Testeur Certifié
Automatisation des tests - Stratégie
Syllabus

Version 1.0

International Software Testing Qualifications Board

Notice de copyright

Historique de révision

Version	Date	Remarques
Syllabus v1.0	2024/05/03	CT-TAS v1.0 soumise à l'AG
Syllabus v1.0 FR	2024/10/01	Version française

Table des matières

Remerciements

Ce document a été officiellement publié par l'Assemblée Générale de l'ISTQB® le 3 mai 2024.

Il a été produit par le groupe de travail sur l'automatisation des tests du groupe de travail spécialisé de l'International Software Testing Qualifications Board: Graham Bath (Specialist Working Group Chair) Andrew Pollner (Specialist Working Group Vice Chair and Test Automation Task Force Chair), Péter Földházi, Patrick Quilter, Gergely Ágnecz, László Szikszai. Réviseurs du groupe de travail sur l'automatisation des tests : Armin Beer, Armin Born, Geza Bujdoso, Renzo Cerquozzi, Jan Giesen, Arnika Hryszko, Kari Kakkonen, Gary Mogyorodi, Chris van Bael, Carsten Weise, Marc-Florian Wendland.

Revue technique : Gary Mogyorodi

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

Horváth Ágota, Laura Albert, Prasunkumar Banerjee, Jürgen Beniermann, Armin Born, Piet de Roo, Nicola De Rosa, Dingguofu Ding Guofu, Elizabeta Fourneret, Jan Giesen, Erik Haartmans, Matthias Hamburg, Tobias Horn, Mattijs Kemmink, Ilia Kulakov, Ashish Kumar, Vincenzo Marrazzo, Marton Matyas, Patricia McQuaid, Smitha Mohandas, Ingvar Nordström, Sreeja Padmakumari, Nishan Portoyan, Meile Posthuma, Swapnil Shah, Péter Sótér, Szilard Szell, Richard Taylor, Giancarlo Tomasig, Chris Van Bael, Daniel Van der Zwan, Carsten Weise, Marc-Florian Wendland, Claude Zhang.

0 Introduction

0.1 Objectif de ce Syllabus

Ce syllabus constitue la base de l'International Software Testing Qualifications Board pour l'Automatisation des tests - Stratégie de niveau Spécialiste (CTAL-TAS). L'ISTQB® fournit ce syllabus comme suit :

1. Aux Membres de l'ISTQB®, pour qu'ils le traduisent dans leur langue locale et 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 élaborer des questions d'examen dans leur langue locale, adaptées aux objectifs d'apprentissage de ce syllabus.
3. Aux organismes de formation, pour produire des supports de cours et déterminer les méthodes d'enseignement appropriées.
4. Aux candidats à la certification, pour se préparer à l'examen de certification (soit dans le cadre d'un cours de formation, soit de manière indépendante).
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 de tests pour des livres et des articles.

0.2 L'Automatisation des tests – Stratégie dans le test logiciel

La certification de niveau spécialiste en Automatisation des tests - Stratégie s'adresse à toute personne impliquée dans les tests de logiciels et l'automatisation des tests. Il s'agit notamment des testeurs, des analystes de test, des ingénieurs en automatisation des tests, des consultants en test, des architectes de test, des Test Managers et des développeurs de logiciels. Cette certification spécialisée convient également à toute personne souhaitant acquérir une compréhension de base de l'automatisation des tests, comme les chefs de projet, les responsables qualité, les responsables du développement de logiciels, les analystes métier, les directeurs informatiques et les consultants en management.

Le syllabus de niveau spécialiste en Automatisation des tests - Stratégie (CT-TAS) présente les multiples facteurs qui entrent en jeu lors de la planification de l'automatisation des tests au sein d'une organisation. Les aspects d'implémentation de l'ingénierie technique des méthodes d'automatisation des tests et des meilleures pratiques ne sont pas dans le périmètre de ce syllabus et sont couvertes dans le syllabus de niveau Avancé Automatisation des tests - Ingénierie (CTAL-TAE) de l'ISTQB.

La stratégie d'automatisation des tests aborde les besoins d'automatisation des tests au-delà de ceux qui relèvent de l'implémentation d'outils techniques et des défis d'intégration. Une vue stratégique de l'automatisation des tests fournit une vision de l'implémentation à travers les projets au sein d'une organisation d'une manière systématique et cohérente qui, en fin de compte, teste la valeur pour l'organisation.

0.2 Parcours de carrière des testeurs et des ingénieurs en automatisation des tests

Le programme de l'ISTQB® apporte un soutien aux professionnels du test à tous les stades de leur carrière en leur offrant des connaissances à la fois étendues et approfondies. Les personnes qui obtiennent la certification ISTQB® Automatisation des tests - Stratégie peuvent également être intéressées par la certification Automatisation des tests - Ingénierie (CTAL-TAE).

Les personnes qui obtiennent la certification ISTQB® Automatisation des tests - Ingénierie peuvent également être intéressées par les principaux niveaux Avancé (Analyste de test, 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 ses compétences en matière de pratiques de test dans un environnement en mode Agile pourrait envisager les certifications Testeur technique Agile ou Conduite des tests en mode Agile à l'échelle. La filière Spécialiste offre une plongée approfondie dans les domaines qui ont des approches de test et des activités de test spécifiques, par exemple, dans la stratégie d'automatisation des tests, les tests de performance, les tests de sécurité, les tests d'IA et les tests d'applications mobiles, ou lorsque des connaissances spécifiques au domaine sont requises (par exemple, les tests de logiciels automobiles ou les tests de jeux). Veuillez consulter le site www.istqb.org pour obtenir les dernières informations sur le programme de testeur certifié de l'ISTQB®.

0.3 Objectifs métier

Cette section énumère les objectifs métier attendus d'un candidat ayant obtenu la certification en Automatisation des tests - Stratégie.

Un candidat ayant obtenu la certification en automatisation des tests peut……

TAS-B01	Comprendre les facteurs des logiciels et des systèmes qui influencent le succès de l'automatisation des tests.
TAS-B02	Identifier les coûts et les risques liés à l'implémentation d'une solution d'automatisation des tests.
TAS-B03	Comprendre les rôles et responsabilités des personnes qui contribuent à l'automatisation des tests.
TAS-B04	Planifier l'intégration de l'automatisation des tests à travers les niveaux de test.
TAS-B05	Identifier les considérations stratégiques pour l'implémentation de l'automatisation des tests dans différents modèles de cycle de vie du développement logiciel.
TAS-B06	Comprendre l'applicabilité et la viabilité de l'automatisation des tests.
TAS-B07	Planifier des solutions d'automatisation des tests qui répondent aux besoins de l'organisation.
TAS-B08	Comprendre les stratégies de déploiement de l'automatisation des tests.
TAS-B09	Comprendre les dépendances de l'automatisation des tests vis à vis de l'environnement de test.
TAS-B10	Comprendre les coûts de mise en place et de maintenabilité de l'automatisation des tests.
TAS-B11	Apprendre quelles métriques d'automatisation des tests aident à la prise de décision.
TAS-B12	Identifier comment l'automatisation des tests apporte de la valeur au projet et à l'organisation.
TAS-B13	Identifier les exigences en matière de reporting de l'automatisation des tests pour répondre aux besoins des parties prenantes.
TAS-B14	Définir les activités de transition entre les tests manuels et l'automatisation des tests.
TAS-B15	Définir une stratégie d'automatisation des tests qui garantisse que les projets partagent les actifs et les méthodes afin d'assurer une implémentation cohérente dans l'ensemble de l'organisation.

0.5 Objectifs d'apprentissage examinables et niveau cognitif de connaissance

Les objectifs d'apprentissage soutiennent les objectifs métier et sont utilisés pour créer les examens de Testeur certifié Automatisation des tests - Stratégie.

En général, tous les contenus de ce syllabus sont examinables aux niveaux K2, K3 et K4, à l'exception de l'Introduction et des Annexes. En d'autres termes, le candidat peut être amené à reconnaître, mémoriser ou rappeler un mot-clé ou un concept mentionné dans l'un des huit chapitres. Les niveaux des objectifs d'apprentissage spécifiques sont indiqués au début de chaque chapitre et classés comme suit :

- K2 : Comprendre
- K3 : Appliquer

Des détails supplémentaires et des exemples d'objectifs d'apprentissage sont donnés dans l'annexe A.

Tous les termes listés comme mots-clés juste en dessous des titres de chapitres doivent être retenus, même s'ils ne sont pas explicitement mentionnés dans les objectifs d'apprentissage.

0.6 L'examen d'automatisation des tests - stratégie

L'examen du certificat d'Automatisation des tests - Stratégie est basé sur ce syllabus. Les réponses aux questions de l'examen peuvent nécessiter l'utilisation d'exigences basées 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 concernant l'examen de certification en Automatisation des tests - Stratégie, se référer au document Structures et règles d'examen v1.1 compatibles avec les syllabus de niveaux Fondation et Avancé et les modules Spécialistes.

Le critère d'entrée pour passer l'examen d'Automatisation des tests - Stratégie est que les candidats aient un intérêt pour les tests de logiciels et l'automatisation des tests. Cependant, il est fortement recommandé aux candidats de

- Disposer d'une expérience minimale en matière de développement et de test de logiciels, par exemple six mois d'expérience en tant qu'ingénieur de test de logiciels ou en tant que développeur de logiciels.
- Suivre un cours accrédité conçu selon les recommandations de l'ISTQB® (accrédité 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 en automatisation des tests de l'ISTQB®.

0.7 Accréditation

Un Membre de l'ISTQB® peut accréditer les organismes de formation dont le matériel de cours suit ce syllabus. Les organismes de formation doivent obtenir les directives d'accréditation auprès du Membre ou de l'organisme qui effectue l'accréditation. Un cours accrédité est reconnu comme étant conforme à ce syllabus et peut comporter un examen de l'ISTQB®.

Les directives d'accréditation pour ce syllabus suivent les directives générales d'accréditation publiées par le groupe de travail sur la Gestion des Processus et la conformité.

0.8 Gestion des normes

Certaines normes sont référencées dans le syllabus de l'Automatisation des tests - Stratégie (par exemple, IEEE et ISO). Le but de ces références est de fournir un framework (comme dans les références à ISO 25010 concernant les caractéristiques de qualité) ou de fournir une source d'information supplémentaire si le lecteur le souhaite. Veuillez noter que le syllabus utilise les documents de normes comme référence. Les documents de normes ne sont pas destinés à l'examen. Pour plus d'informations sur les normes, reportez-vous au chapitre 7 "Références".

0.9 Actualisation

L'industrie du logiciel évolue rapidement. Pour faire face à ces changements et permettre aux parties prenantes d'accéder à des informations pertinentes et actuelles, les groupes de travail de l'ISTQB ont créé des liens sur le site web www.istqb.org, qui renvoient à des documents de support et à des modifications apportées aux normes. Ces informations ne sont pas examinables dans le cadre du syllabus de l'Automatisation des tests - Stratégie.

0.10 Niveau de détail

Le niveau de détail de ce syllabus permet des cours et des examens cohérents au niveau international. Afin d'atteindre cet objectif, le syllabus se compose :

- Des objectifs pédagogiques généraux décrivant l'intention de l'ingénieur en automatisation des tests de niveau Spécialiste.
- Une liste de termes que les étudiants doivent être en mesure de rappeler.
- Des objectifs d'apprentissage pour chaque domaine de connaissance, décrivant le résultat d'apprentissage cognitif à atteindre.
- Une description des concepts clés, y compris des références à des sources telles que la littérature ou les normes reconnues.

Le contenu du syllabus n'est pas une description de l'ensemble du domaine de connaissance des tests de logiciels ; il reflète le niveau de détail à couvrir dans les cours de formation en automatisation des tests. Il se concentre sur les concepts et techniques de test qui peuvent s'appliquer à tous les projets logiciels, y compris ceux qui suivent les méthodes Agile. Ce syllabus ne contient pas d'objectifs d'apprentissage spécifiques liés aux tests en mode Agile, mais il aborde la façon dont ces concepts s'appliquent dans les projets en mode Agile et dans d'autres types de tests.

0.11 Organisation de ce syllabus

Il y a six chapitres dont le contenu peut faire l'objet d'un examen. L'en-tête de chaque chapitre précise la durée du chapitre ; le calendrier n'est pas fourni au-dessous du niveau du chapitre. Pour les formations accréditées, le syllabus exige un minimum de 12,75 heures d'enseignement, réparties sur les six chapitres comme suit :

- Chapitre 1 : 45 minutes - Introduction et objectifs de la stratégie d'automatisation des tests.
  - Le testeur comprend les concepts de l'automatisation des tests et apprend les critères de sélection des tests pour les projets candidats.
  - Le testeur comprend les facteurs qui définissent une implémentation réussie de l'automatisation des tests.
- Chapitre 2 : 60 minutes - Ressources pour l'automatisation des tests
  - Le testeur apprend les différentes solutions qui sont disponibles pour l'automatisation des tests et l'investissement relatif pour chacune d'entre elles.
  - Les licences logicielles pour les outils d'automatisation des tests sont couvertes.
  - Les testeurs comprennent les compétences nécessaires à l'automatisation des tests.
- Chapitre 3 : 225 minutes - Se préparer à l'automatisation des tests.
  - Le testeur apprend comment l'automatisation des tests est utilisée à travers et au sein des niveaux de test.
  - Les stratégies d'automatisation des tests pour distribuer adéquatement les tests et réaliser le shift-left et le shift-right sont couvertes.
  - Le testeur apprend comment l'automatisation des tests soutient les projets patrimoniaux et en mode Agile.
  - L'automatisation des tests dans le cadre de DevOps et des pratiques de tests en continu est couverte.
  - Le testeur comprend comment définir les critères d'utilisation de l'automatisation des tests, y compris les tests les plus adaptés à l'automatisation des tests.
- Chapitre 4 : 135 minutes - Stratégies organisationnelles de déploiement et de version de l'automatisation des tests.
  - Le testeur apprend comment l'automatisation des tests peut améliorer le temps de mise sur le marché.
  - Le testeur comprend comment développer des tests automatisés pertinents d'un point de vue opérationnel et le reporting des défauts.
  - La stratégie d'automatisation des tests et l'atténuation des risques est couverte.
  - Le testeur apprend à connaître l'environnement de l'automatisation des tests et ses dépendances.
  - L'intégration de l'automatisation des tests et des données de test à un système sous test est couverte.

- Chapitre 5 : 150 minutes - Analyse d'impact de l'automatisation des tests
    - Les métriques de l'automatisation des tests et le reporting pour aider à éclairer les décisions sont couverts.
    - Un testeur apprend comment effectuer un retour sur investissement pour l'automatisation des tests.
    - Les objectifs d'une organisation et d'un projet pour utiliser l'automatisation des tests sont couverts.
    - Le testeur apprend comment analyser les rapports de test et informer les décideurs d'une manière claire et compréhensible.
- Chapitre 6 : 150 minutes - Stratégies d'implémentation et d'amélioration de l'automatisation des tests.
    - Le testeur apprend comment passer du test manuel à l'automatisation des tests et aux tests en continu.
    - L'évaluation de l'automatisation des tests pour l'amélioration continue est couverte.

