la conception d'une interface utilisateur pour son utilisabilité) peut être utilisée pour trouver les problèmes d'utilisabilité dans la conception afin qu'ils puissent être pris en charge dans le cadre d'un processus de conception itérative. Il s'agit de faire examiner l'interface par un petit groupe d'évaluateurs et de juger de sa conformité aux principes reconnus de l'utilisabilité (l'heuristique). Les revues sont plus efficaces lorsque l'interface utilisateur est visible. Par exemple, des exemples de captures d'écran sont généralement plus faciles à comprendre et à interpréter qu'une simple description des fonctionnalités données par un écran particulier. La visualisation est importante pour une revue adéquate de l'utilisabilité.

**Enquêtes et questionnaires d'utilisabilité**
Des techniques d'enquête et de questionnaire peuvent être appliquées pour recueillir des observations et des commentaires concernant le comportement des utilisateurs avec le système. Enquêtes normalisées et accessibles au public telles que SUMI (Software Usability Measurement Inventory) et WAMMI (Website Analysis and MeasureMent Inventory) permettent l'analyse comparative par rapport à une base de données de mesures d'utilisabilité antérieures. En outre, puisque SUMI fournit des mesures tangibles de l'utilisabilité, cela peut fournir un ensemble de critères d'achèvement / acceptation.

### 4.2.6 Test de portabilité

Les tests de portabilité se rapportent à la mesure du degré par lequel un composant ou un système logiciel peut être transféré dans son environnement prévu, soit initialement, soit à partir d'un environnement existant.

La classification ISO 25010 des caractéristiques de qualité du produit comprend les sous-caractéristiques suivantes de la portabilité:

- Facilité d'installation
- Adaptabilité
- Facilité de remplacement

Les tâches d'identification des risques et de conception des tests pour les sous-caractéristiques de portabilité est partagée entre l'Analyste de Test et l'Analyste Technique de Test (voir [ISTQB_ALTTA_SYL] Section 4.7).

#### 4.2.6.1 Test de facilité d'installation

Les tests de facilité d'installation sont effectués sur le logiciel et les procédures écrites utilisées pour installer et désinstaller le logiciel sur son environnement cible.

Les objectifs typiques des tests portés par l'Analyste de Test incluent:

- Valider que différentes configurations du logiciel peuvent être installées avec succès. Lorsqu'un grand nombre de paramètres peuvent être configurés, l'Analyste de Test peut concevoir des tests à l'aide de la technique des tests par paires pour réduire le nombre de combinaisons de paramètres testées et se concentrer sur des configurations d'intérêt particulières (p. ex., celles fréquemment utilisées).
- Tester la justesse fonctionnelle des procédures d'installation et de désinstallation.
- Effectuer des tests fonctionnels à la suite d'une installation ou d'une désinstallation pour détecter les défauts qui pourraient avoir été introduits (p. ex., configurations incorrectes, fonctions non disponibles).
- Identifier des problèmes d'utilisabilité dans les procédures d'installation et de désinstallation (p. ex., valider que les utilisateurs reçoivent des instructions compréhensibles et des messages de rétroaction/erreur lors de l'exécution de la procédure).).

#### 4.2.6.2 Test d'adaptabilité

Les tests d'adaptabilité vérifient si une application donnée peut fonctionner correctement dans tous les environnements cibles prévus (matériel, logiciel, middleware, système d'exploitation, etc.). L'Analyste de Test prend en charge les tests d'adaptabilité en concevant des tests qui identifient les combinaisons des environnements cibles prévus (p. ex., versions de différents systèmes d'exploitation mobiles pris en charge, différentes versions de navigateurs qui peuvent être utilisées). Ces environnements sont ensuite testés à l'aide d'une sélection de cas de test fonctionnels qui exercent les différents composants présents dans l'environnement.

