Testeur certifié niveau Avancé
Analyste de test
(CTAL-TA)
Syllabus

v4.0

International Software Testing Qualifications Board

Notice de copyright

Historique de révision

Version	Date	Remarques
V4.0 FR	15/08/2025	Traduction française
v4.0	02/05/2025	Mise à jour majeure avec révision globale et mise à jour du périmètre
v3.1.2	31/01/2022	Errata : Correction de défauts mineurs de formatage, de grammaire et de formulation
v3.1.1	15/05/2021	Errata : La notice de copyright a été adaptée aux normes actuelles de l'ISTQB®.
v3.1	03/03/2021	Mise à jour mineure avec réécriture de la section 3.2.3 et diverses améliorations de la formulation
v3.0 (2019)	19/10/2019	Mise à jour majeure avec révision globale et réduction du périmètre.
v2.0 (2012)	19/10/2012	Première version sous la forme d'un syllabus CTAL-TA séparé.

Table des matières

Remerciements

L'Assemblée générale de l'ISTQB® a officiellement publié ce document le 2 mai 2025.

Il a été réalisé par une équipe de l'International Software Testing Qualifications Board: Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (Product Owner), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

L'équipe remercie Julia Sabatine pour sa relecture, Gary Mogyorodi pour sa révision technique, Daniel Pol'an pour son soutien technique constant, ainsi que l'équipe de réviseurs et les Membres de l'ISTQB® pour leurs suggestions et leur contribution.

Les personnes suivantes ont participé à la revue, aux commentaires et au vote de ce syllabus :

Édition actuelle (v4.0): Gergely Ágnecz, Laura Albert, Dani Almog, Somying Atiporntham, Andre Baumann, Lars Bjørstrup, Ralf Bongard, Earl Burba, Filipe Carlos, Renzo Cerquozzi, Kanitta Chantaramanon, Alessandro Collino, Nicola De Rosa, Aneta Derkova, Dingguofu, Ole Chr. Hansen, Zheng Dandan, Karol Frühauf, Dinu Gamage, Chen Geng, Sabine Gschwandtner, Ole Chr. Hansen, Zsolt Hargitai, Tharushi Hettiarachchi, Ágota Horváth, Arnika Hryszko, Caroline Julien, Beata Karpinska, Ramit Manohar Kaul, Mattijs Kemmink, László Kvintovics, Thomas Letzkus, Marek Majerník, Donald Marcotte, Dénes Medzihradszky, Imre Mészáros, Krisztián Miskó, Gary Mogyorodi, Ebbe Munk, Maysinee Nakmanee, Ingvar Nordström, Tal Pe'er, Kunlanan Peetijade, Sahani Pinidiya, Lukáš Piška, Nishan Portoyan, Meile Posthuma, Yasassri Ratnayake, Randall Rice, Adam Roman, Marc Rutz, Jan Sabak, Vimukthi Saranga, Sumuduni Sasikala, Gil Shekel, Radoslaw Smilgin, Péter Sótér , Helder Sousa, Richard Taylor, Benjamin Timmermans, Giancarlo Tomasig, Robert Treffny, Shun Tsunoda, Stephanie Ulrich, François Vaillancourt, Linda Vreeswijk, Carsten Weise, Marc-Florian Wendland, Paul Weymouth, Elżbieta Wiśniewska, John Young, Claude Zhang.

Edition 2021-2022 (v3.1): Gery Ágnecz, Armin Born, Chenyifan, Klaudia Dussa-Zieger, Chen Geng (Kevin), Istvan Gercsák, Richard Green, Ole Chr. Hansen, Zsolt Hargitai, Andreas Hetz, Tobias Horn, Joan Killeen, Attila Kovacs, Rik Marselis, Marton Matyas, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Pe'er, Palma Polyak, Nishan Portoyan, Meile Posthuma, Stuart Reid, Murian Song, Péter Sótér, Lucjan Stapp, Benjamin Timmermans, Chris van Bael, Stephanie van Dijck, Paul Weymouth.

Par la suite, Tal Pe'er, Stuart Reid, Marc-Florian Wendland et Matthias Hamburg ont suggéré des améliorations formelles, grammaticales et rédactionnelles qui ont été implémentées et publiées dans les Errata 3.1.1 et 3.1.2.

Edition 2019 (v3.0): Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, 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, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

Edition 2012 (v2.0): Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina

Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

0 Introduction

0.1 Objectif de ce syllabus

Ce syllabus constitue la base de la qualification internationale de test du logiciel pour l'Analyste de test de niveau Avancé. L'ISTQB® fournit ce syllabus comme suit :

1. Aux membres, pour qu'ils le traduisent dans leur langue locale et pour qu'ils accréditent les organismes de formation. Les membres peuvent adapter le syllabus à leurs besoins linguistiques particuliers et modifier les références pour les adapter à leurs publications locales.
2. Aux organismes de certification, pour qu'ils élaborent des questions d'examen dans leur langue locale, adaptées aux objectifs d'apprentissage de ce syllabus.
3. Aux organismes de formation, pour produire du matériel de formation et déterminer les méthodes d'enseignement appropriées.
4. Aux candidats à la certification, pour préparer l'examen de certification (soit dans le cadre d'une formation, soit de manière autonome).
5. A la communauté internationale de l'ingénierie des logiciels et des systèmes, pour faire progresser la profession de testeur de logiciels et de systèmes et comme base pour des livres et des articles.

0.2 L'Analyste de test de niveau Avancé dans les tests de logiciels

La certification ISTQB® d'Analyste de test de niveau Avancé (CTAL-TA) permet d'acquérir les compétences nécessaires pour tester des logiciels de manière structurée et approfondie tout au long du cycle de vie du développement logiciel. Elle détaille le rôle et les responsabilités de l'Analyste de test à chaque étape d'un processus de test standard et développe les techniques de test importantes. La certification d'Analyste de test de niveau Avancé s'adresse aux personnes titulaires d'une certification de niveau Fondation qui souhaitent développer davantage leur expertise en matière d'analyse de test et de techniques de test.

Dans ce syllabus, l'Analyste de test est compris comme un rôle qui :

- Se concentre davantage sur les besoins métier du client que sur les aspects techniques des tests.
- Effectue principalement des tests fonctionnels, mais contribue également à des tests non fonctionnels axés sur l'utilisateur, tels que des tests d'utilisabilité, d'adaptabilité, de facilité d'installation ou d'interopérabilité
- Utilise des techniques de test boîte noire et des tests basés sur l'expérience plutôt que des techniques de test boîte blanche
- Améliore l'efficacité des tests en utilisant des techniques de prévention des défauts.

0.3 Parcours de carrière des testeurs

Le programme de l'ISTQB® soutient les professionnels du test à tous les stades de leur carrière. Les personnes qui obtiennent la certification ISTQB® Analyste de test niveau Avancé peuvent également être intéressées par les autres niveaux Avancés principaux (Analyste technique de test et Management des tests) et, par la suite, par le niveau Expert (Management des tests ou Amélioration du processus de test). Toute personne cherchant à développer des compétences en matière de pratiques de test dans un

domaine d'environnement Agile pourrait envisager les certifications Testeur Technique Agile ou conduite des tests en mode Agile à l'échelle (ATLaS). En outre, des filières spécialisées proposent des produits de certification axés sur des technologies et des approches de test spécifiques, des caractéristiques de qualité et des niveaux de test spécifiques, ou des tests dans des domaines industriels particuliers.
Veuillez consulter le site www.istqb.org pour obtenir les dernières informations sur le programme de certification des testeurs de l'ISTQB®.

0.4 Objectifs métier

Cette section énumère les objectifs métier attendus d'un candidat ayant obtenu la certification Analyste de test niveau Avancé.
Un Analyste de test de niveau Avancé certifié peut . . .

Code	Description
TA-BO1	Soutenir et effectuer les tests appropriés sur la base du cycle de vie du développement logiciel suivi
TA-BO2	Appliquer les principes du test basé sur les risques
TA-BO3	Sélectionner et appliquer les techniques de test appropriées pour soutenir la réalisation des objectifs de test
TA-BO4	Fournir une documentation présentant un niveau de détail et de qualité approprié
TA-BO5	Déterminer les types de tests fonctionnels appropriés à effectuer
TA-BO6	Contribuer aux tests non fonctionnels
TA-BO7	Contribuer à la prévention des défauts
TA-BO8	Améliorer l'efficience du processus de test grâce à l'utilisation d'outils.
TA-BO9	Spécifier les exigences en matière d'environnements de test et de données de test

0.5 Objectifs d'apprentissage et niveau cognitif des connaissances

Les objectifs d'apprentissage soutiennent les objectifs métier et sont utilisés pour créer les examens de Testeur Certifié Analyste de test de niveau Avancé.

En général, tous les contenus de ce syllabus sont examinables, à l'exception de l'Introduction et des Annexes. Les questions de l'examen confirmeront la connaissance des mots clés au niveau K1 (voir ci-dessous) ou des objectifs d'apprentissage au niveau de connaissance respectif.

Les objectifs d'apprentissage spécifiques et leurs niveaux de connaissance sont indiqués au début de chaque chapitre et classés comme suit :

- K2 : Comprendre

- K3 : Appliquer

- K4 : Analyser

Pour tous les termes listés comme mots-clés juste en dessous des titres de chapitre, le nom et la définition corrects du glossaire de l'ISTQB® doivent être mémorisés (K1), même s'ils ne sont pas explicitement mentionnés dans un objectif d'apprentissage.

De plus amples détails et des exemples d'objectifs d'apprentissage sont donnés dans la section 7.

0.6 L'examen de certification d'Analyste de test de niveau Avancé

L'examen de certification d'Analyste de test de niveau Avancé sera basé sur ce syllabus. Les réponses aux questions de l'examen peuvent exiger l'utilisation de matériel basé sur plus d'une section de ce syllabus. Toutes les sections du syllabus sont examinables, à l'exception de l'Introduction et des Annexes. Les normes et les livres sont inclus comme références, mais leur contenu n'est pas examinable au-delà de ce qui est résumé dans le syllabus lui-même à partir de ces normes et de ces livres.

Pour plus de détails, se référer au document Structures et règles de l'examen compatible avec le niveau Avancé Analyste de test v4.0.

Le critère d'entrée pour suivre le niveau Avancé testeur certifié Analyste de test de l'ISTQB® est que les candidats soient intéressés par les tests de logiciels. Cependant, il est fortement recommandé aux candidats :

- d'avoir au moins une expérience minimale soit dans le développement de logiciels, soit dans les tests de logiciels, par exemple six mois d'expérience en tant que testeur système ou des tests d'acceptation utilisateurs ou en tant que développeur de logiciels.
- de suivre un cours accrédité selon les normes de l'ISTQB® (par l'un des Membres reconnus par l'ISTQB).

Note sur les exigences d'admission : le certificat de niveau Fondation de l'ISTQB® doit être obtenu avant de passer l'examen de certification de niveau Avancé Analyste de test.

0.7 Accréditation

Un Membre de l'ISTQB® peut accréditer les organismes de formation dont le matériel de formation est conforme à ce syllabus. Les organismes de formation doivent obtenir les directives d'accréditation auprès du Membre ou de l'organisme qui effectue l'accréditation. Une formation accréditée est reconnue comme étant conforme à ce syllabus et permet d'inclure un examen de l'ISTQB® à la suite de la formation. Les directives d'accréditation pour ce syllabus suivent les directives générales d'accréditation publiées par le groupe de travail Processes Management and Compliance de l'ISTQB®.

0.8 Gestion des normes

Les organisations internationales de normalisation telles que l'IEEE et l'ISO ont publié des normes associées aux caractéristiques de qualité et aux tests de logiciels. Ces normes sont référencées dans ce syllabus. Le but de ces références est de fournir un framework (comme dans les références à ISO/IEC 25010 concernant les caractéristiques de qualité) ou de fournir une source d'information supplémentaire si le lecteur le souhaite. Veuillez noter que les syllabus de l'ISTQB® utilisent des documents normalisés comme référence. Les normes ne sont pas destinées à l'examen. Reportez-vous à la section 6 - Références pour plus d'informations sur celles-ci.

0.9 Niveau de détail

Le niveau de détail de ce syllabus permet des formations et des examens cohérents à l'échelle internationale. Pour atteindre cet objectif, le syllabus se compose :

- Des objectifs pédagogiques généraux décrivant l'objectif de l'Analyste test de niveau Avancé.
- D'une liste de termes que les étudiants doivent être capables de se rappeler
- Des objectifs d'apprentissage pour chaque domaine de connaissance, décrivant les résultats

d'apprentissage cognitifs à atteindre.

- D'une description des concepts clés, y compris des références à des sources telles que la littérature acceptée ou les normes.

Le contenu du syllabus ne décrit pas l'ensemble des domaines de connaissance des tests de logiciels ; il reflète le niveau de détail à couvrir dans les formations de niveau Avancé Analyste de test.

Le syllabus utilise la terminologie (c'est-à-dire le nom et la signification) des termes utilisés dans les tests de logiciels et l'assurance qualité selon le glossaire de l'ISTQB® (ISTQB-Glossary, 2024).
Pour la terminologie des disciplines connexes, veuillez vous référer aux glossaires respectifs : IREB-CPRE pour l'ingénierie des exigences (IREB-Glossary, 2024), et IEEE-Pascal pour l'ingénierie logicielle (Computer Society et al., 2024).

0.10 Organisation du syllabus

Il y a cinq chapitres dont le contenu peut faire l'objet d'un examen. L'en-tête de chaque chapitre précise le temps alloué au chapitre ; le calendrier n'est pas indiqué au-dessous du niveau du chapitre. Pour les formations accréditées, le syllabus exige un minimum de 20,25 heures de cours (1215 minutes), réparties sur les cinq chapitres comme suit :

- Chapitre 1 (225 minutes) : Les tâches de l'Analyste de test dans le processus de test.
  - Le candidat apprend comment l'Analyste de test est impliqué dans les différents cycles de vie du développement logiciel.
  - Le candidat apprend comment l'Analyste de test est impliqué dans les différentes activités de test.
  - Le candidat apprend les tâches effectuées par l'Analyste de test en relation avec les produits d'activités.
- Chapitre 2 (90 minutes) Les tâches de l'Analyste de test dans le test basé sur les risques
  - Le candidat apprend comment l'analyste de test contribue à l'analyse des risques produit.
  - Le candidat apprend comment analyser l'impact des changements pour déterminer le périmètre des tests de régression.
- Chapitre 3 (615 minutes) Analyse de test et conception des tests
  - Le candidat apprend les techniques de test basées sur les données, telles que les tests de domaine, les tests combinatoires et les tests aléatoires.
  - Le candidat apprend les techniques de test basées sur le comportement, telles que les tests CRUD, les tests de transition d'état et les tests basés sur des scénarios.
  - le candidat apprend les techniques de test basées sur les règles, telles que les tests par tables de décisions et les tests métamorphiques.
  - Le candidat apprend à connaître les tests basés sur l'expérience, tels que les tests basés sur les sessions et le crowd testing.
  - Le candidat apprend à sélectionner les techniques de test appropriées pour atténuer les risques liés au produit.
- Chapitre 4 (60 minutes) Tester les caractéristiques de qualité
  - Le candidat apprend à effectuer plusieurs types de tests fonctionnels.
  - Le candidat apprend à utiliser la connaissance spécifique de la compatibilité pour contribuer aux types de tests non fonctionnels, tels que les tests d'utilisabilité, les tests de flexibilité et les tests de compatibilité.

- Chapitre 5 (225 minutes) Prévention des défauts logiciels
  - Le candidat apprend à connaître les différentes pratiques de prévention des défauts.
  - Le candidat apprend les différentes approches qui soutiennent le confinement de phase.
  - Le candidat apprend comment atténuer la récurrence des défauts.

1 Les tâches de l'Analyste de test dans le processus de test – 225 minutes

Mots-clés

cas de test de haut niveau, mot-clé, test piloté par les mots-clés, cas de test de bas niveau, cycle de vie du développement logiciel, analyse de test, analyste de test, cas de test, condition de test, données de test, conception de test, environnement de test, exécution de test, implémentation de test, oracle de test, script de test, testware.

Objectifs d'apprentissage pour le chapitre 1 :

1.1 Le test dans le cycle de vie du développement logiciel

TA-1.1.1 (K2) *Résumer la participation de l'analyste de test dans différents cycles de vie du développement logiciel*

1.2 Participation aux activités de test

TA-1.2.1 (K2) *Résumer les tâches effectuées par l'analyste de test dans le cadre de l'analyse de test*

TA-1.2.2 (K2) *Résumer les tâches effectuées par l'analyste de test dans le cadre de la conception des tests*

TA-1.2.3 (K2) *Résumer les tâches effectuées par l'analyste de test dans le cadre de l'implémentation des tests*

TA-1.2.4 (K2) *Résumer les tâches effectuées par l'analyste de test dans le cadre de l'exécution des tests*

1.3 Tâches liées aux produits d'activités

TA-1.3.1 (K2) *Distinguer les scénarios de test de haut niveau des scénarios de test de bas niveau*

TA-1.3.2 (K2) *Expliquer les critères de qualité pour les cas de test*

TA-1.3.3 (K2) *Donner des exemples d'exigences relatives à l'environnement de test*

TA-1.3.4 (K2) *Expliquer le problème de l'oracle de test et les solutions possibles*

TA-1.3.5 (K2) *Donner des exemples d'exigences relatives aux données de test*

TA-1.3.6 (K3) *Utiliser des tests basés sur des mots-clés pour développer des scripts de test*

TA-1.3.7 (K2) *Résumer les types d'outils permettant de gérer le testware*

Introduction aux tâches de l'Analyste de test dans le processus de test

Le syllabus du niveau Fondation (ISTQB-CTFL, v4.0.1) décrit deux rôles principaux dans les tests : un rôle de management des tests et un rôle de test. Dans ce syllabus, la personne dans un rôle de test qui est responsable de tester les aspects métier du logiciel est appelée un Analyste de test (AT). Bien que cette responsabilité soit rarement attribuée à un poste ou à un rôle dédié, l'AT a des tâches clairement définies et des compétences requises. En termes de niveaux de test, l'AT se concentre sur les tests système, les tests d'acceptation et les tests d'intégration système. En termes de caractéristiques de qualité, les compétences de l'AT se concentrent sur l'aptitude fonctionnelle et couvrent également certaines caractéristiques de qualité non fonctionnelles orientées vers l'utilisateur, telles que l'utilisabilité, l'adaptabilité, la facilité d'installation et l'interopérabilité.

1.1 Le test dans le cycle de vie du développement logiciel.

1.1.1 Implication de l'Analyste de test dans différents cycles du développement logiciel

L'organisation des activités de test peut varier en fonction du cycle de vie du développement logiciel (SDLC) suivi. Par conséquent, l'implication de l'AT dans les activités de test peut varier en fonction du modèle de SDLC adopté.

Dans les **modèles de développement séquentiel**, les activités de développement se déroulent par phases et commencent lorsque la phase précédente est achevée. En général, il y a peu de chevauchement entre ces activités. Par conséquent, les tâches de l'AT changent généralement au fil du temps. Dans les premières phases du SDLC, l'AT soutient la planification des tests. L'AT commence l'analyse de test lorsque la base de test est en cours de production. La conception des tests et leur implémentation suivent parallèlement à la conception et à l'implémentation du logiciel. Enfin, l'AT exécute les tests et soutient la clôture des tests au cours des dernières phases du SDLC.

Les **modèles de développement incrémental** divisent le logiciel en incréments plus petits et plus faciles à gérer. Chaque incrément est développé et testé indépendamment. Par conséquent, l'AT effectue les mêmes activités pour chaque incrément (analyse de test, conception des tests, implémentation des tests, exécution des tests et soutien au management des tests). Toutefois, le travail de l'AT peut être organisé différemment pour chaque incrément. Les tests se concentrent sur les caractéristiques nouvelles ou modifiées. En outre, en raison du risque accru de régression, l'AT doit accorder une attention particulière au refactoring et à l'assemblage des suites de tests de régression.

Dans les **modèles de développement itératif**, le processus de développement est cyclique. Le projet est soumis à des cycles répétés de prototypage, de test, de remaniement et de déploiement. Le rôle de l'AT est dynamique et adaptable. L'AT collabore étroitement avec les développeurs et les représentants des métiers, en s'adaptant à l'évolution du produit. L'AT adapte et modifie les conditions de test et les cas de test au fur et à mesure de l'évolution du logiciel et fournit un retour d'information afin d'améliorer le processus de test à chaque itération. Plus les itérations sont fréquentes, plus la maintenance et le développement continus des tests de régression par l'AT sont essentiels.

Un SDLC peut combiner des éléments de divers modèles et des techniques et approches spécifiques (par exemple, le développement logiciel en mode Agile combine des aspects des modèles itératif et incrémental). Dans ce cas, la participation de l'AT dépendra des caractéristiques spécifiques du SDLC et de la manière dont elles sont combinées. Une bonne pratique commune à tous les modèles de SDLC consiste à impliquer l'AT dès les phases initiales du SDLC.

1.2 Implication dans les activités de test

Le syllabus du niveau Fondation (ISTQB-CTFL, v4.0.1) décrit sept activités au sein du processus de test. L'AT se concentre principalement sur quatre d'entre elles : l'analyse de test, la conception des tests, l'implémentation des tests et l'exécution des tests.

Les tâches de l'AT dans ces quatre activités sont décrites en détail dans les sections suivantes.

1.2.1 Analyse de test

Pendant l'analyse de test, l'AT vérifie la complétude de la base de test et collecte toute information supplémentaire pertinente pour tester. Cela inclut non seulement la documentation mais aussi des informations verbales, par exemple des conversations lors de la rédaction collaborative de User Story (voir ISTQB-CTFL (v4.0.1), Section 4.5.1). Les modifications de la base de test peuvent conduire à des ajustements du périmètre de test en coordination avec le management des tests.

Pour procéder efficacement à l'analyse de test, l'AT vérifie les critères d'entrée suivants :

- La planification des tests a été effectuée, et le périmètre de test, les objectifs de test et l'approche de test sont clairs.
- La base de test (contenant des informations telles que les exigences ou les User Story) est définie.
- Les risques produits déjà identifiés ont été évalués et documentés si nécessaire.

L'AT évalue la base de test afin d'identifier les éventuels défauts qu'elle contient et d'en évaluer la testabilité, fournissant ainsi un retour d'information précoce aux Product Owners. Cela peut inclure la modélisation du comportement du système en fonction des techniques de test à appliquer (voir section 3, section 5.2.1). Des techniques de revue sont également appliquées dans le cadre du processus (voir section 5.2.2). S'ils ne sont pas directement corrigés, les défauts concernant la base de test doivent être documentés. En outre, l'AT détermine les oracles de test nécessaires (voir section 1.3.4).

L'AT définit et hiérarchise les conditions de test pour chaque élément de test dans le périmètre. Les conditions de test répondent aux objectifs de test (voir ISTQB-CTFL (v4.0.1), Section 1.1.1) et doivent être traçables aux éléments de la base de test. Le périmètre et l'objectif des conditions de test prennent en compte les risques produits. Dans les modèles de développement incrémental ou itératif, il s'agit notamment de déterminer le périmètre des tests de régression sur la base d'une analyse d'impact. Dans le développement logiciel Agile, les conditions de test peuvent être exprimées sous forme de critères d'acceptation qui reflètent les risques des User Stories.