1 Introduction et objectifs de la stratégie d'automatisation des tests – 45 minutes (K2)

Mots clés

architecture d'automatisation des tests, framework d'automatisation des tests, stratégie d'automatisation des tests

Objectifs d'apprentissage pour le chapitre 1:

1.1 Facteurs de succès d'un projet d'automatisation des tests

CT-TAS-1.1.1 (K2) Expliquer les objectifs et la pertinence de l'automatisation des tests.

CT-TAS-1.1.2 (K2) Identifier les facteurs de succès techniques d'un projet d'automatisation des tests.

CT-TAS-1.1.3 (K2) Résumer les critères d'investissement appropriés dans la sélection des projets candidats à l'automatisation des tests.

1.1 1.1 Facteurs de succès d'un projet d'automatisation des tests

1.1.1 Définir les objectifs d'une stratégie d'automatisation des tests

Lors de la définition de la stratégie d'un projet d'automatisation des tests, il convient d'aborder les points suivants :

- Définir l'objectif visé.
- Identifier les risques.
- Définir le périmètre.
- Identifier les parties prenantes concernées.
- Sélectionner les outils d'automatisation.
- Concevoir l'architecture d'automatisation des tests (TAA).
- Identifier les environnements.

Les objectifs de l'automatisation des tests peuvent inclure :

- L'amélioration de l'efficience des tests.
- Une couverture plus large et plus profonde.
- L'amélioration de la qualité globale du SUT.
- La réduction du coût total et du délai de mise sur le marché.
- La réalisation de tests que les testeurs manuels ne peuvent pas effectuer.
- La réduction de la durée d'exécution du test.
- L'augmentation de la fréquence des tests.

Étant donné que le développement logiciel en mode Agile offre des cycles de déploiement plus rapides et que les applications cloud sont plus répandues, le développement exige un retour d'information plus précoce et plus rapide sur la qualité du SUT. En conséquence de ce changement, l'automatisation des tests a plus d'intérêt et de pertinence dans les projets modernes, compte tenu de sa nature et des objectifs de test.

1.1.2 Identifier les facteurs de succès techniques d'un projet d'automatisation des tests

Les facteurs de succès suivants s'appliquent aux projets d'automatisation des tests qui se concentrent sur les facteurs ayant un impact sur le succès à long terme d'un projet. L'utilisation d'un projet pilote permet de déterminer les outils et la viabilité des technologies sélectionnées.

Les facteurs de succès de l'automatisation des tests sont notamment les suivants :

- Testabilité du SUT
  - Permettre à l'automatisation des tests d'accéder aux interfaces du SUT.
- Une stratégie d'automatisation des tests définie.

    - La stratégie doit être applicable et personnalisable ; ses objectifs doivent pouvoir être atteints dans le respect des contraintes de temps et de coût, et elle doit être tenue à jour.
- TAA
    - Clarté de ce qu'il faut implémenter et de la manière de le faire
    - Pour plus d'informations, voir le CTAL-TAE Syllabus, section 3.1.1

- Framework d'automatisation des tests (TAF)
    - Le TAF est facile à utiliser, bien documenté et maintenable. Il soutient une approche cohérente de l'automatisation des tests. Pour plus d'informations, voir le syllabus CTAL-TAE, section 3.1.3.
    - Rapports de tests définis et implémentés.
    - Dépannage facile.
    - Environnement de test approprié.
    - Cas de test automatisés documentés.
    - Traçabilité des tests automatisés par rapport aux définitions et aux exigences des cas de test.
    - Maintenance aisée.
    - Tests automatisés actualisés.
    - Plan de déploiement clair.
    - Plan d'exécution du test clair.
    - Tests retirés au fur et à mesure des besoins.
    - Gestion efficace des exceptions.

Avant de commencer le projet d'automatisation des tests, il est important d'analyser les chances de succès du projet en considérant les facteurs en place et les facteurs manquants, en gardant à l'esprit les risques de l'approche de test choisie ainsi que le contexte du projet. Tous les facteurs ne sont pas exigés et, dans la pratique, il est rare que tous les facteurs soient réunis.

1.1.3 Résumer les critères d'investissement appropriés pour la sélection des projets candidats à l'automatisation des tests

La mise en place de l'automatisation des tests sur un projet implique un investissement et, dans la plupart des cas, un coût important. Il convient de tenir compte de ces éléments, ainsi que de la nature du projet et de l'opportunité d'utiliser l'automatisation des tests dans le cadre du projet.

Il faut prendre en compte les critères d'investissement suivants avant d'introduire l'automatisation des tests :

- Coût de l'introduction : Outre le travail nécessaire pour mettre en place l'automatisation des tests, des exigences supplémentaires seront requises pour le projet, car il peut être nécessaire d'embaucher de nouveaux ingénieurs en automatisation des tests (TAE), d'acheter du nouveau matériel ou de mettre en place une formation.
- Phase actuelle du cycle de vie du développement logiciel (SDLC) du projet : Il est préférable de commencer le plus tôt possible afin que l'automatisation des tests puisse apporter plus de valeur plus rapidement
- Durée prévue/planifiée du projet/développement logiciel : Pour un projet plus court, il se peut qu'il n'y ait pas assez de ressources pour commencer, ou qu'il ne reste pas assez de temps pour que l'automatisation des tests apporte de la valeur.

- Coût de maintenance : dans le cas d'un départ de zéro, la mise en place d'une solution d'automatisation des tests (TAS) prendra du temps, de même que la nécessité d'en assurer la maintenance.

Si l'investissement que représente l'introduction de l'automatisation des tests sur un projet est acceptable, la préparation à l'automatisation des tests peut commencer. Cela inclut la sélection de la bonne approche de test et des outils d'automatisation des tests. La responsabilité première de ce processus incombe à un architecte de test ou à un Test Manager, qui dispose des connaissances nécessaires en matière d'automatisation des tests pour prendre les décisions qui s'imposent.

2 Ressources d'automatisation des tests – 60 minutes (K2)

Mots clés

ingénieur en automatisation des tests (TAE), solution d'automatisation des tests (TAS)

Objectifs d'apprentissage pour le chapitre 2:

2.1 Coûts et risques liés à l'implémentation d'une solution d'automatisation des tests

CT-TAS-2.1.1 (K2) Comparer des solutions techniques alternatives du point de vue du coût de possession.

CT-TAS-2.1.2 (K2) Expliquer les considérations relatives au modèle de licence pour les outils d'automatisation des tests.

CT-TAS-2.1.3 (K2) Donner des exemples de facteurs à prendre en compte lors de la définition d'une stratégie d'automatisation des tests.

2.2 Rôles et responsabilités dans l'automatisation des tests

CT-TAS-2.2.1 (K2) Résumer les rôles et les compétences nécessaires pour une solution d'automatisation des tests réussie.

2.1 Coûts et risques liés à l'implémentation d'une solution d'automatisation des tests

2.1.1 Comparer les solutions techniques alternatives en fonction du coût de possession

En termes de possession, une approche courante est une solution interne personnalisée basée sur des outils open-source ou commerciaux. D'autres approches alternatives incluent des solutions commerciales non personnalisables ou la signature d'un contrat avec une société d'externalisation pour que le travail soit effectué par le TAE.

La création de toutes les TAS en interne garantit que l'ensemble des coûts, des ressources, des risques et de la gouvernance se trouvent au sein de l'organisation (par exemple, les membres de l'équipe/les développeurs et les ressources matérielles et logicielles nécessaires à la solution proprement dite). Un facteur clé est que, en suivant cette approche et en allouant du temps à cette activité, la TAS peut être développée sans qu'il soit nécessaire de passer un contrat avec un fournisseur ou une société d'externalisation, puisque chaque TAE requis et ses connaissances sont disponibles au sein de l'organisation. Toutefois, pour mener à bien cette activité, l'organisation doit disposer des bons TAE, recrutés ou formés, qui peuvent piloter le développement de la TAS.

Dans le cas d'une solution basée sur un fournisseur, la possession sera partagée entre le client et le fournisseur en fonction des détails de leur contrat. S'il existe déjà dans l'organisation un outil de test qui a déjà été testé et qui répond à toutes les exigences, et s'il n'y a pas de besoin de caractéristiques supplémentaires pour cet outil de test, cette approche sera plus facile à adopter. Les testeurs de l'organisation seront en mesure de travailler de manière productive une fois qu'ils auront reçu la formation nécessaire de la part du fournisseur, et il y aura généralement des experts en la matière (SMEs - NDT: Subject Matter Experts) désignés en tant que responsables de produits au sein de l'organisation. Le risque de cette approche est que si des travaux supplémentaires doivent être effectués sur l'outil de test (par exemple, correction des défauts de l'outil et demandes de caractéristiques supplémentaires), cela prendra du temps et nécessitera des négociations avec le fournisseur pour que ces travaux soient effectués à temps et de la bonne manière, ce qui peut affecter le travail des testeurs qui travaillent avec cet outil de test.

Si l'organisation ne souhaite pas mettre en place une équipe propriétaire, l'externalisation est une solution recommandée. En travaillant avec des sociétés externes, le client n'a pas besoin d'embaucher ou d'acheter du matériel ou des logiciels, ni de recruter des employés possédant les exigences requises. Tout le travail et les coûts supplémentaires seront pris en charge par la société externe, de même que les licences liées aux outils. Des attentes et des métriques mesurables doivent être définies dans le contrat afin de rendre les progrès transparents et visibles. Cette approche est suggérée dans le cas où l'organisation ne prévoit pas d'investir des efforts dans l'embauche de TAE en raison de la brièveté du projet, des coûts ou d'autres considérations.

2.1.2 Expliquer les considérations relatives au modèle de licence pour les outils d'automatisation des tests

La problématique des licences est un aspect important à prendre en considération lorsque l'automatisation des tests est mise en place. Chaque modèle de licence mentionné ci-dessous a un

impact différent sur l'utilisabilité et les facteurs de coût, par exemple les coûts d'exécution des tests et la difficulté à mettre en place l'environnement de développement.

Les modèles de licence comprennent :

- **Open-source** : Dans de nombreux cas, les organisations utilisent des outils open-source pour atteindre leurs objectifs de test dans l'automatisation des tests. La raison principale est qu'il n'y a pas de coûts de licence ni de frais de maintenance pour l'utilisation des outils de test. De nombreux utilisateurs et organisations contribuent au développement des outils open-source, ce qui facilite la collecte d'informations et le soutien. La plupart des licences open-source permettent également aux gens de modifier les outils si nécessaire ou même de les republier sans restrictions significatives.
- **Licence par utilisateur/machine** : Les outils commerciaux ont souvent une licence par utilisateur(s) ou par machine(s). En termes de coûts, les outils peuvent être utilisés de manière efficiente lorsque l'organisation connaît le nombre de TAE qui travailleront à long terme sur un projet donné. Souvent, plus une organisation commande de licences, plus elle bénéficie d'une tarification avantageuse.
- **Licence flottante** : Il s'agit d'une licence qui peut être partagée entre plusieurs personnes de l'organisation pour être utilisée à des moments différents. Elle peut être très utile lorsqu'il y a de nombreux TAE qui travaillent sur des machines différentes. Le nombre de licences est calculé en fonction de l'utilisation simultanée, et non du nombre total de TAE ou des machines spécifiques sur lesquelles ils exécutent leurs tests.
- **Licence d'exécution** : De nombreux outils commerciaux disposent de licences d'exécution pour l'exécution de l'automatisation des tests. Ce type de licence se rencontre chez les fournisseurs de cloud qui hébergent des outils d'automatisation des tests, plusieurs systèmes d'exploitation (OS) et versions de navigateur en tant que service. Il existe également des fournisseurs de cloud qui donnent accès à des fermes d'appareils qui disposent d'une variété d'appareils mobiles, de plateformes, de versions et de réseaux cellulaires. Ceux qui utilisent ces services ne sont facturés que pour le temps qu'ils les utilisent pour exécuter les tests. Ils sont donc facturés sur la base d'une licence d'exécution.

2.1.3 Fournir des exemples de facteurs à prendre en compte lors de la définition d'une stratégie d'automatisation des tests

De nombreux facteurs peuvent influencer les décisions concernant l'implémentation de l'automatisation des tests et sa stratégie.

Les facteurs les plus cruciaux sont :

- Les contraintes de calendrier.
- Le niveau d'expertise nécessaire et le nombre de TAE pour développer la TAS.
- Le matériel de test.
- Les licences d'utilisation des outils de test.
- L'adaptabilité.

- La maintenance.
- Le support pour différentes plateformes (par exemple, Web, ordinateur de bureau et/ou mobile).
- Le support à l'intégration continue/à la livraison continue (CI/CD).
- L'intégration de la gestion des tests et le reporting des tests.

Le premier et le plus important facteur de coût est le calendrier du travail à effectuer et de l'automatisation des tests elle-même.

Si le délai est court, une approche souvent utilisée consiste à n'embaucher que quelques TAE compétents pour créer la TAS. Dans l'optique d'une feuille de route et d'un calendrier plus longs, l'organisation peut décider du nombre de TAE requis avec plus de contexte. Lorsque le SUT se développe, s'améliore et devient plus complexe, la décision peut être prise d'engager davantage de TAE pour implémenter et maintenir la TAS et les tests.

Les outils et leurs licences sont d'autres facteurs de coût importants. Le budget doit tenir compte des exigences en matière d'outils de test, de matériel et de formation supplémentaire pour les TAE. Ces facteurs sont étroitement liés à l'intégration à d'autres outils et systèmes afin d'assurer toutes les fonctions supplémentaires autres que les tests, telles que le chargement des résultats des tests dans le système de gestion des tests ou le déclenchement d'un build dans le système de gestion de la configuration. Pour ces fonctions supplémentaires, il existe des outils, gratuits ou payants, et des bibliothèques à utiliser, mais ceux-ci doivent être pris en compte lors de la création de la stratégie d'automatisation des tests et inclus dans le budget.

Le nombre d'environnements de test et d'agents dont dispose une équipe de développement dépend toujours du contexte. Plus le SUT et le calendrier des versions sont vastes et complexes, plus l'organisation aura besoin de ressources, ce qui augmentera également les coûts. Une approche récente, pour limiter les coûts des ressources, consiste à utiliser des fournisseurs de cloud en payant pour des ressources matérielles de test à la demande et des licences d'exécution. Cette approche doit être utilisée avec précaution car elle peut s'avérer contreproductive. Il arrive que des machines soient laissées en fonctionnement et deviennent plus coûteuses qu'en local, sur un matériel propre. Cette approche peut réduire les coûts élevés de maintenance et d'exploitation pour l'équipe TAE.

2.2 Rôles et responsabilités dans l'automatisation des tests

2.2.1 Résumer les rôles et les compétences nécessaires à la réussite d'une solution d'automatisation des tests

Pour une TAS réussie, les organisations ont besoin de TAE compétents qui devraient avoir de solides connaissances en matière de programmation et d'architecture technique.