#### 4.2.6.3 Test de facilité de remplacement

Les tests de facilité de remplacement se concentrent sur la capacité des composants ou versions logicielles dans un système à être échangés contre d'autres. Cela peut être particulièrement pertinent pour les architectures système basées sur l'Internet des objets, où l'échange de différents appareils matériels et/ou installations logicielles est un phénomène courant. Par exemple, un périphérique matériel utilisé dans un entrepôt pour enregistrer et contrôler les niveaux de stock peut être remplacé par un périphérique matériel plus avancé (par exemple, avec un meilleur scanner) ou le logiciel installé peut être mis à niveau avec une nouvelle version qui permet d'émettre automatiquement des commandes de remplacement de stock au système d'un fournisseur.

Les tests de facilité de remplacement peuvent être effectués par l'Analyste de Test en parallèle avec des tests d'intégration fonctionnelle où plus d'un composant alternatif est disponible pour l'intégration dans le système complet.

# Revues - 120 mins.

## Mots clés

Revues basées sur des checklists

## Objectifs d'apprentissage pour Revues

### 5.1 Introduction

Pas d'objectifs d'apprentissage

### 5.2 Utiliser des checklists dans les revues

TA-5.2.1 (K3) Identifier des problèmes dans une spécification d'exigence selon les informations de checklists fournies dans le syllabus

TA-5.2.2 (K3) Identifier des problèmes dans une User Story selon les informations de checklists fournies dans le syllabus

## 5.1 Introduction

Un processus de revue réussi exige la planification, la participation et le suivi. Les Analyste de Tests doivent participer activement au processus de revue, en fournissant leurs points de vue. Lorsqu'elles sont effectuées correctement, les revues peuvent être le plus grand et le plus rentable contributeur de la qualité globale fournie.

## 5.2 Utiliser des checklists dans les revues

Mener des revues avec des checklists est la technique la plus courante utilisée par un Analyste de Test lors de la revue de la base de test. Les checklists sont utilisées lors des revues pour rappeler aux participants de vérifier des points spécifiques au cours de la revue. Elles peuvent également aider à dépersonnaliser la revue (p. ex., « C'est la même checklist que nous utilisons pour chaque revue. Nous ne ciblons pas particulièrement votre produit d'activités. »).

Des revues basées sur des checklists peuvent être effectuées de manière générique pour toutes les revues ou se concentrer sur des caractéristiques qualité, des zones ou des types de documents spécifiques. Par exemple, une checklist générique peut vérifier les propriétés générales du document telles que l'identifiant unique, l'absence de la mention «à compléter », la mise en forme appropriée et d'autres éléments de conformité. Une checklist spécifique pour un document d'exigences peut contenir des vérifications de l'utilisation appropriée des termes « doit » et « devrait », des vérifications de la testabilité de chaque exigence énoncée, et ainsi de suite.

Le format des exigences peut également indiquer le type de checklist à utiliser. Un document d'exigences qui est au format textuel aura des critères de revue différents d'un document basé sur des diagrammes.

Les checklists peuvent également être orientées vers un aspect particulier,:

- Un ensemble de compétences programmeur/architecte ou un ensemble de compétences testeur
  - Dans le cas de l'Analyste de Test, la checklist des compétences des testeurs serait la plus appropriée
  - Ces checklist peuvent inclure des éléments tels que indiqués aux sections 5.2.1 et 5.2.2
- Un certain niveau de risque (p. ex., dans les systèmes critiques pour la sécurité) – les checklists incluent généralement les informations spécifiques nécessaires au niveau de risque
- Une technique de test spécifique - la checklist se concentrera sur l'information nécessaire pour une technique particulière (p. ex, les règles à représenter dans une table de décision)
- Un élément de spécification particulier, tel qu'une exigence, un cas d'utilisation ou une User Story – ces checklists sont discutées dans les sections suivantes et ont généralement une approche différente de celles utilisées par un Analyste technique de Test pour la revue du code ou de l'architecture

### 5.2.1 Revues des exigences

Les éléments suivants sont un exemple de ce qu'une checklist axée sur les exigences pourrait inclure:

- Source de l'exigence (p. ex., personne, département)
- Testabilité de chaque exigence
- Critères d'acceptation pour chaque exigence
- Disponibilité d'une structure d'appel des cas d'utilisation, le cas échéant
- Identifiant unique de chaque exigence/cas d'utilisation/User Story

