Testeur Certifié

Syllabus Niveau Avancé

Version 2007FR

International Software Testing Qualifications Board

Comité Français des Tests Logiciels

Groupe de travail niveau avancé : Bernard Homès (président), Graham Bath, Rex Black, Sigrid Eldh, Jayapradeep Jiothis, Paul Jorgensen, Vipul Kocher, Judy McKay, Klaus Olsen, Randy Rice, Jürgen Richter, Eric Riou du Cosquer, Mike Smith, Geoff Thompson, Erik Van Veenendaal; 2006-2007.

Traduction française : Eric Riou du Cosquer (responsable), Alain Ribault, Gilles Douat, Arnaud Poivre, Anna Stevenson, Jean-Marie Hallouet, Bertrand Cornanguer, Cyrille Riviere, Olivier Denoo, Bertrand Batoge, Annabelle Ayrault, Bernard Homès.

Historique des modifications

Version	Date	Remarques
ISEB v1.1	04SEP01	ISEB Practitioner Syllabus
ISTQB 1.2E	SEP03	ISTQB Advanced Level Syllabus from EOQ-SG
V2007	12OCT07	ISTQB Certified Tester Advanced Level syllabus version 2007
V2007FR	31DEC08	Testeur Certifié Niveau Avancé, version Française

Table des matières

Remerciements

Ce document a été produit par le groupe de travail niveau avancé de l'International Software Testing Qualification Board: Bernard Homès (chairman), Graham Bath, Rex Black, Sigrid Eldh, Jayapradeep Jiothis, Paul Jorgensen, Vipul Kocher, Judy McKay, Thomas Mueller, Klaus Olsen, Randy Rice, Jürgen Richter, Eric Riou du Cosquer, Mike Smith, Geoff Thompson, Erik Van Veenendaal.

Le groupe de travail remercie l'équipe de revue et tous les comités nationaux pour leurs suggestions et apports.

Au moment de la finalisation du syllabus avancé, le groupe de travail du niveau avancé avait pour membres:(par ordre alphabétique):

Graham Bath, Rex Black, Robert Bender, Chris Carter, Maria Clara Choucair, Sigrid Eldh, Dorothy Graham, Bernard Homès (chair), Jayapradeep Jiothis, Vipul Kocher, Anastasios Kyriakopoulos, Judy McKay, Thomas Mueller, Klaus Olsen, Avinoam Porat, Meile Posthuma, Erkki Pöyhönen, Jürgen Richter, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Maud Schlich, Rajesh Sivaraman, Mike Smith, Michael Stahl, Geoff Thompson, Erik Van Veenendaal.

Les personnes suivantes ont participé aux revues, commentaires et votes pour ce syllabus:

Bernard Homès (chair)

Reto Armuzzi
Sue Atkins
Graham Bath
Paul Beekman
Armin Beer
Rex Black
Francisca Blunschi
Armin Born
Con Bracke
Chris Carter
Maria Clara Choucair
Robert Dankanin
Piet de Roo
Sigrid Eldh
Tim Edmonds
Erwin Engelsma
Graham Freeburn
Dorothy Graham
Brian Hambling
Jeff B Higgott
Bernard Homès
Rob Hendriks
Dr Suhaimi Ibrahim
Phillip Isles
Pr. Paul C. Jorgensen
Vipul Kocher
Anastasios Kyriakopoulos
Junfei Ma
Fergus McClachlan
Judy McKay
Don Mills
Gary Mogyorodi
Richard Morgan
Silvio Moser
Ernst Müller
Reto Müller
Thomas Müller
Peter Mullins
Beat Nagel
Richard Neeve
Klaus Olsen
Dale Perry
Helmut Pichler
Jörg Pietzsch
Avionam Porat
Iris Pinkster
Horst Pohlmann
Meile Posthuma
Eric Riou du Cosquer
Stefan Ruff
Hans Schaefer
Maud Schlich
Rajesh Sivaraman
Mike Smith
Katja Stalder
Neil Thompson
Benjamin Timmermans
Chris van Bael
Jurian van de Laar
Marnix van den Ent
Mark van der Zwan
Stephanie van Dijck
Jan van Moll
Erik Van Veenendaal
Roland Weber
Phillip Whettlock
Derek Young
Mike Young
Wenqiang Zheng.