En fonction de la taille et de la maturité du projet, il devrait également y avoir au moins un expert du domaine compétent (par exemple, un chef de test (Test Lead), un architecte et un analyste métier) qui comprend le domaine d'activité réel et les objectifs du test. Le rôle de cette personne est d'aider à conduire et à créer le concept, basé sur les objectifs du test, et une feuille de route pour la TAS actuelle. Le management de l'équipe et les compétences non techniques seront nécessaires pour former, motiver et construire l'équipe pour le travail réel. [CTEL-TM-MTT]

Un TAE doit avoir de solides compétences techniques et des connaissances sur les différents SDLC, ainsi que sur l'architecture du SUT et son environnement de développement. Outre les aspects techniques, un TAE doit avoir la capacité de coopérer avec les analystes de test et les autres parties prenantes sur les objectifs de l'automatisation des tests. Étant donné qu'il n'est pas possible de tester de manière exhaustive, il en va de même pour l'automatisation des tests : une couverture à 100 % de l'automatisation des tests n'est pas réalisable. Le temps et les efforts étant limités, les TAE doivent être en mesure de hiérarchiser les conditions de test les plus impactantes à couvrir du point de vue du métier et de l'investissement en contribuant à l'évaluation des risques.

3 Se préparer à l'automatisation des tests – 225 minutes (K3)

Mots clés

Tests d'API, tests de composants, tests de contrats, shift-left, shift-right, système sous test, approche d'automatisation des tests, condition de test, double de test, niveau de test, pyramide des tests.

Objectifs d'apprentissage pour le chapitre 3:

3.1 Intégration aux niveaux de test

CT-TAS-3.1.1 (K2) Différencier les distributions d'automatisation des tests.

CT-TAS-3.1.2 (K2) Sélectionner une approche d'automatisation des tests basée sur l'architecture du système sous test.

CT-TAS-3.1.3 (K3) Démontrer les moyens d'optimiser la répartition de l'automatisation des tests pour réaliser des approches shift-left et shift-right.

3.2 Considérations stratégiques dans les différents modèles de cycle de vie du développement logiciel

CT-TAS-3.2.1 (K2) Expliquer comment les projets d'automatisation des tests se conforment aux anciens modèles de cycle de vie du développement logiciel.

CT-TAS-3.2.2 (K2) Expliquer comment les projets d'automatisation des tests se conforment aux meilleures pratiques de développement logiciel en mode Agile qui soutiennent l'automatisation des tests.

CT-TAS-3.2.3 (K3) Préparer les projets d'automatisation des tests à se conformer aux meilleures pratiques DevOps qui soutiennent l'automatisation des tests en continu.

3.3 Application et viabilité de l'automatisation des tests

CT-TAS-3.3.1 (K2) Expliquer les critères permettant de déterminer la pertinence des tests pour l'automatisation des tests.

CT-TAS-3.3.2 (K2) Identifier les défis que seule l'automatisation des tests peut relever.

CT-TAS-3.3.3 (K2) Identifier les conditions de test difficiles à automatiser.

3.1 Intégration aux niveaux de test

3.1.1 Différencier les distributions d'automatisation des tests

Mike Cohn a conçu le concept original d'une pyramide d'automatisation des tests décrivant trois niveaux : unitaire (l'ISTQB l'appelle "de composants"), service, et interface utilisateur (IU). Depuis, de nombreuses variantes sont apparues, et elles partagent toutes les mêmes principes de base de description du rôle des niveaux de test et de distribution des cas de test à travers ces niveaux (voir le Syllabus CTFL de l'ISTQB, section 5.1.6 Pyramide des tests).

Dans la pyramide d'automatisation des tests originale de Mike Cohn, l'accent est mis sur les types de tests effectués par la TAS. Le niveau de service peut être décomposé en trois types de tests : les tests d'intégration des composants, les tests de contrat et les tests d'API.

Les tests unitaires (l'ISTQB les appelle tests de composants) servent à valider les composants individuels, en se concentrant sur la qualité du code. Les tests d'intégration des composants permettent de valider l'interface utilisateur (UI) et l'API en exploitant des doubles de test, tels que les mocks et les bouchons. Le test de contrat permet la validation des contrats entre les services. Les tests d'API se concentrent sur la validation fonctionnelle de services donnés avec des données réelles par le biais de connexions aux services réels. Le test de l'interface graphique est un test de bout en bout d'un système, en interaction avec son interface graphique.

Il est utile de dessiner l'état actuel des tests comme ligne de base et l'état ciblé. Cela permet de comprendre clairement ce qui manque ou ce qui constitue un niveau de test acceptable. On indiquera ainsi la quantité de tests effectués à chaque niveau de test et on précisera s'il s'agit de tests manuels ou automatisés. L'état ciblé dépend également du calendrier et de ce qui est réalisable dans ce délai. S'il est réalisable, la forme de la pyramide devrait alors être la forme ciblée.

Exemples de distributions de test :

- **Pyramide :** Il s'agit d'une répartition équilibrée des tests, avec moins de tests effectués sur les niveaux de test supérieurs et davantage sur les niveaux de test inférieurs avec des tests stables et plus rapides. Si les ressources disponibles et le calendrier le permettent, il s'agit souvent de la pyramide de l'état cible.
- **Cornet de glace :** Il s'agit de la version inversée de la pyramide. Il y a une quantité équilibrée de tests de services, mais les tests dépendent fortement de la capacité à trouver la majorité des défauts dans le niveau de test de l'interface utilisateur, qui est généralement plus coûteux à automatiser en raison de sa complexité. En raison de l'absence de tests de composants, les défauts sont trouvés plus tard dans le SDLC.
- **Sablier :** Les tests pèsent lourd sur les niveaux de test les plus élevés et les plus bas, et les tests de niveau de service sont le plus souvent absents, ce qui se traduit par des défauts d'intégration. Si la logique Métier est fournie par des API (c'est-à-dire des services), alors de nombreux tests de l'interface utilisateur peuvent facilement être déplacés vers les niveaux de test inférieurs.
- **Parapluie :** Les tests dépendent entièrement des tests coûteux de l'interface utilisateur. Il en résulte une lenteur dans la résolution des défauts, une maintenance coûteuse des cas de test et de la TAS. S'il n'est pas techniquement possible d'implémenter des cas de test de niveau inférieur, alors s'éloigner de la forme parapluie pourrait ne pas être réalisable, et l'accent devrait

être mis sur l'optimisation de la suite d'automatisation des tests de l'IU, la réduction du temps d'exécution des tests, et l'amélioration de la stabilité.

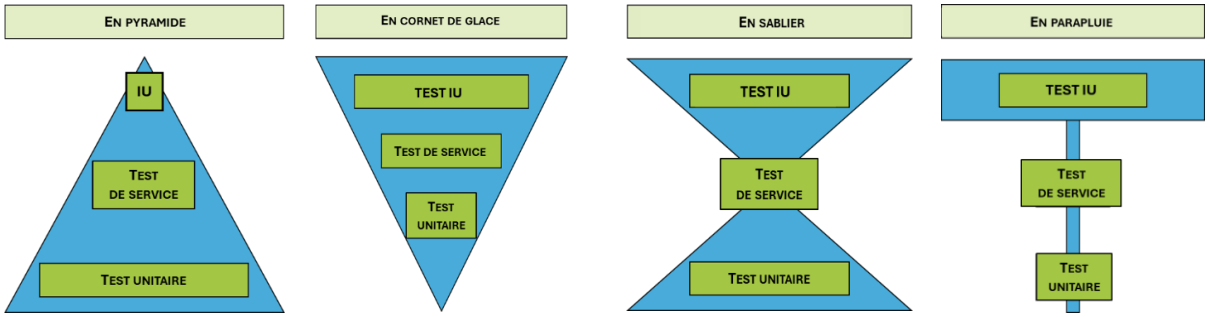

Figure 1: Exemples de distributions de test

3.1.2 Choisir une approche d'automatisation des tests en fonction de l'architecture du système sous test

Il existe de nombreuses façons de définir les niveaux de test dans une stratégie de test. Celle à choisir dépend fortement de plusieurs facteurs tels que la culture de l'organisation, la maturité globale du génie logiciel, le SDLC suivi et le fait que le SUT ait une architecture mainframe ou microservices.

Bien que la forme pyramidale soit considérée comme la répartition idéale des tests, elle n'est pas toujours réalisable, ou prend plus de temps pour passer à cet état final. Une bonne pratique consiste à adopter une forme réaliste comme état ciblé, par exemple en passant de la forme parapluie à la forme sablier. Une fois cet objectif atteint, un nouvel état cible peut être déterminé.

Le choix de la distribution appropriée de l'automatisation des tests pour les systèmes centraux diffère de celui du développement logiciel moderne et dépend à nouveau de nombreux facteurs, car certains niveaux peuvent ne pas être du tout automatisables. L'accès aux ordinateurs centraux se fait traditionnellement par l'émulation de terminal (c'est-à-dire par des écrans verts). La validation des processus par lots est possible mais limitée. Les tests de composants sont plus difficiles à réaliser. Comme il n'y a que peu ou pas de microservices présents, il n'est pas toujours possible de valider la communication des données entre les interfaces à l'aide de tests d'API. Les tests par l'intermédiaire de lots, de bases de données et d'une interface graphique sont plus fréquents.

Les organisations qui modernisent leurs solutions patrimoniales, en évoluant lentement vers une architecture de microservices, peuvent introduire des tests API et des tests de contrat dans les parties modernisées du système.

3.1.3 Démontrer les moyens d'optimiser la distribution de l'automatisation des tests pour réaliser les approches shift-left et shift-right.

Une fois que la distribution de l'état actuel est identifiée et que la distribution de l'état ciblé est sélectionnée, une feuille de route des améliorations peut être planifiée. Cela permet de déterminer ce qu'il faut tester avec l'automatisation (c'est-à-dire le périmètre de l'automatisation des tests) et comment

tester (c'est-à-dire la stratégie d'automatisation des tests). Un backlog hiérarchisé des éléments à implémenter et les critères d'entrée pour l'automatisation des tests doivent être déterminés.

Dans le cas d'une répartition déséquilibrée de l'automatisation des tests, il est recommandé d'adopter une approche ascendante. Si la couverture du code est faible, c'est le signe d'un manque de cas de tests de composants, et des cas de tests de composants supplémentaires doivent être écrits. Dans un deuxième temps, la qualité des tests de composants doit être revue et, si nécessaire, améliorée. L'automatisation des tests peut être améliorée en suggérant les meilleures pratiques, en suivant une technique de développement piloté par les tests et en organisant des ateliers sur les techniques de test.

L'introduction de tests de composants pour les tests de l'interface utilisateur et de l'API, en tirant parti de doubles de tests (par exemple, mocks, bouchons) permet de s'éloigner des tests coûteux, lents et peu fiables. La suppression de la dépendance à l'égard des services et des données réels améliore la cohérence de l'exécution des tests et fournit un retour d'information précoce facile à intégrer dans les pipelines CI/CD.

Ces dernières années, les tests de contrat ont pris une place de plus en plus importante. En validant le contrat entre un fournisseur et un consommateur, il est plus facile de trouver plus tôt les causes racine des défauts. Les équipes ne dépendront plus des tests d'API, ni des tests d'interface utilisateur pour détecter les défauts des services sous-jacents, et pourront au contraire diminuer le nombre de ces cas de test. Concevoir un graphe d’appels pour les APIs est une façon intelligente pour suivre les APIs connectées entre elles et pour identifier les producteurs et les consommateurs d’une API donnée. Cela permet de mieux identifier les points douloureux potentiels d'un système.

Alors que le shift-left est une approche qui consiste à déplacer les tests plus tôt dans le SDLC, le shift-right déplace les tests plus tard, lorsque le SUT a été versionné, comme un moyen d'évaluer la performance dans un environnement de test de pré-production ou de production. En appliquant le shift-right, les testeurs peuvent surveiller les performances de l'application et de l'API tout en obtenant un retour d'information de la part des utilisateurs réels. Bien que l'automatisation des tests soit plus importante dans une approche shift-left, si elle est combinée avec l'observabilité, alors l'automatisation des tests peut aider à prendre des décisions plus rapides sur la question de savoir si une version complète peut aller de l'avant, ou si la version candidate doit être annulée.

Les tests shift-right visent à :

- Faciliter la compréhension des préférences des utilisateurs.
- Soutenir les versions canari et les lancements obscurs afin de perturber le moins possible les fonctionnalités des utilisateurs.
- Identifier rapidement les défauts dans la production
- Contribuer à étendre le périmètre et l'utilisation de l'automatisation des tests.
- Augmenter la couverture

3.2 Considérations stratégiques dans les différents modèles de cycle de vie du développement logiciel

3.2.1 Expliquer comment les projets d'automatisation des tests se conforment aux anciens modèles de cycle de vie du développement logiciel

Dans le modèle en cascade, la phase de test vient après les phases d'analyse des exigences, de conception du système et d'implémentation. En raison de cet ordre strict, les activités d'automatisation des tests commencent plus tard. Des cycles plus longs entre les tests signifient moins d'opportunités de tirer parti de l'automatisation des tests, et le retour d'information de l'automatisation des tests arrive généralement trop tard, ce qui se traduit par un retour sur investissement (ROI) plus faible.

Dans le modèle en V, toute la planification des tests ainsi que la préparation de la documentation ont lieu dans les premières phases. Par exemple, la phase de conception de l'architecture produit la conception des tests d'intégration, ce qui permet de tester plus tôt que dans le modèle en cascade. Malheureusement, l'implémentation effective d'une TAS intervient toujours plus tard dans le cycle de développement logiciel, et bien que le retour sur investissement de l'automatisation des tests soit plus élevé que dans le modèle en cascade, il reste en retrait par rapport aux pratiques modernes de développement logiciel en mode Agile.

3.2.2 Expliquer comment les projets d'automatisation des tests sont conformes aux meilleures pratiques de développement logiciel en mode Agile qui soutiennent l'automatisation des tests

L'un des idéaux du développement logiciel en mode Agile est de parvenir à l'automatisation des tests en cours d'exécution. Cela signifie déterminer toutes les activités d'automatisation des tests nécessaires dans le cadre des critères de sortie pour chacune des User Story. Cela inclut les définitions des cas de test, l'implémentation des cas de test automatisés, les mises à jour du TAF et, dans certains cas, l'intégration à un pipeline CI/CD. Les organisations qui n'appliquent pas correctement les pratiques Agile n'incluent pas d'estimation de l'effort de test et les administrent dans un ticket séparé ou ne les pilotent pas du tout. En réalisant l'automatisation des tests en sprint, les équipes Agile s'assurent qu'elles sont prêtes à déployer le périmètre convenu au cours ou à la fin de chaque sprint. Si une équipe n'est pas mature d'un point de vue Agile, son premier objectif devrait être de tester en cours de sprint, tandis que l'automatisation serait en retard d'un sprint. À partir de là, l'équipe peut s'efforcer de parvenir à l'automatisation des tests au cours du même sprint (NDT: in-sprint).

3.2.3 Préparer les projets d'automatisation des tests à se conformer aux meilleures pratiques DevOps pour parvenir à des tests continus.

Le développement logiciel en mode Agile se concentre sur l'organisation du travail, tandis que le DevOps est responsable de la livraison des logiciels de bout en bout. Pour ce faire, les activités de build, d'intégration, de test, de déploiement et de production sont automatisées. Cela facilite les tests continus grâce à des boucles de rétroaction qui garantissent une amélioration continue.

L'accent est davantage mis sur l'implémentation de cas de test automatisés de bas niveau, y compris les tests de composants, d'intégration de composants et de contrats. En s'appuyant moins sur les données et

les services réels, les tests seront exécutés en moins de temps. Si possible, l'automatisation des tests devrait être exécutée dans le même pipeline où le SUT est construit.

Différentes suites de tests sont déclenchées après chaque phase de développement et de build (par exemple, locale, pull request, merge (fusion) et deploiement (deploy)).