L'AT peut procéder par étapes, en commençant par des conditions de test de haut niveau telles que : « fonctionnalité de l'écran x ». Ensuite, l'AT définit des conditions de test plus détaillées telles que : « L'écran x rejette les numéros de compte dont le numéro est trop court d'un chiffre ».
Cette approche soutient une couverture suffisante et permet de commencer rapidement la conception des tests, par exemple pour les User Stories qui doivent encore être remaniées.

L'AT implique les parties prenantes dans la révision des conditions de test afin de s'assurer que la base de test est clairement comprise et que les tests sont alignés sur les objectifs de test.

1.2.2 Conception des tests.

La conception des tests décrit comment effectuer les tests pour atteindre les objectifs de test énoncés. Cela se fait généralement à l'aide de cas de test. La manière dont la conception des tests est réalisée dépend de nombreux facteurs, notamment la couverture requise, la base de test, le SDLC, les contraintes du projet, ainsi que les connaissances et l'expérience des testeurs.

Pendant la conception des tests, l'AT détermine dans quels domaines les scénarios de test de bas niveau ou les scénarios de test de haut niveau sont appropriés (voir section 1.3.1). Dans les deux cas, l'AT doit

définir des critères de réussite/échec clairs. L'AT conçoit les cas de test pour les conditions de test nouvelles ou modifiées en fonction des critères de qualité (voir section 1.3.2). Pour les tests de régression, la sélection de scénarios de test haut niveau existants ou l'adaptation de scénarios de test bas niveau existants en fonction de leur priorité est généralement suffisante.

L'AT capture la traçabilité entre la base de test, les conditions de test et les cas de test. Dans les tests basés sur l'expérience, les cas de test ne sont pas toujours documentés ; à la place, les conditions de test (entre autres) peuvent guider l'exécution du test. L'AT peut concevoir certains cas de test sur la base d'objectifs de test haut niveau.

En plus de ces tâches, l'AT définit les exigences de l'environnement de test (voir section 1.3.3) et identifie, crée et spécifie les exigences pour les données de test (voir section 1.3.5).
L'AT utilise les critères de sortie définis dans la planification des tests pour déterminer quand suffisamment de cas tests ont été conçus. Cependant, d'autres critères de sortie tels que les niveaux de risque résiduels ou les contraintes du projet (par exemple, le budget ou le temps) indiquent également quand la conception des tests peut se terminer.
La conception des tests peut être soutenue par des outils mais devrait être agnostique en termes d'outils et de technologies pour rester indépendante de ces derniers. La conception des tests applique une approche systématique utilisant des techniques de test ou est ad-hoc dans le cas contraire.
Les cas de test jouent un rôle de communication et doivent être compréhensibles par les parties prenantes concernées. Comme un cas de test n'est pas toujours exécuté par son auteur, les autres testeurs doivent comprendre comment l'exécuter, ses objectifs de test (c'est-à-dire les conditions de test sous-jacentes) et son importance. Les cas de test doivent également être compréhensibles par les développeurs qui peuvent implémenter les tests ou les réexécuter en cas de défaillance et par les auditeurs qui peuvent avoir à les approuver.

1.2.3 Implémentation des tests

Lors de l'implémentation des tests, l'AT fournit le testware nécessaire à l'exécution des tests. L'AT peut organiser les procédures de test et les scripts de test en suites de tests ou suggérer des cas de test pour l'automatisation. Définir les procédures de test nécessite d'identifier soigneusement les contraintes et les dépendances qui peuvent influencer l'ordre d'exécution des tests. Outre les étapes contenues dans les cas de test, les procédures de test comprennent des étapes pour la mise en place de toutes les préconditions initiales (par exemple, le chargement des données de test à partir d'un référentiel de données), la vérification des résultats attendus et des postconditions, et la réinitialisation des étapes après l'exécution (par exemple, la réinitialisation de la base de données, de l'environnement et du système).

L'AT hiérarchise les procédures de test et les scripts de test pour l'exécution du test, en se basant sur les critères de hiérarchisation identifiés lors de l'analyse des risques et de la planification des tests. Il identifie les procédures de test ou les scripts de test qui doivent être exécutés sur la version actuelle de l'objet de test. Cela permet d'exécuter ensemble des tests connexes (par exemple, pour de nouvelles caractéristiques ou des tests de régression) dans un run de test spécifique. L'AT met à jour la traçabilité entre la base de test et d'autres testware tels que les procédures de test, les scripts de test et les suites de tests.

L'AT peut aider le Test Manager (TM) à définir un calendrier d'exécution des tests, y compris l'allocation des ressources, pour permettre une exécution efficiente des tests en définissant l'ordre d'exécution des tests (voir ISTQB-CTFL, v4.0.1, section 5.1.5).

L'AT crée des données d'entrée et d'environnement à charger dans les bases de données et autres référentiels (voir section 1.3.5). Ces données doivent être « adaptées à l'objectif » pour soutenir les objectifs spécifiques du test.

L'AT doit également vérifier que l'environnement de test est entièrement mis en place et prêt pour l'exécution du test (voir section 1.3.3). La meilleure façon de procéder consiste à concevoir et à exécuter un smoke test. L'environnement de test doit révéler les défauts de l'objet de test grâce à l'exécution du test,

fonctionner normalement en l'absence de défaillances et reproduire de manière adéquate, si nécessaire, l'environnement de production ou l'environnement de l'utilisateur final.

Le niveau de détail et la complexité associée du travail effectué pendant l'implémentation des tests peuvent être influencés par le niveau de détail des conditions de test et des cas de test. Dans certains cas, des dispositions réglementaires s'appliquent, et le testware doit apporter la preuve de la conformité avec les normes applicables (par exemple, RTCA DO- 178C, 2011).

1.2.4 Exécution des tests

L'exécution des tests est réalisée conformément au calendrier d'exécution des tests. Les tâches typiques de l'AT sont l'exécution des tests, la comparaison des résultats réels avec les résultats attendus, l'analyse des anomalies, le rapport des défauts et l'enregistrement des résultats de tests.

L'AT exécute les tests manuellement. Cela comprend les tests exploratoires, l'exécution des procédures de test, les tests de régression et les tests de confirmation. Pour les tests exploratoires, l'AT peut utiliser des tests basés sur des sessions avec chartes de test (voir section 3.4.1). L'AT peut également exécuter des scripts de tests automatisés, mais il peut incomber aux développeurs, aux ingénieurs en automatisation des tests (TAE) ou aux analystes techniques de tests (AT) d'exécuter des scripts de tests automatisés et de les analyser en cas de défaillance.

L'AT analyse les anomalies qui se produisent dans les exécutions de tests manuels ou automatisés afin d'en établir les causes probables. Une anomalie peut être la conséquence d'un défaut dans un objet de test. Cependant, d'autres raisons peuvent également exister, notamment des préconditions manquantes, des données de test incorrectes, des défauts dans les scripts de test ou dans l'environnement de test, ou une mauvaise compréhension des spécifications. L'AT consigne les résultats réels de l'exécution du test, communique les défauts sur la base des défaillances observées et établit un rapport à ce sujet si nécessaire.

L'AT met à jour la traçabilité entre la base de test et d'autres testware en tenant compte des résultats des tests. Cette information permet de transformer les résultats des tests en informations de haut niveau sur le risque ou la couverture, ce qui permet aux parties prenantes de prendre des décisions éclairées. Par exemple, cela permet de clarifier pour les parties prenantes combien de cas de test liés à une condition de test ont passé ou échoué.
En plus de ces tâches typiques, l'AT évalue les résultats des tests, ce qui comprend les tâches suivantes :

- Reconnaissance des regroupements de défauts, qui peuvent indiquer la nécessité de tester davantage une partie particulière de l'objet de test (voir section 5.3.1).
- Réexécution manuelle des tests automatisés qui ont échoué pour s'assurer que l'automatisation des tests n'a pas produit un résultat faux positif.
- Proposition de tests supplémentaires sur la base de ce qui a été appris au cours des tests précédents.
- Identification de nouveaux risques à partir des informations obtenues lors de l'exécution des tests.
- Proposition d'améliorations à la conception des tests ou à l'implémentation des tests (par exemple, des améliorations aux procédures de test) ou même au système sous test.
- Proposition d'améliorations aux suites de tests de régression, y compris le refactoring, les ajustements de périmètre et l'automatisation des tests (voir section 2.2.1).

1.3 Tâches liées aux produits d'activités

L'AT doit veiller à la qualité des produits d'activités dont il a la responsabilité. Cela inclut les cas de test, les environnements de test, les données de test, les oracles de test et les scripts de test. Dans cette section, le syllabus aborde les tâches de l'AT liées aux testware et les types d'outils permettant de gérer les testware.

1.3.1 Cas de test de haut niveau et cas de test de bas niveau

Un cas de test de haut niveau (également appelé cas de test abstrait ou cas de test logique) décrit les conditions dans lesquelles l'objet de test est examiné en indiquant les conditions de test couvertes par le cas de test. Les cas de test haut niveau permettent donc de s'assurer que les tests couvrent toutes les conditions de test pertinentes. Les cas de test de haut niveau ne contiennent pas d'informations concrètes sur les préconditions, les données d'entrée, les résultats attendus ou les postconditions. Celles-ci sont toutes exprimées à un niveau abstrait (par exemple, « commander plus d'un livre, le prix de la commande donnant lieu à une remise ; résultat attendu : la remise est attribuée »).

Un cas de test de bas niveau (également appelé cas de test concret ou cas de test physique) est le remaniement détaillé d'un cas de test de haut niveau. Les cas de test de bas niveau décrivent les données qui doivent être préparées, les actions que le testeur doit effectuer (si nécessaire) et le résultat attendu concret. Les cas de test de bas niveau contiennent des préconditions spécifiques, des données d'entrée, des résultats attendus et des postconditions (par exemple, « commander les livres B1 (10 €) et B2 (20 €), le prix total de la commande étant de 30 € ; résultat attendu : remise de 10 %, prix total : 27 € »).

En règle générale, un AT conçoit d'abord des cas de test de haut niveau, qui servent de base au développement de cas de test de bas niveau. Un cas de test haut niveau peut être implémenté dans un ou plusieurs cas de test bas niveau. Parfois, le cas de test peut rester de haut niveau, les informations concrètes étant déterminées par l'AT pendant l'exécution du test.

Par exemple, les cas de test de haut niveau peuvent guider l'AT lors de la création des objectifs de test dans une charte de test pour les tests basés sur des sessions, permettant aux AT d'élaborer sur ces objectifs pendant l'exécution du test (voir section 3.4.1).

Le passage des scénarios de test de haut niveau aux scénarios de test de bas niveau est plus qu'un simple remplissage de valeurs concrètes. Il s'agit également d'une étape entre le concept et la technique. Il est souvent différé de la conception des tests à l'implémentation des tests, en particulier si des données de tests spécifiques sont nécessaires. L'AT doit s'assurer que tout ce qui est nécessaire à l'exécution des cas de test bas niveau est connu (Koomen et al., 2006). Dans la pratique, de nombreux cas de test sont hybrides, concrets sous certains aspects et abstraits sous d'autres. Cela est souvent dû à un compromis entre la maintenabilité et la compréhensibilité des cas de test.

1.3.2 Critères de qualité pour les cas de test.

Négliger la qualité des cas de test peut entraîner de nombreux problèmes, tels que des coûts de maintenabilité élevés, une compréhension réduite ou des retards d'exécution. Les critères de qualité pour les cas de test sont un premier pas vers des cas de test plus maintenables. Ils comprennent:

- **Exactitude**. Un cas de test doit faciliter la vérification précise des conditions de test sur lesquelles il est basé.
- **Faisabilité**. Un cas de test doit pouvoir être exécuté.
- **Nécessité**. Chaque cas de test doit couvrir un objectif de test clair, tel qu'exprimé dans son titre ou son résumé. Les doublons doivent être évités. Les cas de test ne doivent pas être conçus pour les choses qui ne doivent pas être testées.

- **Facilité de compréhension**. Les cas de test peuvent être revus, modifiés et exécutés par des personnes autres que l'auteur. L'AT doit rédiger les cas de test dans un langage et un format compréhensibles par toutes les parties prenantes concernées, sans expliquer ce qui est évident. Les cas de test complexes doivent être simplifiés ou scindés.
- **Traçabilité**. Les cas de test doivent être traçables aux conditions de test, aux exigences et aux risques pour permettre à l'AT de les tenir à jour (voir ISTQB-CTFL (v4.0.1), Section 1.4.4).
- **Cohérence**. La cohérence du langage, de la mise en forme et de la structure facilite la compréhension et la maintenance des cas de test. L'AT peut utiliser un glossaire à cette fin.
- **Précision**. Il ne doit y avoir qu'une seule interprétation d'un cas de test afin d'éviter des résultats de test de type faux négatifs et faux positifs. Les termes ambigus tels que « approprié », « selon les besoins » ou « plusieurs » doivent être évités.
- **Complétude**. Tous les attributs nécessaires (par exemple, voir ISO/IEC/IEEE 29119-3, 2021) doivent être présents, y compris les données de test requises (voir section 1.3.5) et un résultat attendu clair afin d'éviter tout doute lorsqu'on le compare avec le résultat réel.
- **Concision**. La granularité des cas de test (c'est-à-dire un grand cas de test avec de nombreuses actions de test par opposition à plusieurs cas de test plus petits) doit correspondre à la base de test et aux conditions de test. Il est préférable d'utiliser des cas de test plus petits, axés sur quelques éléments de couverture, car ils facilitent la détection des causes des défaillances, peuvent être combinés avec souplesse dans des procédures de test et des suites de tests, et une défaillance lors de l'exécution du test ne bloquerait pas les tests ultérieurs.

Le format et le niveau de détail des cas de test dépendent du projet et du contexte du produit et doivent être convenus au sein de l'équipe de test.

1.3.3 Exigences relatives à l'environnement de test

L'environnement de test est un facteur critique de succès pour l'exécution des tests, qu'ils soient manuels ou automatisés. L'implémentation de l'environnement de test a un impact sur la testabilité, la détection des défauts, les coûts globaux des tests et la fiabilité des résultats des tests.

Un cas de test qui passe ou échoue dans l'environnement de test doit avoir le même résultat de test lorsqu'il est exécuté en production. Idéalement, un environnement de test est robuste, prévisible et intégré au framework d'automatisation des tests, si nécessaire. L'AT peut définir les exigences de l'environnement de test au cours de la conception des tests sur la base de l'analyse :

- des conditions de test, des cas de test et des exigences en matière de données de test, de sorte que les exigences en matière d'environnement de test décrivent les conditions nécessaires à la mise en place et à la maintenabilité de l'environnement de test pour garantir que les préconditions de l'exécution du test sont remplies.
- des niveaux de test et des types de test, qui influencent le compromis entre la flexibilité de l'environnement de test et la similarité avec l'environnement de production
- de la disponibilité et de l'indépendance des composants et des systèmes, qui peuvent indiquer la nécessité d'utiliser des doubles des tests (par exemple, des bouchons ou des pilotes)

Les exigences relatives à l'environnement de test décrivent les éléments de l'environnement de test, qui peuvent être classés en plusieurs types, tels que le matériel, le middleware, les logiciels, les services virtualisés, le réseau, les interfaces, les outils, la sécurité, la configuration et le lieu. Pour chaque élément de l'environnement de test, les exigences doivent comprendre les informations suivantes (ISO/IEC/IEEE 29119-3, 2021) :

- Identifiant unique - utilisé à des fins de traçabilité.
- Description - suffisamment détaillée pour l'implémenter comme requis
- Responsabilité - décrit qui est responsable de sa disponibilité
- Période nécessaire - identifie quand et pour combien de temps l'élément est nécessaire
- Fidélité : degré de représentativité ou d'écart de cet élément par rapport à l'environnement de production.

Les exigences doivent également porter sur les besoins généraux de l'environnement de test, notamment la configuration, la sauvegarde et la restauration, les besoins en matière de sécurité, la capacité à modifier l'environnement de test, ainsi que les rôles et les autorisations (Koomen et al., 2006).

L'AT doit documenter les exigences de l'environnement de test de manière claire, compacte et cohérente. L'AT peut utiliser des diagrammes ou des tableaux. Pour éviter les redondances et la documentation inutile, l'AT peut faire référence à des environnements de test existants et se concentrer sur les besoins spécifiques du niveau de test. Les parties prenantes concernées (par exemple, les développeurs, les AT, les ingénieurs en automatisation des tests, les analystes métier, les sponsors et les Product Owners) devraient également revoir, approuver et mettre à jour les exigences de l'environnement de test.

1.3.4 Déterminer les oracles de test

Un oracle de test est nécessaire pour déterminer les résultats attendus lors des tests dynamiques. Idéalement, la base de test fournira les oracles de test (par exemple, dans une spécification textuelle ou formelle). L'expérience humaine ou la connaissance de l'objet de test peuvent également servir d'oracle de test. Un oracle de test automatisé peut être exigé pour des raisons d'efficacité (par exemple, si les données d'entrée sont générées automatiquement ou si les oracles humains sont coûteux).

En fonction de la qualité et de la complétude de la base de test ou des caractéristiques du système, un oracle de test rentable peut ne pas être disponible. C'est ce qu'on appelle le *problème de l'oracle de test*. Les facteurs typiques contribuant au problème de l'oracle de test sont la complexité des données, le non-déterminisme (par exemple, les systèmes basés sur l'IA), le comportement probabiliste et les exigences manquantes ou ambiguës.

Quelques solutions connues au problème de l'oracle de test comprennent (Barr et al., 2014) :

- Les pseudo-oracles qui sont des systèmes développés indépendamment qui remplissent la même spécification que l'objet de test (par exemple, les systèmes patrimoniaux, les versions simplifiées des objets du test). Développer un pseudo-oracle dédié pour un objet de test complexe peut être coûteux mais est typique pour les systèmes critiques.
- Les tests basés sur des modèles qui peuvent formaliser l'oracle de test en tant que partie du modèle de test. Ils permettent de générer des sorties attendues et de dériver des tests à partir du modèle. Les techniques de test basées sur le comportement impliquent souvent l'implémentation d'un oracle de test automatisé (voir section 3.2.2 et section 3.2.3).
- Les tests basés sur les propriétés qui utilisent des propriétés spécifiées de l'objet de test pour vérifier les relations entre l'entrée et le résultat attendu des cas de test individuels. Si une telle relation n'est pas respectée, le cas de test échoue. Cette solution est bien adaptée à l'automatisation des tests, mais son efficacité dépend des relations, qui peuvent être difficiles à déterminer.
- Les tests métamorphiques (voir section 3.3.2).

- Les oracles humains qui utilisent la capacité de l'homme à déterminer les résultats attendus. Les ressources humaines peuvent être coûteuses et rares. Toutefois, les oracles humains sont privilégiés dans certaines approches de test (par exemple, les tests exploratoires).

Les assertions peuvent être intégrées dans le code d'automatisation des tests ou dans l'objet de test lui-même pour implémenter un oracle de test automatisé. Il s'agit d'instructions exécutables qui vérifient l'état ou le comportement de l'objet de test. Lorsqu'elles sont intégrées dans l'objet de test, elles ne vérifient généralement que ce qui est nécessaire à la poursuite de la tâche.

1.3.5 Exigences en matière de données de test

Lors de la conception des tests, l'AT identifie et demande les données de test qui peuvent nécessiter une préparation ou un approvisionnement en vue de l'exécution du test. Les considérations comprennent l'objectif, le format et le contexte d'utilisation. (La norme ISO/IEC/IEEE 29119-3, 2021, section 8.5, décrit également les exigences en matière de données de test. Les aspects clés sont les suivants :

- **Similitude avec les données de production**. Les données de production reflètent les données du monde réel mais peuvent manquer de variabilité. Les données synthétiques permettent une variabilité contrôlée. Elles devraient refléter les principaux aspects des canevas, des distributions et des valeurs aberrantes des données de production, mais peuvent négliger certains défauts qui ne se produisent qu'avec des données réelles. Pour les systèmes qui ne disposent pas de données de production, les données synthétiques doivent refléter des scénarios métier et techniques réalistes. Les personae aident en fournissant des profils réalistes, centrés sur l'utilisateur, qui guident la création de données reflétant divers scénarios et comportements d'utilisateurs.
- **Confidentialité**. Les données de test sensibles (par exemple, les informations personnelles) nécessitent une protection. Les données pseudonymisées remplacent les informations personnelles par des identifiants artificiels. Les données anonymisées suppriment les informations permettant d'identifier les personnes. Si nécessaire, l'AT doit respecter les réglementations relatives à la protection des données telles que le RGPD dans l'Union européenne (Commission, 2016) ou l'HIPAA aux États-Unis (Health et al., 2024).
- **Objectif**. Les données de test sont cruciales pour déterminer les préconditions et les résultats attendus qui ont un impact sur l'état du système et sa configuration (par exemple, l'heure et la date du système). Elles permettent également de spécifier les autorisations des utilisateurs et d'établir des relations entre les produits, les départements et les catégories.
- **Critères de couverture**. Les données de test doivent correspondre aux critères de couverture de la technique de test choisie. Outre des données de test valides, cela peut également nécessiter des données de test non valides, par exemple pour les tests négatifs.
- **Format des données**. La gestion systématique des données (par exemple, dans les tests API) peut nécessiter des données structurées (par exemple, CSV, JSON, XML ou base de données).
- **Traçabilité**. La traçabilité assure la maintenabilité des données de test lorsque des modifications sont apportées aux cas de test.
- **Maintenabilité**. Les données de test codées en dur dans les scénarios de test de bas niveau doivent être évitées pour faciliter la détection des défauts et la maintenance. Les cas de test doivent séparer la logique de test des données de test (voir section 1.3.2).
- **Dépendances**. Les données dépendantes nécessitent une série d'étapes pour leur création.
- **Disponibilité**. La virtualisation des services peut remédier aux données manquantes en simulant des services absents ou inaccessibles pour interagir avec des systèmes ou des services externes.
- **Sensibilité au temps et vieillissement des données**. Les cas de test impliquant des données obsolètes ou sensibles au temps peuvent avoir un impact sur le comportement du système de manière inattendue ou inexacte.

1.3.6 Développer des scripts de test en utilisant des tests basés sur des mots-clés

Si des tests basés sur des mots-clés sont utilisés, l'AT crée des scripts de test à l'aide de mots-clés. L'implémentation des scripts de test incombe à l'Analyste Technique de test (ATT), à l'ingénieur en Automatisation de test (TAE), ou à un développeur.

L'AT identifie et spécifie les mots-clés en analysant la base de test ou en collaborant avec les parties prenantes. Les mots-clés peuvent être classés en deux catégories : action et vérification. Les mots-clés d'action doivent interagir avec l'objet de test (par exemple, exécuter des fonctions, soumettre des données, naviguer à l'intérieur de l'objet de test), l'environnement de test (par exemple, établir des configurations et activer des simulateurs), ou d'autres composants ou systèmes (par exemple, déclencher une interface de l'objet de test). Les mots-clés de vérification représentent des assertions permettant d'évaluer si le résultat réel produit par l'objet de test correspond au résultat attendu.

Les mots-clés résident dans au moins deux couches d'abstraction : la couche du domaine et la couche de l'interface de test. Les mots-clés de la couche du domaine correspondent à des actions liées au métier et reflètent la terminologie du domaine d'application. Ils font abstraction des détails techniques de l'interface de l'objet de test. Les mots-clés de la couche de l'interface de test communiquent avec les objets du test, les éléments de l'environnement de test ou d'autres composants ou systèmes via leur interface de test. Ils résident dans la couche d'abstraction la plus basse. Des couches intermédiaires supplémentaires peuvent soutenir la maintenabilité des mots-clés.