- Version de chaque exigence/cas d'utilisation/User Story
- Traçabilité pour chaque exigence à partir des exigences métier/marketing
- Traçabilité entre les exigences et/ou les cas d'utilisation (le cas échéant)
- Utilisation d'une terminologie cohérente (p. ex., utilisation d'un glossaire))

Il est important de se rappeler que si une exigence n'est pas testable, ce qui signifie qu'elle est définie de telle sorte que l'Analyste de Test ne puisse pas déterminer comment la tester, alors il y a un défaut dans cette exigence. Par exemple, une exigence selon laquelle « Le logiciel doit être très convivial » n'est pas testable. Comment l'Analyste de Test peut-il déterminer si le logiciel est convivial, ou même très convivial ? Si, au lieu de cela, l'exigence dit "Le logiciel doit se conformer aux normes d'utilisabilité énoncées dans le document sur les normes d'utilisabilité, version xxx", et si le document sur les normes d'utilisabilité existe vraiment, alors il s'agit d'une exigence testable. Il s'agit également d'une exigence primordiale, car cette exigence s'applique à tous les éléments de l'interface. Dans ce cas, cette exigence pourrait facilement engendrer de nombreux cas de test individuels dans une application non triviale. La traçabilité de cette exigence, ou peut-être du document sur les normes d'utilisabilité, aux cas de test est également essentielle parce que si la spécification d'utilisabilité référencée devait changer, tous les cas de test devront être examinés et mis à jour au besoin.

Une exigence est également impossible à tester si le testeur est incapable de déterminer si le test a réussi ou échoué, ou est incapable de construire un test qui peut réussir ou échouer. Par exemple, « le système doit être disponible 100 % du temps, 24 heures par jour, 7 jours par semaine, 365 (ou 366) jours par an » n'est pas testable.

Une checklist simple[1] pour l'examen des cas d'utilisation peut inclure les questions suivantes:

- Le comportement principal (chemin) est-il clairement défini?
- Tous les comportements alternatifs (chemins) sont-ils identifiés, complets avec la gestion des erreurs?
- Les messages d'interface utilisateur sont-ils définis?
- Y a-t-il un seul comportement principal (normalement oui, sinon il y a plusieurs cas d'utilisation)?
- Chaque comportement est-il testable?

### 5.2.2 Revues de User Stories

Dans un projet Agile, les exigences prennent généralement la forme de User Stories. Ces User Stories représentent de petites unités de fonctionnalité démontrable. Alors qu'un cas d'utilisation est une transaction utilisateur qui traverse plusieurs zones de fonctionnalité, une User Story est une fonctionnalité plus isolée et est généralement précisée par le temps qu'il faut pour le développer. Une checklist[2] pour une User Story peut inclure les éléments suivants ::

- La User Story est-elle appropriée pour l'itération/sprint cible?
- La User Story est-elle écrite du point de vue de la personne qui le demande?
- Les critères d'acceptation sont-ils définis et testables?
- La caractéristique est-elle clairement définie et distincte?
- La User Story est-elle indépendante des autres?
- La User Story est-elle priorisée?
- La User Story suit-elle le format couramment utilisé:
  En tant que < type d'utilisateur >, je veux < un certain objectif > afin de < une raison > [Cohn04]

[1]La question de l'examen fournira un sous-ensemble de la checklist des cas d'utilisation avec laquelle répondre à la question
[2]La question de l'examen fournira un sous-ensemble de la checklist de la User Story avec laquelle répondre à la question

Si la User Story définit une nouvelle interface, l'utilisation d'une checklist de User Story générique (comme celle ci-dessus) et d'une checklist détaillée de l'interface utilisateur serait appropriée.