Si les testeurs ont la capacité de construire des suites de tests d'interface utilisateur et d'API plus importantes, celles-ci sont exécutées séparément pour apporter une valeur ajoutée. Il est suggéré que les efforts de test manuel se concentrent sur les tests exploratoires et le retour d'information de l'utilisateur final, en tant qu'activité complémentaire à l'automatisation des tests.

3.3 3.3 Applicabilité et viabilité de l'automatisation des tests

3.3.1 Expliquer les critères permettant de déterminer si les tests se prêtent à l'automatisation.

La sélection des cas de test pour l'automatisation des tests est généralement effectuée par un analyste de test qui comprend quels cas de test peuvent être automatisés, ou par un TAE qui a la connaissance du domaine nécessaire pour prendre de telles décisions.

Les éléments suivants sont pris en compte lors de la sélection et de la hiérarchisation des cas de test pour l'automatisation des tests :

- Est-il techniquement possible d'implémenter les cas de test de manière automatisée ?
- Existe-t-il des défis techniques qui ont un impact sur la livraison des cas de test automatisés ? L'équipe est-elle préparée et bien formée pour effectuer le travail d'implémentation ?
- L'effort de codage fournit-il un retour sur investissement adéquat ? (voir section 5.1.1)
- L'exécution fréquente des cas de test présente-t-elle un intérêt ?
- S'agit-il d'un test fonctionnel ou non fonctionnel ? Fait-il partie de la suite de smoke tests, de la suite de tests de régression ou de la suite de tests de confirmation ?
- Le cas de test est-il reproductible ?
- Le cas de test est-il facile à maintenir lorsque le SUT change à la suite de mises à jour ?
- Le cas de test couvre-t-il des flux de travail métier fréquemment utilisés ?
- Existe-t-il un chevauchement fonctionnel entre les tests permettant la facilité de réutilisation des étapes de test et des données de test ?

3.3.2 Identifier les défis que seule l'automatisation des tests peut relever

Certains tests ne peuvent être réalisés qu'avec l'automatisation des tests. Il s'agit notamment des catégories suivantes :

1. L'exécution manuelle des tests prend plus de temps qu'il n'en faut.
2. L'exécution des cas de test doit être synchronisée.
3. Les résultats des tests doivent être disponibles dans un pipeline.
4. De grands fichiers de log doivent être analysés pour détecter les défauts.
5. La précision du calendrier des tests est exigée.
6. Des permutations de tests sont exigées entre plusieurs systèmes d'exploitation, navigateurs, appareils, lieux ou configurations.
7. Un volume important d'exécutions et/ou de données d'entrée est nécessaire pour maximiser la couverture.
8. Tous les tests non fonctionnels qui nécessitent un pilotage et une analyse automatisée, ou la contribution d'un grand nombre d'utilisateurs, tels que les tests de stress ou de fiabilité.

3.3.3 Identifier les conditions de test difficiles à automatiser

Certaines conditions de test font de l'automatisation des cas de test un défi difficile à relever, voire une tâche impossible. Il existe de nombreux exemples concrets. Certains d'entre eux sont énumérés ci-dessous :

- Validation des exigences de conception, y compris la cohérence de l'interface utilisateur à travers différentes plateformes. L'aspect général et la convivialité d'un logiciel sont subjectifs et nécessitent la revue d'un humain.
- Tout cas de test qui implique une trop grande interaction humaine. Dans le secteur de la finance, un bon exemple serait une demande de prêt. Dans ce processus, il peut y avoir des cas de certaines réglementations ou conditions (par exemple, un revenu insuffisant et des données personnelles incorrectes). Dans de telles situations, les agents doivent revérifier la demande de prêt réelle et prendre des décisions manuelles ou contacter le client.
- Des difficultés techniques bloquant les activités d'automatisation des tests, telles que des restrictions au niveau du système d'exploitation. Un exemple est le logiciel natif du système d'exploitation qui envoie des messages textuels aux utilisateurs. Les interactions ou les validations de ces messages textuels ne sont pas possibles avec les outils d'automatisation des tests de l'interface utilisateur, car le système d'exploitation ne permet pas d'interactions en dehors du SUT ciblé.
- Des conditions de test avec une longue dépendance temporelle rendraient l'automatisation des tests inefficace et sont souvent difficiles à mettre en place correctement par rapport à l'exécution du test manuel. À titre d'exemple, le testeur doit se connecter au SUT, rafraîchir le contenu de la page d'accueil et attendre deux heures pour que le SUT se déconnecte automatiquement en raison d'un délai d'expiration de la session. Si le temps écoulé ne peut pas être modifié manuellement (par exemple, par des simulations, des réponses côté serveur et des changements de temps au niveau du système d'exploitation), il n'est pas conseillé d'implémenter de tels cas de test de manière automatisée.

4 Stratégies d'automatisation des tests de déploiement et de livraison au niveau de l'organisation – 135 minutes (K2)

Mots clés

composant, test de confirmation, point de contrôle de qualité

Objectifs d'apprentissage pour le chapitre 4:

4.1 Planification de la solution d'automatisation des tests

CT-TAS-4.1.1 (K2) Identifier les moyens par lesquels l'automatisation des tests soutient un délai de mise sur le marché plus court.

CT-TAS -4.1.2 (K2) Identifier les façons dont l'automatisation des tests aide à vérifier les défauts rapportés conformément aux exigences.

CT-TAS -4.1.3 (K2) Définir des approches permettant le développement de scénarios pertinents du point de vue de l'exploitation pour l'automatisation des tests.

4.2 Stratégies de déploiement

CT-TAS-4.2.1 (K2) Définir une stratégie d'automatisation des tests.

CT-TAS -4.2.2 (K2) Identifier les risques liés à l'automatisation des tests lors du déploiement.

CT-TAS -4.2.3 (K2) Définir des approches pour atténuer les risques liés au déploiement des tests.

4.3 Dépendances vis à vis de l'environnement de test

CT-TAS-4.3.1 (K2) Définir les composants de l'automatisation des tests dans l'environnement de test

CT-TAS -4.3.2 (K2) Identifier les composants de l'infrastructure et les dépendances de l'automatisation des tests

CT-TAS -4.3.3 (K2) Définir les exigences en matière de données et d'interfaces de l'automatisation des tests pour l'intégration dans le système sous test

4.1 Planification de la solution d'automatisation des tests

4.1.1 Identifier les moyens par lesquels l'automatisation des tests soutient la réduction du délai de mise sur le marché.

L'automatisation des tests permet d'accélérer la mise sur le marché des logiciels en réduisant la durée du cycle de test. Les tests effectués avant la livraison d'un logiciel exigent une vérification et une validation approfondies. Cela peut être particulièrement important lors des tests de régression, car ce type de test favorise la réutilisation, la réduction des coûts et la cohérence entre les exécutions des tests.

L'automatisation des tests permet de réduire l'effort de test manuel et de fournir un retour d'information rapide aux développeurs tout en couvrant le même périmètre de test. Elle permet également de tester plus tôt dans le cycle de vie du développement logiciel si les tests de composants et les tests d'intégration des composants sont automatisés, ce qui n'est généralement pas fait manuellement.

Un point de contrôle de qualité est une mesure intégrée dans votre processus que le logiciel doit respecter avant de pouvoir passer à la phase suivante. La mise en place de points de contrôle qualité basés sur l'automatisation des tests permet d'accélérer le processus de déploiement dans un environnement de pré-production ou de production. Grâce à ces points de qualité, les défauts peuvent être trouvés plus tôt, ce qui raccourcit le délai de mise sur le marché. Le temps d'exécution des tests peut être réduit en tirant parti de l'exécution parallèle des tests et en suivant une approche shift-left. Une approche shift-left promeut une culture de la qualité et encourage les tests plus tôt dans le cycle de vie du développement logiciel. Il peut s'agir de cas de test multiples et indépendants ou de tests inter-navigateurs. Pour plus d'informations, voir les sections 3.1.3, 6.1.2 et 6.2.1.

4.1.2 Identifier les moyens par lesquels l'automatisation des tests aide à vérifier les rapports de défauts rapportés

Les tests de confirmation effectués à la suite d'une correction de code permettent de corriger un rapport de défaut. Un testeur suit généralement les étapes de test nécessaires pour reproduire le défaut afin de vérifier qu'il n'existe plus.

Les défauts ont une façon de se réintroduire dans les versions suivantes (par exemple, cela peut indiquer un problème de gestion de la configuration ou du référentiel de code) et, par conséquent, les tests de confirmation sont des candidats appropriés pour l'automatisation des tests et peuvent être ajoutés à la suite de tests de régression existante.

Un test de confirmation automatisé a généralement un périmètre fonctionnel restreint. L'implémentation peut avoir lieu à n'importe quel moment une fois qu'un défaut est rapporté et que les étapes de test nécessaires pour le reproduire sont comprises.

Le suivi de l'automatisation des tests de confirmation permet d'établir des rapports sur le temps et le nombre de cycles de test consacrés à la résolution des défauts.

En vérifiant les corrections sur plusieurs plateformes, appareils, navigateurs et versions de systèmes d'exploitation grâce à l'automatisation des tests, le temps consacré aux tests est considérablement réduit.

4.1.3 Définir des approches permettant le développement de scénarios pertinents du point de vue de l'exploitation pour l'automatisation des tests.

Le test d'acceptation opérationnelle garantit que le logiciel est prêt pour les systèmes de production et est généralement effectué juste avant une livraison. Cet effort vise à effectuer une validation finale des systèmes, composants et autres infrastructures du SUT, et à tester son aptitude à la production. Cette étape est nécessaire car, malgré tous les efforts du TAE, rien ne garantit que le SUT se comportera de la même manière en dehors de l'environnement de test et en production.

Un bon plan de test d'exploitation comprendra des tests de fiabilité, de tolérance aux défauts, d'intégrité et de maintenabilité. Les approches d'automatisation des tests suivantes peuvent être utilisées :

- Analyse statique du code - Outils automatisés qui analysent le code pour en déceler la sécurité et les vulnérabilités. Les résultats de test sont stockés en vue d'une analyse des tendances au fil du temps.
- Tests de bout en bout - Des scripts de test automatisés qui réalisent des scénarios utilisateur de bout en bout et exercent tous les aspects de l'environnement de test afin de découvrir les éventuels défauts.
- Tests de basculement - Scripts de test automatisés qui testent spécifiquement ce qui se passe lorsque le matériel d'une application tombe en panne. Quelques exemples : comment l'application se rétablit-elle en cas de dysfonctionnement des serveurs physiques, des serveurs cloud, des réseaux, des disques informatiques et d'autres composants matériels. Les scripts de test sont conçus pour mesurer la réussite ou la défaillance de la capacité du SUT à se rétablir automatiquement, ce qui est particulièrement important pour les organisations qui implémentent l'ingénierie du chaos.
- Tests de back-up et restauration - Scripts de test automatisés qui testent la réussite de la réalisation d'une sauvegarde de la version actuelle, puis du retour à un point antérieur (c'est-à-dire une version plus ancienne du logiciel).
- Tests d'efficacité de performance (par exemple, tests de charge) - Scripts de test automatisés qui peuvent être utilisés pour simuler de nombreux utilisateurs exerçant le SUT en même temps. Les résultats des tests sont stockés à des fins de comparaison et de tendance dans le temps.
- Revue de la documentation opérationnelle - Des scripts de test automatisés peuvent être utilisés pour cette activité afin de comparer les versions de la documentation du SUT et de signaler aux équipes de développement si une mise à jour est nécessaire en fonction des nouvelles caractéristiques ajoutées à une version particulière.
- Tests de sécurité - Des scripts de test automatisés peuvent être créés en combinaison avec des outils de test de sécurité conformes aux normes de l'industrie pour évaluer le SUT, de manière statique et dynamique. Les résultats des tests sont stockés au fil du temps à des fins de comparaison et d'analyse des tendances.
- Pilotage basé sur l'accord de niveau de service (SLA) d'une organisation - Les scripts de tests automatisés utilisés pendant le SDLC peuvent être réaffectés au pilotage des opérations de production et à l'envoi d'alertes en cas d'échec d'un test automatisé. Il s'agit d'une mesure proactive permettant de rattraper les pannes de production avant que les utilisateurs réels n'en fassent l'expérience.

L'automatisation des tests pour la validation de logiciels opérationnels peut apporter d'immenses bénéfices, en particulier si les tests automatisés peuvent être exécutés de manière répétée et dans des environnements multiples. Des suites d'automatisation des testss dédiées peuvent être créées afin que les résultats des tests puissent être générés et comparés avec les exécutions de tests précédentes. La cohérence est ainsi assurée lorsque de nouvelles versions du SUT sont mises en circulation. Une suite de tests automatisée fiable comprenant des conditions de test spécifiques à l'exploitation peut réduire les coûts au fil du temps.

4.2 Stratégies de déploiement d'automatisation des tests

4.2.1 Définir une stratégie de déploiement d'automatisation des tests

Une bonne stratégie d'automatisation des tests prendra en compte, sans s'y limiter, l'environnement de test global, les outils de test disponibles, l'accès au SUT, le stockage des scripts de test et les autres dépendances, ainsi que l'approvisionnement en données de test. Avec ces considérations de haut niveau à l'esprit, un TAE peut commencer à réfléchir à une stratégie de développement et de déploiement de la TAS.

Environnement de test - Les TAE doivent réfléchir à la manière dont ils vont accéder au SUT dans les différents environnements de test. Lorsqu'ils commencent à développer leur testware d'automatisation des tests, qu'ils utilisent une approche pilotée par les mots-clés ou une autre solution, ils doivent réfléchir à la manière dont cette solution s'exécutera dans plusieurs environnements de test. Par exemple, un script de test doit être développé de manière à pouvoir être exécuté dans un environnement de test et dans un environnement de pré-production avec un minimum de modifications. En règle générale, il suffit de modifier un localisateur de ressources uniformes (URL) pour une application Web et le script de test s'exécute de la même manière quel que soit l'environnement de test dans lequel il se trouve.

Outils - Les TAE devront également s'interroger sur les outils qu'ils utilisent pour construire la TAS. S'il s'agit d'un outil commercial, il y a de fortes chances qu'ils doivent comprendre quel est son modèle de licence. Le TAE peut trouver que son environnement de test doit avoir accès au serveur de licence de l'outil. Il faut en tenir compte lorsque le script de test est destiné à être utilisé dans plusieurs environnements de test. Ce n'est pas parce que le serveur de licences est disponible dans l'environnement de test qu'il le sera nécessairement dans l'environnement de pré-production.

Accès au logiciel - Il est important de comprendre comment accéder au SUT. Les scripts de test peuvent être conçus pour accepter des paramètres de sorte que des utilisateurs spécifiques ou des identifiants avec un accès spécial puissent être rapidement mis à jour lorsque les points d'extrémité du SUT changent. Cet aspect devient à nouveau très important lorsque l'on passe à des environnements de test différents et que l'URL doit changer. En outre, il peut être nécessaire d'utiliser des identifiants différents (par exemple, via des comptes d'utilisateur et de test, des données biométriques et des cartes à puce) en fonction de l'environnement de pré-production. La conception d'un bon script de test automatisé est suffisamment souple pour que des paramètres simples puissent être définis et que les scripts de test soient exécutés comme prévu, quel que soit l'environnement de test.