Les mots-clés peuvent être atomiques ou composés d'autres mots-clés. La structure et la couche d'abstraction d'un mot-clé sont des attributs indépendants. Cependant, les mots-clés composés résident souvent à un niveau d'abstraction plus élevé, tandis que les mots-clés atomiques ont tendance à résider sur la couche d'interface de test.

Les tâches de l'AT dans les tests pilotés sur les mots-clés incluent :

- la spécification des mots-clés et de leurs paramètres
- la spécification des cas de test par mots-clés, c'est-à-dire des scripts de test utilisant des mots-clés
- la spécification des étapes supplémentaires des scripts de test à l'aide de mots-clés, telles que les préconditions, les actions de vérification et le nettoyage de l'environnement de test après le test
- la maintenance des cas de test par mots-clés pour refléter les changements apportés à l'objet de test
- l'exécution des scripts de test par mots-clés, que ce soit de manière automatisée ou manuelle.
- l'analyse des cas de test par mots-clés ayant échoué afin de déterminer la cause de la défaillance.

Lors de l'analyse de la base du test, l'AT recherche les interactions entre l'objet de test et son environnement (par exemple, les utilisateurs, les autres systèmes et les appareils). Considérons, par exemple, la User Story « En tant que membre, je veux m'authentifier pour avoir accès aux installations », avec le critère d'acceptation « Des cartes de membre valides peuvent être utilisées pour l'authentification ». L'AT peut spécifier un mot-clé (d'action) « Authentification du membre » avec un paramètre « carte de membre » et un mot-clé de vérification « Vérification de l'accès ». L'AT vérifie si les mots-clés identifiés résident dans la couche d'abstraction appropriée.

Lors de la spécification des mots-clés, l'AT doit garder à l'esprit que les mots-clés doivent :

- contenir un verbe (+ un nom)
- utiliser la forme impérative du verbe (+ nom)
- être uniques dans leur signification
- être documentés de manière adéquate
- refléter le vocabulaire du domaine d'application
- être réutilisables

Lors de la composition des cas de test par mots-clés, l'AT peut reconnaître certains mots-clés manquants et les spécifier immédiatement. Les cas de test par mots-clés peuvent être rédigés sous différents formats, tels que des listes ou des tableaux.

Les mots-clés peuvent changer au cours de la durée de vie d'un projet et sont susceptibles d'être spécifiés de manière redondante (Rwemalika et al., 2019). Les recommandations mentionnées dans cette section visent à éviter la redondance et à réduire les efforts de maintenance.

Bien que les tests basés sur des mots-clés soient une approche d'automatisation des tests, les tests manuels peuvent également en bénéficier. S'ils sont appliqués pour les tests manuels, ils soutiennent efficacement une transition ultérieure des tests manuels vers les tests automatisés.

Plus de détails sur l'automatisation de l'exécution des tests et les tests pilotés par mots-clés sont disponibles dans (ISTQB-TTA, v4.0), (ISTQB-TAE, v2.0), et (ISO/IEC/IEEE 29119-5, 2016).

1.3.7 Outils utilisés pour la gestion du testware.

Une bonne gestion des tests à l'aide d'outils peut aider l'AT à soutenir le processus de test dans son ensemble. Les outils peuvent fournir l'état des produits d'activités et soutenir le pilotage des tests et le contrôle des tests. En cas de défaillances du système sous test, ils permettent à l'AT de vérifier les résultats des exécutions de test précédentes et d'analyser le moment où les défauts se sont produits. Voici quelques types d'outils clés pour le management du testware :

- Outils de gestion des tests pour fournir un référentiel de tous les éléments du testware pertinents (par exemple, les conditions de test, les cas de test, les scripts de test, les suites de test et les runs de test). Ces outils facilitent la traçabilité (par exemple, au moyen d'une matrice de traçabilité), la récupération des cas de test, le calendrier des exécutions de test, l'enregistrement des résultats des tests et le reporting global de l'avancement et de la qualité des tests.
- Outils de gestion des défauts pour consigner, classer par ordre de priorité et suivre le processus de résolution des défauts.
- Outils de gestion des données de test pour créer et maintenir les données de test, y compris la protection des données sensibles (voir section 1.3.5).
- Outils de gestion des configurations pour faciliter les activités liées aux tests dans le cadre des processus de développement, de gestion de version et d'exploitation, y compris la gestion de la configuration et de la disponibilité des environnements de test.
- Outils de gestion des exigences pour contenir et suivre les exigences de haut niveau, en veillant à ce qu'elles soient clairement définies, contrôlées par version et tracées tout au long du cycle de développement du logiciel.

L'AT soutient le management du testware en :

- analysant le projet et la version pour sélectionner le sous-ensemble correct du testware (par exemple, une spécification identifiée par sa version pour correspondre à la version du SUT)
- définissant une structure fonctionnelle (par exemple, par caractéristiques ou modules) ou technique (par exemple, par type de test ou environnement) appropriée pour l'organisation des

cas de test dans l'outil de management des tests

- ajoutant des métadonnées aux cas de test (par exemple, des informations sur l'effort lié à l'exécution du test ou sur l'environnement de test spécifique requis).
- assurant la traçabilité entre les exigences, les conditions de test, les tests, les runs de test et les défauts
- sélectionnant des suites de tests correctes pour les tests de régression (pour l'exécution manuelle/automatisée des tests)
- veillant à la gestion de la configuration des cas de test, y compris l'identification des cas de test obsolètes.

Les outils permettent d'organiser, de suivre et d'assurer la qualité du processus de test. L'AT soutient cet effort en sélectionnant, structurant et maintenant les cas de test, en assurant la traçabilité et en gérant les versions des cas de test pour des tests et un contrôle des tests efficaces.

2 Les tâches de l'Analyste de test dans le test basés sur les risques – 90 minutes

Mots-clés

analyse d'impact, risque produit, test de régression, analyse des risques, évaluation des risques, contrôle des risques, identification des risques, atténuation des risques, suivi des risques, test basé sur les risques.

Objectifs d'apprentissage pour le chapitre 2 :

2.1 Analyse des risques

TA-2.1.1 (K2) *Résumer la contribution de l'analyste de test à l'analyse des risques produit*

2.1 Contrôle des risques

TA-2.2.1 (K4) *Analyser l'impact des changements pour déterminer le périmètre des tests de régression*

Introduction aux tâches de l'Analyste de test dans le test basé sur les risques.

Le test basé sur les risques est une approche de test qui priorise les efforts de test sur la base des niveaux de risque des éléments du test. Le Test Manager détermine cette approche. L'AT joue un rôle actif en l'implémentant. Le test basé sur les risques est plus amplement décrit dans les documents suivants (ISTQB-TM, v3.0).

2.1 Analyse des risques

Selon (ISTQB-CTFL, v4.0.1, Section 5.2), l'analyse des risques implique l'identification des risques et l'évaluation des risques. Ce syllabus se concentre sur la façon dont l'AT devrait participer aux activités d'analyse des risques pour s'assurer que le test basé sur les risques sont implémentés correctement.

2.1.1 La contribution de l'Analyste de test à l'analyse des risques produit

L'AT possède souvent une connaissance unique du système, ainsi qu'une expérience et une connaissance intuitive de ce qui se passe généralement de manière incorrecte, de l'impact que cela a et de la manière dont les tests peuvent atténuer les risques. Cela fait de l'AT une partie prenante précieuse pour l'analyse des risques produit.

Lors de l'**identification des risques**, l'AT contribue par son expérience et ses connaissances aux rétrospectives, aux ateliers sur les risques, au brainstorming et à la création de checklists. L'AT peut également mener des entretiens avec les parties prenantes pour mieux comprendre ce qu'elles considèrent comme les risques les plus importants de leur point de vue.

Lors de l'**évaluation des risques**, l'AT, avec d'autres parties prenantes, contribue à la détermination du niveau de risque en estimant plusieurs facteurs tels que :

- La fréquence d'utilisation et la criticité des caractéristiques affectées
- La criticité des objectifs métiers concernés
- Les dommages financiers, environnementaux et de réputation
- La qualité de la base de test
- Les besoins juridiques ou de sûreté

L'AT aide également à catégoriser les risques produit en fonction des caractéristiques de qualité impactées, par exemple en utilisant le modèle de qualité produit de la norme ISO/IEC 25010 (2023). Souvent, le niveau de risque n'est pas uniformément réparti sur l'objet de test. Dans de tels cas, l'AT devrait décomposer l'objet de test en éléments de test (par exemple, les composants, les interfaces et les caractéristiques) et évaluer un risque donné pour chaque élément de test séparément.

Enfin, dans le cadre de l'évaluation des risques produit, l'AT propose des activités de test appropriées pour atténuer chaque risque produit identifié. Ces activités peuvent comprendre des tests statiques et des tests dynamiques. En fonction de facteurs tels que le niveau de risque, le type d'élément de test associé et la caractéristique de qualité concernée, l'AT indique les niveaux de test, les types de test, les techniques de test, les niveaux d'indépendance des tests et l'exhaustivité des tests requis. Dans l'esprit du shift-left, l'AT indique quelles activités de test peuvent atténuer le risque le plus tôt pour minimiser l'effort de test.

2.2 Contrôle des risques

Selon (ISTQB-CTFL, v4.0.1, Section 5.2.4), le contrôle des risques implique l'atténuation des risques et le suivi des risques. L'AT comprend les fonctionnalités du système et les risques potentiels qui y sont liés. Par conséquent, le rôle de l'AT est crucial dans plusieurs actions d'atténuation des risques mentionnées dans (ISTQB-CTFL, v4.0.1, Section 5.2.4) :

- La réalisation de revues est abordée dans la section 5.2.2 de ce syllabus.
- L'application des techniques de test et des niveaux de couverture appropriés est abordée à la section 3.5.
- L'application des types de tests appropriés est abordée au chapitre 4.
- L'exécution des tests de régression est abordée ci-dessous.

De plus, comme les risques et les niveaux de risque ne sont pas statiques et évoluent dans le temps, le test basé sur les risques implique un pilotage régulier des tests. Dans un cycle de vie itératif, cette surveillance est effectuée à une fréquence déterminée par l'équipe (vraisemblablement une fois par itération). Dans d'autres cycles de vie, sa fréquence est fixée par la personne responsable de la gestion du risque produit, souvent le Test Manager. L'AT y contribue en mettant à jour le référentiel des risques sur la base des changements effectués et en ajustant les actions d'atténuation des risques mentionnées ci-dessus.

2.2.1 Déterminer le périmètre des tests de régression

L'objectif principal des tests de régression est de garantir la confiance dans la qualité de l'objet de test après une modification. Cependant, il peut être impossible d'exécuter tous les tests de régression pendant la phase de tests de régression en raison de contraintes (par exemple, des restrictions de temps, de budget, d'environnement de test ou de données de test). Ce problème concerne principalement les tests exécutés manuellement, mais il s'applique également aux tests automatisés, par exemple lorsque les cycles de test sont courts et qu'il existe de nombreux tests automatisés de longue durée. Il est donc nécessaire de sélectionner des tests de régression appropriés sur la base de critères spécifiques. Il est essentiel de réviser le périmètre des tests de régression à chaque cycle de test. La revue peut conclure que la suite de tests de régression existante doit être ajustée.

La technique la plus fiable pour sélectionner les tests automatisés est une analyse d'impact, que des outils peuvent soutenir. Ces outils sont basés sur un système de gestion de configuration automatisé. Ils enregistrent les éléments de configuration qui sont activés pendant l'exécution de chaque cas de test. En cas de modification, ils suivent les éléments de configuration qui ont été modifiés et sélectionnent les tests de régression qui interagissent avec les éléments modifiés.

Ce faisant, ces outils garantissent qu'après une modification d'un élément de configuration, les tests se concentrent sur les éléments dans lesquels ils sont le plus susceptibles de trouver des défaillances (Juergens et al., 2018).

En ce qui concerne l'exécution manuelle des tests, il n'y a pas eu de preuve concluante qu'une technique de sélection des tests de régression soit clairement supérieure, car les résultats dépendent de divers facteurs (Engström et al., 2010). Par conséquent, l'AT doit décider de la technique à utiliser en fonction de la situation donnée. Les techniques couramment utilisées sont les suivantes :

- **Sélection de tests basée sur les risques**, où l'AT maintient la maintenabilité de la suite de tests de régression à un référentiel de risques. Lorsqu'un changement est effectué et que le référentiel des risques est mis à jour en conséquence, l'AT ajuste la suite de tests de régression pour couvrir les niveaux de risque les plus élevés.

- **Tests basés sur l'historique**, où l'AT évalue les exécutions de test passées et détermine celles qui ont révélé des défauts ou qui étaient sensibles à des changements similaires à celui passé depuis la dernière exécution des tests. La réexécution des tests correspondants dans le cadre du test de régression augmente la probabilité d'exposer des défauts similaires. L'AT peut également inclure certains tests qui n'ont pas été exécutés depuis longtemps pour s'assurer qu'ils passent toujours.
- **Tests basés sur la couverture**, où l'AT sélectionne un petit nombre de tests qui atteignent la plus grande couverture possible sur la base de la (des) technique(s) de test choisie(s). Le nombre de tests doit être soigneusement équilibré avec l'augmentation de la couverture par test.
- **Matrice de traçabilité des exigences**, qui est utilisée pour évaluer l'impact des changements d'exigences sur les tests associés. Elle peut s'avérer particulièrement utile lorsque des exigences nouvelles ou modifiées ont un impact indirect sur des fonctionnalités existantes. L'AT sélectionne des tests de régression pour les fonctionnalités directement affectées et pour les caractéristiques associées afin de couvrir d'éventuels effets secondaires involontaires. Dans le cadre du développement logiciel Agile, on peut procéder de la même manière en sélectionnant des tests qui couvrent les critères d'acceptation impactés par des User Stories nouvelles ou modifiées.
- **Tests basés sur des profils opérationnels**, où l'AT sélectionne les cas de test de régression à exécuter en fonction des canevas d'utilisation de l'objet de test. Par exemple, lorsqu'on teste une boutique en ligne, l'un de ces tests peut consister pour l'utilisateur à se connecter, à rechercher des produits, à les ajouter au panier et à passer la commande. Lorsqu'une application est modifiée de manière significative, cette technique permet d'obtenir un aperçu rapide de la fonctionnalité globale du système. S'il en résulte un trop grand nombre de tests, l'AT donne la priorité aux canevas d'utilisation critiques qui se produisent souvent et couvrent des fonctionnalités et des processus métiers essentiels.
- **Analyse d'impact**, qui peut également être utilisée pour sélectionner les cas de test de régression à exécuter manuellement si l'AT sait lesquels d'entre eux interagissent avec les éléments de configuration modifiés.

Il est souvent nécessaire d'utiliser une combinaison de techniques de sélection pour obtenir une suite de tests de régression plus complète et plus efficace. Cependant, l'AT doit soigneusement équilibrer le besoin de couverture avec une taille gérable de la suite de tests. Après chaque cycle de test, l'AT analyse les résultats du test pour déterminer l'efficacité des techniques appliquées (voir section 5.3.1). La fois suivante, l'AT conserve les techniques efficaces et remplace celles qui sont inefficaces. Cela permet d'améliorer en permanence la sélection des tests de régression au fil du temps. Elle est essentielle dans les modèles de développement itératif et incrémental, où les changements sont fréquents.

3 Analyse de test et conception des tests – 615 minutes

Mots-clés

test basé sur des checklists, technique de test basée sur le comportement, test combinatoire, crowd testing, test CRUD, technique de test basée sur les données, test basé sur les tables de décision, test de domaine, partition d'équivalence, test basé sur l'expérience, relation métamorphique, test métamorphique, test aléatoire, technique de test basée sur des règles, test basé sur des scénarios, test basé sur une session, test de transition d'état, charte de test.

Objectifs d'apprentissage pour le chapitre 3 :

3.1 3.1 Techniques de test basées sur les données

TA-3.1.1 (K3) Appliquer le test de domaine

TA-3.1.2 (K3) Appliquer le test combinatoire

TA-3.1.3 (K2) Résumer les avantages et les limites des tests aléatoires

3.2 3.2 Techniques de test basées sur le comportement

TA-3.2.1 (K2) Expliquer le test CRUD

TA-3.2.1 (K3) Appliquer le test de transition d'état

TA-3.2.3 (K3) Appliquer le test basé sur des scénarios

3.3 Techniques de test basées sur des règles

TA-3.3.1 (K3) Appliquer le test par table de décision

TA-3.3.2 (K3) Appliquer le test métamorphique

3.3 3.3 Tests basés sur l'expérience

TA-3.4.1 (K3) Préparer des chartes de test pour le test basé sur des sessions

TA-3.4.2 (K3) Préparer des checklists qui soutiennent le test basé sur l'expérience

TA-3.4.3 (K2) Donner des exemples de bénéfices et de limites du crowd testing

3.4 3.4 Appliquer les techniques de test les plus appropriées

TA-3.5.1 (K4) Sélectionner les techniques de test appropriées pour atténuer les risques produit dans une situation donnée

TA-3.5.2 (K2) Expliquer les bénéfices et les risques de l'automatisation de la conception des tests

Introduction à l'analyse de test et à la conception des tests

Les techniques de test sont principalement utilisées dans l'analyse de test et la conception des tests. Les techniques de test abordées dans ce chapitre couvrent les techniques de test boîte noire et les techniques de test basées sur l'expérience. Les techniques de test boîte blanche sont discutées dans (ISTQB-TTA, v4.0).
Ce syllabus subdivise les techniques de test boîte noire en trois catégories basées sur les conditions de test sous-jacente modélisant :

- les éléments des données (basées sur les données)
- les éléments du comportement dynamique (basées sur le comportement)
- les éléments des règles du comportement statique (basées sur les règles)

3.1 Techniques de test basées sur les données

Dans cette section, le terme « domaine » est utilisé pour représenter l'ensemble des données d'entrée d'un élément de test. Les données d'entrée provenant de différents domaines entraînent différents comportements de l'élément du test. Les techniques de test basées sur les données visent à vérifier que l'implémentation traite correctement des domaines spécifiques. Le syllabus de niveau Fondation (ISTQB-CTFL, v4.0.1, Section 4.2) couvre deux techniques de test qui peuvent être catégorisées comme basées sur les données : les partitions d'équivalence (EP) et l'analyse des valeurs limites (BVA). Ce syllabus introduit trois autres techniques de test basées sur les données :

- le test de domaine - étend l'EP et la BVA à des domaines avec des paramètres multiples et des partitions complexes.
- le test combinatoire - se concentre sur les interactions de plusieurs paramètres dans un domaine multidimensionnel
- le test aléatoire - sélectionne des entrées aléatoires dans le domaine sur la base d'une distribution de probabilité spécifiée.

Notez que le test aléatoire peut généralement se référer à la fois à des données d'entrée aléatoires et à des événements aléatoires. Ce syllabus ne traite que du test avec des données d'entrée aléatoires.

3.1.1 Test de domaine

Le test de domaine permet de vérifier si l'élément du test se comporte comme spécifié sur les partitions d'équivalence du domaine et à leurs frontières. Dans ce cas, les partitions d'équivalence sont définies à l'aide d'expressions qui combinent des conditions atomiques à l'aide d'opérateurs booléens (par exemple, AND, OR et NOT) et impliquent une ou plusieurs variables en interaction. Chaque condition atomique définit une frontière de la partition d'équivalence. Les frontières fermées sont formées par des expressions relationnelles avec les opérateurs ≤, ≥ ou =, et les frontières ouvertes sont formées par des expressions relationnelles avec les opérateurs <, >, ou /=.
Par exemple, l'expression taille > 1,29 mètre ET poids / taille^2 ≥ 30 définit une partition d'équivalence d'un domaine bidimensionnel de deux variables qui a deux frontières, une ouverte et une fermée.
Les tests de domaine visent à identifier les défauts dans les implémentations des partitions d'équivalence (par exemple, un opérateur ou une constante incorrecte) en sélectionnant les éléments de couverture appropriés qui peuvent révéler ces défauts. Les critères de couverture dans les tests de domaine font référence aux points ON, OFF, IN et OUT :

- Pour une frontière fermée, un point ON se trouve sur cette frontière. Pour une frontière ouverte, un point ON se trouve à l'intérieur de la partition d'équivalence et est le plus proche de la frontière selon la précision donnée.
- Pour une frontière fermée, un point OFF se trouve à l'extérieur de la partition d'équivalence et est le plus proche de la frontière selon la précision donnée. Pour une frontière ouverte, un point OFF se trouve sur cette frontière.
- Un point IN appartient à la partition d'équivalence et n'est pas un point ON.
- Un point OUT se situe en dehors de la partition d'équivalence et n'est pas un point OFF.

Chacun de ces types de points est lié à la frontière de la partition d'équivalence. Un même point peut avoir différents types pour différentes frontières.

Les critères de couverture sont les suivants :

Couverture de domaine simplifiée (Jeng et al., 1994), qui exige les éléments de couverture suivants :

- Un point ON et un point OFF pour chaque frontière définie par l'un des opérateurs <, ≤, > ou ≥.
- Un point ON et deux points OFF situés de part et d'autre de la frontière pour l'opérateur =.
- Un point OFF et deux points ON situés de part et d'autre de la frontière pour l'opérateur ≠.

Chaque point OFF doit être aussi proche que possible du point ON correspondant, conformément à la précision donnée.

Couverture fiable du domaine (Forgács et al., 2024), qui exige les éléments de couverture suivants :

- Un point ON, un point OFF, un point IN et un point OUT pour chaque frontière définie par l'un des opérateurs <, ≤, > ou ≥.
- Un point ON et deux points OFF situés de part et d'autre de la frontière pour l'opérateur =.
- Un point OFF et deux points ON situés de part et d'autre de la frontière pour l'opérateur ≠.

Pour les deux critères de couverture examinés, le nombre d'éléments de couverture dépend linéairement du nombre de frontières. Il peut être optimisé pour les deux critères de couverture en utilisant le même élément de couverture pour différentes frontières.
Par exemple, toutes les frontières d'une partition d'équivalence peuvent utiliser un point IN commun ou une paire de points ON et OFF d'une frontière peut être utilisée comme points OFF et ON pour la frontière de la partition d'équivalence adjacente. Pour les domaines avec de nombreuses frontières, un algorithme d'optimisation avancé est disponible (Forgács et al., 2024).
La couverture de domaine fiable résulte en une quantité légèrement plus importante d'éléments de couverture que la couverture de domaine simplifiée, mais peut détecter considérablement plus de défauts de domaine (Site of Software Test Design, 2020).
Le test de domaine peut être appliqué à n'importe quel niveau de test. Il généralise la BVA et l'EP (voir ISTQB-CTFL, v4.0.1, section 4.2) à des domaines plus complexes. Diverses approches des tests de domaine peuvent être trouvées dans des manuels comme (Beizer, 1990, Ch. 6 ; Binder, 2000, Section 10.2.4 ; Kaner ; Padmanabhan, et al., 2013 ; Jorgensen, 2014, Ch. 5).

3.1.2 Test combinatoire

Certaines défaillances logicielles sont dues à des combinaisons spécifiques de valeurs de paramètres, connues sous le nom de défaillances d'interaction. Les tests combinatoires visent à révéler ces défaillances en explorant ces combinaisons de valeurs de paramètres.