Ce document a été officiellement validé lors de l'assemblée générale de l'ISTQB le 12 octobre 2007.

0. Introduction à ce syllabus

0.1 L'ISTQB

L'International Software Testing Qualifications Board (ci-après nommé ISTQB®) est constitué de Comités Membres représentant des pays ou régions du monde entier. à la date de la publication de ce syllabus, l'ISTQB® était constitué de 33 comités membres. Davantage de détails sur la structure et les membres de l'ISTQB peuvent être trouvés sur www.istqb.org.

Objectifs de ce document

Ce syllabus forme la base de la Qualification Internationale en Tests de Logiciels au Niveau Avancé. L'ISTQB® fourni ce syllabus comme suit :

1. Aux Comités Membres, pour traduction dans leur langue et pour l'accréditation des fournisseurs de formation. Les Comités Nationaux peuvent adapter le syllabus aux particularités de leur langage et modifier les références pour les adapter à leurs publications locales.
2. Aux Comités d'Examens, pour en dériver des questions d'examen adaptées aux objectifs d'apprentissage de chaque module dans la langue locale.
3. Aux fournisseurs de formation, pour concevoir les cours et déterminer les méthodes de formation appropriées.
4. Aux candidats à la certification, pour préparer l'examen (dans le cadre d'une formation ou indépendamment).
5. A la communauté internationale du logiciel et de l'ingénierie des systèmes, pour faire progresser la profession de testeur de logiciels et de systèmes, et comme base pour des livres et articles.

L'ISTQB® peut autoriser d'autres entités à utiliser ce syllabus pour d'autres objectifs à condition qu'elles demandent et obtiennent une autorisation écrite préalable.

Le Testeur Certifié Niveau Avancé en Test de Logiciels

La qualification Niveau Avancé s'adresse aux personnes ayant atteint un stade avancé dans leur carrière de testeur de logiciels. Cela inclus des personnes ayant des rôles tels que testeur, analyste de test, ingénieur en test, consultant en test, gestionnaire de test, testeur en test d'acceptation et développeurs de logiciels. Cette qualification Niveau Avancé est également adéquate pour toute personne souhaitant acquérir une compréhension plus approfondie du test de logiciels, comme les chefs de projets, les responsables qualité, les responsables d'équipes de développement, les analystes métier, les directeurs IT et les consultants en management. Pour obtenir la certification Niveau Avancé, les candidats doivent détenir la certification Niveau Fondation et satisfaire aux conditions du Comité d'Examen concerné en ce qui concerne l'expérience pratique requise pour pouvoir être qualifié au Niveau Avancé. Veuillez vous adresser a votre Comité d'Examen ou Comité National respectif pour connaître les critères spécifiques sur l'expérience pratique requise.

Niveaux de connaissance

Des niveaux de connaissance pour chaque chapitre sont répartis de façon à être clairement identifiés pour chaque module. Des détails et exemples d'objectifs d'apprentissage sont donnés en section 0.3.

Le contenu du syllabus, les termes et les principaux éléments (objectifs) de tous les standards listés doivent au minimum être acquis au niveau K1 (Se souvenir) même si cela n'est pas mentionné dans les objectifs de formation.

Examen

Touts les examens de Certification Avancée doivent être basés sur ce syllabus et sur le syllabus Niveau Fondation. Les réponses aux questions peuvent demander l'utilisation d'éléments présentés dans

différentes sections des syllabus. Toutes les sections de ce syllabus et du syllabus Niveau Fondation peuvent être examinés.

Le format d'examen est défini par les consignes d'examen avancé de l'ISTQB®. Un Comité National peut adopter d'autres schémas d'examen s'il le souhaite.

Les examens peuvent être passés à l'issue d'une formation accréditée ou de façon indépendante (par exemple dans un centre d'examen). Les examens peuvent être passés par écrit ou sur ordinateur mais ils doivent impérativement se dérouler sous la surveillance et selon les consignes d'une personne mandatée par le Comité National ou d'Examen.