Stockage des scripts de test - Le TAE devra décider d'un emplacement central pour stocker et gérer les scripts d'automatisation des tests . Une bonne stratégie consisterait à intégrer un référentiel de code source qui utilise la gestion de la configuration. De cette manière, les scripts de test peuvent être accessibles à partir de plusieurs environnements de test, à condition qu'ils aient accès au référentiel du

code source, et des versions peuvent être créées pour la version spécifique du SUT. Toutes les configurations et dépendances peuvent être sauvegardées dans le même référentiel que les scripts de test et offrent une portabilité optimale. En bref, la TAS, le TAF et tous les cas de test peuvent être stockés et gérés dans des référentiels.

Approvisionnement des données - Il sera important de comprendre si un script de test automatisé dépend de certaines données de test déjà existantes dans l'environnement de test où le SUT est testé. Une bonne conception des scripts de test évitera autant que possible les données statiques (c'est-à-dire les ensembles de données fixes). Cependant, il y aura des situations où cela ne sera pas possible. Dans ces cas, le TAE devra trouver une solution pour que les données de test soient préchargées ou utiliser l'automatisation des tests pour créer des scripts de test de configuration qui génèrent les données de test nécessaires avant que les scripts de test réels ne s'exécutent et ne testent la SUT. Les deux approches présentent des avantages et des inconvénients. D'une part, il est pratique qu'un administrateur précharge les données de test. Cependant, le TAE dépend alors d'une autre ressource qui doit se rendre disponible pour le soutenir. En étant autonome avec un script de test configuré, il n'est plus nécessaire de dépendre d'une autre ressource. Cependant, l'élaboration de ces scripts de test prend du temps et devient un autre élément de la solution qui doit être maintenu.

En tenant compte de tous les éléments mentionnés ci-dessus, la stratégie d'automatisation des tests sera en mesure de fournir une planification et un contrôle appropriés des activités d'automatisation des tests.

4.2.2 Identifier les risques de l'automatisation des tests lors du déploiement

Les problèmes techniques peuvent entraîner des risques produit et des risques projet (pour plus d'informations, voir le syllabus CTFL, section 5.2.2). Les problèmes techniques typiques comprennent :

- Trop d'abstractions peuvent conduire à des difficultés de compréhension de ce que le code d'automatisation des tests fait réellement (par exemple, avec des mots-clés dans l'approche pilotée par les mots-clés).
- Les tables de données de test peuvent devenir trop volumineuses/complexes/commodes à migrer vers d'autres environnements de test, ce qui se traduit par des résultats d'état incohérents.
- La dépendance de la TAS à l'égard de certaines bibliothèques de systèmes d'exploitation ou d'autres composants qui peuvent ne pas être disponibles dans tous les environnements de test du SUT.

Les risques typiques d'un projet de déploiement comprennent :

- Les problèmes de personnel : Il peut être difficile de trouver les bonnes personnes pour assurer la maintenance de l'automatisation des tests.
- Maintenance non planifiée de la TAS en raison de mises à jour du SUT qui entraînent une opérabilité incorrecte de la TAS.
- Retards dans l'introduction de l'automatisation des tests.
- Retards dans la mise à jour de la TAS en fonction des modifications apportées au SUT.
- La TAS ne peut pas capturer les objets non standard de l'interface utilisateur.

- Permettre aux cas de test obsolètes de rester dans les suites de tests, ce qui fait perdre du temps à l'exécution des tests.

Les points de défaillance potentiels du projet TAS sont les suivants :

- Migration vers un environnement de test différent.
- Déploiement dans un environnement de production
- Oubli du fait que l'automatisation est un logiciel qui doit également être testé.

Il est important de réaliser que divers problèmes techniques, risques projet et points de défaillance potentiels peuvent compromettre le succès des projets d'automatisation des tests. Les problèmes techniques les plus courants sont la complexité due à des abstractions excessives, les difficultés liées à la gestion des données de test et les dépendances à l'égard de composants spécifiques. Les risques du projet doivent également tenir compte des difficultés de recrutement, des problèmes de maintenance dus aux mises à jour des systèmes, des retards de déploiement et de la présence de cas de test obsolètes. En outre, les points de défaillance potentiels, tels que la migration vers différents environnements et négliger la nécessité de tester le logiciel d'automatisation lui-même, doivent être pris en compte dans le cadre d'une planification adéquate. Dans l'ensemble, le pilotage et les mesures proactives garantiront la réussite des projets d'automatisation des tests.

4.2.3 Définir des approches pour atténuer les risques de déploiement

De nombreuses stratégies d'atténuation des risques peuvent être mises en œuvre pour faire face à ces risques. Certaines d'entre elles sont présentées ci-dessous.

La TAS a son propre cycle de vie de développement, qu'il s'agisse d'un développement interne ou d'une solution acquise. Il ne faut pas oublier que la TAS, comme tout autre logiciel, doit faire l'objet d'une gestion de configuration et que ses caractéristiques doivent être documentées. Dans le cas contraire, il devient extrêmement difficile de déployer différentes parties et de les faire fonctionner ensemble ou dans certains environnements de test.

En outre, il doit exister une procédure de déploiement documentée, claire et facile à suivre. Cette procédure dépend de la version ; elle doit donc également être incluse dans la gestion de la configuration.

Il y a deux cas distincts lors du déploiement d'une TAS:

1. Premier déploiement
2. Déploiement de maintenance - la TAS existe déjà et une mise à jour doit être déployée.

Les risques liés au premier déploiement sont les suivants :

- La durée totale d'exécution de la suite de tests peut être supérieure à la durée d'exécution des tests planifiée pour le cycle de tests. Dans ce cas, il est important de veiller à planifier suffisamment de temps pour que la suite de tests soit exécutée dans son intégralité avant le début du prochain cycle test prévu.
- Il existe des problèmes d'installation et de configuration des environnements de test (par exemple, installation de la base de données et charge initiale, et démarrage/arrêt des services). La TAS a besoin d'un contexte de test (c'est-à-dire un ensemble de données prédéfini) pour créer les préconditions nécessaires à l'exécution des cas de test automatisés dans l'environnement de test.

Pour les déploiements de maintenance, il y a d'autres considérations à prendre en compte. La TAS elle-même doit évoluer et ses mises à jour doivent être déployées dans la production. Avant de déployer une version mise à jour de la TAS en production, il faut la tester. Il est donc nécessaire de vérifier les nouvelles fonctionnalités, de s'assurer qu'une suite de tests peut être exécutée sur la TAS mise à jour, que les rapports de test peuvent être envoyés et qu'il n'y a pas de défauts de performance ou d'autres problèmes de qualité. Dans certains cas, il peut être nécessaire de modifier l'ensemble de la suite de tests pour l'adapter à la dernière version de la TAS.

4.3 4.3 Dépendances vis-à-vis de l'environnement de test

4.3.1 Définir les composants d'automatisation des tests dans l'environnement de test.

Les composants d'automatisation des tests sont généralement constitués d'outils, de machines virtuelles, de scripts d'automatisation des tests, de conteneurs et de configurations. L'environnement de test comprend également le SUT. Les composants d'automatisation des tests dans l'environnement de test peuvent être définis comme suit :

SUT - Il s'agit d'une partie évidente de l'environnement de test. Le SUT peut être testé dans son ensemble ou divisé en sous-composants (par exemple, API, interface Web et base de données).

Plateforme - La plateforme décrit l'endroit où les composants d'automatisation des tests sont hébergés. Cela inclut l'infrastructure cloud, le réseau, les machines virtuelles et les conteneurs qui peuvent être utilisés pour rendre l'automatisation des tests efficiente et portable.

Cas de test et suites de test - Cela décrit les cas de test individuels qui sont constitués d'étapes de test qui dirigent à la fois des actions automatisées et manuelles. Les suites de tests décrivent un regroupement logique de cas de tests afin qu'ils puissent être exécutés ensemble de manière efficace.

Outils - Différents outils d'automatisation des tests sont utilisés pour plusieurs raisons. Une collecte d'outils inclurait ceux pour l'automatisation des tests de l'interface utilisateur, les tests des points d'extrémité de l'API, la génération de données, la surveillance, la gestion des exigences, la gestion des défauts, les outils de logging et de reporting, et les outils pour la génération de tendances basées sur des métriques.

TAF - Il s'agit de tous les éléments qui entrent dans la conception du TAF. Les TAF comprennent des pilotes, des bibliothèques communes, des modèles de cas de test automatisés, des scripts de chargement/approvisionnement de données, de la documentation sur l'utilisation des composants du TAF et des didacticiels qui aident les TAE à utiliser le TAF. Pour plus d'informations, voir le CTAL-TAE Syllabus, section 3.1.

La maintenance des composants de test est une considération majeure, car une complication excessive de l'environnement de test peut résulter en des heures de temps perdues à corriger des défauts dans le TAF au lieu de bénéficier de la solution. Il est important de trouver la bonne combinaison d'outils, de configuration et de portabilité de la plate-forme pour rendre les composants aussi réutilisables que possible.

4.3.2 Identifier les composants de l'infrastructure et les dépendances de l'automatisation des tests.

Il existe un certain nombre de composants d'infrastructure et de dépendances à connaître lors de l'assemblage de l'automatisation des tests. Collectivement, ils couvrent tous les prérequis nécessaires à l'exécution d'une TAS. Les principaux composants et dépendances sont les suivants :

Machines hôtes - Il peut s'agir de machines virtuelles, de serveurs physiques, d'ordinateurs portables et d'appareils (par exemple, tablettes et mobiles). Le logiciel d'automatisation des tests y est installé et c'est là que les scripts de test sont créés et exécutés.

Réseau - C'est ce qui permet à la TAS d'accéder au SUT. Il peut également inclure la mise en réseau de plusieurs machines hôtes afin d'assurer l'exécution en parallèle de tests automatisés. En règle générale, les machines hôtes doivent se trouver sur le même réseau et être configurées correctement pour communiquer entre elles.

Plateforme - L'automatisation des tests, comme tout autre logiciel, peut s'exécuter sur des plateformes cloud ou être conçue pour fonctionner dans des conteneurs. Tant que la plateforme fournit des autorisations et un accès à l'OS sous-jacent, tous les outils et dépendances nécessaires peuvent être installés.

Dépendances logicielles - Pour que les outils d'automatisation des tests fonctionnent correctement, il est nécessaire de comprendre toutes les autres dépendances de l'automatisation des tests. Par exemple, un outil d'automatisation des tests particulier peut nécessiter que la dernière version d'un langage de programmation soit d'abord installée sur la machine hôte. Les TAE doivent tenir compte de ces dépendances avant et après la sélection d'un outil.

SUT - Une fois que les composants, les dépendances et l'infrastructure globale ont été pris en compte et configurés correctement, la dernière étape consiste à garantir l'accès au SUT. Outre le réseau, il est également nécessaire de réfléchir à la manière de s'interfacer avec le SUT. Par exemple, dans une application basée sur le web, un navigateur doit être installé sur la machine hôte. Le type de navigateur et sa version doivent être pris en compte.

4.3.3 Définir les exigences en matière de données d'automatisation des tests et d'interfaces.

Deux considérations auxquelles les TAE doivent penser avant de développer des scripts d'automatisation des tests sont la manière dont ils prévoient de s'interfacer avec le SUT et les dépendances des données des cas de test. Voici quelques exemples :

API - Si le SUT utilise une API, celle-ci peut être testée au niveau du point final au lieu de nécessiter une interface utilisateur au niveau de l'application. Cela peut nécessiter une plus grande compréhension des points finaux et des communications de données qui sont cachés par une interface utilisateur. Le TAE devra comprendre dans quel ordre appeler les API pour créer un processus métier et comment corréler les données pour que le cas de test reste intact. Les API exposées à l'internet sont également appelées API web ou services web.

Interface avec la base de données - Certains outils d'automatisation des tests ont la capacité de s'interfacer directement avec la base de données sous-jacente du SUT. Des scripts de test peuvent être écrits pour vérifier les données dans les colonnes et les lignes afin de s'assurer que les procédures scriptées et les autres règles de la base de données sont configurées correctement. Cela peut exiger que des valeurs de données spécifiques existent déjà dans la base de données pour tester la fiabilité.

Compatibilité des interfaces - L'utilisation de tests de contrats permet de s'assurer que deux systèmes distincts (par exemple, deux microservices) sont compatibles et peuvent communiquer l'un avec l'autre. Les tests de contrats peuvent être pilotés par le consommateur, par le producteur et bidirectionnels. Des détails supplémentaires peuvent être obtenus dans le syllabus du CTAL-TAE, section 5.1.3.

En outre, les TAE doivent étudier attentivement la manière dont ils vont s'interfacer avec le SUT et comprendre les dépendances des données dans les cas de test. Qu'il s'agisse de tester par le biais d'API, d'interfaces de base de données ou d'assurer la compatibilité des interfaces entre les systèmes, l'attention portée à ces considérations est essentielle pour une automatisation des tests robuste et efficace. En prenant en compte ces facteurs de manière réfléchie, les TAE peuvent améliorer la fiabilité et l'efficience de leurs efforts d'automatisation, contribuant en fin de compte à la qualité et au succès du SUT.

5 Analyse de l'impact de l'automatisation des tests – 150 minutes (K3)

Mots clés

couverture, rapport de test

Objectifs d'apprentissage pour le chapitre 5 :

5.1 Investissement dans la mise en place et la maintenance de l'automatisation des tests

CT-TAS-5.1.1 (K3) Montrer le retour sur investissement de la construction d'une solution d'automatisation des tests.

5.2 Métriques d'automatisation des tests

CT-TAS-5.2.1 (K2) Classifier les métriques pour l'automatisation des tests.

5.3 La valeur de l'automatisation des tests au niveau du projet et de l'organisation

CT-TAS-5.3.1 (K3) Identifier les considérations organisationnelles pour l'utilisation de l'automatisation des tests.

CT-TAS-5.3.2 (K3) Analyser les caractéristiques du projet qui aident à déterminer l'implémentation optimale des objectifs de test de l'automatisation des tests.

5.4 Décisions prises à partir des rapports d'automatisation des tests

CT-TAS-5.4.1 (K2) Analyser les données des rapports de tests pour éclairer la prise de décision.

5.1 Investissement dans la mise en place et la maintenance de l'automatisation des tests

5.1.1 Montrer le retour sur investissement de la construction d'une solution d'automatisation des tests

Il est important d'estimer les coûts d'installation et de maintenance de l'automatisation des tests avant de procéder à l'implémentation d'un projet. Au-delà des valeurs que l'automatisation peut apporter à un projet, il est bénéfique de comprendre le calcul du retour sur investissement du projet.

Le calcul du ROI peut fournir un retour d'information significatif à tout moment du cycle de vie d'un projet en démontrant le rendement d'une activité par rapport à l'effort investi. Le calcul du retour sur investissement peut être ajusté en incluant les coûts des testeurs manuels et les coûts des TAE en multipliant simplement le temps passé par leurs coûts de main d'œuvre respectifs.

Pour calculer le retour sur investissement, il faut déterminer l'investissement dans l'automatisation des tests (c'est-à-dire le temps et le coût) et les économies réalisées grâce à elle :

ROI = Économie / Investissement

Il est important de noter que les économies et l'investissement peuvent être calculés en tenant compte de différentes métriques et données, dans différentes mesures et unités. Dans le périmètre de ce syllabus, un modèle simple est utilisé pour démontrer l'approche, avec des unités de temps et non de coût. Si certaines activités ne sont mesurées qu'en coût, ce montant peut être converti en temps en utilisant un taux spécifique au projet.