Dans les tests combinatoires, les conditions de test combinent généralement des paramètres de

configuration ou des valeurs de données d'entrée. Il existe donc deux approches principales des tests combinatoires. La première approche utilise des combinaisons de paramètres de configuration, et chacune de ces combinaisons peut être testée à l'aide du même cas de test. La seconde approche utilise des combinaisons de valeurs de données d'entrée, qui font ensuite partie de cas de tests complets, créant ainsi une suite de tests pour le système sous test (D. R. Kuhn et al., 2013).

Une paire paramètre-valeur spécifique se compose d'un paramètre et de sa valeur (par exemple, '(couleur, rouge)').

Les critères de couverture combinatoires sont notamment les suivants (Ammann et al., 2008), (Forgács et al., 2019) :

- **Couverture du choix de base** qui suppose que certaines paires paramètre-valeur sont plus importantes que d'autres. Une paire paramètre-valeur de base est choisie pour chaque paramètre, et un élément de couverture de base est la combinaison des paires paramètre-valeur de base. Les éléments de couverture suivants sont créés à partir de l'élément de couverture de base pour chaque paramètre en remplaçant sa paire paramètre-valeur de base par chaque valeur hors base.
- **Couverture par paires**, dans laquelle les éléments de couverture sont des paires de paires paramètre-valeur pour deux paramètres quelconques. Il existe des outils permettant de générer des éléments de couverture. Toutefois, il est généralement difficile de trouver un ensemble minimal de cas de test permettant d'obtenir une couverture par paires.

Pour les paramètres ayant de nombreuses valeurs, la méthode EP peut d'abord être appliquée pour réduire ce nombre et l'ensemble des combinaisons qui en résultent. La saisie des paramètres et de leurs valeurs dans un arbre de classification ou un modèle de caractéristiques (voir IREB-Glossary, 2024) soutient cette activité. Le nombre final de cas de test peut être affecté par des contraintes entre les paires paramètre-valeur, par des combinaisons ajoutées manuellement et connues pour être problématiques, ou en raison de combinaisons invalides ou infaisables.

L'idée essentielle pour les tests combinatoires est que tous les paramètres ne contribuent pas à une défaillance et que la plupart des défaillances sont déclenchées par une seule valeur de paramètre ou par des interactions entre un nombre relativement faible de paramètres (Cohen et al., 1994). Cela correspond à l'hypothèse de l'effet de couplage, qui indique que la détection de défauts simples dans un programme permet souvent de découvrir des défauts complexes (Offutt, 1992).

Dans une étude limitée (D. Kuhn et al., 2004), les résultats ont montré qu'environ 97 % des défaillances sont causées par seulement une ou deux conditions en interaction, ce qui indique que les tests par paires sont une technique de test efficace.

De plus amples informations sur les tests combinatoires, y compris d'autres critères de couverture comme le diff-pair-t ou le test n-wise, peuvent être trouvées dans (Ammann et al., 2008), (Forgács et al., 2019). Des outils soutenant les tests combinatoires sont également disponibles dans (Czerwonka, 2004).

3.1.3 Test aléatoire

Le test aléatoire consiste à sélectionner des données de test de manière aléatoire à partir du domaine d'entrée de l'élément testé, sur la base d'une distribution de probabilité spécifiée. À des fins de validation, il est recommandé d'utiliser une distribution basée sur des profils opérationnels. À des fins de vérification, la distribution doit être indépendante de l'utilisation afin d'éviter tout biais. Les résultats attendus peuvent être ajoutés aux cas de test en utilisant un oracle de test, ce qui nécessite le plus souvent un oracle de test automatisé.

Les tests aléatoires peuvent être guidés ou non guidés. Dans les tests aléatoires non guidés, la distribution de probabilité reste fixe tout au long du processus. Les tests aléatoires guidés, illustrés par des techniques telles que les tests aléatoires adaptatifs (Huang et al., 2019), ajustent la distribution en fonction de valeurs sélectionnées au préalable, qui évoluent au fil du temps. Les tests aléatoires guidés visent à couvrir efficacement le domaine d'entrée, en tenant compte du fait que les défauts se regroupent souvent dans

des régions spécifiques du domaine.
Les tests aléatoires ne disposent pas de critères de couverture reconnus. Par conséquent, les critères de sortie ne peuvent s'appuyer que sur le nombre de tests exécutés, le temps de test ou des mesures similaires d'achèvement.

Les tests aléatoires sont particulièrement utiles lorsque les connaissances du domaine sont limitées ou lorsqu'un volume important de données de test est nécessaire. Ils sont rentables et fournissent, en termes probabilistes, des informations sur la fiabilité de l'objet de test. Les tests aléatoires permettent d'éviter les biais tels que les défauts négligés lors des tests manuels en raison d'une confiance mal placée dans certains codes ou fonctionnalités. Cependant, les tests aléatoires présentent également plusieurs défis et limites, notamment la négligence de la sémantique des données, le risque de passer à côté de défauts liés à la signification des données, la possibilité de négliger certains défauts, la génération de tests redondants, la dépendance à un oracle de test automatisé et des résultats aléatoires conduisant à des résultats de test incohérents. Il est essentiel de trouver un équilibre entre les avantages et les limites des tests aléatoires dans chaque contexte de test.

Traditionnellement, les tests aléatoires sont considérés comme moins efficaces que d'autres techniques de test. Ces dernières années, cette hypothèse a fait l'objet de nombreuses études empiriques. Il a été constaté que dans les circonstances mentionnées ci-dessus, les tests aléatoires peuvent être plus efficaces et plus efficients que d'autres techniques de test basées sur les données (Arcuri et al., 2012), (Wu et al., 2020). Les tests aléatoires sont également utilisés dans le fuzz testing et le chaos engineering.

3.2 Techniques de test basées sur le comportement

Les techniques de test basées sur le comportement dérivent des cas de test à partir des spécifications du comportement dynamique (c'est-à-dire dépendant de l'état) de l'élément du test. Ce syllabus traite de trois techniques de test basées sur le comportement :

- Test CRUD
- Test de transition d'état
- Test basé sur des scénarios

Le test CRUD et le test basé sur des scénarios étendent la gamme des techniques de test boîte noire connues (ISTQB-CTFL, v4.0.1, section 4.2). Ce syllabus complète le test de transition d'état avec des critères de couverture supplémentaires.

3.2.1 Test CRUD

Le test CRUD vérifie le cycle de vie des entités de données traitées par l'élément du test. CRUD signifie créer, lire, mettre à jour et supprimer (NDT: Create, Read, Update, Delete). Il s'agit des quatre opérations de base que les fonctions peuvent effectuer sur les entités.

La matrice CRUD donne un aperçu du cycle de vie des entités de données. Ses colonnes représentent les entités et ses lignes représentent les fonctions. Supposons qu'une fonction exécute une ou plusieurs opérations particulières de création, lecture, mise à jour ou suppression sur une entité donnée. Ceci est représenté dans la matrice en utilisant les initiales de ces opérations : C, R, U ou D. Pour créer une matrice CRUD, l'AT détermine pour chaque fonction laquelle des quatre opérations elle effectue sur quelles entités. Une attention particulière doit être accordée à l'opération de lecture, souvent implicitement liée aux opérations C-U-D.

Le test CRUD se compose de deux parties (Koomen et al., 2006) :

- **Le test de complétude** est un test statique qui vérifie si toutes les opérations possibles (c'est-à-dire C, R, U et D) se produisent avec chaque entité (c'est-à-dire si le cycle de vie complet a été implémenté pour chaque entité). L'absence d'une opération est une anomalie qui doit être

examinée.

- **Le test de cohérence** est un test dynamique visant à intégrer les différentes fonctions et à vérifier si l'entité est utilisée de manière cohérente. Il vérifie que les fonctions interagissent correctement lors du traitement d'une entité. Les cas de test doivent couvrir toutes les opérations de la matrice CRUD. En outre, des tests négatifs doivent également être inclus (par exemple, la lecture d'une entité qui n'a pas encore été créée). Les cas de test sont conçus par entité en combinant des fonctions afin de couvrir l'ensemble de son cycle de vie.

La couverture CRUD est mesurée par le nombre d'opérations exécutées par la suite de tests divisé par le nombre total d'opérations dans la matrice CRUD. Une couverture CRUD plus rigoureuse peut considérer des combinaisons spécifiques d'opérations comme des éléments de couverture (par exemple, après chaque U, tous les R possibles doivent être couverts).

Les tests CRUD sont principalement utilisés au niveau système. Ils se concentrent sur les défauts dans le comportement de l'élément du test dans le traitement des entités, tels que les violations de l'intégrité des données, les défauts de contrôle d'accès ou les incohérences des données.

3.2.2 Test de transition d'état

De nombreux systèmes complexes sont sensibles à l'état (c'est-à-dire que la réaction du système à un événement dépend de l'état actuel du système). Les systèmes embarqués, les systèmes basés sur le dialogue, les systèmes de contrôle ou les systèmes qui traitent des entités et de leurs cycles de vie sont des exemples de systèmes sensibles à l'état.

Un état simplifie les détails internes complexes, ce qui permet aux parties prenantes de comprendre plus facilement le comportement attendu. Lors de l'analyse des tests, l'AT doit s'assurer que le modèle basé sur les états représente le comportement attendu de l'élément du test au niveau de détail requis pour le test. Si la base de test contient déjà un tel modèle, l'AT doit vérifier s'il contient les conditions de test et, si nécessaire, l'adapter ou concevoir un nouveau modèle.

Dans les modèles basés sur les états, les nœuds représentent les états et les arêtes représentent les transitions d'état. Les transitions d'état sont déclenchées par des événements et peuvent inclure des conditions de garde et des actions (voir ISTQB-CTFL, v4.0.1, section 4.2.4). Plusieurs variantes de ces modèles sont utilisées, telles que les machines à états finis étendues (Bochmann et al., 1994), les diagrammes d'états de Harel (Harel, 1987) ou les machines à états UML (OMG® UML, 2017).

Outre les critères de couverture des transitions d'états abordés dans (ISTQB-CTFL, v4.0.1), deux autres critères sont examinés ci-dessous, pour lesquels une grande efficacité en matière de détection des défauts a été prouvée empiriquement :

- **La couverture N-switch** s'applique à des séquences valides de N+1 transitions consécutives, également appelées N-switches (Chow, 1978). La couverture 0-switch est égale à la couverture des transitions valides (voir ISTQB-CTFL, v4.0.1, section 4.2.4). Un 1-switch est une paire de transitions entrantes et sortantes à un état. L'extension des N-switches à leur extrémité par des transitions d'état valides ultérieures donne N+1 switches. La couverture 0-switch et 1-switch est fréquemment utilisée dans la pratique. Une couverture 2-switch de 100 % ou plus n'est indiquée que pour un risque élevé de défaillance dû à des séquences d'événements inattendues, car le nombre de N-switches peut croître de manière exponentielle avec N.
- **La couverture aller-retour** (Ammann et al., 2008) s'applique aux chemins qui forment des boucles dans un modèle basé sur les états. Un round-trip est une boucle dans laquelle les états de début et de fin sont identiques et aucun autre état de cette boucle ne se produit deux fois. Les éléments de couverture sont les round-trips. (Antoniol et al., 2002) ont trouvé ce critère très efficace pour détecter les défauts.

Les tests de transition d'état sont bien adaptés à l'automatisation à l'aide d'outils de tests basés sur des modèles. Comme la plupart des techniques de test boîte noire, les tests de transition d'état contribuent à la prévention des défauts en détectant les défauts dans la spécification pendant la modélisation. Les tests de transition d'état peuvent être appliqués à n'importe quel niveau de test.

D'autres critères de couverture des transitions d'états sont abordés dans (Rechtberger et al., 2022). Un modèle basé sur les états récent est le modèle action-état abordé dans (Forgács et al., 2024).

3.2.3 Test basé sur des scénarios.

Le test basé sur des scénarios évalue le comportement de l'élément du test dans des scénarios réalistes. Des techniques telles que la recherche utilisateur, les User Stories, les personae et la cartographie du parcours utilisateur (voir section 11) peuvent aider à identifier des scénarios pertinents. Dans le test basé sur des scénarios, l'AT crée un modèle de scénario basé sur des séquences d'actions qui constituent des workflows à travers l'élément du test (cf. ISO/IEC/IEEE 29119-4, 2021). Ce syllabus traite des deux modèles suivants : les diagrammes d'activité et les cas d'utilisation. D'autres modèles incluent les organigrammes, les diagrammes BPMN (OMG® BPMN, 2013), les diagrammes de séquence ou les diagrammes de collaboration (OMG® UML, 2017).

Ces modèles ne sont pas abordés dans ce syllabus. Veuillez vous référer à (ISTQB- AcT, v1.0) pour plus de détails.

Un **diagramme d'activité** est une représentation graphique du workflow au sein d'un système. Les diagrammes d'activité sont particulièrement utiles pour modéliser les processus métier, mais peuvent également modéliser le flux de contrôle. Les diagrammes d'activité étendent la notation des organigrammes, permettant ainsi la modélisation de la concurrence. Les principaux éléments des diagrammes d'activité sont les nœuds de début et de fin, les actions, les transitions, les nœuds de décision, les nœuds de fusion, les nœuds de bifurcation, les nœuds de jonction et les couloirs.

Un **cas d'utilisation** est une description textuelle ou graphique des actions représentant les interactions entre un utilisateur et un système ou entre des systèmes. Trois types de scénarios sont distingués dans le modèle :

- **Scénario principal (**appelé « chemin heureux » NDT: "happy path") : séquence d'actions typique et attendue menant à la réalisation d'un objectif spécifique du point de vue de l'utilisateur. Un cas d'utilisation ne peut avoir qu'un seul scénario principal.
- **Extension** (ou scénario alternatif) : séquence d'actions, autre que le scénario principal, qui conduit finalement à la réalisation de l'objectif du scénario principal.
- **Exception :** séquence d'actions qui ne permet pas d'atteindre l'objectif du scénario principal en raison d'une action inattendue (par exemple, une utilisation anormale ou une entrée non valide).

Dans les tests basés sur des scénarios, l'AT conçoit des cas de test pour couvrir les scénarios (c'est-à-dire les éléments de couverture), souvent en suivant une hiérarchisation basée sur les risques. Lorsque le modèle de scénario ne contient pas de boucles, chaque scénario peut être testé avec un cas de test distinct (c'est-à-dire que tous les scénarios possibles dans le modèle peuvent être exercés avec une suite de tests). Le nombre de scénarios potentiels (chemins) peut être infini lorsqu'il existe des boucles. Dans ce cas, la couverture de boucle simple peut être appliquée au modèle de scénario. La couverture de boucle simple nécessite de tester chaque boucle lorsqu'elle est exécutée zéro fois (c'est-à-dire ignorée), avec exactement une itération, avec plus d'une itération (c'est-à-dire un nombre typique d'itérations) et avec le nombre maximal d'itérations (si possible).

La **couverture basée sur des scénarios** est mesurée par le nombre de scénarios exécutés divisé par le nombre de tous les scénarios identifiés. Les scénarios peuvent être identifiés en appliquant divers critères de couverture au modèle de scénario (voir Koomen et al., 2006). Les critères de couverture peuvent nécessiter de tester chaque scénario plus d'une fois. Par exemple, un scénario peut nécessiter

une couverture EP ou BVA supplémentaire des variables apparaissant dans le scénario. Dans de tels cas, un scénario peut devoir être testé par plusieurs cas de test.

Les tests basés sur des scénarios sont souvent effectués dans le cadre de tests système ou de tests d'acceptation. Ils prennent la forme de tests de bout en bout axés sur l'aptitude fonctionnelle d'un système (voir section 4.1) du point de vue de l'utilisateur. Toutefois, les tests basés sur des scénarios peuvent également être utilisés à d'autres niveaux de test (par exemple, les tests d'intégration des composants avec des scénarios basés sur les protocoles d'interaction ou les tests de composants de classes orientées objet sensibles à l'état avec des scénarios invoquant diverses méthodes) et dans les tests non fonctionnels (par exemple, les scénarios peuvent constituer les éléments des profils opérationnels utilisés dans les tests de fiabilité, de flexibilité ou de compatibilité).

3.3 Techniques de test basées sur des règles

Les techniques de test basées sur des règles vérifient l'implémentation du comportement sans état d'un élément du test, spécifié par des règles qui sont valides quel que soit son état (par exemple, des règles métier). Dans ce syllabus, deux techniques de test basées sur des règles sont abordées, à savoir :

- Les tests par tables de décisions
- Les tests métamorphiques

Le syllabus de niveau Fondation (ISTQB-CTFL, v4.0.1) couvre les bases des tests par tables de décisions. Ce syllabus aborde des sujets plus avancés. La terminologie et la notation utilisées sont conformes à la norme (OMG® DMN, 2024).

3.3.1 Test par tables de décisions

Dans les tests par tables de décisions, l'AT commence généralement par créer une table de décision complète ou par analyser une table existante obtenue à partir de la base de test. Le nombre de règles d'une table de décision complète est le produit du nombre de valeurs de ses conditions. Ce nombre croît de manière exponentielle avec le nombre de conditions et leurs valeurs, ce qui motive la minimisation des tables de décisions.

Les tables de décision peuvent être minimisées en fusionnant des règles en utilisant l'opérateur « – ». Il est recommandé de ne pas tenir compte des règles irréalisables (c'est-à-dire les règles avec une combinaison de valeurs de conditions qui ne peuvent jamais se produire) lors de la fusion des règles. Souvent, les règles irréalisables sont supprimées de la table de décision avant la fusion. Un algorithme de minimisation systématique possible recherche les règles équivalentes en termes d'actions qui ne diffèrent que par une seule condition et couvrent toutes ses valeurs possibles. Ces règles sont fusionnées et la valeur de condition différente est remplacée par « – ». Le résultat de la minimisation systématique peut dépendre de l'ordre dans lequel les colonnes sont minimisées, ce qui ne conduit pas toujours à une solution optimale. L'AT doit vérifier si une minimisation supplémentaire est possible.

Il incombe à l'AT de revoir la table de décision, éventuellement avec d'autres parties prenantes, en fonction des critères suivants :

- cohérence (c'est-à-dire que si deux règles différentes s'appliquent à la même combinaison de valeurs des conditions, elles sont alors équivalentes en termes d'actions)
- faisabilité (c'est-à-dire qu'elle ne contient aucune règle irréalisable)
- complétude (c'est-à-dire qu'aucune combinaison réalisable de valeurs des conditions ne manque)
- exactitude (c'est-à-dire que les règles modélisent le comportement prévu du système)

En outre, il est conseillé que les règles ne se chevauchent pas (c'est-à-dire que pour toute combinaison de valeurs des conditions, une seule règle s'applique).

Des règles peuvent se chevaucher lorsque la table de décision d'origine est déjà minimisée ou si les règles sont fusionnées de manière incorrecte.
La *procédure de somme de contrôle* utilise le nombre de règles dans une table de décision minimisée pour indiquer les chevauchements et les lacunes. Pour chaque règle de la table minimisée, le nombre de règles qu'elle représente dans la table de décision d'origine est calculé. Une règle sans « – » dans les conditions (c'est-à-dire avec des valeurs individuelles pour toutes les conditions) obtient une note de 1. Chaque valeur « – » multiplie la note de la règle par le nombre de valeurs individuelles pour la condition correspondante.

La somme des notes des règles correspond à la somme de contrôle de la table de décision minimisée. Si la somme de contrôle est inférieure à celle de la table de décision d'origine, la table de décision minimisée est incomplète. Si la somme de contrôle est supérieure, certaines règles se chevauchent ou des règles supplémentaires existent (par exemple, des combinaisons de conditions irréalisables). Si la minimisation a été effectuée correctement, les sommes de contrôle sont égales. Des sommes de contrôle égales ne garantissent pas à elles seules que la table de décision minimisée est équivalente à la table d'origine.

La **couverture de la table de décision** est mesurée en divisant le nombre de colonnes exercées par le nombre total de colonnes réalisables dans la table de décision. Lors de l'implémentation des cas de test résultant d'une règle de table de décision, l'AT doit implémenter les conditions et les actions. L'AT doit décider comment implémenter les valeurs de condition « – » pour une règle donnée, car « – » représente au moins deux valeurs. Toutefois, si le niveau de risque associé à la table de décision est élevé, l'AT doit s'abstenir de minimiser et doit mesurer la couverture des colonnes réalisables dans la table de décision complète.

3.3.2 Test métamorphique

Le test métamorphique (MT) est une technique visant à générer des cas de test sur la base d' un cas de test source existant. Un ou plusieurs cas de test de suivi sont générés en modifiant (métamorphisant) le cas de test source sur la base d'une relation métamorphique (MR). La MR définit une propriété de l'élément du test et décrit comment une modification des entrées d'un cas de test se reflète dans les résultats attendus du cas de test.

L'AT combine les cas de test source et de suivi d'une MR en une procédure de test, avec une évaluation conjointe des résultats. S'ils remplissent la relation métamorphique, le test passe, sinon il échoue. En cas d'échec, le débogage qui suit doit déterminer lequel des cas de test individuels impliqués a échoué.

Prenons par exemple une fonction qui détermine la moyenne d'une série de nombres. Un cas de test source est créé avec une série de nombres et une moyenne attendue, et le cas de test est exécuté pour confirmer qu'il passe. Une MR pourrait indiquer que toute permutation de la série de nombres donne la même moyenne.

En utilisant cette MR, l'AT peut créer plusieurs cas de test de suivi, chacun avec le même ensemble de nombres en entrée, mais dans un ordre différent. Le résultat attendu reste le même.

Pour la même fonction moyenne, l'AT peut également utiliser une autre MR, qui stipule que si chaque nombre de la série est multiplié par le même nombre, x, alors le résultat attendu sera également multiplié par x. Avec cette MR, l'AT peut créer un nombre quelconque de cas de test de suivi à partir d'un cas de test source en choisissant différentes valeurs de x. Cela peut être particulièrement utile lorsque la conception des tests et l'exécution des tests sont automatisées. L'AT peut également combiner deux ou plusieurs MR pour créer des cas de test de suivi (par exemple, permuter et multiplier par 2).

Le MT peut également être utilisé en présence d'un problème d'oracle de test (voir section 1.3.4). Dans

une telle situation, les résultats attendus du cas de test source et des cas de test de suivi ne sont pas disponibles, de sorte que leurs résultats de test ne peuvent pas être évalués individuellement. Un exemple peut être un programme actuariel basé sur l'IA qui prédit l'âge au décès à partir d'un grand ensemble de données. La MR pourrait indiquer que si le nombre de cigarettes fumées augmente, l'âge prévu du décès devrait diminuer.

Actuellement, il n'existe aucune mesure de couverture reconnue pour le MT qui fournisse des critères de sortie utiles. Il ne suffit pas de couvrir chaque MR une seule fois, car cela ne permet d'obtenir qu'une vérification partielle des résultats attendus. L'AT peut combiner le MT avec des tests aléatoires afin de générer de nombreux scénarios de test de bas niveau et des cas de test de suivi pour la même MR. Par exemple, dans le calcul de moyenne mentionné ci-dessus, un générateur de nombres aléatoires peut être utilisé pour générer diverses entrées pour le cas de test source, ainsi que des permutations et des multiplicateurs aléatoires pour les cas de test de suivi.