Accréditation

Un Comité Membre de l'ISTQB® peut accréditer des organismes de formation dont le matériel des cours respecte ce syllabus. Les organismes de formation doivent obtenir les consignes d'accréditation auprès du comité national ou de l'instance en charge d'exécuter les accréditations. Un cours accrédité est officiellement reconnu conforme à ce syllabus et est autorisé à inclure un examen de l'ISTQB® come partie de ce cours.

D'autres précisions pour les organismes de formation sont données en Annexe C - Notes pour les organismes de formation.

Niveau de détail

Le niveau de détail dans ce syllabus permet la mise en place de formations et examens internationaux cohérents. A cette fin, le syllabus contient :

- Des objectifs d'apprentissage généraux décrivant l'intention du Niveau Avancé
- Des objectifs d'apprentissage pour chaque domaine de connaissance, décrivant les résultats cognitifs d'enseignement et la mentalité à acquérir
- Une liste des informations à enseigner incluant une description, et des références à des sources additionnelles si besoin
- Une liste de termes que les étudiants doivent se rappeler et avoir compris
- Une description des concepts clé à enseigner, incluant des sources comme des normes ou de la littérature reconnue.

Le contenu du syllabus n'est pas une description de l'ensemble du domaine de connaissance en test de logiciels; il reflète le niveau de détail devant être couvert par les cours et formations du Niveau Avancé.

Organisation du syllabus

Il y a 10 chapitres majeurs, chacun avec une introduction qui indique comment ils s'appliquent aux différents types de professionnels des tests (modules).

Pour les besoins des formations, les sections 0.3 à 0.6 fournissent des informations sur les objectifs d'apprentissage pour chaque module, par chapitre. Ces sections indiquent également le temps minimum devant être affecté à la formation de ces sujets.

Il est fortement recommandé de lire le syllabus et prendre connaissance des objectifs d'apprentissage du chapitre concerné. Ceci permettra au lecteur de bien comprendre ce qui est attendu et ce qui est essentiel pour chaque chapitre et pour chacun des trois modules.

Termes & Définitions

Plusieurs termes utilisés dans la littérature le sont de façon interchangeable. Les définitions des termes utilisés dans ce Syllabus Niveau Avancé sont disponibles dans le Glossaire des termes utilisés en test logiciel publié par l'ISTQB® et le Comité Français du Test Logiciel1.

Approche

Le test peut être abordé de plusieurs façons, comme celles basées sur les spécifications, la structure du code, les données, les risques, les processus, les standards et listes de taxonomie similaires. Plusieurs

[1] Ce glossaire publié par le CFTL est la traduction du glossaire officiel de l'ISTQB® et donne également la correspondance entre les termes anglais et leur traduction en français

processus et outils offrent du support aux processus de test ; des méthodes sont disponibles pour améliorer les processus déjà en place.

Ce Syllabus Niveau Avancé est organisé selon les approches proposées dans la norme ISO 9126, avec une séparation entre les approches fonctionnelles, non fonctionnelles et de support. Des processus de support et des méthodes d'amélioration sont mentionnées. La sélection de ces approches, processus et méthodes est faite selon une base arbitraire destinée à fournir une base solide aux testeurs et gestionnaires de tests du Niveau Avancé.

0.2 Attentes

La certification niveau avancé décrite dans ce syllabus sera examinée selon trois descriptions de tâches principales, chacune représentant les responsabilités de base et les attentes au sein d'une organisation. Dans toute organisation, les responsabilités et tâches associées peuvent être réparties entre différents individus ou assurées par une seule personne. Les différentes responsabilités sont présentées ci-dessous.