En général, les économies réalisées grâce à l'automatisation des tests sont dues au fait que les mêmes tests peuvent être exécutés en beaucoup moins de temps que s'ils étaient exécutés manuellement. Cela signifie également qu'ils peuvent être exécutés plus souvent. Par conséquent, le nombre de tests exécutés peut être augmenté.

Pour calculer les économies réalisées, vous devez prendre en compte les métriques suivantes :

- Temps nécessaire pour exécuter un cas de test manuellement.
- Durée d'exécution d'un cas de test automatisé.
- Nombre de cas de test.
- Nombre d’exécutions de test.

Pour calculer l'investissement, vous devez tenir compte des métriques suivantes :

- Temps de mise en place de l'automatisation des tests.
- Temps moyen pour développer des scripts de test automatisés.
- Nombre de scripts de test automatisés implémentés.
- Durée moyenne de maintenance d'un script de test automatisé.
- Durée d'exécution d'un script de test automatisé.
- Pourcentage de scripts de test automatisés ayant échoué.

- Nombre de cas de test définis.
- Nombre d'exécutions de test.

En adaptant le modèle décrit au développement logiciel en mode Agile, les sprints d'un projet peuvent être prévus comme illustré dans le graphique ci-dessous. On peut déterminer le sprint à partir duquel l'automatisation des tests a rentabilisé son investissement en utilisant le graphique..

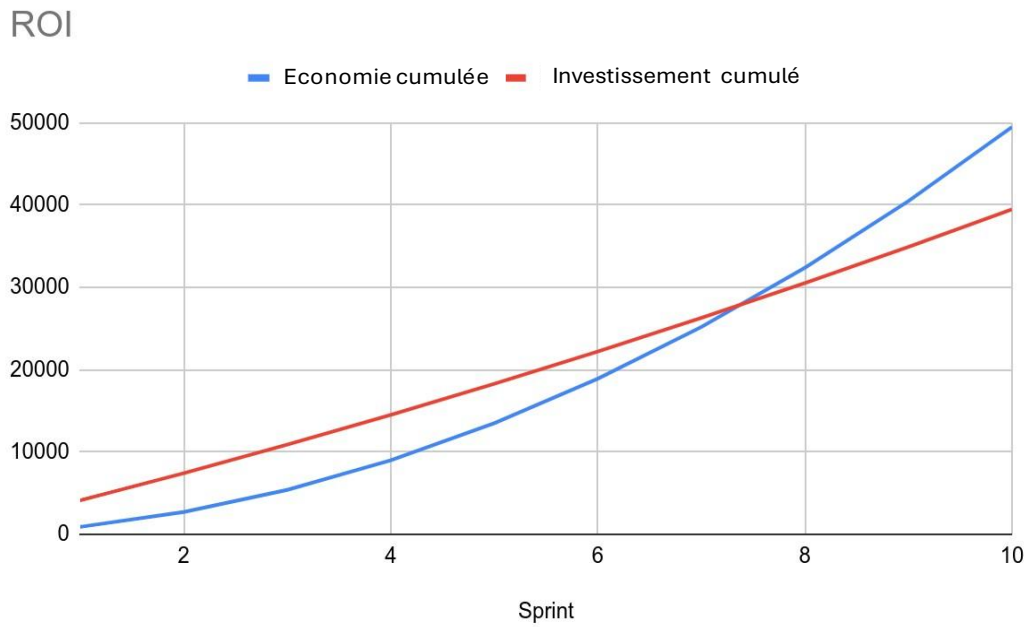

Figure 2: Exemple de calcul de retour sur investissement montrant le point de retour sur investissement

Gardez à l'esprit qu'il existe certaines connexions entre les métriques mesurées pour calculer le ROI, notamment :

- Si la durée planifiée du projet est inférieure au point d'inflexion du ROI, il ne vaut pas la peine d'introduire l'automatisation des tests. Dans ce cas, l'exécution manuelle des tests permet d'économiser du temps et des efforts.
- Comme l'investissement dépend considérablement du temps d'exécution des tests automatisés, le fait d'appliquer la pyramide des tests et d'implémenter les cas de test au bon niveau de test permet de réduire ce temps d'exécution et d'améliorer le ROI.

5.2 Métriques d'automatisation des tests

5.2.1 Classifier les métriques pour l'automatisation des tests

L'analyse des tendances sur la base de données factuelles facilite la prise de décision. En collectant les métriques d'une TAS, les testeurs sont en mesure d'évaluer la TAS et de prendre des décisions sur :

- L'adéquation de la TAS au projet.
- L'adaptabilité de la TAS en vue d'étendre ses fonctionnalités à de nouvelles conditions de test.
    - La modification d'un flux d'utilisateurs dans le SUT.
    - Un changement dans la manière dont les tests sont effectués.
- La maintenabilité de l'automatisation des tests en raison des défauts constatés dans la TAS.

Le coût de la mesure doit être aussi bas que possible, ce qui peut souvent être réalisé en automatisant la collecte et le reporting des métriques. Voici quelques exemples :

Ratio réussite-échec

Il s'agit d'une métrique courante qui mesure le rapport entre les tests automatisés qui ont réussi et les tests automatisés qui ont échoué à atteindre le résultat attendu. Le ratio réussite-échec = le nombre de tests passés / le nombre de tests échoués.

Rapport entre les défaillances et les défauts

Un problème courant avec les tests automatisés est qu'un grand nombre d'entre eux peuvent échouer pour la même raison, c'est-à-dire un seul défaut dans le SUT. La mesure du nombre de tests automatisés qui échouent pour un défaut donné peut aider à déterminer où se situe le problème. En outre, on peut collecter le nombre de défaillances par défaut et suivre la raison de la défaillance : défaut dans le sous-système, dans l'ensemble du système, problème avec les données de test ou l'infrastructure.

Temps d'exécution de l'automatisation des tests

L'une des métriques les plus faciles à déterminer est le temps nécessaire à l'exécution des tests automatisés. Au début de la TAS, cela peut ne pas être important, mais au fur et à mesure que le nombre de cas de tests automatisés augmente, cette métrique peut devenir très importante. Cette métrique inclut le temps de build de la TAS.

Nombre de cas de test automatisés

Cette métrique peut être utilisée pour montrer les progrès réalisés par le projet d'automatisation des tests. Mais il faut tenir compte du fait que le nombre de cas de test automatisés ne révèle pas beaucoup d'informations ; par exemple, il n'indique pas le niveau de couverture.

Couverture fonctionnelle de l'automatisation des tests

Cette couverture indique le pourcentage d'exigences fonctionnelles couvertes par les cas de tests automatisés.

Couverture du code

La couverture du code indique combien de lignes de code sont exercées par les tests de bas niveau (c'est-à-dire de composants).

Il n'existe pas de pourcentage absolu indiquant une couverture adéquate, et une couverture de code de 100 % est souvent difficile à atteindre, sauf pour les logiciels les plus simples. Cependant, il est généralement admis qu'une plus grande couverture est préférable car elle augmente la confiance dans le SUT. Au fur et à mesure de l'ajout de tests automatisés de bas niveau, la couverture du code devrait augmenter.

5.3 La valeur de l'automatisation des tests au niveau du projet et de l'organisation

5.3.1 Identifier les considérations organisationnelles pour l'utilisation de l'automatisation des tests

Avant de commencer l'automatisation des tests sur n'importe quel projet, il faut identifier les éléments suivants au sein de l'organisation :

Politiques et pratiques en matière de développement logiciel

Il est bon de vérifier comment les équipes de développeurs travaillent et quel type de documentation est présent. Toute documentation disponible peut être utile pour identifier comment l'automatisation des tests peut être connectée aux processus des équipes de développement. La documentation peut inclure la spécification technique du SUT, le logiciel et les outils de développement utilisés, ou toute politique disponible concernant le développement, comme les directives de revue de code, les normes de codage définies et les processus de merge.

Projets d'automatisation des tests actifs existants et leur statut

Dans le cas d'un projet de développement en cours, il peut y avoir un ou plusieurs projets de développement de TAS en cours par différentes équipes. Une recommandation générale pour les décideurs est de vérifier ces projets existants et leur statut, pour voir si l'un d'entre eux correspond aux objectifs de test définis pour la nouvelle TAS. En analysant la solution, on peut déterminer s'il faut réutiliser l'une des TAS existantes ou en créer une nouvelle en fonction des besoins actuels.

Les experts en matière d'organisation de l'automatisation des tests

Il est recommandé de rechercher des experts du sujet qui peuvent aider au déploiement d'une nouvelle TAS au sein de l'organisation. Ceux-ci peuvent partager leurs expériences et leçons apprises sur leurs TAS et leur déploiement. À partir de là, il est possible d'identifier les différents risques à prendre en compte ou à éviter pour que le déploiement de la TAS soit une réussite.

Disponibilité des environnements de test

Dans le cas des grandes organisations, il est généralement recommandé de recueillir des informations sur les environnements de test existants, leur utilisation et leur disponibilité. Ces informations sont cruciales pour éviter de ruiner le travail d'une autre équipe en déployant une nouvelle TAS et en utilisant le système sans notification ni accord. Souvent, les besoins d'un projet nécessitent un nouvel environnement de test, de sorte que les environnements existants peuvent être utilisés comme référence pour la création d'un nouvel environnement, et le travail peut être effectué plus facilement si les équipes et les personnes responsables sont identifiées et disponibles.

Outils de test et licences

Il est toujours utile d'obtenir des informations sur les outils et les licences dont dispose actuellement l'organisation. En identifiant ceux-ci, les coûts et les délais, la planification d'une TAS peut être réduite. De plus, les outils nécessaires à un nouveau projet peuvent déjà être disponibles. Par exemple, si les tests en cloud sont mis en place par l'intermédiaire d'un fournisseur défini, et que les licences sont disponibles pour le nouveau projet, il n'est pas judicieux d'utiliser un autre fournisseur de cloud. Il est recommandé d'utiliser les mêmes outils et les mêmes licences pour réduire les coûts du projet.

5.3.2 Analyser les caractéristiques du projet qui aident à déterminer l'implémentation optimale des objectifs de test de l'automatisation des tests

Plusieurs caractéristiques majeures du projet peuvent définir une méthode de travail optimale et aider à définir des objectifs d'automatisation des tests réussis.

Domaine

Il est important de comprendre que chaque domaine diffère soit par ses réglementations, soit par ses normes. Par exemple, les réglementations et les risques ne sont pas les mêmes dans le domaine du tourisme que dans celui de la finance ou des soins de santé. Il est toujours recommandé de vérifier les différentes normes et restrictions du domaine pour s'assurer que les objectifs d'automatisation des tests prévus sont conformes.

Plates-formes

En termes d'objectifs d'automatisation des tests, il est également souligné d'évaluer quelles plateformes le projet couvre et où il serait bénéfique de faire de l'automatisation des tests. La planification de l'automatisation des tests de plateformes multiples peut être plus difficile car il peut y avoir un besoin de solutions multiples pour couvrir les différentes plateformes. Dans de nombreux cas, comme pour le mobile et le web, les mêmes outils peuvent être utilisés. Mais c'est la planification qui déterminera la facilité de réutilisation de la TAS.

Langage de programmation et pile technologique

Les détails de l'implémentation d'un projet, comme son langage de programmation et sa pile technologique, déterminent également le type d'objectifs d'automatisation des tests qu'une organisation devrait définir. Il est recommandé d'utiliser les mêmes langages de programmation que les développeurs sur un projet donné. Ce faisant, la collaboration peut être beaucoup plus facile, et l'apprentissage croisé en effectuant des revues de code ensemble améliorera encore la qualité du SUT et la connaissance des TAE.

Maturité du projet

L'analyse de la maturité du projet permet d'obtenir des informations pour concevoir l'objectif d'automatisation des tests idéal. En identifiant les différents facteurs tels que le nombre de cas de test, les pipelines CI/CD existants, le nombre de testeurs et de TAE, différents objectifs de test et une feuille de route peuvent être créés. Par exemple, s'il y a beaucoup de cas de test manuels qui ont une priorité élevée et que leur temps d'exécution manuelle est long, il est important de viser à les automatiser en premier pour économiser du temps et des efforts. Outre les facteurs susmentionnés, le calendrier du projet est également très important. Si le projet est court ou se terminera bientôt et qu'il n'y a pas beaucoup de temps pour faire les choses, il n'est pas utile de planifier une TAS importante et robuste. En revanche, s'il s'agit d'un nouveau projet, il convient d'élaborer une feuille de route incrémentale, étape par étape, basée sur le produit minimum viable.

L'adhésion des parties prenantes

Souvent, l'automatisation des tests n'est pas exploitée parce que les parties prenantes clés n'y adhèrent pas. Cela peut être dû à la crainte de voir la vélocité tomber en dessous de leur cible, à des délais serrés, à la budgétisation ou à d'autres préoccupations. Après analyse de la maturité du projet, s'il y a de la place pour implémenter une TAS, une partie prenante stratégique doit identifier les risques produit, énumérer les bénéfices de l'introduction de l'automatisation des tests, et proposer un plan approprié. Ce plan doit mettre en évidence les jalons à atteindre avec et par l'automatisation des tests et indiquer comment la

testabilité du SUT doit être améliorée pour introduire avec succès une TAS qui répondra aux risques identifiés. Enfin, ce plan doit être présenté aux parties prenantes.

Connaissance de l'équipe et expérience pertinente

Le facteur le plus important et le plus pertinent pour une automatisation des tests réussie est la compétence et l'expérience des testeurs. Il est avantageux de travailler avec les testeurs et d'utiliser leurs connaissances pour définir des objectifs de test avec lesquels les analystes métier et eux-mêmes sont à l'aise. Souvent, les Test Managers ou les leaders de test créent une matrice de compétences et d'aptitudes pour identifier les connaissances disponibles au sein des équipes, et pour identifier les lacunes en même temps. Ces lacunes peuvent être identifiées avec différents objectifs tels que la formation et les tâches d'implémentation assignées.

Support du responsable des tests et budgétisation

En fonction de la taille et de la maturité du projet, le budget disponible et le support du responsable des tests doivent être pris en compte lors de la définition des objectifs de test pour l'automatisation des tests. Il est important de définir des objectifs de test qui peuvent être supportés, ce qui en retour permettra d'obtenir le support du responsable des tests.

Lors de la proposition d'une stratégie d'automatisation des tests au management, la documentation doit être concise, définissant clairement les lacunes actuellement identifiées ou les coûts futurs pour que l'automatisation des tests soit implémentée correctement. Une liste de recommandations et de leurs bénéfices soulignant la valeur métier et les réductions de coûts encouragera le responsable des tests à approuver le développement ou l'amélioration d'une TAS.

Caractéristiques de qualité

La norme ISO/IEC 25010:2011 définit les caractéristiques de qualité qui sont énumérées dans le syllabus CTFL de l'ISTQB, section 2.2.2. Types de test. Ces caractéristiques de qualité peuvent ensuite être utilisées pour évaluer la TAS actuelle, ou déterminer les métriques qui seront collectées par la TAS.