Le MT peut être utilisé pour la plupart des éléments du test et appliqué à des tests fonctionnels et non fonctionnels (par exemple, tests de charge, génération de charge par des cas de test de suivi utilisant des MR, ou tests de facilité d'installation avec divers paramètres d'installation pouvant être sélectionnés dans plusieurs séquences). Il s'agit d'une technique de test privilégiée pour les systèmes basés sur l'IA (ISTQB-AI, v1.0).

Pour plus de détails, voir également la norme (ISO/IEC/IEEE 29119-4, 2021) ou les articles de synthèse (Segura ; Towey, et al., 2020) et (Segura ; Fraser, et al., 2016).

3.4 Test basé sur l'expérience

Un AT utilise des approches de test basées sur l'expérience et des techniques de test, en tirant parti de son expertise et de ses expériences passées pour guider les tests. Ce chapitre détaille l'utilisation de la documentation par l'AT dans les tests basés sur des sessions et les tests basés sur une checklist (voir ISTQB-CTFL, v4.0.1, sections 4.4.2 et 4.4.3). En outre, le crowd testing est abordé. Toutes les techniques de test basées sur l'expérience peuvent être appliquées dans le cadre de tests collaboratifs, comme l'acceptance test-driven development (ATDD). Pour plus de détails, voir (ISTQB-CTFL, v4.0.1, section 4.5).

3.4.1 Chartes de test soutenant le test basé sur des sessions.

Dans le cadre des tests exploratoires, une charte de test définit la mission d'une session de test, en décrivant son périmètre et ses objectifs, ainsi que des informations telles que les limitations, les délais et les risques. Elle sert de feuille de route pour les tests, en structurant les sessions de test. Les chartes de test aident les AT à rester concentrés sur des domaines ou des caractéristiques spécifiques à tester, tout en leur laissant la flexibilité nécessaire pour explorer le système si nécessaire. La charte de test ne spécifie pas les suites de tests qui seront exécutées lors de chaque session de test.

Lors de la préparation d'une charte de test pour des tests basés sur des sessions, l'AT doit tenir compte de certains facteurs qui influencent la conception de la charte de test, en particulier :

- les facteurs liés aux clients et aux exigences (par exemple, les exigences énoncées par les clients, les cas d'utilisation métier du système, les exigences de qualité) et les cartographies du parcours utilisateur (c'est-à-dire l'interaction de l'utilisateur avec le système au fil du temps ; NDT « User Journey Maps »)
- les facteurs liés au produit (par exemple, les flux fonctionnels, les principaux objectifs du produit, les caractéristiques du produit, la conception du logiciel et les interfaces)
- les facteurs liés à la gestion de projet (par exemple, les contraintes de temps, l'objectif du projet, l'effort estimé et la valeur métier)

Une charte de test consiste en une mission qui décrit l'objectif du test et diverses informations supplémentaires. Un format léger populaire pour une mission est « Explorer [cible] avec [ressources]

pour découvrir [informations] »
(Hendrickson, 2013), où [cible] décrit ce qui doit être exploré (par exemple, zone, caractéristique, risque, composant, et exigence), [ressources] décrit ce que l'AT utilisera (par exemple, données de test, configurations, outils, restrictions, heuristiques et dépendances), et [informations] explique quel type d'information l'AT vise à trouver (par exemple, évaluations des caractéristiques de qualité, type de défauts attendus et violations des normes).
Les chartes de test peuvent contenir, sans s'y limiter, les informations supplémentaires suivantes (Ghazi et al., 2017) :

- Informations organisationnelles (par exemple, durée de la session de test, date et heure de début, nom du testeur)
- Objectifs du test (par exemple, motivation du test et mission de la charte de test)
- Périmètre du test (par exemple, les domaines spécifiques d'intérêt au sein du système sous test, le niveau de test, les techniques de test à utiliser, les idées de test, les critères de sortie, les priorités, ce que la charte de test est censée couvrir et une description de ce qui ne sera pas testé)
- Critères d'entrée (c'est-à-dire les préconditions qui doivent être remplies pour pouvoir démarrer la session de test)
- Informations relatives au produit (par exemple, définition, données et workflow entre les composants, et architecture du système)
- Limitations (c'est-à-dire ce que le produit ne doit jamais faire)
- Description de l'environnement de test
- Sources de données existantes, informations sur le produit et outils de test qui faciliteraient les tests
- Informations historiques (par exemple, défauts précédemment trouvés tels que des défauts de compatibilité et d'interopérabilité, questions en suspens se rapportant à des anomalies existantes et canevas de défaillance liés aux tests effectués dans le passé)
- Contraintes et risques (par exemple, réglementations, règles et normes utilisées)

Au cours d'une session de test, l'AT enregistre des logs de test qui comprennent des questions, des observations ou des idées pour des tests futurs, ainsi que les résultats du test. Ces données sont consignées dans une fiche de session.
Le périmètre et le niveau de détail des informations incluses dans les chartes de test spécifiques peuvent varier et affecter le degré de flexibilité de l'AT. Par exemple, définir uniquement les objectifs généraux du test laisse une grande marge d'exploration, tandis qu'ajouter des informations sur les techniques de test à utiliser peut limiter l'AT. D'un autre côté, ajouter des informations (par exemple, ce que le système ne peut absolument pas faire) peut aider à réduire le risque de reporting de faux positifs.

3.4.2 Checklists soutenant les techniques de test basées sur l'expérience

Le test basé sur une checklist est une technique de test largement utilisée en raison de son adaptabilité, sa simplicité et son efficacité pour garantir la qualité des logiciels. En utilisant des checklists, l'AT s'assure de couvrir tous les aspects essentiels connus d'un élément de test, ce qui évite d'oublier des zones critiques. Elles introduisent également une cohérence entre les cycles de test et entre les différents AT. Lorsqu'il utilise une checklist, l'AT se concentre sur les aspects importants. Les checklists sont une forme d'enregistrement des expériences passées de l'AT en matière de défaillances et de défauts. Elles servent de rappel ou de source d'inspiration si l'AT est à court d'idées (par exemple, lors de tests exploratoires). L'AT gagne du temps en réutilisant une checklist standard ou une checklist créée par l'un de ses pairs, mais uniquement si ces checklists sont pertinentes pour l'élément de test. Les checklists contribuent à réduire le travail nécessaire pour documenter les cas de test, ce qui peut être un atout considérable lorsque les exigences et les logiciels sont en constante évolution.

La préparation des checklists relève souvent de la responsabilité de l'AT. Les checklists soutiennent les

tests basés sur l'expérience, car elles aident à organiser, structurer et guider les tests. La création d'une checklist réutilisable, maintenable, claire et efficace demande des efforts.

Les étapes suivantes peuvent servir de lignes directrices pour établir une checklist appropriée pour les tests basés sur l'expérience :

Tout d'abord, l'AT détermine le périmètre, les objectifs et le format de la checklist, car ceux-ci influencent la profondeur des tests et le niveau de détail requis. Une checklist « lire-faire » contient les principaux éléments à prendre en compte pour un certain processus (par exemple, les éléments de la checklist contiennent des entrées invalides concrètes pour un champ de saisie à vérifier). Une checklist « faire-confirmer » sert d'aide pour guider le processus de réflexion, en fournissant des idées de tests basés sur l'expérience pour explorer plus en détail une application (par exemple, un élément de la checklist peut nécessiter de vérifier si les résultats de la recherche sont pertinents).

Ensuite, l'AT collecte les informations nécessaires pour définir les éléments de la checklist. Cela peut inclure la collecte d'informations auprès de professionnels expérimentés, la consultation de bibliothèques de défauts et de taxonomies des défauts (Beizer, 1990), (Kaner ; Falk, et al., 1999), la revue de la documentation pertinente et l'analyse des risques, des cas de test et des scénarios potentiels. Les éléments de la checklist doivent être clairs, spécifiques, sans ambiguïté, cohérents, pertinents, maintenables, exploitables et mesurables. Ils doivent être formulés sous forme de questions auxquelles on peut répondre par « oui », « non » ou « sans objet ». Ils doivent être classés par priorité en fonction de leur importance, de leur impact potentiel et de leur niveau de risque. Les checklists ne sont pas des guides pratiques exhaustifs. Il s'agit plutôt d'outils simples et rapides destinés à des professionnels expérimentés.

Enfin, l'AT structure et organise la checklist en classant les éléments de la checklist en groupes logiques basés sur les domaines fonctionnels, les rôles des utilisateurs, les niveaux de test ou d'autres critères pertinents. La création de catégories distinctes peut être particulièrement utile pour une checklist longue.

Dans la mesure du possible, il convient d'utiliser des templates / gabarits et des normes. L'AT peut gagner du temps en utilisant des templates existants ou des checklists prédéfinies conformes aux normes et aux meilleures pratiques du secteur.

Une checklist n'est jamais définitive. L'AT la révise et la remanie en permanence, en l'adaptant pour refléter les nouvelles constatations, les changements de priorités, les commentaires des autres testeurs ou les leçons apprises lors des cycles de test précédents. En partageant la checklist avec les autres testeurs, l'AT favorise la cohérence et la collaboration et les aide à mieux comprendre les éléments de test et les domaines critiques sur lesquels se concentrer pendant les tests.

3.4.3 Crowd Testing

Le crowd testing distribue les tests parmi un groupe de testeurs internes ou externes issus de divers horizons et situés à différents endroits. Il peut s'agir d'un moyen rentable de valider l'utilisabilité et qui couvre à la fois les caractéristiques de qualité fonctionnelles et non fonctionnelles (Alyahya, 2020), (Leicht et al., 2017).

Voici quelques-uns des bénéfices du crowd testing :

- **Diversité des environnements** de test. Les testeurs peuvent se trouver dans divers emplacements géographiques et utiliser diverses configurations d'environnement avec une grande variété d'appareils, de navigateurs et de conditions réseau.
- **Plus de flexibilité**. Facilement évolutif pour gérer de nombreux tests en peu de temps.
- **Rentabilité**. Le crowd testing est généralement moins coûteux que le maintien d'une équipe de test interne importante et diversifiée ou le recours à des services de test externes.
- **Retour d'information rapide**. Les testeurs peuvent fournir un retour d'information rapide, ce qui permet d'identifier et de corriger rapidement les défaillances.
- **Perspective réelle de l'utilisateur**. Les testeurs peuvent être des utilisateurs réels de l'application et peuvent mieux fournir des informations sur son expérience utilisateur et son

utilisabilité. Cela peut être particulièrement utile dans les tests d'acceptation des utilisateurs.

- **Variabilité**. Comme les tests sont exécutés à chaque fois par une grande variété de testeurs, ils ne sont pas aussi reproductibles. Bien que cela puisse être une limitation, cela se traduit également par une couverture plus large, ce qui augmente les chances de trouver des défauts.

Certaines des limitations du crowd testing sont les suivantes :

- **Qualité des tests peu fiable**. La qualité des tests peut varier considérablement en fonction des compétences des testeurs individuels, bien que cela puisse ne pas être pertinent, par exemple, lorsque l'objectif est d'obtenir un retour d'information sur l'expérience utilisateur.
- **Difficultés de communication**. La coordination entre de nombreux testeurs situés dans différents endroits, avec des fuseaux horaires, des différences culturelles et des barrières linguistiques, peut s'avérer difficile.
- **Sécurité**. Le partage de logiciels avec des testeurs externes présente des risques pour la sécurité et la confidentialité des données. Des mesures appropriées peuvent atténuer ces risques, permettant une utilisation responsable du crowd testing sans divulguer de détails sensibles ni faciliter le plagiat.
- **Documentation et reporting**. Il peut être difficile de garantir une documentation complète des tests et de gérer un grand nombre de constatations, y compris les doublons et les faux positifs, lorsqu'on a affaire à un groupe de testeurs important et diversifié.

Le crowd testing est une approche qui ne remplace pas les ATs appliquant des techniques de test, mais qui peut augmenter la couverture de divers environnements de test.

3.5 Appliquer les techniques de test les plus appropriées

Les tests doivent être aussi efficaces et efficients que possible dans le contexte donné. À cette fin, l'AT soutient le Test Manager dans la sélection de la ou des techniques de test les plus appropriées. En outre, l'AT peut utiliser l'automatisation pour soutenir les activités de test. Cela comprend l'automatisation de la conception des tests, décrite dans cette section, et le soutien à l'automatisation de l'exécution des tests, décrite dans la section 1.3.6.

3.5.1 Sélectionner les techniques de test pour atténuer les risques produit

Le choix de la ou des techniques de test les plus appropriées est crucial pour une atténuation efficace et efficiente des risques produits et dépend de nombreux facteurs, notamment les suivants.

Les **objectifs du test** (voir ISTQB-CTFL, v4.0.1, Section 1.1.1) spécifient les aspects de l'objet de test à évaluer. Ils guident le choix des techniques de test et dépendent du type de système (par exemple, les tests basés sur le domaine pour les calculs numériques en ingénierie par opposition aux tests par tables de décisions dans la gestion du risque de crédit).

Les **risques produit** sont associés à des défauts potentiels. Ces défauts peuvent être détectés au mieux en utilisant des techniques de test particulières, car la plupart des techniques de test se concentrent sur la détection de types spécifiques de défauts (voir Sections 3.1, 3.2, 3.3 et 3.4 de ce syllabus). Par exemple :

- Les techniques de test basées sur les données peuvent détecter des défauts dans le traitement des données, l'implémentation du domaine, les interfaces utilisateur, les calculs et les combinaisons de paramètres.
- Les techniques de test basées sur le comportement peuvent détecter les défauts liés aux exigences

des utilisateurs, tels que les caractéristiques manquantes, les défauts de communication et les défauts de traitement.

- Les techniques de test basées sur les règles peuvent détecter les défauts dans la logique et les flux de contrôle.

L'analyse des risques aide également à déterminer l'approche de test appropriée, par exemple :

- Critères de sortie basés sur la couverture. Plus le niveau de risque est élevé, plus la couverture peut être rigoureuse (par exemple, toutes les combinaisons dans la couverture combinatoire au lieu de la couverture par paires). Cependant, l'AT doit toujours tenir compte du compromis entre le niveau de couverture et l'effort de test requis.
- Les tests basés sur l'expérience peuvent être utilisés lorsque la définition de la couverture est difficile, lorsque le niveau de risque est faible ou lorsque le calendrier du projet est serré.

Base de test. Si la spécification de l'objet de test utilise des modèles, l'AT peut utiliser des techniques de test basées sur ces modèles. S'il est impossible ou difficile de dériver un oracle de test à partir de la base de test, des techniques de test telles que les tests métamorphiques ou les techniques de test basées sur l'expérience peuvent être appliquées.

La **connaissance des types de défauts** récurrents peut indiquer le choix de techniques de test qui se concentrent sur la détection de ces défauts (par exemple, les tests basés sur une checklist). Si l'on s'attend à découvrir des défauts similaires à ceux des itérations ou des projets précédents, il peut être raisonnable d'utiliser les mêmes techniques de test qui ont déjà fait leurs preuves.

Connaissances et expérience du testeur. Si l'AT n'est pas familier avec une technique de test donnée, il n'est pas recommandé de l'utiliser dans des missions de test critiques. La connaissance du domaine peut également avoir une incidence sur le choix des techniques de test. Par exemple, une connaissance faible ou inexistante du domaine indique que des techniques de test telles que les tests exploratoires peuvent être inefficaces.

Cycle de vie du développement logiciel utilisé. Un modèle de développement séquentiel est propice à l'utilisation de techniques plus formelles. En revanche, un modèle de développement itératif peut être plus approprié pour l'adoption de techniques de test légères (par exemple, des techniques de test basées sur l'expérience) ou lorsque la conception des tests peut être automatisée.

Exigences du client et du contrat. Les contrats peuvent exiger explicitement la réalisation de tests spécifiques (par exemple, en termes de niveaux ou de types de tests), ce qui influence le choix des techniques de test (par exemple, des critères d'acceptation avec un ensemble de scénarios fournis par le client suggèrent l'utilisation d'une technique de test basée sur des scénarios).

Exigences réglementaires. Lorsqu'un projet suit une norme, il peut nécessiter l'utilisation de techniques de test spécifiques. Par exemple, la norme (ISO 26262, 2018) exige l'utilisation de techniques de test telles que la partition d'équivalence, l'analyse des valeurs limites ou l'estimation des erreurs, en fonction du niveau ASIL (Automotive Safety Integrity Level) attribué à un objet de test.

Les **contraintes du projet**, telles que le temps et le budget, peuvent affecter l'utilisation de techniques chronophages ou nécessitant des ressources coûteuses.

Les techniques de test sont souvent combinées pour augmenter l'efficience et l'efficacité de la détection des défauts. Par exemple :

- La BVA peut être utilisée pour les conditions de garde dans les tests de transition d'état.
- Les tests de domaine peuvent être utilisés dans les tests basés sur des scénarios pour déterminer

la valeur d'une condition à partir d'une table de décisions ou d'une variable apparaissant dans un système sous test.

- Les tests basés sur des scénarios peuvent être combinés avec la couverture des décisions, une technique de test boîte blanche abordée dans (ISTQB-TTA, v4.0), afin de couvrir rigoureusement les décisions dans le processus métier. Pour un exemple, voir les tests de cycle de processus dans (Koomen et al., 2006).
- Les tests basés sur des scénarios peuvent être combinés avec la couverture aller-retour afin de traiter les risques spécifiques des activités cycliques des processus métier.

3.5.2 Bénéfices et risques de l'automatisation de la conception des tests

L'AT peut utiliser des outils pour appliquer des techniques de test, en particulier des techniques de test boîte noire. Lors de l'automatisation de la conception des tests, l'AT crée un modèle de test et génère automatiquement certains éléments du testware à partir de ce modèle. Par exemple, l'AT conçoit un modèle de transition d'état et laisse un outil de test basé sur des modèles générer des cas de test pour une couverture aller-retour.

L'automatisation de la conception des tests améliore souvent l'efficience et l'efficacité des tests. Ses bénéfices incluent :

- **Prévention des défauts**. La modélisation pour les tests est un moyen efficace d'évaluer la qualité de la base de test (voir section 5.2.1).
- **Capacité étendue**. L'automatisation permet d'appliquer des techniques de test et des critères de couverture plus complexes, tels que les tests combinatoires, les tests aléatoires ou la couverture N-switch, réduisant ainsi le risque de code non testé.
- **Compréhensibilité améliorée**. Les critères de sélection des tests spécifiés dans un outil renvoient plus clairement et plus visiblement aux conditions de test et justifient de manière plus compréhensible la couverture générée.
- **Moins de travail répétitif**. Le testware peut être généré à partir du modèle de test, ce qui réduit le travail manuel et répétitif tel que la spécification des tests.
- **Moins d'efforts de maintenance**. Le modèle de test est la seule source de vérité utilisée pour dériver le testware. Par conséquent, seul le modèle de test doit être maintenu.
- **Moins de testware défectueux**. Le travail manuel est sujet aux erreurs. Le testware généré par des outils est de meilleure qualité et plus cohérent.
- **Collaboration améliorée au sein de l'équipe**. Les parties prenantes peuvent réviser le modèle de test pour trouver des défauts ou mieux comprendre les conditions de test.
- **Traçabilité améliorée**. Il est plus facile de relier les éléments d'un modèle de test aux conditions de test qu'aux cas de test eux-mêmes. Si l'outil le soutient, les cas de test générés hériteront de ces liens, améliorant ainsi la traçabilité globale des tests.
- **Différents formats de sortie**. Le testware peut être généré dans différents formats de sortie selon les besoins d'autres outils et des activités ultérieures.

L'AT doit également tenir compte des risques liés à l'automatisation de la conception des tests. Il s'agit notamment de l'oubli de conditions de test qui ne figurent pas dans le modèle, de la sous-estimation de l'effort de maintenance du modèle de test, de la difficulté pour les parties prenantes à comprendre le modèle et des risques génériques liés à l'automatisation des tests (voir ISTQB- CTFL, v4.0.1, section 6.2).

4 Tester les caractéristiques de qualité – 60 minutes

Mots-clés

adaptabilité, compatibilité, flexibilité, pertinence fonctionnelle, complétude fonctionnelle, exactitude fonctionnelle, aptitude fonctionnelle, tests fonctionnels, facilité d'installation, capacité d'interaction, interopérabilité, utilisabilité, expérience utilisateur

Objectifs d'apprentissage pour le chapitre 4:

4.1 Test fonctionnel

TA-4.1.1 (K2) Distinguer les tests d'exactitude fonctionnelle, de pertinence fonctionnelle et de complétude fonctionnelle

4.2 Test d'utilisabilité

TA-4.2.1 (K2) Expliquez comment l'analyste de test contribue aux tests d'utilisabilité

4.3 Test de flexibilité

TA-4.3.1 (K2) Expliquer comment l'analyste de test contribue aux tests d'adaptabilité et de facilité d'installation

4.4 Test de Compatibilité

TA-4.4.1 (K2) Expliquer comment l'analyste de test contribue aux tests d'interopérabilité

Introduction au test des caractéristiques de qualité

Ce syllabus utilise le modèle de qualité logicielle fourni dans la norme ISO 25010 (ISO/IEC 25010, 2023) comme guide et traite des caractéristiques de qualité qui sont ciblées par l'AT. Cependant, la terminologie de l'utilisabilité utilisée diffère de celle de cette norme afin d'être cohérente avec le syllabus Tests d'utilisabilité (ISTQB-UT, v1.0).

L'annexe F donne un aperçu du modèle de qualité de la norme ISO 25010 actuelle, des changements par rapport à la version précédente et des syllabi ISTQB® connexes qui se concentrent sur le test de caractéristiques de qualité spécifiques.

4.1 Test fonctionnel

Le test fonctionnel constitue l'une des principales tâches de l'AT. Alors que (ISTQB-CTFL, v4.0.1) résume brièvement le test fonctionnel comme un type de test, ce syllabus entre davantage dans les détails et aborde les sous-caractéristiques de l'aptitude fonctionnelle.

4.1.1 Sous-caractéristiques de l'aptitude fonctionnelle

Le modèle de qualité des produits de la norme ISO 25010 (ISO/IEC 25010, 2023) distingue trois sous-caractéristiques de l'aptitude fonctionnelle. Bien que toutes puissent être évaluées par des tests fonctionnels, toutes les activités de test fonctionnel ne les traitent pas aussi bien. L'AT doit être en mesure de sélectionner les niveaux de test et les techniques de test appropriés pour traiter les risques produit liés à une sous-caractéristique spécifique de l'aptitude fonctionnelle.

Les tests de complétude fonctionnelle doivent couvrir toutes les tâches spécifiées du logiciel et les objectifs des utilisateurs prévus. La question clé est de savoir si tout ce qui est demandé est implémenté.

La complétude fonctionnelle doit être abordée dès que possible en révisant la spécification des exigences dans les modèles de développement séquentiel et en discutant des User Stories, y compris les critères d'acceptation, lors de la rédaction collaborative des User Stories dans le développement logiciel en mode Agile. Dans les tests système, les tests d'intégration système et les tests d'acceptation, la complétude fonctionnelle peut être testée de manière dynamique.

Les techniques de test basées sur le comportement, telles que les tests basés sur des scénarios, sont bien adaptées, bien que d'autres techniques de test boîte noire soient également appropriées. La traçabilité entre la base de test, les conditions de test et les cas de test est essentielle pour déterminer le niveau de complétude fonctionnelle atteint.