0.2.1 Gestionnaire de Tests Niveau Avancé.

Les professionnels, Gestionnaires de Tests Niveau Avancé, doivent être capables de :

- Définir les objectifs de test globaux et la stratégie pour les systèmes à tester
- Planifier, ordonner et suivre les tâches
- Décrire et organiser les activités nécessaires
- Sélectionner, acquérir et affecter les ressources adéquates aux tâches
- Sélectionner, organiser et diriger les équipes de test
- Organiser la communication entre les membres des équipes de tests, et entre les équipes de tests et les autres parties prenantes
- Justifier les décisions et fournir les informations de reporting quand nécessaire

0.2.2 Analyste de Tests Niveau Avancé.

Les Analystes de Tests Niveau Avancé doivent être capables de :

- Structurer les tâches définies dans la stratégie de test en termes d'exigences et domaines métier
- Analyser le système avec un niveau de détail suffisant pour répondre aux attentes de l'utilisateur sur la qualité
- Évaluer les exigences du système pour déterminer le domaine de validité
- Préparer et exécuter les activités adéquates, et communiquer sur leur avancement
- Fournir les preuves nécessaires pour supporter les évaluations
- Implémenter les outils et techniques nécessaires pour atteindre les objectifs définis.

0.2.3 Analyste Technique de Test Niveau Avancé.

Les Analystes de Test Technique Niveau Avancé doivent être capables de :

- Structurer les tâches définies dans la stratégie de test en fonction des exigences techniques
- Analyser la structure interne du système avec le niveau de détail suffisant pour répondre au niveau de qualité attendu
- Évaluer le système en terme de caractéristiques techniques de qualité telles que la performance, la sécurité, etc.
- Préparer et exécuter les activités adéquates, et communiquer sur leur avancement
- Piloter et exécuter les activités de test technique
- Fournir les preuves nécessaires pour supporter les évaluations
- Implémenter les outils et techniques nécessaires pour atteindre les objectifs définis.

0.3 Objectifs d'Apprentissage / Niveau de Connaissance

Les objectifs d'apprentissage suivant sont définis selon leur utilisation dans ce syllabus. Chaque sujet du syllabus sera examiné selon l'objectif d'apprentissage qui lui est associé.

Niveau 1: Se souvenir (K1)
Le candidat doit reconnaître et se souvenir d'un terme ou un concept.
Mots-clés: Se souvenir, reconnaître, mémoriser, savoir.
Exemple :
Peut reconnaître la définition d'une "défaillance" comme:

- « la non délivrance du service à un utilisateur final ou à tout autre partie prenante » ou
- « une divergence constatée du composant ou du système par rapport au résultat, à la sortie, ou au service attendu »

Niveau 2: Comprendre (K2)
Le candidat peut sélectionner les raisons ou explications pour des affirmations liées au sujet traité, et peut résumer, différencier, classer et donner des exemples réels (par exemple comparer des termes) sur les concepts ou procédures de test (expliquer des séquences de tâches).
Mots-clés: Résumer, classer, comparer, relier, opposer, donner des exemples, interpréter, traduire, représenter, déduire, conclure.
Exemples :
Peut expliquer les raisons pour lesquelles les tests doivent être exécutés aussi tôt que possible :

- Pour trouver des défauts quand il est meilleur marché de les supprimer.
- Pour trouver d'abord les défauts les plus importants.

Peut expliquer les similitudes et les différences entre les tests d'intégration et les tests système :

- Similitudes : tester plus d'un composant, et peut tester des aspects non-fonctionnels.
- Différences : les tests d'intégration se concentrent sur les interfaces et les interactions, les tests système se focalisent sur les aspects de systèmes entiers, tels le processus de bout en bout.

Niveau 3: Appliquer (K3)
Le candidat peut sélectionner l'application correcte d'un concept ou d'une technique et l'appliqué à un contexte donné. K3 s'applique normalement à la connaissance de la procédure. Cela n'implique pas d'actes créatifs comme l'évaluation d'une application logicielle ou la création d'un modèle pour un logiciel particulier. Lorsque l'on considère un modèle donné et que l'on couvre dans le syllabus les étapes de la procédure permettant de créer des cas de test à partir du modèle, il s'agit du niveau K3.
Mots-clés: Implémenter, exécuter, utiliser, suivre une procédure, appliquer une procédure
Exemples:

- Peut identifier les valeurs limites pour des transitions valides et invalides.
- Peut utiliser la procédure générique de création des cas de test pour sélectionner les cas de test à partir d'un diagramme de transition d'état (et un ensemble de cas de test) de façon à couvrir toutes les transitions.