5.4 Décisions prises sur la base des rapports d'automatisation des tests

5.4.1 Analyser les données des rapports de tests pour éclairer la prise de décision

Le format et le contenu d'un rapport d'automatisation des tests peuvent varier en fonction des parties prenantes qui le reçoivent. Il peut être créé pour les parties prenantes du management, de l'exploitation ou de la technique. Des informations supplémentaires peuvent être trouvées dans le syllabus du CTAL-TAE, section 6.1.3.

Lorsqu'un rapport d'automatisation des tests est reçu, il peut être incorporé dans un rapport de test plus large, ou il peut être consolidé, puis transmis à un niveau supérieur au sein de la structure organisationnelle.

Les parties prenantes n'accordent pas toutes la même valeur à un tel rapport d'automatisation des tests. Une approche stratégique consiste à identifier les métriques clés qui sont importantes pour les parties prenantes concernées en mettant l'accent sur ces métriques importantes.

Les données collectées grâce à l'automatisation peuvent aider à :

- Identifier les tendances et effectuer une analyse des causes racines.
- Déplacer les efforts d'automatisation des tests vers la maintenance.
- Orienter les efforts d'automatisation des tests vers l'amélioration et le développement d'un TAF.
- Ajouter des capacités à la TAS dans son ensemble.
- Augmenter les approches shift-left et shift-right des tests.
- Étendre la couverture fonctionnelle de l'automatisation des tests dans les futurs sprints.
- Se concentrer davantage sur les regroupements de défauts.
- Conseiller les développeurs sur les points à améliorer dans le code.
- Donner des conseils sur les processus généraux du cycle de vie logiciel.
- Modifier le contenu et le format des futurs rapports d'automatisation des tests.

Sur la base des informations décrites ci-dessus, les TAE, en collaboration avec d'autres parties prenantes, peuvent identifier les lacunes et certains points d'amélioration dans la couverture de l'automatisation des tests et les résultats de test existants.

6 Stratégies d'implémentation et d'amélioration de l'automatisation des tests – 150 minutes (K3)

Mots clés

couverture, précondition, suite de tests

Objectifs d'apprentissage pour le chapitre 6:

6.1 Activités de transition du test manuel vers le test en continu

CT-TAS-6.1.1 (K2) Décrire les facteurs et les activités de planification du passage du test manuel à l'automatisation des tests.

CT-TAS-6.1.2 (K2) Décrire les facteurs et les activités de planification dans la transition de l'automatisation des tests vers le test en continu.

6.2 Comprendre les facteurs et les activités de planification dans la transition de l'automatisation des tests vers le test en continu.

CT-TAS-6.2.1 (K3) Mener une évaluation des actifs et des pratiques d'automatisation des tests afin d'identifier les domaines d'amélioration.

6.1 Activités de transition du test manuel vers le test en continu

6.1.1 Décrire les facteurs et les activités de planification du passage du test manuel à l'automatisation des tests.

L'opportunité la plus simple pour passer du test manuel à l'automatisation des tests est de cibler les tests de régression. Une suite de tests de régression se développe au fur et à mesure que les tests fonctionnels et non fonctionnels d'aujourd'hui deviennent les tests de régression de demain. Ce n'est qu'une question de temps avant que le nombre de tests de régression ne dépasse le temps et les ressources dont dispose une équipe de test manuelle traditionnelle.

Coûts de transition

Pendant la transition du test manuel vers le test automatisé, le projet doit s'attendre à une augmentation des coûts, car les tests manuels et les tests automatisés ont lieu simultanément. Une fois que les tests automatisés sont considérés comme remplaçant adéquatement ou supplantant les activités d'exécution des tests manuels, davantage d'efforts peuvent être consacrés aux tests exploratoires et à la définition de cas de test supplémentaires pour l'automatisation des tests, ce qui a un coût différent par rapport aux tests de régression manuels.

Chevauchement fonctionnel

Il y a chevauchement fonctionnel lorsque les développeurs de scripts de test incluent exactement les mêmes étapes d'automatisation des tests dans différents cas de test. Par exemple, la plupart des cas de test commenceront par une séquence d'étapes de test de connexion. Il peut s'agir de la saisie d'un nom d'utilisateur, d'un mot de passe et de la sélection d'un bouton de connexion. L'ajouter dans chaque cas de test augmente les activités de maintenance. Si une étape de test supplémentaire est ajoutée au processus de test, le même changement devra être mis à jour dans chaque cas de test. Une meilleure approche consiste à faire du processus de connexion un composant d'automatisation des tests reproductible et à faire en sorte que tous les cas de test fassent référence à cette fonctionnalité. Voir le syllabus du CTAL-TAE, section 3.1.5 pour plus de détails sur le canevas du modèle de flux.

Partage des données

Les tests partagent souvent des données de test. Cela peut se produire lorsque les tests utilisent le même enregistrement de données de test pour exécuter différentes fonctionnalités du SUT. Par exemple, le cas de test "A" vérifie les vacances disponibles d'un employé, tandis que le cas de test "B" vérifie les cours suivis par l'employé dans le cadre de ses objectifs de développement de carrière. Chaque cas de test utilise le même employé mais vérifie des paramètres différents. Dans un environnement de test manuel, les données de test de l'employé seraient typiquement dupliquées plusieurs fois dans chaque cas de test manuel qui vérifie les données de test de l'employé. Toutefois, dans un test automatisé, les données de test qui sont partagées devraient, dans la mesure du possible, être stockées et accessibles à partir d'une source unique afin d'éviter la duplication ou l'introduction d'erreurs.

Interdépendance des tests

Lors de l'exécution de tests de régression complexes, un test peut dépendre d'un ou plusieurs autres tests. Cette situation peut être assez fréquente. Par exemple, un nouvel " ID de commande " est créé à la suite d'une étape du test. Les tests suivants peuvent vouloir vérifier que : a) la nouvelle commande est correctement affichée dans le système, b) il est possible de modifier la commande, ou c) la suppression de la commande est réussie. Dans chaque cas, la valeur " ID de la commande " qui est créée dynamiquement dans le premier test doit être capturée pour être réutilisée par les tests ultérieurs. Selon la conception de

la TAS, ce problème peut être résolu. Si les cas de test ultérieurs ne trouvent pas l'"ID de la commande", ils échoueront.

Les préconditions de l'exécution du test

Trop souvent, les TAE commencent immédiatement le développement des scripts de test sans comprendre les préconditions qui permettent de s'assurer qu'un cas de test peut être exécuté de manière fiable. Cela peut s'avérer extrêmement difficile lorsqu'il s'agit de tester l'exécution d'un cas de test sur le même SUT dans plusieurs environnements de test. Parmi les exemples de préconditions figurent les noms d'utilisateur, les rôles de compte, les valeurs des tables de données et d'autres données spécifiques qui rendent le cas de test reproductible. Une bonne stratégie comprendra une analyse en amont pour comprendre quelles informations doivent exister dans le SUT avant d'automatiser un cas de test particulier. En outre, la stratégie peut être améliorée en automatisant la création de préconditions avant que les cas de test réels ne soient exécutés, afin de gagner du temps. Il peut s'agir d'exécuter des scripts de test préconditionnés qui remplissent le SUT à partir de l'interface utilisateur ou des processus de lot automatisés qui chargent les données de test dans une base de données.

Couverture fonctionnelle

Identifiez les lacunes fonctionnelles dans les tests qui peuvent être candidats à l'automatisation des tests, comme expliqué dans le chapitre 3. 100% des cas de test manuels qui sont automatisés ne représentent pas 100% de tous les cas de test possibles qui peuvent être automatisés avec des outils de test. Au fur et à mesure que davantage de tests sont automatisés, les testeurs récupèrent du temps d'exécution de test qu'ils peuvent utiliser pour identifier des tests supplémentaires du SUT afin d'augmenter la couverture.

Tests exécutables

Avant de convertir un test de régression manuel en un test de régression automatisé, il est important de vérifier que le test de régression manuel fonctionne correctement. Cela fournit alors le point de départ correct pour assurer une conversion réussie en un test de régression automatisé. Si le test de régression manuel ne s'exécute pas correctement, c'est peut-être parce qu'il a été mal écrit, qu'il utilise des données de test non valides, qu'il n'est plus à jour ou qu'il n'est pas synchronisé avec le SUT actuel, ou encore en raison d'un défaut du SUT. L'automatisation de ce test avant de comprendre et/ou de résoudre la cause racine de la défaillance créera un test de régression automatisé non fonctionnel, ce qui est un gaspillage et une perte de productivité. Il est important de démontrer que les nouveaux tests automatisés apportent une fonctionnalité équivalente, afin de donner confiance dans les tests automatisés qui remplaceront les anciens tests manuels.

6.1.2 Décrire les facteurs et les activités de planification dans la transition de l'automatisation des tests vers le test en continu.

Le test en continu implique l'utilisation des ressources de test beaucoup plus fréquemment que dans les SDLC traditionnels. Pour ce faire, il faut commencer à exécuter en permanence des suites de tests immédiatement après que les modifications du code ont été effectuées et mises à disposition dans les environnements de test. Cela permet d'obtenir un retour d'information immédiat et de réduire le coût des défauts en les détectant plus tôt dans le processus. Cela peut se faire avec des outils de développement sophistiqués et la volonté de déplacer les tests plus tôt dans le processus de build.

L'adaptation de la TAS pour le test en continu exige ce qui suit :

- Les suites de tests doivent être modifiées pour être exécutées sur la TAS mise à jour : apportez les modifications nécessaires aux suites de tests et testez-les avant de les déployer sur la TAS.
- Les bouchons, pilotes et interfaces utilisés dans les tests doivent être modifiés pour s'adapter à la TAS mise à jour : apportez les changements nécessaires au harnais de test et testez-le avant de le déployer sur la TAS.
- L'infrastructure doit être modifiée pour s'adapter à la TAS mise à jour : faites un audit des composants de l'infrastructure qui doivent être modifiés.
- La TAS mise à jour présente des défauts supplémentaires ou des défauts de performance : effectuez une analyse des risques par rapport aux bénéfices. Si les défauts découverts rendent impossible la mise à jour de la TAS, il peut être préférable de ne pas procéder à la mise à jour ou d'attendre une prochaine version de la TAS. Si les défauts sont négligeables par rapport aux bénéfices qu'il y aurait à les corriger, la TAS peut encore être mise à jour.
- Veillez à créer des notes de livraison des défauts connus afin d'en informer les TAE et les autres parties prenantes et essayez d'obtenir une estimation de la date à laquelle les défauts seront corrigés.

Tous les points mentionnés dans cette section deviennent particulièrement importants lors de l'utilisation de CI/CD. La mise en place de pipelines et l'automatisation du processus de build s'inscrivent naturellement dans le cadre de la TAS en cours de développement. Si l'outil d'orchestration du build se trouve dans l'environnement de test adéquat, le pipeline peut être étendu pour inclure des tests automatisés afin de vérifier le SUT juste après le déploiement. Cela exige que l'outil d'automatisation des tests soit correctement configuré et puisse accéder au SUT, et qu'il puisse accéder au référentiel de code et aux scripts d'approvisionnement des données.

6.2 Stratégie d'automatisation des tests dans l'ensemble de l'organisation

6.2.1 Evaluation des actifs et des pratiques d'automatisation des tests afin d'identifier les domaines d'amélioration.

Comme pour toute autre activité de développement, il est important d'avoir une stratégie pour mettre en pause le développement de la TAS et rechercher des opportunités de refactoriser la solution. Les domaines à surveiller sont l'implémentation initiale, la maintenance et la capacité à évaluer la solution du point de vue de la reproductibilité. Le nombre d'heures consacrées au développement de la TAS, le nombre d'heures consacrées à la correction de la TAS et le gain de temps réalisé par les testeurs par rapport aux tests manuels sont autant d'éléments qu'il convient de surveiller. Si le temps consacré à la maintenance de la TAS est supérieur au temps consacré aux tests manuels, cela indique que quelque chose n'est pas correct.

Avant de procéder au premier déploiement d'une TAS, il est important de s'assurer qu'elle peut fonctionner dans son propre environnement, qu'elle est isolée des changements aléatoires et que les cas de test peuvent être mis à jour et gérés. La TAS et son infrastructure doivent être maintenues. Dans le cas d'un premier déploiement, les étapes de base suivantes sont nécessaires :

- Indiquer, au moyen des outils de couverture du code, dans quelle mesure le code est exercé par la suite de tests des composants, et où il existe des lacunes qui peuvent être couvertes par des tests de composants supplémentaires.

- Déterminer la couverture fonctionnelle du SUT en créant une matrice de traçabilité des exigences, qui permet de découvrir les fonctionnalités qui ne sont encore couvertes par aucun cas de test.
- Définir une utilisation cohérente de l'infrastructure de la TAS à travers les projets ou l'ensemble de l'organisation.
- Développer une stratégie de gestion de configuration cohérente pour les suites de tests.
- Créer des lignes directrices communes pour le développement des TAS en s'appuyant sur les meilleures pratiques décrites dans le syllabus du CTAL-TAE, section 3.1.4
- Implémenter des préconditions. Souvent, un test ne peut pas être exécuté avant la mise en place de préconditions. Ces préconditions peuvent inclure la sélection de la bonne base de données ou des données de test à partir desquelles tester ou la définition des valeurs ou paramètres initiaux. Un grand nombre de ces étapes d'initialisation requises pour établir les préconditions d'un test peuvent être automatisées. Cela permet d'obtenir une solution plus fiable et plus sûre lorsque ces étapes ne peuvent pas être omises avant l'exécution des tests. Lorsque les tests de régression sont convertis en automatisation des tests, ces préconditions doivent faire partie du processus d'automatisation des tests.

Lorsque des mises à jour incrémentales de la TAS ont lieu pour de nouvelles caractéristiques ou à des fins de maintenance, il convient d'envisager ce qui suit :

- Évaluer les mises à jour des outils de test ou d'autres outils de test plus récents qui peuvent fournir des capacités étendues pour la TAS.
- Évaluer les moyens d'optimiser davantage les caractéristiques et les performances de la TAS.
- Identifier les possibilités de décomposer davantage et de modulariser les scripts de test afin d'améliorer la facilité de réutilisation.
- Assurer la connaissance et la prise de conscience des composants réutilisables et de leur utilisation cohérente.
- Collecter des éléments probants sur les domaines d'amélioration potentiels et fournir des recommandations et leurs bénéfices.
- Évaluer et corriger les zones de chevauchement fonctionnel. Lors de l'automatisation des tests de régression existants, une bonne pratique consiste à identifier tout chevauchement fonctionnel existant entre les cas de test et, dans la mesure du possible, à réutiliser les composants d'automatisation des tests développés précédemment.
- Évaluer les cas de test manuels supplémentaires pour déterminer s'il est possible de les automatiser et créer des éléments du backlog pour l'implémentation.
- Mettre en évidence les possibilités d'amélioration de la conception des tests et de la gestion des données de test.
- Veiller à ce que toutes les suites de tests existantes soient adaptées à la dernière version de la TAS.
- Réduire le périmètre des tests automatisés aux plus critiques, si l'automatisation des tests entraîne une mise en file d'attente du pipeline ; c'est-à-dire créer une suite de smoke test. La suite de tests de régression plus large peut être déclenchée séparément ou exécutée à la demande.