Les tests d'exactitude fonctionnelle permettent de déterminer si les résultats réels sont corrects (par exemple, exacts, précis et cohérents) pour des entrées valides et non valides. Il est essentiel de trouver un oracle de test efficace qui fournisse les résultats attendus en détail.

L'exactitude fonctionnelle peut être testée à n'importe quel niveau de test. En termes de shift left, la plupart des tests d'exactitude fonctionnelle doivent avoir lieu lors des tests de composants et des tests d'intégration des composants. Même si l'AT n'est pas responsable de ces niveaux de test, il doit y contribuer afin d'atteindre au mieux les objectifs de test.

Toutes les techniques de test boîte noire, les techniques de test basées sur l'expérience et les tests basés sur la collaboration sont adaptés.

Les tests de pertinence fonctionnelle vérifient que les fonctions facilitent la réalisation des tâches et des objectifs spécifiés. L'accent est mis sur la question de savoir si tout ce qui est implémenté répond aux besoins des utilisateurs.

Les tests de pertinence fonctionnelle peuvent inclure des revues de la conception de l'interface utilisateur, en particulier pour les applications interactives. Les tests dynamiques commencent par des tests système et des tests d'acceptation dans les modèles de développement séquentiel, ainsi que par des sessions de démonstration dans le développement logiciel en mode Agile.

Les tests exploratoires et les tests basés sur la collaboration sont les plus appropriés. En outre, les techniques de test boîte noire basées sur le comportement sont également adaptées.

4.2 Test d'utilisabilité

L'utilisabilité fait référence à un concept large de caractéristiques de qualité liées à l'utilisateur, couvrant la capacité d'interaction du modèle de qualité des produits (ISO/IEC 25010, 2023) et l'utilité du modèle de qualité en utilisation ISO 25019 (ISO/IEC 25019, 2023). Vous trouverez plus d'informations sur les tests d'utilisabilité dans (ISTQB-UT, v1.0), (ISO 9241-210, 2019) et (UXQB-FL, v4.01).

4.2.1 Contribution de l'analyste de test aux tests d'utilisabilité

Les tests d'utilisabilité se concentrent généralement sur l'évaluation des aspects suivants :

- capacité d'interaction - permettre aux utilisateurs d'accomplir des tâches dans des contextes d'utilisation spécifiques (ISO/IEC 25010, 2023) de manière efficace, efficiente et satisfaisante
- expérience utilisateur - prendre en compte les perceptions des utilisateurs avant, pendant et après leur interaction avec l'objet de test
- accessibilité - garantir que les utilisateurs handicapés, issus de milieux culturels divers ou confrontés à des barrières linguistiques peuvent utiliser le système de manière efficace et efficiente

L'AT peut collaborer aux tests d'utilisabilité dès les premières phases en utilisant ses connaissances des groupes d'utilisateurs cibles, de leurs objectifs, de leur contexte d'utilisation, des difficultés potentielles d'utilisation du système ou de l'expérience utilisateur négative.

L'AT peut contribuer aux principales techniques d'évaluation de l'utilisabilité comme suit :

- **Les revues d'utilisabilité** sont effectuées par des experts en utilisabilité afin d'identifier les problèmes potentiels d'utilisabilité et les écarts par rapport aux critères établis. Les revues d'utilisabilité peuvent varier de revues informelles à des inspections. L'AT peut adapter les critères de revue aux besoins spécifiques des groupes d'utilisateurs, aux objectifs et priorités particuliers du métier et au contexte d'utilisation (par exemple, en adaptant une checklist générique d'utilisabilité).
- **Les sessions de test d'utilisabilité** impliquent les futurs utilisateurs ou leurs représentants qui tentent de résoudre des tâches prédéfinies afin d'évaluer si les tâches peuvent être accomplies de manière efficace, efficiente et satisfaisante. L'AT peut contribuer à la conception de scénarios pour les sessions de test d'utilisabilité en fonction des personae, des groupes d'utilisateurs ou des profils opérationnels.
- **Les questionnaires ou enquêtes auprès des utilisateurs**, comprenant des évaluations et des commentaires, mesurent la satisfaction des utilisateurs. Parmi les exemples de questionnaires destinés aux utilisateurs, on peut citer SUMI (Software Usability Measurement Inventory, 1991) et WAMMI (Website Analysis and Measurement Inventory, 1999). L'AT peut aider à concevoir un questionnaire et à évaluer les réponses afin de répondre aux objectifs spécifiques des utilisateurs ciblés dans leur contexte d'utilisation.

Dans les tests d'accessibilité, un objectif courant est de vérifier la conformité aux normes. La norme internationale « Directives sur l'accessibilité du contenu Web » (WCAG, 2023) définit trois niveaux de conformité (A, AA et AAA) qui représentent des degrés croissants d'accessibilité du contenu Web. Les normes nationales comprennent l'Equality Act du Royaume-Uni (gouvernement britannique, 2010) et l'Americans Disabilities Act des États-Unis (ministère américain de la Justice, 2010). L'AT peut identifier le niveau de conformité requis et les besoins spécifiques du groupe cible visé en analysant le contexte d'utilisation.

4.3 Test de flexibilité

Le test de flexibilité (également appelé test de portabilité) vérifie que l'objet de test peut être adapté aux changements de son contexte d'utilisation ou de son environnement système. Le modèle de qualité des produits ISO 25010 (ISO/IEC 25010, 2023) distingue les sous-caractéristiques suivantes de la flexibilité :

- adaptabilité
- passage à l'échelle
- facilité d'installation
- facilité de remplacement

La flexibilité comporte des aspects techniques et non techniques. Cette section se concentre sur la manière dont l'AT peut contribuer aux tests d'adaptabilité et de facilité d'installation. Le passage à l'échelle et la facilité de remplacement sont liés à des aspects techniques et sont abordés respectivement dans (ISTQB-PT, v1.0) et (ISTQB-TTA, v4.0).

4.3.1 Contribution de l'analyste de test à l'évaluation de l'adaptabilité et de la facilité d'installation.

Le test d'adaptabilité vérifie que l'objet de test peut être adapté ou transféré vers le matériel, le logiciel ou tout autre environnement opérationnel ou d'utilisation prévu.

L'AT soutient les tests d'adaptabilité en identifiant les environnements cibles prévus (par exemple, les versions des systèmes d'exploitation mobiles supportés et les versions des navigateurs pouvant être utilisés) et en concevant des tests qui couvrent les combinaisons de ces environnements. Comme cela nécessite des données de test représentant diverses configurations de paramètres d'environnement, des techniques de test telles que les tests combinatoires sont souvent utilisées (voir section 3.1.2). Un autre exemple est l'intégration de divers composants de plate-forme dans des projets clients afin de garantir la compatibilité entre les environnements cibles. En fonction du risque produit, l'AT conçoit et exécute des smoke tests ou une suite de tests plus complète afin de vérifier que l'objet de test est correctement adapté à l'environnement cible.

En suivant les bonnes pratiques en matière de tests d'adaptabilité (par exemple, définir les environnements cibles dès le début, utiliser des tests combinatoires, effectuer des smoke tests sur les nouveaux environnements et assurer le suivi des défauts spécifiques à l'environnement), l'AT peut découvrir des défauts susceptibles de limiter la durée de vie et la facilité d'utilisation du logiciel (par exemple, la facilité d'utilisation sur différentes tailles d'écran). Le travail de l'AT dans le domaine des tests d'adaptabilité doit également être soutenu par des tests automatisés multiplateformes implémentés par des ingénieurs en automatisation des tests. Des tests d'adaptabilité rigoureux permettent de détecter rapidement les problèmes spécifiques à l'environnement, ce qui contribue à éviter les mises à jour ou les refontes majeures fréquentes après le déploiement et à réduire les coûts de maintenance.

Le test de facilité d'installation vérifie que l'objet de test peut être installé, désinstallé, mis à jour et reconfiguré correctement dans des environnements spécifiés. Le test de facilité d'installation va au-delà

de la simple vérification de l'exécution complète de la procédure d'installation.

Les objectifs typiques du test de facilité d'installation pour l'AT sont les suivants :

- Vérifier que les procédures d'installation sont exécutées correctement sous différentes configurations de paramètres d'environnement. Cela rend utiles les techniques de test telles que les tests combinatoires, similaires aux tests d'adaptabilité.
- Concevoir et exécuter des tests pour déterminer si l'objet de test fonctionne correctement après l'installation ou la mise à jour
- Vérifier la facilité d'installation, de désinstallation ou de mise à jour du logiciel par les utilisateurs. Cela inclut la revue de la documentation d'installation.
- Tester le comportement lié aux autorisations, en particulier pour les applications mobiles (voir ISTQB-MAT, v1.0).

4.4 Test de compatibilité

Le test de compatibilité vérifie si un objet de test est compatible avec d'autres composants ou systèmes lorsqu'il est utilisé. Le modèle de qualité des produits ISO 25010 (ISO/IEC 25010, 2023) distingue deux sous-caractéristiques de la compatibilité : l'interopérabilité et la coexistence. Les tests de compatibilité peuvent donc être subdivisés en plusieurs types de tests :

- Le test d'interopérabilité, qui vérifie la compatibilité avec les composants ou les systèmes avec lesquels l'objet de test est destiné à interagir. Il comprend généralement des tests fonctionnels boîte noire, ils relèvent donc généralement de la responsabilité de l'AT.
- Le test de coexistence, qui vérifie que l'objet de test peut partager son environnement cible avec d'autres composants ou systèmes sans interférence. Ce type de test technique est abordé dans le syllabus Analyste Technique de Test (ISTQB-TTA, v4.0).

4.4.1 Contribution de l'analyste de test à l'interopérabilité

L'objectif du test d'interopérabilité est de vérifier que deux ou plusieurs composants ou systèmes peuvent échanger des informations et utiliser mutuellement les informations qui ont été échangées. Lorsque l'échange d'informations implique une transformation des données, le test d'interopérabilité doit inclure la vérification de cette transformation.

Le test d'interopérabilité est particulièrement important lorsque plusieurs systèmes doivent collaborer, partager des données ou effectuer des tâches ensemble. C'est notamment le cas dans les architectures logicielles modernes telles que les solutions cloud, les Web services, les microservices, la conteneurisation et l'Internet des objets.

L'interopérabilité se produit généralement à différents niveaux architecturaux. L'AT doit comprendre les interactions possibles afin de définir les conditions de test appropriées pour les couvrir. Toutes les interactions ne peuvent pas être documentées. L'AT peut récupérer indirectement des informations sur les interactions à partir de la documentation relative à l'architecture et à la conception. Il est donc essentiel de comprendre cette documentation afin de s'assurer que tous les aspects importants des interactions seront testés.

Le test d'interopérabilité peut détecter des défauts dans :

- les transformations de données pour l'échange de données
- l'interprétation ou l'utilisation des données échangées
- les flux de communication et les protocoles
- la conformité aux normes

- la fonctionnalité de bout en bout
- la documentation de conception

Le test d'interopérabilité est généralement appliqué lors des tests d'intégration. Les techniques de test boîte noire, telles que les techniques de test basées sur les données qui se concentrent sur les données échangées, les techniques de test basées sur le comportement pour interpréter ou utiliser les données, et les techniques de test de fonctionnalité de bout en bout ou basées sur des règles pour la transformation des données sont bien adaptées aux tests d'interopérabilité. Les techniques de test basées sur l'expérience peuvent utilement compléter les tests boîte noire. Les cas de test issus des tests fonctionnels ou des tests d'adaptabilité peuvent être réutilisés pour les tests d'interopérabilité.

Exemples de normes générales en matière d'interopérabilité : (ISO 15745, 2003) et (ISO 16100, 2009). L'ETSI (ETSI EG 202 237 v1.2.1, 2010) fournit un exemple de méthodologie concrète de test d'interopérabilité dans le domaine des télécommunications.

5 Prévention des défauts logiciels – 225 minutes

Mots-clés

revue ad-hoc, revue basée sur une checklist, prévention des défauts, tests basés sur des modèles, lecture basée sur la perspective, technique de revue, revue basée sur les rôles, analyse des causes racines, revue basée sur des scénarios, résultat de test

Objectifs d'apprentissage pour le chapitre 5 :

5.1 Pratiques de prévention des défauts

TA-5.1.1 (K2) Expliquer comment l'analyste test peut contribuer à la prévention des défauts

5.2 Soutien au confinement de phase

TA-5.2.1 (K3) Utiliser un modèle de l'objet de test pour détecter les défauts dans une spécification

TA-5.2.2 (K3) Appliquer une technique de revue à une base de test pour trouver des défauts

5.3 Atténuer la récurrence des défauts

TA-5.3.1 (K4) Analyser les résultats des tests afin d'identifier les améliorations potentielles en matière de détection des défauts

TA-5.3.2 (K2) Expliquer comment la classification des défauts soutient l'analyse des causes racines

Introduction à la prévention des défauts logiciels

L'objectif de la prévention des défauts est d'implémenter des actions qui réduisent la probabilité de (ré)apparition de défauts dans les produits d'activités et atténuent la propagation des défauts aux phases suivantes du SDLC. Ces efforts se traduisent par divers bénéfices importants, notamment une réduction des coûts et de la main-d'œuvre, une augmentation de la productivité et une amélioration de la qualité des produits. La prévention des défauts est la responsabilité de toute l'équipe. L'AT peut utiliser ses connaissances et son expérience spécifiques pour y contribuer.
Les pratiques de prévention des défauts comprennent :

- La prévention de l'introduction de défauts, qui fait partie des activités d'assurance qualité
- La prévention de la propagation des défauts aux phases suivantes du SDLC (voir section 5.2)
- La prévention de la récurrence des défauts (voir section 5.3)

5.1 Pratiques de prévention des défauts

5.1.1 Contribution de l'analyste de test à la prévention des défauts

L'AT peut contribuer à la prévention des défauts de plusieurs façons, en utilisant ses connaissances du domaine, son expertise en matière de tests et ses compétences analytiques. Voici quelques exemples :

- Participer à l'analyse des risques – en veillant à ce que les risques identifiés soient correctement atténués (par exemple, en sélectionnant les techniques de test les plus adéquates).
- Revoir les exigences, les modèles et les spécifications – permet de détecter rapidement les défauts dans la base de test et d'empêcher qu'ils ne se retrouvent dans le code, réduisant ainsi considérablement le coût de leur correction.
- Participer à des rétrospectives – permet l'identification d'améliorations potentielles dans l'analyse de test, la conception des tests, l'implémentation des tests et l'exécution des tests (par exemple, en utilisant des techniques de test plus efficaces, en ciblant les tests sur des zones à risque spécifiques ou en améliorant les données de test et les environnements de test afin de réduire à la fois les résultats faux positifs et les résultats faux négatifs) pour mieux empêcher les défauts de passer inaperçus.
- Collecter et évaluer des données sur les défauts – soutient l'analyse des causes racines et l'amélioration des processus en collectant des données détaillées sur les défauts afin de permettre leur classification et leur analyse statistique, facilitant ainsi l'analyse des causes racines.
- Participer à l'analyse des causes racines – empêche la réapparition des défauts en proposant des mesures correctives pour traiter les causes racines identifiées.

En plus de participer à la prévention des défauts, l'AT évalue (généralement en consultation avec le Test Manager) si les mesures proposées ont eu l'effet souhaité. Voici quelques exemples de métriques qui aident à évaluer l'efficacité de ces mesures :

- Efficacité de suppression des défauts (DRE :Defect Removal Efficiency) – mesure le rapport entre les défauts supprimés avant la version et le nombre total de défauts. Le dénominateur (inconnu) peut être estimé ou remplacé par le nombre total de défauts trouvés jusqu'à un certain moment (par exemple, jusqu'à 6 mois après la version). Un DRE élevé indique que moins de

défauts aboutissent en production, ce qui peut impliquer de meilleures pratiques de prévention des défauts. Cependant, le DRE ne fait pas la distinction entre les défauts interceptés par la prévention et ceux détectés.

- Efficacité du confinement de phase (PCE : Phase Containment Effectiveness) – mesure le nombre de défauts introduits et supprimés dans la même phase par rapport au nombre total de défauts introduits dans cette phase. Une PCE élevée indique que moins de défauts passent aux phases ultérieures.
- Coût de la qualité – illustre la relation entre la prévention des défauts, la détection des défauts et les coûts de suppression (voir ISTQB-TM, v3.0, section 3.2.1).

5.2 Soutien au confinement de phase

L'objectif du confinement de phase est de détecter et de supprimer les défauts dans la même phase du SDLC où ils ont été introduits. Dans le développement logiciel en mode Agile, l'approche « shift left » peut être utilisée de manière similaire.

Cette politique réduit le coût de la qualité. La base de test étant une entrée importante pour l'analyse de test et la conception de test, l'AT peut contribuer au mieux au confinement de phase en évaluant la qualité de la base de test.

Une attention précoce à la qualité de la base de test minimisera les efforts ultérieurs et empêchera la propagation des défauts aux phases suivantes. Ce syllabus traite de deux options courantes permettant à l'AT de trouver des défauts dans la base de test : la modélisation à des fins de test et la revue de la base de test en utilisant diverses techniques de revue (ISO/IEC 20246, 2017).

5.2.1 Utiliser des modèles pour détecter les défauts

La modélisation fournit des abstractions d'un système à un certain niveau de précision et de détail. Elle aide les parties prenantes à mieux comprendre le système en cours de développement. Comme nous le verrons ci-dessous, la modélisation peut soutenir le confinement de phase d'au moins trois façons :

- Détection des défauts dans les spécifications
- Détection des défauts dans les modèles
- Détection des défauts dans les objets du test en utilisant des tests basés sur des modèles

Détection des défauts dans les spécifications. Les spécifications sont souvent fournies sous forme de texte informel. L'AT peut représenter formellement ces spécifications en utilisant des modèles. Lors de la création d'un modèle, les conditions de test (par exemple, les exigences) sont cartographiées sur des éléments du modèle et reliées à ceux-ci pour assurer la traçabilité. La formalisation et la visualisation des conditions de test dans un modèle permettent de révéler efficacement les défauts tels que les lacunes, les incohérences ou les ambiguïtés. La force de la modélisation réside dans le fait que la spécification est transformée tandis que les revues la vérifient dans sa forme originale. En conséquence, l'AT contribue également à trouver des solutions appropriées pour les défauts détectés.

Les modèles basés sur les données comprennent les modèles de domaine et les modèles de tests combinatoires. Ils permettent à l'AT de détecter des défauts de domaine tels que des partitions qui se chevauchent, des lacunes dans la couverture du domaine, des partitions vides ou des combinaisons de paramètres manquantes ou incorrectes.

Les modèles basés sur le comportement comprennent les matrices CRUD, les modèles basés sur l'état ou les modèles de scénario. Ils permettent à l'AT de détecter des cycles de vie d'entités incomplets ou incohérents, des transitions d'état manquantes ou défectueuses, des blocages, des boucles sans fin, un comportement système ambigu ou incohérent, ou une gestion des exceptions manquante.

Les modèles basés sur des règles, tels que les tables de décision ou les relations métamorphiques, permettent à l'AT de détecter des défauts dans les règles métier, tels que des omissions, des incohérences, des ambiguïtés, des redondances, des scénarios sujets à erreur ou une logique métier complexe.

La modélisation peut également détecter des défauts tels que des incohérences dans le nommage, les valeurs des données (par exemple, les limites) et les entrées ou sorties. Elle peut également identifier des informations manquantes, incomplètes, ambiguës ou inutiles.

Détection des défauts dans les modèles. Si la base de test contient des modèles, l'AT peut analyser ces modèles pour détecter les défauts. Ce syllabus traite de la détection des défauts dans trois modèles typiques utilisés dans les tests logiciels : les diagrammes de transition d'état (voir section 3.2.2), les diagrammes d'activité (voir section 3.2.3) et les tables de décisions (voir section 3.3.1). Voici quelques exemples de défauts dans les modèles mentionnés ci-dessus :

- diagrammes de transition d'états – états manquants/incorrects ; transitions incorrectes ; conditions de garde ou actions incorrectes ; états redondants ou inaccessibles et comportement non déterministe
- diagrammes d'activité – actions manquantes, inaccessibles ou sans issue ; ordre incorrect des actions ; conditions de garde incorrectes, non exclusives ou incomplètes dans les nœuds de décision ; points de synchronisation manquants ou flux parallèles incorrectement synchronisés pouvant entraîner un comportement indésirable
- tables de décision – règles qui se chevauchent, incohérentes ou irréalisables ; incomplétude (par exemple, combinaisons de conditions manquantes ou actions manquantes pour une combinaison donnée de conditions)

Tous les modèles peuvent également contenir des défauts tels que des erreurs de syntaxe, des fautes de frappe, des doublons et un nommage incohérent des éléments du modèle.

Détection des défauts en utilisant des tests basés sur des modèles (MBT). Dans cette approche de test, un outil MBT conçoit et génère des cas de test et d'autres outils de conception de test basés sur un modèle MBT et des critères de sélection de test définis par l'AT. Le MBT est efficace pour détecter les anomalies dans la spécification car il permet une couverture complète et une exploration systématique du comportement attendu de l'objet de test selon le modèle MBT. Les modèles MBT, en particulier les modèles graphiques, favorisent la communication avec les parties prenantes, créant une perception et une compréhension communes de la base de test. Vous trouverez plus de détails sur le MBT dans le syllabus (ISTQB-MBT, v1.1).

5.2.2 Application des techniques de revue

Une base de test bien définie sert de pierre angulaire pour garantir la qualité des produits d'activités et la réussite des projets. La revue de la base de test permet d'identifier et de traiter les défauts à un stade précoce, empêchant ainsi qu'ils ne se propagent aux phases suivantes.

L'AT peut employer diverses techniques de revue pendant la revue individuelle pour identifier les défauts dans la base de test. Sélectionner la technique de revue la plus appropriée peut améliorer l'efficience et l'efficacité de la revue. Cette sélection doit prendre en compte des facteurs tels que les objectifs de la revue, les objectifs du projet, les ressources disponibles, le type de base de test, les risques associés, le

domaine métier et la culture de l'entreprise.

Ci-dessous, ce syllabus traite de cinq techniques de revue couramment utilisées par l'AT.

La **revue ad-hoc** est effectuée de manière informelle par les réviseurs, sans processus structuré. Les réviseurs reçoivent peu ou pas de directives sur la manière dont la tâche doit être effectuée. La revue ad-hoc nécessite peu de préparation et dépend fortement des compétences des réviseurs. Au cours de la revue, les réviseurs lisent la base de test et documentent les anomalies qu'ils rencontrent. Cette technique, si elle n'est pas gérée, peut entraîner un volume élevé de rapports d'anomalies en double provenant de plusieurs réviseurs.

La **revue basée sur une checklist** consiste à évaluer la base de test par rapport à une checklist prédéfinie. Les checklists rappellent aux réviseurs de vérifier des points spécifiques et peuvent dépersonnaliser la revue. Les checklists peuvent être génériques ou spécifiques aux caractéristiques de qualité, aux objectifs de test ou au type de base de test. L'AT adapte la checklist au type de base de test, au niveau de risque ou à la condition de test. Cela garantit que la revue se concentrera sur les aspects les plus pertinents de la base de test. Les checklists doivent être régulièrement mises à jour avec les défauts précédemment manqués. Le fait de tenir les checklists à jour permet d'éviter de négliger les anomalies nouvellement identifiées. Les checklists ne sont pas exhaustives.

Par conséquent, l'AT ne se limite pas à vérifier les éléments répertoriés. Cela maximise la détection des défauts et permet à l'AT de détecter les anomalies que les checklists ne couvrent pas explicitement.