Niveau 4: Analyser (K4)
Le candidat peut décomposer l'information relative à la procédure ou à la technique en différentes parties pour une meilleure compréhension et peut faire la différence entre des faits et les conclusions qui en sont tirées. Une application classique est de proposer, après l'analyse d'un document, d'un logiciel ou de la situation d'un projet, les actions appropriées pour résoudre un problème ou une tâche.
Mots-clés: Analyser, différencier, sélectionner, structurer, cibler, attribuer, déconstruire, évaluer, juger, contrôler, coordonner, créer, synthétiser, générer, supposer, planifier, plan, concevoir, construire, produire.
Exemple :

- Analyser les risques produit et proposer des activités correctives et préventives pour les atténuer
- Décrire quelles parties d'un rapport d'incident sont factuelles et quelles parties ont été déduites des résultats.

Références (pour les niveaux cognitifs des objectifs d'apprentissage)
Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain,* David McKay, Co. Inc.
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.

0.4 Objectifs d'Apprentissage pour les Gestionnaires de Test

Cette section fournit la liste d'objectifs d'apprentissage détaillés pour le module Gestionnaire de test. En général, toutes les parties de ce syllabus sont examinables au niveau K1. Cela signifie que le candidat doit reconnaître, se souvenir de et mémoriser un terme ou un concept. C'est pour cela que seuls les objectifs de niveau K2, K3 et K4 sont mentionnés dans la suite de cette section.

Introduction au syllabus pour le gestionnaire de test – [60 minutes]
(Intégrant une révision du syllabus niveau fondation de l'ISTQB)

Chapitre 1: Éléments de base du test logiciel – [150 minutes]
1.2 Le test dans le cycle de vie du logiciel
- (K2) Décrire comment le test fait partie intégrante de toute activité de développement logiciel ou de maintenance.
- (K4) Analyser les modèles de cycles de vie logiciel et identifier les tâches et activités de test les plus appropriées à exécuter (faire la distinction entre le test et les activités de développement)

1.3 Systèmes spécifiques
- (K2) Expliquer en donnant des exemples les spécificités du test de systèmes de systèmes
- (K2) Expliquer pourquoi les trois principales issues du test de systèmes critiques du point de vue de la sécurité sont requises pour montrer la conformité aux règlementations

1.4 Métriques et mesurage
- (K2) Décrire et comparer les principales métriques relatives au test.
- (K3) Contrôler les activités de test en mesurant les objets et le processus de test.

Chapitre 2: Processus de test – [120 minutes]
2.3 Planification et contrôle des tests
- (K2) Décrire en donnant des exemples, comment les stratégies de test affectent la planification des tests
- (K2) Comparer les livrables du test et expliquer, en donnant des exemples, les relations entre les livrables du test et ceux du développement
- (K2) Classifier les activités de contrôle des tests destinées à déterminer si la mission du test, ses stratégies et ses objectifs ont été atteints

2.5 Implémentation et exécution des tests
- (K2) Expliquer les pré-conditions à l'exécution des tests
- (K2) Expliquer, en donnant des exemples, les avantages et inconvénients d'une implémentation des tests réalisée tôt, en considérant différentes techniques de test.
- (K2) Expliquer pourquoi les utilisateurs et/ou les clients peuvent être impliqués dans l'exécution des tests
- (K2) Décrire comment le niveau des logs de test peut varier selon le niveau de test

2.6 Évaluer les critères de sortie et informer
- (K2) Résumer l'information qu'il est nécessaire de collecter pendant le processus de test pour permettre une documentation et une évaluation liée aux critères de sortie

2.7 Activités de clôture des tests
- (K2) Résumer les quatre groupes d'activité de clôture des tests
- (K3) Capitaliser sur les retours d'expérience pendant la phase de clôture des tests afin d'identifier des domaines à améliorer ou à répéter