### 5.2.3 Adapter les checklists

Une checklist peut être adaptée en fonction des éléments suivants ::

- Organisation (p. ex., compte tenu des politiques, des normes, des conventions, des contraintes juridiques de l'entreprise)
- Projet / effort de développement (p. ex., objectifs, normes techniques, risques)
- Produit d'activités en cours de revue (p. ex., les revues de codes peuvent être adaptées à des langages de programmation spécifiques)
- Niveau de risque du produit d'activités en cours de revue
- Techniques de test à utiliser

De bonnes checklists trouveront des problèmes et aideront également à entamer des discussions sur d'autres éléments qui n'auraient peut-être pas été mentionnés spécifiquement dans la checklist. L'utilisation d'une combinaison de checklists est un moyen solide de s'assurer qu'une revue permettra d'obtenir un produit d'activités de la plus haute qualité. La mise en œuvre de revues fondées sur des checklists avec des checklists normalisées telles que celles mentionnées dans le syllabus Niveau Fondation et l'élaboration de checklists spécifiques à l'organisation, comme celles indiquées ci-dessus, aideront l'Analyste de Test à être efficace dans les revues.

Pour de plus amples renseignements sur les revues et les inspections, voir [Gilb93] et [Wiegers03]. D'autres exemples de checklists peuvent être obtenus à partir des références de la section 7.4.

# Outils de test et automatisation - 90 mins.

## Mots clés

tests dirigés par les mots clés, préparation des données de test, conception des tests, exécution des test, script de test

## Objectifs d'apprentissage pour Outils de test et automatisation

### 6.1 Introduction

Pas d'objectifs d'apprentissage

### 6.2 Automatisation dirigée par les mots clés

TA-6.2.1 (K3) Pour un scénario donné, déterminer les activités appropriées pour un Analyste de Test dans un projet d'automatisation dirigée par les mots clés

### 6.3 Types d'outils de test

TA-6.3.1 (K2) Expliquer l'utilisation et les types d'outils de test appliqués dans la conception des tests, la préparation des données de test et l'exécution des tests

## 6.1 Introduction

Les outils de test peuvent grandement améliorer l'efficacité et la précision des tests. Les outils de test et les approches d'automatisation qui sont utilisés par un Analyste de Test sont décrits dans le présent chapitre. Il convient de noter que l'Analyste de Test travaille en collaboration avec les Développeurs, les Ingénieurs en Automatisation des Tests et l'Analyste Technique de Tests pour créer des solutions d'automatisation des tests. L'automatisation axée sur les mots clés implique en particulier l'Analyste de Test et tire parti de son expérience avec le métier et la fonctionnalité du système.

De plus amples informations sur l'automatisation des tests et le rôle de l'Ingénieur en Automatisation des Tests sont fournis dans le syllabus ISTQB niveau avancé Ingénieur en automatisation des tests [ISTQB_ALTAE_SYL].

## 6.2 Automatisation dirigée par les mots clés

L'approche de test dirigée par les mots clés est l'une des principales approches d'automatisation des tests et implique l'Analyste de Test pour la fourniture des principales entrées: mots clés et données.

Les mots clés (parfois appelés mots d'action) sont principalement, mais pas exclusivement, utilisés pour représenter des interactions métier de haut niveau avec un système (p. ex., « annuler commande »). Chaque mot clé est généralement utilisé pour représenter un certain nombre d'interactions détaillées entre un acteur et le système sous test. Des séquences de mots clés (y compris des données de test pertinentes) sont utilisées pour spécifier les cas de test. [Buwalda02]

Dans l'automatisation des tests, un mot clé est implémenté sous la forme d'un ou de plusieurs scripts de test exécutables. Des outils lisent les cas de test écrits sous la forme d'une séquence de mots clés appelant les scripts de test appropriés qui implémentent la fonctionnalité du mot clé. Les scripts sont implémentés d'une manière très modulaire pour permettre une association facile à des mots clés spécifiques. Des compétences en programmation sont nécessaires pour implémenter ces scripts modulaires.