La **revue basée sur des scénarios** consiste à simuler un processus ou une activité afin d'identifier les anomalies et de remanier la base de test. Cette technique de revue est particulièrement efficace lorsque la base de test est présentée sous forme de scénarios, par exemple sous forme de diagrammes de cas d'utilisation ou d'activité. Dans ce cas, les réviseurs peuvent effectuer des « exécutions à blanc » basées sur l'utilisation prévue du produit d'activités. Les scénarios fournissent des lignes directrices précieuses, mais les réviseurs ne sont pas tenus de s'en tenir aux scénarios documentés et doivent également réfléchir au-delà des scénarios afin d'identifier davantage d'anomalies.

La **revue basée sur les rôles** consiste à attribuer des rôles ou des responsabilités spécifiques aux réviseurs. Les rôles typiques sont basés sur des types d'utilisateurs finaux spécifiques (par exemple, utilisateur expérimenté, inexpérimenté, senior ou enfant) ou sur des rôles spécifiques au sein de l'organisation (par exemple, administrateur ou utilisateur régulier). Chaque rôle peut être décrit par un persona (c'est-à-dire un personnage concret mais fictif conçu pour représenter les caractéristiques, les besoins, les objectifs et les préférences d'un groupe particulier d'utilisateurs). En répartissant les responsabilités entre les réviseurs en fonction de leurs rôles, les revues basées sur les rôles permettent aux individus de se concentrer sur des aspects spécifiques de la base de test, garantissant ainsi une couverture complète tout en évitant la duplication des anomalies.

La **lecture basée sur la perspective** implique de réviser la base de test à partir de différentes perspectives ou points de vue (par exemple, concepteur, testeur, responsable marketing, administrateur et utilisateur final). Cela conduit à une revue individuelle plus approfondie avec moins de duplication des anomalies entre les réviseurs. En outre, la lecture basée sur la perspective exige que les réviseurs tentent d'utiliser la base de test en cours de revue pour générer le produit d'activités qu'ils en tireraient. Par exemple, un testeur tenterait de générer des projets de tests d'acceptation basés sur les spécifications des exigences afin de voir si toutes les informations nécessaires sont incluses.

5.3 Atténuer la récurrence des défauts

Un AT peut aider de manière proactive à minimiser la récurrence des défauts dans le logiciel. Ce syllabus aborde deux approches liées à l'atténuation de la récurrence des défauts : l'analyse des résultats des tests pour améliorer l'analyse des tests et la conception des tests, et le soutien à l'analyse des causes racines avec la classification des défauts.

5.3.1 Analyse des résultats des tests pour améliorer la détection des défauts

Les résultats des tests permettent à l'AT d'identifier les défaillances, mais ils lui fournissent également un retour d'information qui l'aide à améliorer l'efficacité de la détection des défauts. Certaines techniques couramment utilisées pour analyser les résultats des tests sont décrites ci-dessous.

Analyse des regroupements de défauts prévus par rapport aux défauts réels. Quelques composants contiennent généralement la plupart des défauts (voir ISTQB-CTFL, v4.0.1, section 1.3). Après les tests, l'AT peut prédire les zones sujettes aux défauts et comparer les regroupements de défauts prévus par rapport aux regroupements de défauts réels. En cas de divergences, des tests plus rigoureux peuvent être appliqués aux zones où plus de défauts que prévu ont été trouvés. Lors de la détermination des regroupements, des critères mesurables doivent garantir la clarté et la cohérence (par exemple, la densité de défauts et la sévérité des défauts). Parmi ces critères, la sévérité des défauts doit jouer un rôle important. Les petits regroupements de défauts critiques ou majeurs sont généralement plus importants (c'est-à-dire nécessitent des tests plus rigoureux) que les grands regroupements de défauts mineurs ou cosmétiques.

Analyse du pourcentage de détection des défauts (DDP). Le DDP est l'une des mesures les plus importantes de l'efficacité des tests pour un niveau de test. Lors du calcul du DDP, le nombre de défauts manqués doit être limité à ceux que le niveau de test considéré aurait pu détecter. Des limites claires pour le comptage des défauts, telles que des limites temporelles (par exemple, les défauts constatés dans un délai défini après la livraison) et des critères d'exclusion (par exemple, les défauts dans des composants tiers ou des environnements clients spécifiques), doivent être établis pour garantir la cohérence. Un DDP faible pour un niveau de test donné indique un pourcentage élevé de défauts manqués, ce qui signifie que la détection des défauts est inefficace. Dans ce cas, l'AT doit analyser les raisons et proposer des mesures pour l'améliorer, en rendant le niveau de test plus ciblé et plus rigoureux.

Il est préférable de diviser le DDP par niveaux de sévérité, car la priorité de réduction des défauts manqués dépend généralement de leur sévérité.

L'analyse de la couverture structurelle évalue dans quelle mesure les tests ont exercé des domaines spécifiques de l'objet de test. L'identification des zones à faible couverture permet à l'AT de cibler les efforts de test dans ces zones. L'augmentation de leur couverture permet de découvrir de nouveaux défauts précédemment manqués. La couverture structurelle, telle que la couverture des instructions, la couverture des branches ou la couverture des neurones, est généralement mesurée à l'aide d'outils de test. Lors de la sélection des zones supplémentaires à couvrir, il convient de tenir compte de leur niveau de risque.

L'analyse des écarts de test évalue dans quelle mesure les tests ont exercé les modifications récentes du code. Cela permet à l'AT de concentrer ses efforts de test supplémentaires dans les domaines particulièrement sujets aux erreurs (c'est-à-dire les nouvelles modifications qui n'ont pas été testées du tout) au lieu de cibler tous les domaines qui ont une faible couverture (par exemple, le code qui n'a pas changé depuis longtemps et qui a été testé pour les versions précédentes).

Analyse du schéma d'apparition des défauts. Le nombre ou la densité des défauts trouvés au cours des phases successives d'un projet (par exemple, les itérations) peuvent être comparés à des schémas décrivant la distribution théorique de ces valeurs dans le temps. Le modèle de Rayleigh (Elsayed, 2021) est un exemple classique de schéma d'arrivée des défauts. Il présente un seul pic et est asymétrique vers la droite, ce qui montre que le nombre attendu de défauts trouvés augmente d'abord avec le temps, puis, après avoir atteint sa valeur maximale, diminue lentement vers zéro. L'analyse d'un tel schéma permet de déduire la qualité des cas de test existants et leur potentiel d'amélioration. Par exemple, supposons que la détection des défauts reste à un niveau constant et faible alors que le schéma suggère qu'elle devrait augmenter. Dans ce cas, cela peut signifier que les tests existants sont trop peu performants et incapables

de détecter des défauts supplémentaires.

Les méthodes analytiques décrites ci-dessus utilisent diverses métriques liées aux défauts, telles que le nombre de défauts ou le DDP. Ces métriques sont calculées sur la base des résultats des tests (par exemple, nombre de tests passés/échoués), des rapports de défauts et des métriques structurelles (par exemple, couverture du code ou densité de défauts). Notez toutefois que ces métriques ne sont pas toujours aussi faciles à lire à partir des résultats des tests qu'elles peuvent le paraître. Par exemple, le nombre de tests échoués n'est pas nécessairement le même que le nombre de défauts détectés par ces tests. Pour calculer le nombre réel de défauts détectés, les résultats du processus de débogage doivent être soigneusement analysés, car la relation entre les résultats des tests et les défauts peut être multiple. Plusieurs tests peuvent détecter le même défaut ou un test peut détecter plusieurs défauts. De plus, la sévérité des défauts peut différer de la criticité des tests, car un cas de test critique peut échouer en raison d'un défaut cosmétique.

5.3.2 Soutien à l'analyse des causes racines grâce à la classification des défauts

L'analyse des causes racines (RCA : Root Cause Analysis ou, en français : ACR) est une technique qui permet d'identifier et de traiter les causes sous-jacentes ou fondamentales d'un défaut plutôt que ses seuls symptômes. L'ACR soutient une approche structurée de l'amélioration de la qualité, et son objectif principal est de prévenir la récurrence des défauts. L'AT utilise diverses techniques pour identifier les causes racines des défauts et des défaillances (par exemple, les taxonomies des défauts, la technique des cinq pourquoi, les diagrammes de cause à effet et l'analyse de Pareto).

L'ACR classique consiste à faire étudier un défaut de manière très détaillée par des experts en la matière après l'avoir résolu. Cependant, il y a généralement de nombreux défauts à analyser. Il serait donc très inefficace et fastidieux de planifier des mesures préventives pour chaque défaut. Une façon d'aborder ce problème consiste à classer les défauts, puis à effectuer l'ACR pour les types de défauts qui se produisent.

La classification des défauts repose sur la reconnaissance du fait que les défauts individuels capturent une grande quantité d'informations sur le processus de développement et le système sous test. La classification des défauts permet à l'AT d'extraire des informations sur divers aspects du processus de développement à partir du défaut et de les transformer en une mesure du processus. Cela donne ensuite un aperçu des types d'erreurs commises pendant le développement, ce qui est utile pour l'amélioration du processus. La classification des défauts comble le fossé entre les statistiques quantitatives sur les défauts et l'ACR qualitative. Pour soutenir efficacement l'ACR, les défauts doivent être classés de manière uniforme tout au long du SDLC, depuis les premiers tests jusqu'à la production.

L'AT doit soutenir son organisation dans la normalisation de la classification des défauts logiciels. Cela améliorera la communication et l'échange d'informations sur les défauts entre les développeurs et les organisations, facilitant ainsi l'ACR.

Voici quelques exemples de méthodes de classification des défauts :

- la classification orthogonale des défauts (ODC) (Chillarege, 1992), qui classe chaque défaut en attributs orthogonaux (c'est-à-dire mutuellement exclusifs), collectés à la fois lorsqu'un défaut est signalé et lorsqu'il est corrigé
- IEEE 1044 (IEEE 1044, 2009), une norme de classification des anomalies logicielles fournissant un ensemble principal d'attributs pour la classification des défaillances et des défauts
- la classification basée sur la sévérité, qui classe les défauts en fonction de leur sévérité (par exemple, critique, majeur, mineur, trivial)
- les modèles de taxonomie des défauts, tels que (Beizer, 1990) ou (Catolino et al., 2019)

Les défauts peuvent également être cartographiés en attributs de qualité en utilisant des modèles de qualité

logicielle tels que (ISO/IEC 25010, 2023) ou le modèle FURPS (Grady et al., 1987).
Vous trouverez plus d'informations sur l'analyse des causes racines dans (ISTQB-ITP, v1.0).

6 Références

Normes

ETSI EG 202 237 v1.2.1: Internet Protocol Testing (IPT), 2010. Standard. European Telecommunications Standards Institute.

IEEE 1044: Standard Classification for Software Anomalies, 2009. Standard. Institute of Electrical and Electronics Engineers.

ISO 15745: Industrial automation systems and integration – Open systems application integration framework, 2003. Standard. International Organization for Standardization.

ISO 16100: Industrial automation systems and integration – Manufacturing software capability profiling for interoperability, 2009. Standard. International Organization for Standardization.

ISO 26262: Road vehicles – functional safety – Part 6: Product development at the software level, 2018. Standard. International Organization for Standardization.

ISO 9241-210: Ergonomics of human-system interaction, part 210: Human-centred design for interactive systems, 2019. Standard. International Organization for Standardization.

ISO/IEC 20246: Software and systems engineering – Work product reviews, 2017. Standard. International Organization for Standardization.

ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model, 2023. Standard. International Organization for Standardization.

ISO/IEC 25019: Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality-in-use model, 2023. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-3: Software testing, Part 3: Test documentation, 2021. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-4: Software testing, Part 4: Test techniques, 2021. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-5: Software testing, Part 5: Keyword-Driven Testing, 2016. Standard. International Organization for Standardization.

OMG® BPMN: Business Process Model and Notation, version 2.0, 2013. Standard. Object Management Group, OMG®.

OMG® DMN: Decision Model and Notation, version 1.5, 2024. 2024-01. Standard. Object Management Group.

OMG® UML: Unified Modeling Language, version 2.5, 2017. Standard. Object Management Group.

RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 2011. Standard. Radio Technical Commission for Aeronautics, RTCA Inc.

Documents ISTQB®

NDT: les références suivantes concernent les documents en anglaise, disponibles sur www.istqb.org mais les versions françaises correspondantes se trouvent, pour une grande partie de ces documents, sur www.cftl.fr, le site du Comité Français des Tests Logiciels.

ISTQB-ACT, ISTQB®, 2019. *Certified Tester Specialist Mobile Acceptance Testing: Syllabus*. Version 1.0.

ISTQB-AI, ISTQB®, 2021. *Certified Tester AI Testing: Syllabus*. Version 1.0.

ISTQB-CTFL, ISTQB®, 2024. *Certified Tester Foundation Level: Syllabus*. Version 4.0.1.

ISTQB-ITP, ISTQB®, 2011. *Certified Tester Expert Level Improving the Test Process: Syllabus*. Version 1.0.

ISTQB-MAT, ISTQB®, 2019. *Certified Tester Specialist Mobile Application Testing: Syllabus*. Version 1.0.

ISTQB-MBT, ISTQB®, 2024. *Certified Tester Model-Based Tester: Syllabus*. Version 1.1.

ISTQB-PT, ISTQB®, 2018. *Certified Tester Performance Testing: Syllabus*. Version 1.0.

ISTQB-TAE, ISTQB®, 2024. *Certified Tester Test Automation Engineering: Syllabus*. Version 2.0.

ISTQB-TM, ISTQB®, 2024. *Certified Tester Advanced Level Test Management: Syllabus*. Version 3.0.

ISTQB-TTA, ISTQB®, 2021. *Certified Tester Advanced Level Technical Test Analyst: Syllabus*. Version 4.0.

ISTQB-UT, ISTQB®, 2018. *Certified Tester Usability Testing: Syllabus*. Version 1.0.

Livres

AMMANN, Paul; OFFUTT, Jeff, 2008. *Introduction to Software Testing*. Cambridge University Press.

BEIZER, Boris, 1990. *Software Testing Techniques (2nd Ed.)* International Thomson Computer Press.

BINDER, Robert V., 2000. *Testing Object-Oriented Systems*. Addison-Wesley.

ELSAYED, Elsayed A., 2021. *Reliability Engineering*. Wiley.

FORGÁCS, Istvan; KOVÁCS, Attila, 2019. *Practical Test Design: Selection of traditional and automated test design techniques*. BCS, The Chartered Institute for IT.

FORGÁCS, Istvan; KOVÁCS, Attila, 2024. *Modern Software Testing Techniques*. Apress.

GRADY, Robert; CASWELL, Deborah, 1987. *Software Metrics: Establishing a Company-wide Program*. Prentice Hall.

HENDRICKSON, Elisabeth, 2013. *Explore It!* Pragmatic Bookshelf.

JORGENSEN, Paul, 2014. *Software Testing. A Craftsman's Approach*. CRC Press.

KANER, Cem; FALK, Jack; NGUYEN, Hung Q., 1999. *Testing Computer Software*. Wiley.

KANER, Cem; PADMANABHAN, Sowmya; HOFFMAN, Douglas, 2013. *The Domain Testing Workbook*. Context Driven Press.

KOOMEN, Tim; AALST, Leo van der; BROEKMAN, Bart; VROON, Michiel, 2006. *TMap Next for result-driven testing*. UTN Publishers.

KUHN, D. Richard; YU, Lei; KACKER, Raghu N., 2013. *Introduction to Combinatorial Testing*. CRC Press.

Articles

ALYAHYA, Sultan, 2020. Crowdsourced Software Testing: A Systematic Literature Review. *Information and Software Technology*. Vol. 127, p. 106363. Available from doi: 10.1016/j.infsof.2020.106363.

ANTONIOL, Giuliano; BRIAND, Lionel; DI PENTA, Massimiliano; LABICHE, Yvan, 2002. A case study using the round-trip strategy for state-based class testing. *Proceedings 13th International Symposium*

on Software Reliability Engineering, pp. 269–279. isbn 0-7695-1763-3. Available from doi: 10.1109/ISSRE.2002.1173268.

ARCURI, Andrea; IQBAL, Muhammad Zohaib; BRIAND, Lionel, 2012. Random Testing: Theoretical Results and Practical Implications. *IEEE Transactions on Software Engineering*. Vol. 38, pp. 258–277. Available from doi: 10.1109/TSE.2011.121.

BARR, Earl; HARMAN, Mark; MCMINN, Phil; SHAHBAZ, Muzammil; YOO, Shin, 2014. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*. Vol. 41, no. 5, pp. 507–525. Available from doi: 10.1109/TSE.2014.2372785.

BOCHMANN, Gregor; PETRENKO, Alexandre, 1994. Protocol Testing: Review of Methods and Relevance for Software Testing. *ISSTA '94: Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*, pp. 109–124. Available from doi: 10.1145/186258.187153.

CATOLINO, Gemma; PALOMBA, Fabio; ZAIDMAN, Andy; FERRUCCI, Filomena, 2019. Not All Bugs Are the Same: Understanding, Characterizing, and Classifying Bug Types. *Journal of Systems and Software*. Vol. 152, pp. 165–181. Available from doi: 10.1016/j.jss.2019.03.002.

CHILLAREGE, R. et al., 1992. Orthogonal Defect Classification - A Concept for In-Process Measurements. *IEEE Transactions on Software Engineering*. Vol. 18, pp. 943–956. Available from doi: 10.1109/ 32.177364.

CHOW, Tsun, 1978. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering*. Vol. 4, pp. 178–187. Available from doi: 10.1109/TSE.1978.231496.

COHEN, D.M.; DALAL, Siddhartha; KAJLA, A.; PATTON, G.C., 1994. The Automatic Efficient Test Generator (AETG) system. *Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering*, pp. 303–309. isbn 0-8186-6665-X. Available from doi: 10.1109/ISSRE.1994.341392.

ENGSTRÖM, Emelie; RUNESON, Per; SKOGLUND, Mats, 2010. A systematic review on regression test selection techniques. *Information and Software Technology*. Vol. 52, pp. 14–30. Available from doi: 10.1016/j.infsof.2009.07.001.

GHAZI, Ahmad Nauman; GARIGAPATI, Ratna; PETERSEN, Kai, 2017. Checklists to Support Test Charter Design in Exploratory Testing. *Agile Processes in Software Engineering and Extreme Programming. XP 2017*. isbn 978-3-319-57632-9. Available from doi: 10.1007/978-3-319-57633-6_17.

HAREL, David, 1987. Statecharts: A Visual Formalism For Complex Systems. *Science of Computer Programming*. Vol. 8, pp. 231–274. Available from doi: 10.1016/0167-6423(87)90035-9.

HUANG, Rubing; SUN, Weifeng; XU, Yinyin; CHEN, Haibo; TOWEY, Dave; XIA, Xin, 2019. A Survey on Adaptive Random Testing. *IEEE Transactions on Software Engineering*. Vol. 47, no. 10, pp. 2052–2083. Available from doi: 10.1109/TSE.2019.2942921.

JENG, Bingchiang; WEYUKER, Elaine, 1994. A Simplified Domain-Testing Strategy. *ACM Transactions on Software Engineering and Methodology*. Vol. 3, pp. 254–270. Available from doi: 10.1145/196092.193171.

JUERGENS, Elmar; PAGANO, Dennis; GOEB, Andreas, 2018. Test Impact Analysis: Detecting Errors Early Despite Large, Long-Running Test Suites. *Whitepaper, CQSE GmbH*.

KUHN, D.; WALLACE, Dolores; JR, A.M., 2004. Software Fault Interactions and Implications for Software Testing. *IEEE Transactions on Software Engineering*. Vol. 30, pp. 418–421. Available from doi: 10.1109/TSE.2004.24.

LEICHT, Niklas; BLOHM, Ivo; LEIMEISTER, Jan Marco, 2017. Leveraging the Power of the Crowd for Software Testing. *IEEE Software*. Vol. 34, pp. 62–69. Available from doi: 10.1109/MS.2017.37.

OFFUTT, A. Jefferson, 1992. Investigations of the software testing coupling effect. *ACM Transactions on Software Engineering and Methodology*. Vol. 1, pp. 5–20.

RECHTBERGER, Vaclav; BURES, Miroslav; AHMED, Bestoun, 2022. Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs. *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Valencia, Spain*, pp. 207–214.

RWEMALIKA, Renaud; KINTIS, Marinos; PAPADAKIS, Mike; LE TRAON, Yves; LORRACH, Pierre, 2019. On the Evolution of Keyword-Driven Test Suites. *12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, pp. 335–345.

SEGURA, Sergio; FRASER, Gordon; SANCHEZ, Ana; RUIZ-CORTÉS, Antonio, 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering*, pp. 46–53.

SEGURA, Sergio; TOWEY, Dave; ZHOU, Zhi Quan; CHEN, Tsong Yueh, 2020. Metamorphic Testing: Testing the Untestable. *IEEE Software*. Vol. 42, no. 9, pp. 805–824.

WU, Huayao; NIE, Changhai; PETKE, Justyna; JIA, Yue; HARMAN, Mark, 2020. An Empirical Comparison of Combinatorial Testing, Random Testing and Adaptive Random Testing. *IEEE Transactions on Software Engineering*. Vol. 46, no. 3, pp. 302–320.

Pages Web

COMMISSION, European, 2016. *General Data Protection Regulation: Regulation (EU) 2016/679* [online]. [visited on 2024-09-15]. Available from: https://commission.europa.eu/law/law-topic/data-protection/legal-framework-eu-data-protection_en?.

COMPUTER SOCIETY, IEEE; JTC 1/SC7, ISO/IEC, 2024. *SE VOCAB: Software and Systems Engineering Vocabulary* [online]. [visited on 2024-09-15]. Available from: https://pascal.computer.org/.

CZERWONKA, Jacek, 2004. *Pairwise Testing: Combinatorial Test Case Generation* [online]. [visited on 2024-09-15]. Available from: www.pairwise.org.

HEALTH, U.S. Department of; SERVICES, Human, 2024. *Health Insurance Portability and Accountability Act: Standards for Privacy of Individually Identifiable Health Information (Privacy Rules)* [online]. [visited on 2024-09-15]. Available from: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html.

IREB-GLOSSARY, IREB®, 2024. *The CPRE Glossary: Core terms of Requirements Engineering* [online]. [visited on 2024-09-15]. Available from: https://glossary.istqb.org.

ISTQB-GLOSSARY, ISTQB®, 2024. *Standard Glossary of Terms Used in Software Testing* [online]. [visited on 2024-09-15]. Available from: https://glossary.istqb.org.

Site of Software Test Design, 2020 [online]. [visited on 2024-09-15]. Available from: https://test-design.org/.

SUMI, 1991. *Software Usability Measurement Inventory: Standard evaluation questionnaire for assessing quality of use* [online]. [visited on 2024-09-15]. Available from: sumi.uxp.ie.

U.S. DEPARTMENT OF JUSTICE, Civil Rights Division, 2010. *Americans with Disabilities Act: Guidance & Resource Materials* [online]. [visited on 2024-09-15]. Available from: www.ada.gov/resources/.

UK GOVERNMENT, Equalities Office, 2010. *Equality Act 2010: guidance: Information and guidance on the Equality Act 2010, including age discrimination and public sector Equality Duty.* [online]. [visited on 2024-09-15]. Available from: www.gov.uk/guidance/equality-act-2010-guidance.