Chapitre 3: Gestion des tests – [1120 minutes]
3.2 Documentation de la gestion des tests
- (K4) Produire les documents de la gestion des tests tels que le plan de test, les spécifications de conception des tests et les procédures de test, en accord avec l'IEEE 829
- (K2) Décrire au moins 4 éléments importants dans la stratégie/approche de test et identifier les documents contenants ces éléments selon l'IEEE 829
- (K2) Illustrer comment et pourquoi des déclinaisons de la stratégie de test peuvent être gérées dans d'autres documents de la gestion des tests

3.3 Modèles de document de plan de test
- (K2) Résumer la structure du plan de test maître de l'IEEE 829

- (K2) Reformuler et interpréter les sujets introduits dans la composition du plan de test par la norme IEEE 829 en respectant l'adaptation à une organisation, aux risques produits et aux risques, à la taille et à la formalité d'un projet

3.4 Estimation des tests

- (K3) Estimer l'effort de test sur l'exemple d'un petit système en utilisant une approche basée sur les métriques et sur l'expérience et en prenant en compte les facteurs ayant une influence sur le coût, la charge et la durée.
- (K2) Comprendre et donner des exemples pour les facteurs listés dans le syllabus qui peuvent provoquer des erreurs dans les estimations

3.5 Planification des tests

- (K2) Expliquer les bénéfices d'une planification organisée tôt et de façon itérative. Assortir l'explication d'exemples.

3.6 Contrôle et surveillance de la progression des tests

- (K2) Comparer les différentes procédures pour contrôler la progression des tests
- (K2) Donner au moins 5 exemples conceptuellement différents sur la façon avec laquelle les informations sur la progression des tests influencent le processus de test.
- (K4) Utiliser les découvertes observées pendant les activités et mesures liées au contrôle et à la surveillance afin de mettre en place un plan d'action d'amélioration du processus de test actuel. Proposer des améliorations.
- (K4) Analyser les résultats de test et déterminer la progression des tests, ceci documenté dans un rapport de contrôle et dans un rapport de test final couvrant les 4 dimensions du reporting.

3.7 Valeur commerciale des tests

- (K2) Donner des exemples (mesures) pour chacune des 4 catégories déterminant le "coût de la qualité"
- (K3) Pour un contexte donné, lister les valeurs quantitatives et/ou qualitatives qui s'appliquent

3.8 Les tests distribués, externalisés et internalisés

- (K2) Lister les risques, les points communs et les différences entre les 3 stratégies d'encadrement (distribué, externalisé et internalisé)

3.9 Les tests basés sur les risques

3.9.1 Introduction au test basé sur les risques

- (K2) Expliquer de différentes façon comment le test basé sur les risques répond aux risques
- (K4) Identifier les risques dans un projet ou un produit et déterminer la stratégie de test adéquate ainsi qu'un plan de test basé sur ces risques

3.9.2 Gestion des risques

- (K3) Effectuer pour un produit une analyse de risque selon le point de vue du testeur en suivant l'approche AMDE (FMEA en anglais)
- (K4) Synthétiser les résultats issus de différentes perspectives sur le risque produits typiquement par les acteurs clés, and utiliser leur jugement collectif afin de mettre en place des activités de test visant à réduire ces risques.

3.9.3 Gestion des risques dans le cycle de vie

- (K2) Décrire des caractéristiques de la gestion du risquent qui impliquent qu'elle soit menée dans un processus itératif
- (K3) Traduire une stratégie de test basée sur les risques en des activités de test et mesurer ses effets pendant le test
- (K4) Analyser et documenter les résultats de test et déterminer / proposer des risques résiduels permettant aux chefs de projet de prendre les bonnes décisions pour la livraison

3.10 Analyse des modes de défaillance et effets (AMDE)

- (K2) Décrire le concept de l'Analyse des modes de défaillance et effets, expliquer avec des exemples son application dans les projets et les bénéfices associés.

3.11 Les enjeux de la gestion des tests

- (K2) Comparer les enjeux de la gestion des tests pour les tests exploratoires, les systèmes de systèmes, et le test de systèmes critiques du point de vue de la sécurité. Montrer leurs bénéfices et inconvénients, leur adéquation et leur impact sur le planning, la couverture, la mesure et le contrôle

Chapitre 4: Techniques de test – [0 minute]

Pas d'objectifs d'apprentissage pour le gestionnaire de test (d'aucun niveau de connaissance).