Voici les principaux avantages de l'automatisation des tests dirigée par les mots clés:

- Les mots clés qui se rapportent à une application ou à un domaine métier particulier peuvent être définis par des experts du domaine. Cela peut rendre la tâche de spécification de cas de test plus efficace.
- Une personne ayant principalement une expertise métier peut bénéficier de l'exécution automatique de cas de test (une fois que les mots clés ont été implémentés en tant que scripts) sans avoir à comprendre le code d'automatisation sous-jacent.
- L'utilisation d'une technique d'écriture modulaire permet une maintenance efficace des cas de test par l'ingénieur d'automatisation de test lorsque des modifications à la fonctionnalité et à l'interface du logiciel en cours de test se produisent [Bath14].
- Les spécifications des cas de test sont indépendantes de leur implémentation.

Les Analyste de Tests créent et maintiennent habituellement les données mot-clé/action. Ils doivent se rendre compte que la tâche de développement de script est encore nécessaire pour l'implémentation des mots clés. Une fois que les mots clés et les données à utiliser ont été définis, la personne chargée de l'automatisation des tests (p. ex., un Analyste Technique de Test ou un Ingénieur en Automatisation des Tests) traduit les mots clés de processus métier et les actions de bas niveau en scripts de test automatisés.

Bien que l'automatisation par mots clés est généralement exécutée pendant les tests système, le développement du code peut commencer dès la conception des tests. Dans un environnement itératif,

en particulier lorsque l'intégration continue/le déploiement continu est utilisé, le développement de l'automatisation des tests est un processus continu.

Une fois les mots clés d'entrée et les données créés, l'Analyste de Test prend la responsabilité d'exécuter les cas de test dirigés par les mots clé et d'analyser les défaillances qui peuvent se produire.

Lorsqu'une anomalie est détectée, l'Analyste de Test doit aider à rechercher la cause de la défaillance et à déterminer si le défaut est dans les mots clés, les données d'entrée, le script d'automatisation lui-même ou dans l'application en cours de test. Habituellement, la première étape dans le dépannage consiste à exécuter le même test avec les mêmes données manuellement pour voir si la défaillance se trouve dans l'application elle-même. Si cela ne provoque pas de défaillance, l'Analyste de Test doit examiner la séquence de tests qui a conduit à la défaillance et déterminer si le problème s'est produit dans une étape précédente (peut-être en introduisant des données d'entrée incorrectes), mais que le défaut n'a été révélé que plus tard dans le traitement. Si l'Analyste de Test n'est pas en mesure de déterminer la cause de la défaillance, les informations de l'exécution doivent être remises à l'Analyste Technique de Test ou au Développeur pour une analyse plus approfondie.

## 6.3 Types d'outils de test

Une grande partie du travail de l'Analyste de Test exige l'utilisation efficace des outils. Cette efficacité est renforcée par:

- Savoir quels outils utiliser
- Savoir que les outils peuvent accroître l'efficacité de l'effort de test (par exemple, en aidant à fournir une meilleure couverture des tests dans le temps imparti)

### 6.3.1 Outils de conception des tests

Les outils de conception de test sont utilisés pour aider à créer des cas de test et des données de test à appliquer pour les tests. Ces outils peuvent fonctionner à partir de documents d'exigences spécifiques, de modèles (p. ex., UML) ou d'entrées fournies par l'Analyste de Test. Les outils de conception des tests sont souvent conçus et développés pour fonctionner avec des formats particuliers et des outils particuliers tels que des outils spécifiques de gestion des exigences.

Les outils de conception des tests peuvent fournir des informations à l'Analyste de Test pour déterminer les types de tests nécessaires pour obtenir un niveau ciblé particulier de couverture de test, la confiance dans le système ou des actions d'atténuation des risques produits. Par exemple, les outils de classification arborescente génèrent (et affichent) l'ensemble des combinaisons nécessaires pour atteindre une couverture complète en fonction d'un critère de couverture donné. Ces informations peuvent ensuite être utilisées par l'Analyste de Test pour déterminer les cas de test qui doivent être exécutés.