7 Références

Normes

Les normes relatives à l'automatisation des tests sont notamment les suivantes:

The Automatic Test Markup Language (ATML) by IEEE (Institute of Electrical and Electronics Engineers) consisting of: ● IEEE Std 1671.1: Test Description ● IEEE Std 1671.2: Instrument Description ● IEEE Std 1671.3: UUT Description ● IEEE Std 1671.4: Test Configuration Description ● IEEE Std 1671.5: Test Adaptor Description ● IEEE Std 1671.6: Test Station Description ● IEEE Std 1641: Signal and Test Definition ● IEEE Std 1636.1: Test Results
ISO/IEC 30130:2016 (E) Software engineering — Capabilities of software testing tools
The Testing and Test Control Notation (TTCN-3) by ETSI (European Telecommunication Standards Institute) and ITU (International Telecommunication Union) consisting of: ● ES 201 873-1: TTCN-3 Core Language ● ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT) ● ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT) ● ES 201 873-4: TTCN-3 Operational Semantics ● ES 201 873-5: TTCN-3 Runtime Interface (TRI) ● ES 201 873-6: TTCN-3 Control Interface (TCI) ● ES 201 873-7: Using ASN.1 with TTCN-3 ● ES 201 873-8: Using IDL with TTCN-3 ● ES 201 873-9: Using XML with TTCN-3 ● ES 201 873-10: TTCN-3 Documentation ● ES 202 781: Extensions: Configuration and Deployment Support ● ES 202 782: Extensions: TTCN-3 Performance and Real-Time Testing ● ES 202 784: Extensions: Advanced Parameterization ● ES 202 785: Extensions: Behaviour Types ● ES 202 786: Extensions: Support of interfaces with continuous signals ● ES 202 789: Extensions: Extended TRI
The UML Testing Profile (UTP) by OMG (Object Management Group) specifying test specification concepts for: ● Test Architecture ● Test Data ● Test Behavior ● Test Logging ● Test Management

ISTQB® Documents

Identifiant	Référence
ISTQB-AL-TM	ISTQB Certified Tester, Advanced Level Syllabus, Test Manager, Version 2.0, October 2012, available from [ISTQB-Web] ISTQB Testeur certifié, syllabus de niveau Avancé, Test Manager, version 2.0, octobre 2012, disponible en français sur le site du CFTL à l'adresse suivante - Documents Associés Aux Certifications - CFTL
ISTQB-EL-TM-MTT	ISTQB Certified Tester, Expert Level Test Management Managing the Test Team, Version 2.0, November 2011, available from [ISTQB-Web]
ISTQB-FL	ISTQB Certified Tester, Foundation Level Syllabus, Version 4.0, April 2023, available from [ISTQB-Web] ISTQB Testeur certifié, Syllabus de niveau Fondation, version 4.0, avril 2023, disponible en français sur le site du CFTL à l'adresse suivante - Documents Documents Associés Aux Certifications - CFTL
ISTQB-PT	ISTQB Certified Tester, Performance Testing Syllabus, December 2018, available from [ISTQB-Web] Testeur certifié de l'ISTQB, Syllabus Tests de performance, décembre 2018, disponible en français sur le site du CFTL à l'adresse suivante - Documents Associés Aux Certifications - CFTL
ISTQB-TAE	ISTQB Certified Tester, Test Automation Engineering Syllabus, February 2024, available from [ISTQB-Web] Testeur certifié de l'ISTQB, Syllabus Automatisation des tests - Ingénierie, février 2024, disponible en français sur le site du CFTL à l'adresse suivante - Documents Associés Aux Certifications - CFTL
ISTQB-Glossary	ISTQB Glossary of terms, available online from [ISTQB-Web] Glossaire de l'ISTQB, en français également sur [ISTQB-Web]

Livres

Paul Baker, Zhen Ru Dai, Jens Grabowski and Ina Schieferdecker, “Model-Driven Testing: Using the UML Testing Profile”, Springer 2008 edition, ISBN-10: 3540725628, ISBN-13: 978-3540725626
Efriede Dustin, Thom Garrett, Bernie Gauf, “Implementing Automated Software Testing: how to save time and lower costs while raising quality”, Addison-Wesley, 2009, ISBN 0-321-58051-6
Efriede Dustin, Jeff Rashka, John Paul, “Automated Software Testing: introduction, management, and performance”, Addison-Wesley, 1999, ISBN-10: 0201432870, ISBN-13: 9780201432879
Mark Fewster, Dorothy Graham, “Experiences of Test Automation: Case Studies of Software Test Automation”, Addison-Wesley, 2012
Mark Fewster, Dorothy Graham, “Software Test Automation: Effective use of test execution tools”, ACM Press Books, 1999, ISBN-10: 0201331403, ISBN-13: 9780201331400
Boby Jose, “Test Automation, a manager's guide”, September 2021, ISBN: 9781780175478
James D. McCaffrey, “.NET Test Automation Recipes: A Problem-Solution Approach”, APRESS, 2006 ISBN-13:978-1-59059-663-3, ISBN-10:1-59059-663-3

Daniel J. Mosley, Bruce A. Posey, “Just Enough Software Test Automation”, Prentice Hall, 2002, ISBN-10: 0130084689, ISBN-13: 9780130084682
Casey Rosenthal, “Chaos Engineering: System Resiliency in Practice by Casey Rosenthal”, April 2020, ISBN-13: 1492043869
Colin Willcock, Thomas Deiß, Stephan Tobies and Stefan Keil, “An Introduction to TTCN-3” Wiley, 2nd edition 2011, ISBN-10: 0470663065, ISBN-13: 978-0470663066

Articles

Robert V. Binder, Suzanne Miller, “Five Keys to Effective Agile Test Automation for Government Programs” August 24, 2017, Software Engineering Institute, Carnegie Mellon University, https://resources.sei.cmu.edu/asset_files/Webinar/2017_018_101_503516.pdf
DoD CIO, Modern Software Practices “DevSecOps Fundamentals Guidebook: Activities & Tools”, Version 2.2, May 2023, https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsActivitesToolsGuidebookTables.pdf?ver=_Sylg1WJB9K0Jxb2XTvzDQ%3d%3d
Naveen Jayachandran, “Understanding roi metrics for software test automation”, 2005, https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=3937&context=etd
Thomas Pestak, William Rowell, PhD, “Automated Software Testing Practices and Pitfalls”, September 2018, https://www.afit.edu/stat/statcoe_files/Automated%20Software%20Testing%20Practices%20and%20Pitfalls%20Rev%201.pdf
Andrew Pollner, Jim Simpson, Jim Wisnowski, “Automated Software Testing Implementation Guide for Managers and Practitioners”, October 2018, https://www.afit.edu/stat/statcoe_files/0214simp%202%20AST%20IG%20for%20Managers%20and%20Practitioners.pdf

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

Les objectifs d'apprentissage suivants sont définis comme s'appliquant à ce syllabus. Chaque sujet du syllabus sera examiné en fonction de l'objectif d'apprentissage qui lui est associé.

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

Niveau 2 : Comprendre (K2)

Le candidat peut sélectionner les raisons ou les explications des instructions liées 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
Classer les outils de test en fonction de leur objectif et des activités de test qu'ils soutiennent.	
Comparer les différents niveaux de test.	Peut être utilisé pour rechercher des similitudes, des différences ou les deux.
Différencier les tests du débogage.	Recherche de différences entre les concepts.
Faire la distinction entre les risques projet et les risques produit.	Permet de classer séparément deux concepts (ou plus).
Expliquer l'impact du contexte sur le processus de test.	
Donner des exemples de raisons pour lesquelles les tests sont nécessaires.	
Déduire la cause racine des défauts à partir d'un profil donné de défaillances.	
Résumer les activités du processus de revue des produits d'activités.	

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 l'analyse des valeurs limites pour dériver des cas de test à partir d'exigences données.	Doit faire référence à une procédure / une technique / un processus, etc.
Implémenter des méthodes de collecte de métriques pour soutenir les exigences techniques et de management.	
Préparer des tests de facilité d'installation pour les applications mobiles.	
Utiliser la traçabilité pour contrôler la progression des tests afin de s'assurer de leur complétude et de leur cohérence avec les objectifs du test, la stratégie de test et le plan 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".

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

9 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 de l'Automatisation des tests - Stratégie.

Objectifs métier d'automatisation des tests - stratégie			B01	B02	B03	B04	B05	B06	B07	B08	B09	B10	B11	B12	B13	B14	B15
Chapitre 1	Introduction et objectifs de la stratégie d'automatisation des tests																
1.1	Facteurs de succès d'un projet d'automatisation des tests																
1.1.1	Expliquer les objectifs et la pertinence de l'automatisation des tests	2	X														
1.1.2	Identifier les facteurs de succès techniques d'un projet d'automatisation des tests	2	X														
1.1.3	Résumer les critères d'investissement appropriés dans la sélection des projets candidats à l'automatisation des tests	2	X														
Chapitre 2	Ressources en matière d'automatisation des tests																
2.1	Coûts et risques liés à l'implémentation d'une solution d'automatisation des tests																
2.1.1	Comparer les solutions techniques alternatives en termes de coût de possession	2		X													
2.1.2	Expliquer les considérations relatives au modèle de licence pour les outils d'automatisation des tests	2		X													
2.1.3	Fournir des exemples de facteurs à prendre en compte lors de la définition d'une stratégie d'automatisation des tests	2		X													
2.2	Rôles et responsabilités dans l'automatisation des tests																

Objectifs métier d'automatisation des tests - stratégie			B01	B02	B03	B04	B05	B06	B07	B08	B09	B10	B11	B12	B13	B14	B15
2.2.1	Résumer les rôles et les compétences nécessaires pour une solution d'automatisation des tests réussie	2			X												
Chapitre 3	Préparation à l'automatisation des tests																
3.1	Intégration aux niveaux de test																
3.1.1	Différencier les distributions de l'automatisation des tests	2				X											
3.1.2	Sélectionner une approche d'automatisation des tests en fonction de l'architecture du système sous test	3				X											
3.1.3	Démontrer comment optimiser la distribution de l'automatisation des tests pour obtenir des approches shift-left et shift-right	2				X											
3.2	Considérations stratégiques dans les différents modèles de cycle du développement logiciel																
3.2.1	Expliquer comment les projets d'automatisation des tests se conforment aux anciens modèles de cycle de vie du développement logiciel	2					X										
3.2.2	Expliquer comment les projets d'automatisation des tests sont conformes aux meilleures pratiques de développement logiciel en mode Agile qui soutiennent l'automatisation des tests	2					X										
3.2.3	Préparer la conformité des projets d'automatisation des tests avec les meilleures pratiques DevOps qui soutiennent l'automatisation des tests en continu	3					X										
3.3	Applicabilité et viabilité de l'automatisation des tests																
3.3.1	Expliquer les critères pour déterminer l'adéquation des tests à l'automatisation des tests	2						X									

Objectifs métier d'automatisation des tests - stratégie			B01	B02	B03	B04	B05	B06	B07	B08	B09	B10	B11	B12	B13	B14	B15
3.3.2	Identifier les défis que seule l'automatisation des tests peut relever	2						X									
3.3.3	Identifier les conditions de test qui sont difficiles à automatiser	2						X									
Chapitre 4	Stratégies d'automatisation des tests pour le déploiement et la mise en production au sein de l'organisation																
4.1	Planification de la solution d'automatisation des tests																
4.1.1	Identifier les façons dont l'automatisation des tests soutient un délai de mise sur le marché plus court	2							X								
4.1.2	Identifier les façons dont l'automatisation des tests aide à vérifier que les défauts rapportés sont conformes aux exigences	2							X								
4.1.3	Définir des approches qui permettent le développement de scénarios pertinents sur le plan de l'exploitation pour l'automatisation des tests.	2							X								
4.2	Stratégies de déploiement																
4.2.1	Définir une stratégie d'automatisation des tests de déploiement	2								X							
4.2.2	Identifier les risques liés à l'automatisation des tests lors du déploiement	2								X							
4.2.3	Définir des approches pour atténuer les risques de déploiement	2								X							
4.3	Dépendances dans l'environnement de test																
4.3.1	Définir les composants de l'automatisation des tests dans l'environnement de test	2									X						

Objectifs métier d'automatisation des tests - stratégie			B01	B02	B03	B04	B05	B06	B07	B08	B09	B10	B11	B12	B13	B14	B15
4.3.2	Identifier les composants de l'infrastructure et les dépendances de l'automatisation des tests	2									X						
4.3.3	Définir les exigences en matière de données et d'interfaces de l'automatisation des tests pour l'intégration dans le système sous test	2									X						
Chapitre 5	Analyse d'impact de l'automatisation des tests																
5.1	Investissement dans la mise en place et la maintenabilité de l'automatisation des tests																
5.1.1	Montrer le retour sur investissement de la construction d'une solution d'automatisation des tests	3										X					
5.2	Métriques de l'automatisation des tests																
5.2.1	Classer les métriques pour l'automatisation des tests	2											X				
5.3	La valeur de l'automatisation des tests au niveau du projet et de l'organisation																
5.3.1	Identifier les considérations organisationnelles pour l'utilisation de l'automatisation des tests	3												X			
5.3.2	Analyser les caractéristiques du projet qui aident à déterminer l'implémentation optimale des objectifs de test de l'automatisation des tests	3												X			
5.4	Décisions prises à partir des rapports d'automatisation des tests																
5.4.1	Analyser les données des rapports de test pour éclairer la prise de décision	2													X		
Chapitre 6	Stratégies d'implémentation et d'amélioration de l'automatisation des tests																
6.1	Transition des activités de test manuel vers le testsen continu																

Objectifs métier d'automatisation des tests - stratégie			B01	B02	B03	B04	B05	B06	B07	B08	B09	B10	B11	B12	B13	B14	B15
6.1.1	Décrire les facteurs et les activités de planification de la transition du test manuelsvers l'automatisation des tests.	2														X	
6.1.2	Décrire les facteurs et les activités de planification dans la transition de l'automatisation des tests vers les tests continus.	2														X	
6.2	Stratégie d'automatisation des tests dans l'ensemble de l'organisation																
6.2.1	Mener une évaluation des actifs et des pratiques d'automatisation des tests afin d'identifier les domaines d'amélioration.	3															X

10 Appendice C – Notes de version

Le syllabus ISTQB® 2024 sur l'automatisation des tests - stratégie est un nouveau syllabus de l'ISTQB® qui combine les aspects stratégiques de la version précédente du syllabus de l'automatisation des tests - ingénierie de 2016 avec des mises à jour supplémentaires et les meilleures pratiques actuelles pour implémenter et mesurer le succès de l'automatisation des tests. Pour cette raison, il n'y a pas de notes de version détaillées par chapitre et section.

11 Appendice D – Termes spécifiques au domaine

Nom du terme	Définition
version canari	Stratégie de déploiement et de test visant à réduire les risques et à vérifier les nouveaux logiciels en ne les livrant qu'à un petit nombre d'utilisateurs.
conteneur	Unité de logiciel qui conditionne le code et ses dépendances, de sorte que le logiciel s'exécute rapidement et de manière fiable dans tous les environnements.
DevOps	Méthodologie qui intègre et automatise le travail du développement logiciel et des opérations informatiques afin d'améliorer et de raccourcir le cycle de vie du développement logiciel.
canevas de modèle de flux	Une vue de haut niveau du domaine de travail, de ses composants et de leurs interconnexions.

12 Index

Tous les termes sont définis dans le glossaire ISTQB® (http://glossary.istqb.org/).