WAMMI, 1999. *Website Analysis and Measurement Inventory: Professional service for analyzing user experience* [online]. [visited on 2024-09-15]. Available from: www.wammi.com.

WCAG, 2023. *Web Content Accessibility Guidelines 2.2: W3C Recommendation* [online]. [visited on 2024-09-15]. Available from: www.w3.org/TR/WCAG22/.

Les références précédentes renvoient à des informations disponibles sur Internet et ailleurs. Bien que ces références aient été vérifiées au moment de la publication de ce syllabus, l'ISTQB® ne peut être tenu responsable si ces références ne sont plus disponibles.

7 Appendice A –Objectifs d'apprentissage/Niveau cognitif de connaissances

Les objectifs d'apprentissage spécifiques de ce syllabus sont indiqués au début de chaque chapitre. Chaque sujet du syllabus sera examiné en fonction de son objectif d'apprentissage.

Les objectifs d'apprentissage commencent par un verbe d'action correspondant à son niveau cognitif de connaissance, comme indiqué ci-dessous.

Niveau 1 : Se souvenir (K1)

Le candidat se souviendra, reconnaîtra et rappellera un terme ou un concept.

Verbes d'action : Se souvenir, reconnaître.

Note : Les syllabi de niveau Avancé n'ont pas d'objectifs d'apprentissage spécifiques de niveau K1. Cependant, le contenu du syllabus dans les chapitres 1 à 5 et les définitions de tous les termes listés comme mots-clés juste en dessous des titres des chapitres doivent être mémorisés (K1), même s'ils ne sont pas explicitement mentionnés dans les objectifs d'apprentissage.

Niveau 2 : Comprendre (K2)
Le candidat peut sélectionner les raisons ou les explications des énoncés liés au sujet, et peut résumer, comparer, classer et donner des exemples pour le concept de test.

Verbes d'action : Classer, comparer, différencier, distinguer, expliquer, donner des exemples, interpréter, résumer.

Exemples	Notes
Faire la différence entre les tests d'exactitude fonctionnelle, de pertinence fonctionnelle et de complétude fonctionnelle.	Recherche des différences entre les concepts.
Expliquer les tests CRUD	
Donner des exemples d'exigences en matière d'environnement de test	
Résumer l'implication de l'analyste de test dans les différents cycles de vie du développement logiciel	

Niveau 3 : Appliquer (K3)
Le candidat peut exécuter une procédure lorsqu'il est confronté à une tâche familière ou sélectionner la procédure correcte et l'appliquer à un contexte donné.

Verbes d'action : Appliquer, implémenter, préparer, utiliser.

Exemples	Notes
Appliquer les tests de domaine	Doit faire référence à une procédure / une technique / un processus, etc.
Préparer des chartes de test pour les tests basés sur des sessions	
Utiliser des tests pilotés par les mots-clés pour développer des scripts de test	Peut être utilisé dans un LO qui veut que le candidat soit capable d'utiliser une technique ou une procédure. Semblable à « appliquer ».

Niveau 4 : Analyser (K4)

Le candidat peut séparer les informations relatives à une procédure ou à une technique en leurs éléments constitutifs pour une meilleure compréhension et peut faire la distinction entre les faits et les déductions. Une application typique consiste à analyser un document, un logiciel ou la situation d'un projet et à proposer des actions appropriées pour résoudre un problème ou une tâche.

Verbes d'action : Analyser, déconstruire, tracer les grandes lignes, hiérarchiser, sélectionner.

Exemples	Notes
Analyser l'impact des changements pour déterminer le périmètre des tests de régression	Examinable uniquement en combinaison avec un objectif mesurable de l'analyse. Doit être de la forme « Analyser X à Y » (ou similaire).
Sélectionner les techniques de test appropriées pour atténuer les risques du produit dans une situation donnée	Requis lorsque la sélection demande une analyse.

Référence

(Pour les niveaux cognitifs des objectifs d'apprentissage)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

8 Appendice B – Matrice de traçabilité des objectifs métier avec les objectifs d'apprentissage.

Cette section énumère la traçabilité entre les objectifs métier et les objectifs d'apprentissage du syllabus Analyste de test au niveau Avancé.

Objectifs métier : Niveau Avancé Analyste de test		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-BO1	Soutenir et effectuer les tests appropriés sur la base du cycle de vie du développement logiciel suivi.	6								
TA-BO2	Appliquer les principes du test basé sur les risques		3							
TA-BO3	Sélectionner et appliquer les techniques de test appropriées pour soutenir la réalisation des objectifs du test			13						
TA-BO4	Fournir une documentation présentant un niveau de détail et de qualité approprié				5					
TA-BO5	Déterminer les types de tests fonctionnels appropriés à effectuer					1				
TA-BO6	Contribuer aux tests non fonctionnels						3			
TA-BO7	Contribuer à la prévention des défauts							5		
TA-BO8	Améliorer l'efficience du processus de test grâce à l'utilisation d'outils.								4	
TA-BO9	Spécifier les exigences en matière d'environnements de test et de données de test.									2

Objectifs métier : Niveau Avancé Analyste de test		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
Numéro LO	Objectifs d'apprentissage (niveau K)									
1	Les tâches de l'Analyste de test dans le processus de test - 225 minutes									
1.1	Les tests dans le cycle de vie du développement logiciel									
TA-1.1.1	Résumer la participation de l'analyste de test dans différents cycles de vie du développement logiciel (K2)	X								
1.2	Participation aux activités de test									
TA-1.2.1	Résumer les tâches effectuées par l'analyste de test dans le cadre de l'analyse de test (K2)	X								
TA-1.2.2	Résumer les tâches effectuées par l'analyste de test dans le cadre de la conception des tests (K2)	X								
TA-1.2.3	Résumer les tâches effectuées par l'analyste de test dans le cadre de l'implémentation des tests ((K2)	X								
TA-1.2.4	Résumer les tâches effectuées par l'analyste de test dans le cadre de l'exécution des tests (K2)	X								
1.3	Tâches liées aux produits d'activités									
TA-1.3.1	Distinguer les scénarios de test de haut niveau des scénarios de test de bas niveau (K2)				X					
TA-1.3.2	Expliquer les critères de qualité pour les cas de test (K2)				X					
TA-1.3.3	Donner des exemples d'exigences relatives à l'environnement de test (K2)				X					X
TA-1.3.4	Expliquer le problème de l'oracle de test et les solutions possibles (K2)			X	X					
TA-1.3.5	Donner des exemples d'exigences relatives aux données de test (K2)				X					X
TA-1.3.6	Utiliser des tests basés sur des mots-clés pour développer des scripts de test (K3)	X							X	
TA-1.3.7	Résumer les types d'outils permettant de gérer le testware (K2)								X	
2	Les tâches de l'Analyste de test dans le test basé sur les risques – 90 minutes									

Objectifs métier : Niveau Avancé Analyste de test		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
2.1	Analyse des risques									
TA-2.1.1	Résumer la contribution de l'analyste de test à l'analyse des risques produit (K2)		X							
2.2	Contrôle des risques									
TA-2.2.1	Analyser l'impact des changements pour déterminer le périmètre des tests de régression (K4)		X						X	
3	Analyse de test et conception des tests – 615 minutes									
3.1	Techniques de test basées sur les données									
TA-3.1.1	Appliquer le test de domaine (K3)			X						
TA-3.1.2	Appliquer le test combinatoire (K3)			X						
TA-3.1.3	Résumer les avantages et les limites du test aléatoire (K2)			X						
3.2	Techniques de test basées sur le comportement									
TA-3.2.1	Expliquer le test CRUD (K2)			X						
TA-3.2.2	Appliquer le test de transition d'état (K3)			X						
TA-3.2.3	Appliquer le test basé sur des scénarios (K3)			X						
3.3	Techniques de test basées sur des règles									
TA-3.3.1	Appliquer le test par table de décision (K3)			X						
TA-3.3.2	Appliquer le test métamorphique (K3)			X						
3.4	Tests basés sur l'expérience									
TA-3.4.1	Préparer des chartes de test pour le test basé sur des sessions (K3)			X						
TA-3.4.2	Préparer des checklists qui soutiennent le test basé sur l'expérience (K3)			X						

Objectifs métier : Niveau Avancé Analyste de test		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-3.4.3	Donnez des exemples des bénéfices et des limites du crowd testing (K2)			X						
3.5	Appliquer les techniques de test les plus appropriées									
TA-3.5.1	Sélectionner les techniques de test appropriées pour atténuer les risques produit dans une situation donnée (K4)		X	X						
TA-3.5.2	Expliquer les bénéfices et les risques de l'automatisation de la conception des tests (K2)								X	
4	Tester les caractéristiques de qualité – 60 minutes									
4.1	Test fonctionnel									
TA-4.1.1	Distinguer les tests d'exactitude fonctionnelle, de pertinence fonctionnelle et de complétude fonctionnelle (K2)					X				
4.2	Test d'utilisabilité									
TA-4.2.1	Expliquez comment l'analyste de test contribue au test d'utilisabilité (K2)						X			
4.3	Test de flexibilité									
TA-4.3.1	Expliquer comment l'analyste de test contribue aux tests d'adaptabilité et d'installabilité (K2)						X			
4.4	Test de compatibilité									
TA-4.4.1	Expliquer comment l'analyste de test contribue au test d'interopérabilité (K2)						X			
5	Prévention des défauts logiciels – 225 minutes									
5.1	Pratiques de prévention des défauts									
TA-5.1.1	Expliquer comment l'analyste de test peut contribuer à la prévention des défauts (K2)							X		
5.2	Soutien au confinement de phase									
TA-5.2.1	Utiliser un modèle de l'objet de test pour détecter les défauts dans une spécification (K3)							X		

Objectifs métier : Niveau Avancé Analyste de test		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-5.2.2	Appliquer une technique de revue à une base de test pour trouver des défauts (K3)							X		
5.3	Atténuer la récurrence des défauts									
TA-5.3.1	Analyser les résultats des tests afin d'identifier les améliorations potentielles en matière de détection des défauts (K4)							X		
TA-5.3.2	Expliquer comment la classification des défauts soutient l'analyse des causes racines (K2)							X		

9 Appendice C – Notes de version

Le syllabus de l'ISTQB® Analyste de test de niveau Avancé v4.0 est une mise à jour majeure. Pour cette raison, il n'y a pas de notes de version détaillées par chapitre et section. Cependant, un résumé des principaux changements est fourni ci-dessous. De plus, dans un document séparé de notes de version, l'ISTQB® fournit une traçabilité détaillée entre les objectifs d'apprentissage des versions v3.1.2 et v4.0 du syllabus Analyste de test de niveau Avancé.

Cette version majeure a apporté les changements suivants :

Structure du syllabus

Tous les objectifs d'apprentissage ont été modifiés pour les rendre atomiques et pour créer une traçabilité univoque entre les objectifs d'apprentissage et le contenu afin d'éviter d'avoir un contenu sans objectif d'apprentissage correspondant. Chaque objectif d'apprentissage est décrit dans une section portant le même numéro (par exemple, TA-1.1.1 est traité dans la section 1.1.1). L'objectif était de rendre cette version plus facile à lire, à comprendre, à apprendre et à traduire, en se concentrant sur l'augmentation de l'utilité pratique et l'équilibre entre les connaissances et les compétences.

Il y a 36 objectifs d'apprentissage dans la v4.0 contre 31 dans la v3.1.2. Plusieurs objectifs d'apprentissage K4 ont été ramenés à K2 ou K3. Dans le cas des objectifs d'apprentissage liés aux techniques de test, la motivation était que l'accent devait être mis sur l'application des techniques de test et non sur l'analyse de la documentation en vue d'une conception des tests plus poussée. Certains objectifs d'apprentissage ont été fusionnés en un seul objectif d'apprentissage. Le tableau ci-dessous présente des statistiques détaillées sur le nombre d'objectifs d'apprentissage et le temps de formation.

Version du syllabus	#K2 LO (temps)	#K3 LO (temps)	#K4 LO (temps)	#Total LO (temps)
actuelle (4.0)	22 (330 min)	11 (660 min)	3 (225 min)	36 (1215 min)
précédente (3.1.2)	16 (240 min)	5 (300 min)	10 (750 min)	31 (1290 min)

Classification des techniques de test

La structure du chapitre 3 reflète une classification plus détaillée des techniques de test par rapport à la v3.1.2. Les techniques de test boîte noire sont désormais classées en trois catégories : basées sur les données, basées sur le comportement et basées sur les règles, en fonction du type de modèle de l'objet de test sous-jacent.

Élargir le périmètre du chapitre 5

L'ancien chapitre 5 (consacré uniquement aux revues) a été élargi pour aborder d'autres formes essentielles de pratiques de prévention des défauts utilisées par l'AT, telles que l'utilisation de modèles pour détecter les défauts dans les spécifications, l'analyse des résultats des tests pour améliorer la détection des défauts, et l'utilisation de la classification des défauts pour soutenir l'analyse des causes racines.

Modification des objectifs d'apprentissage

- Le chapitre 1 a été réorganisé en une section sur les activités de test et une autre sur le testware.
- Le sujet sur les cas de test de haut niveau et les cas de test de bas niveau (ancien TA-1.4.2) a été rétrogradé de K4 à K2 (TA-1.3.1).
- Le sujet K3 sur le test basé sur les risques (anciennement TA-2.1.1) a été divisé en deux, K2 TA-2.1.1 relatif à l'analyse des risques et K4 TA-2.2.1 relatif au contrôle des tests.

- Les partitions d'équivalence (anciennement K4 TA-3.2.1) et l'analyse des valeurs limites (anciennement K4 TA-3.2.2) ont été remplacées par le K3 TA-3.1.1, plus général, sur les tests de domaine.
- Les tests par paires (anciennement K4 TA-3.2.6) et les classifications arborescentes (ancien K2 TA-3.2.5) ont été fusionnés en un seul K3 TA-3.1.2, plus général, sur les tests combinatoires.
- Le test de transition d'état (ancien K4 TA-3.2.4) a été remplacé par le K3 TA-3.2.2, qui se concentre sur la couverture N-switch et aller-retour. Les redondances avec CTFL ont été supprimées.
- Les tests de cas d'utilisation (ancien K4 TA-3.2.7) ont été remplacés par le K3 TA-3.2.3, plus général, sur les tests basés sur des scénarios.
- Les tests par tables de décisions (anciennement K4 TA-3.2.3) ont été rétrogradés au niveau K3 (TA-3.3.1).
- Les tests exploratoires (anciennement K3 TA-3.3.2) ont été remplacés par deux objectifs d'apprentissage distincts : sur la préparation des chartes de test (K3 TA-3.4.1) et sur la préparation des checklists soutenant les tests basés sur l'expérience (K3 TA-3.4.2). Les redondances avec la version actuelle du CTFL v4.0 ont été supprimées.
- Quatre objectifs d'apprentissage relatifs à la comparaison des techniques de test et à l'application de la technique la plus appropriée (anciens : K4 TA-3.2.8, K2 TA-3.3.3, K2 TA-3.4.1, K2 TA-3.3.1) ont été combinés en un seul K4 TA-3.5.1.
- Quatre LO concernant les tests fonctionnels (anciens : K2 TA-4.2.1, K2 TA-4.2.2, K2 TA-4.2.3 et K4 TA-4.2.7) ont été regroupés en un seul : K2 TA-4.1.1. Le sujet a été simplifié parce que les tests fonctionnels sont déjà largement décrits dans le contexte des techniques de test au chapitre 3.
- Les sujets du chapitre 6 (Outils de test) ont été déplacés à la section 1.3, en se concentrant sur des questions plus pratiques. L'ancien K3 TA-6.2.1 « Pour un scénario donné, déterminer les activités appropriées pour un Analyste de test dans un projet de tests pilotés par les mots-clés » a été légèrement modifié en K3 TA-1.3.6 « Utiliser les tests basés sur les mots-clés pour développer des scripts de test ». L'ancien K2 TA-6.3.1 « Expliquer l'utilisation et les types d'outils de test appliqués à la conception des tests, à la préparation des données de test et à l'exécution des tests » a été légèrement modifié en K2 TA-1.3.7 « Résumer les types d'outils appliqués à la gestion des testware ».
- Deux LO similaires sur les revues (anciens K3 TA-5.2.1 et K3 TA-5.2.2) ont été combinés en un seul K3 TA-5.2.2.

Nouveaux sujets

- Critères de qualité pour les cas de test (K2 TA-1.3.2)
- Exigences relatives à l'environnement de test (K2 TA-1.3.3)
- Détermination des oracles de test (K2 TA-1.3.4)
- Exigences relatives aux données de test (K2 TA-1.3.5)
- Tests aléatoires (K2 TA-3.1.3)
- Tests CRUD (K2 TA-3.2.1)
- Tests métamorphiques (K3 TA-3.3.2)
- Crowd testing (K2 TA-3.4.3)

- Bénéfices et risques de l'automatisation de la conception des tests (K2 TA-3.5.2)
- Contributions de l'analyste de test à la prévention des défauts (K2 TA-5.1.1)
- Utiliser des modèles pour détecter les défauts dans les spécifications (K3 TA-5.2.1)
- Analyser les résultats des tests pour améliorer la détection des défauts (K4 TA-5.3.1)
- Soutenir l'analyse des causes racine par la classification des défauts (K2 TA-5.3.2)

Mises à jour des normes

Les changements apportés aux dernières versions des normes internationales relatives à la qualité des logiciels (ISO/IEC 25010, 2023) et aux techniques de test (ISO/IEC/IEEE 29119-4, 2021) ont conduit à l'adaptation du contenu correspondant du syllabus.

10 Appendice D – Liste des Abréviations

Abréviation	Signification
AI (IA : FR)	intelligence artificielle
BPMN	Business Process Model and Notation
BVA	analyse des valeurs limites
CRUD	create, read, update, and delete
CSV	valeur séparée par des virgules
DDP	pourcentage de détection des défauts
DRE	efficience de l'élimination des défauts
EP	partition d'équivalence
GDPR (RGPD : FR)	Règlement général sur la protection des données
JSON	JavaScript Object Notation
MBT	model-based testing
MR	relation métamorphique
MT	test métamorphique
ODC	classification orthogonale des défauts
PCE	efficacité du confinement de phase
ACR (RCA : ENG)	analyse des causes racines
SDLC	cycle de vie du développement logiciel
AT (TA: ENG)	Analyste de test
TTA (ATT : FR)	Analyste Technique de Test
UML	Unified Modeling Language
UX	user experience
XML	Extensible Markup Language

NDT: certains acronymes et leur signification ont été volontairement laissés en anglais en raison de leur usage très répandu constaté dans la communauté professionnelle francophone

11 Appendice E – Termes spécifiques au domaine

Terme	Definition
Matrice CRUD	Une matrice qui indique les types d'action des fonctions sur les entités d'un système.
Sémantique des données	La signification et l'interprétation des données.
Technique des cinq pourquoi	Technique d'interrogation itérative utilisée pour déterminer la cause racine d'un défaut ou d'un problème en répétant cinq fois la question « pourquoi ? », en orientant à chaque fois le « pourquoi » actuel vers la réponse du « pourquoi » précédent.
Analyse de Pareto	Une approche permettant d'identifier les domaines problématiques ou les tâches qui auront le plus de retombées.
Cartographie du parcours de l'utilisateur	Documentation de visualisation de l'expérience utilisateur qui montre les étapes qu'un utilisateur suit dans un processus pour atteindre un objectif.
Recherche utilisateurs	Discipline consistant à se renseigner sur les besoins et les processus de pensée des utilisateurs en étudiant la manière dont ils exécutent des tâches, en observant la façon dont ils interagissent avec un composant ou un système, ou par l'analyse et l'interprétation de données.

12 Appendice F – Modèle de qualité des logiciels

Le tableau ci-dessous présente les caractéristiques de qualité du modèle de qualité des produits ISO/CEI 25010 (ISO/CEI 25010, 2023). Il indique quelles caractéristiques/sous-caractéristiques sont abordées dans ce syllabus (AT/TA) et lesquelles sont couvertes dans d'autres syllabi de l'ISTQB® (Analyste Technique Test (ATT/TTA), Test de Performance (PT), Test d'Utilisabilité (UT), Testeur de logiciels automobiles (AuT), et Testeur de sécurité (SEC)). Si plusieurs syllabus couvrent une caractéristique de qualité, celui qui la couvre le plus en détail est cité en premier. Le tableau compare également le modèle ISO/IEC 25010 actuel avec la version 2011 utilisée dans la version précédente de ce syllabus.

ISO/IEC 25010:2023 (actuel)	**ISO/IEC 25010:2011 (précédent)**	**Notes**	**ISTQB® syllabi**
Aptitude fonctionnelle	**Aptitude fonctionnelle**		**TA**
Complétude fonctionnelle	Complétude fonctionnelle		
Exactitude fonctionnelle	Exactitude fonctionnelle		
Pertinence fonctionnelle	Pertinence fonctionnelle		
Efficacité de performance	**Efficacité de performance**		PT, TTA
Comportement temporel	Comportement temporel		
Utilisation des ressources	Utilisation des ressources		
Capacité	Capacité		
Compatibilité	**Compatibilité**		**TA**, TTA
Co-existence	Co-existence		TTA
Interopérabilité	Interopérabilité		**TA**
Capacité d'interaction	**Utilisabilité**	**Renommé**	UT, **TA**
Reconnaissabilité	Reconnaissabilité		
Facilité d'apprentissage	Facilité d'apprentissage		
Opérabilité	Opérabilité		
Protection contre les erreurs utilisateur	Protection contre les erreurs utilisateur		
Engagement des utilisateurs	Esthétique de l'interface utilisateur	Renommé	
Inclusivité	Accessibilité	Divisé et renommé	
Aide aux utilisateurs			
Auto-description		Nouveau	

ISO/IEC 25010:2023 (actuel)	ISO/IEC 25010:2011 (précédent)	Notes	ISTQB® syllabi
Fiabilité	**Fiabilité**		TTA
Absence de fautes	Maturité	Renommé	
Disponibilité	Disponibilité		
Tolérance aux fautes	Tolérance aux défauts		
Récupérabilité	Récupérabilité		
Sécurité	**Sécurité**		SEC, TTA
Confidentialité	Confidentialité		
Integrité	Integrité		
Non-répudiation	Non-répudiation		
Responsabilité	Responsabilité		
Authenticité	Authenticité		
Résistance		Nouveau	
Maintenabilité	**Maintenabilité**		TTA
Modularité	Modularité		
Réutilisabilité	Réutilisabilité		
Analysabilité	Analysabilité		
Modifiabilité	Modifiabilité		
Testabilité	Testabilité		
Flexibilité	**Portabilité**	**Renommé**	TTA, **TA**, PT
Adaptabilité	Adaptabilité		**TA**, TTA
Passage à l'échelle		Nouveau	PT
Facilité d'installation	Facilité d'installation		**TA**, TTA
Remplaçabilité	Remplaçabilité		TTA
Sûreté		**Nouveau**	AuT
Contrainte opérationnelle		Nouveau	
Identification des risques		Nouveau	
Sécurité en cas de défaillance		Nouveau	
Avertissement de danger		Nouveau	
Intégration sécurisée		Nouveau	

13 Appendice G – Marques déposées

ISTQB® est une marque déposée de l'International Software Testing Qualifications Board.

UML® est une marque déposée de Object Management Group (OMG).

BPMN™ est une marque déposée de Object Management Group (OMG).

14 Index