### 6.3.2 Outils de préparation des données de test

Les outils de préparation des données de test peuvent fournir les avantages suivants:

- Analyser un document tel qu'un document d'exigences ou même le code source pour déterminer les données requises lors des tests pour atteindre un niveau de couverture.
- Prendre un ensemble de données à partir d'un système en production et le brouiller ou l'anonymiser pour supprimer toute information personnelle tout en maintenant l'intégrité interne de ces données. Les données brouillées peuvent ensuite être utilisées pour les tests sans risque de fuite de sécurité ou d'utilisation abusive de renseignements personnels. Ceci est particulièrement important lorsque de grands volumes de données réalistes sont nécessaires, et lorsque les risques pour la sécurité et la confidentialité des données s'appliquent.

- Générer des données de test synthétiques à partir d'ensembles donnés de paramètres d'entrée (p. ex., pour une utilisation dans des tests aléatoires). Certains de ces outils analyseront la structure de la base de données pour déterminer les entrées requises par l'Analyste de Test.

### 6.3.3 Outils d'exécution automatisée des tests

Les outils d'exécution des tests sont utilisés par l'Analyste de Test à tous les niveaux de test pour effectuer des tests automatisés et vérifier les résultats des tests. L'objectif de l'utilisation d'un outil d'exécution de test correspond généralement à un ou plusieurs des objectifs suivants:

- Réduire les coûts (en termes d'effort et/ou de temps)
- Exécuter plus de tests
- Exécuter le même test dans de nombreux environnements
- Pour rendre l'exécution des tests plus répétable
- Pour exécuter des tests qui seraient impossibles à exécuter manuellement (c.-à-d. des tests de validation de grands volumes de données)

Ces objectifs se recoupent souvent avec les principaux objectifs d'accroître la couverture tout en réduisant les coûts.

Le retour sur investissement pour les outils d'exécution des tests est généralement le plus élevé lors de l'automatisation des tests de régression en raison du faible niveau de maintenance prévu et de l'exécution répétée des tests. L'automatisation des « smoke tests » peut également être une utilisation efficace de l'automatisation en raison de l'utilisation fréquente des tests, de la nécessité d'un résultat rapide et, bien que le coût de maintenance puisse être plus élevé, de la capacité d'avoir un moyen automatisé d'évaluer un nouveau build dans un environnement d'intégration continue.

Les outils d'exécution des tests sont couramment utilisés lors des tests système et d'intégration. Certains outils, en particulier les outils de test API, peuvent également être utilisés dans les tests de composants. Tirer parti des outils là où ils sont le plus applicables aidera à améliorer le rendement de l'investissement.

# Références

## 7.1 Standards


- [ISO25010] ISO/IEC 25010 (2014) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models Chapter 4
- [ISO29119-4] ISO/IEC/IEEE 29119-4 Software and Systems Engineering – Software Testing – Part 4, Test Techniques, 2015
- [RTCA DO-178C/ED-12C]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12C, 2013.
  Chapter 1


## 7.2 Documents ISTQB et IREB


- [IREB_CPRE] IREB Certified Professional for Requirements Engineering Foundation Level Syllabus, Version 2.2.2, 2017
- [ISTQB_AGILE_SYL] ISTQB Foundation Agile Software Testing, Version 2014
- [ISTQB_AL_OVIEW] ISTQB Advanced Level Overview, Version 2.0
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2019
- [ISTQB_ALTAE_SYL] ISTQB Advanced Level Test Automation Engineer Syllabus, Version 2017
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 2019
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2018
- [ISTQB_FL_UT] ISTQB Foundation Level Specialist Syllabus Usability Testing, Version 2018
- [ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, Version 3.2, 2018


## 7.3 Ouvrages


[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook (2nd Edition)”, Rocky Nook, 2014, ISBN 978-1-933952-24-6

[Beizer95] Boris Beizer, “Black-box Testing”, John Wiley & Sons, 1995, ISBN 0-471-12094-4

[Black02]: Rex Black, “Managing the Testing Process (2nd edition)”, John Wiley & Sons: New York, 2002, ISBN 0-471-22398-0

[Black07]: Rex Black, “Pragmatic software testing: Becoming an effective and efficient test professional”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Black09]: Rex Black, “Advanced Software Testing, Volume 1”, Rocky Nook, 2009, ISBN 978-1-933-952-19-2

[Buwalda02]: Hans Buwalda, “Integrated Test Design and Automation: Using the Test Frame Method”, Addison-Wesley Longman, 2002, ISBN 0-201-73725-6

[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5

[Copeland04]: Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2004, ISBN 1-58053-791-X

[Craig02]: Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9

[Gilb93]: Tom Gilb, Graham Dorothy, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4

[Koomen06]: Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon "TMap NEXT, for result driven testing", UTN Publishers, 2006, ISBN 90-72194-80-2

[Kuhn16]: D. Richard Kuhn et al, “Introduction to Combinatorial Testing, CRC Press, 2016, ISBN 978-0-429-18515-1

[Myers11]: Glenford J. Myers, “The Art of Software Testing 3rd Edition”, John Wiley & Sons, 2011, ISBN: 978-1-118-03196-4

[Offutt16]: Jeff Offutt, Paul Ammann, Introduction to Software Testing – 2nd Edition, Cambridge University Press, 2016, ISBN 13: 9781107172012,

[vanVeenendaal12]: Erik van Veenendaal, “Practical risk-based testing." Product Risk Management: The PRISMA Method ”, UTN Publishers, The Netherlands, ISBN 9789490986070

[Wiegers03]: Karl Wiegers, “Software Requirements 2”, Microsoft Press, 2003, ISBN 0-735-61879-8

[Whittaker03]: James Whittaker, “How to Break Software”, Addison-Wesley, 2003, ISBN 0-201-79619-8

[Whittaker09]: James Whittaker, “Exploratory software testing: tips, tricks, tours, and techniques to guide test design”, Addison-Wesley, 2009, ISBN 0-321-63641-4

## 7.4 Autres références

Les références suivantes indiquent l’information disponible sur Internet et ailleurs. Même si ces références ont été vérifiées au moment de la publication de ce syllabus de niveau avancé, l’ISTQB ne peut être tenu responsable si les références ne sont plus disponibles.

- Chapitre3
  - Czerwonka, Jacek: www.pairwise.org
  - Bug Taxonomy: www.testingeducation.org/a/bsct2.pdf
  - Sample Bug Taxonomy based on Boris Beizer’s work: inet.uni2.dk/~vinter/bugtaxst.doc
  - Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf
  - Heuristic Risk-Based Testing By James Bach
  - Exploring Exploratory Testing, Cem Kaner and Andy Tinkham, www.kaner.com/pdfs/ExploringExploratoryTesting.pdf
  - Pettichord, Bret, “An Exploratory Testing Workshop Report”, www.testingcraft.com/exploratorypettichord
- Chapitre5
  http://www.tmap.net/checklists-and-templates

# Annexe A

Le tableau suivant est dérivé du tableau complet fourni dans ISO 25010. Il se concentre uniquement sur les caractéristiques de qualité couvertes par le syllabus Analyste de Test et compare les termes utilisés dans ISO 9126 (tel qu'elle est utilisée dans la version 2012 du syllabus) avec ceux de la nouvelle ISO 25010 (tel qu'elle est utilisée dans cette version).

| ISO/IEC 25010 | ISO/IEC 9126-1 | Notes |
|---|---|---|
| **Aptitude fonctionnelle** | **Fonctionnalité** | |
| Complétude fonctionnelle | | |
| Exactitude fonctionnelle | Exactitude | |
| Adéquation fonctionnelle | Aptitude | |
| | Interopérabilité | Changé en compatibilité |
| **Utilisabilité** | | |
| Reconnaissance de la pertinence | Facilité de compréhension | Nouveau nom |
| Facilité d'apprentissage | Facilité d'apprentissage | |
| Opérabilité | Opérabilité | |
| Protection contre les erreurs utilisateur | | Nouvelle sous-caractéristique |
| Esthétique de l'interface utilisateur | Attractivité | Nouveau nom |
| Accessibilité | | Nouvelle sous-caractéristique |
| **Compatibilité** | | Nouvelle définition |
| Interopérabilité | | |
| Coexistence | | Couvert dans Analyste Technique de Test |

# Index