Chapitre 5: Tester les caractéristiques du logiciel – [0 minutes]
Pas d'objectifs d'apprentissage pour le gestionnaire de test (d'aucun niveau de connaissance).

Chapitre 6: Revues – [120 minutes]
6.2 Les principes de revue
- (K2) Expliquer les bénéfices des revues comparées au test dynamique et aux autres techniques de test statique

6.4 Introduire les revues
- (K2) Comparer entre eux les différents types de revue et montrer leurs forces et faiblesses relatives ainsi que leurs champs d'usage.
- (K3) Piloter une équipe de revue pour une revue formelle suivant les étapes identifiées
- (K4) Mettre en place un plan de revue faisant partie intégrante d'un plan de test/qualité pour un projet en considérant les différentes techniques de revue et en les utilisant selon les défauts à trouver, les compétences disponibles et en les alignant avec les approches de test dynamique appropriées.

6.5 Les facteurs de succès pour les revues
- (K2) Expliquer les risques pris si l'on ne prend pas en compte les facteurs techniques, organisationnels et humains pour mener des revues.

Chapitre 7: Gestion des incidents – [80 minutes]
- (K3) Traiter un défaut en suivant la procédure de cycle de vie le la gestion d'un incident proposée par IEEE 1044-1993
- (K3) Évaluer un rapport de défaut selon l'IEEE 1044-1993 et la taxonomie appliquée afin d'améliorer sa qualité
- (K4) analyser les rapports de défaut créés dans le temps et mettre à jour la taxonomie des défauts

Chapitre 8: Normes et processus d'amélioration des tests **– [120 minutes]**
- (K2) Résumer les sources de standard logiciel et expliquer leur utilité pour le test logiciel

8.4 Amélioration du processus de test
- (K3) Écrire et tester un plan d'amélioration en utilisant les étapes génériques impliquant les bonnes personnes
- (K2) Résumer le processus d'amélioration des tests tel que défini par TMM, TPI, CTP, STEP, et les domaines vérification et validation dans CMMI
- (K2) Expliquer les critères d'évaluation des modèles d'amélioration TMM, TPI, CTP, STEP, et les domaines vérification et validation dans CMMI

Chapitre 9: Outils de test et automatisation – [90 minutes]
9.2 Concepts des outils de test
- (K2) Comparer les éléments et les aspects à l'intérieur de chacun des concepts suivants sur les outils de test : "Bénéfices et risques", "Stratégies des outils de test", "Intégration des outils", "Langages d'automatisation", "Oracle de test", "Déploiement d'un outil", "Outils Open Source", "Développement d'outils", et "Classification des outils"
- (K2) Décrire pourquoi et quand il est important de créer une stratégie d'outil de test ou une road-map pour un outil de test
- (K2) Comprendre les différentes phases de la mise en œuvre d'un outil de test

9.3 Les catégories d'outils de test
- (K2) Présenter les catégories d'outils de test par objectifs, usage, forces, risques et exemples
- (K2) Résumer les exigences spécifiques au test de systèmes critiques du point de vue de la sécurité pour les outils de test commerciaux ou Open Source
- (K2) Décrire les aspects importants et les conséquences de différents outils de test et leur implémentation, utilisation et effets sur le processus de test
- (K2) Décrire quand et pourquoi implémenter son propre outil est une option, les bénéfices, risques et conséquences associées

Chapitre 10: Compétences – Composition de l'équipe – [240 minutes]
10.2 Compétences individuelles
- (K3) Utiliser un questionnaire donné afin de déterminer les forces et les faiblesses des membres de l'équipe en ce qui concerne l'utilisation des systèmes logiciels, les connaissances du domaine et du

métier, les activités du développement de système, le test logiciel et les compétences relationnelles.

10.3 Dynamique de l'équipe de test

- (K3) Réaliser une analyse d'écart afin de déterminer les compétences techniques et logicielles requises pour un poste ouvert dans une organisation.

10.4 Mettre en place le test dans une organisation

- (K2) Décrire les différentes organisations possibles et les comparer avec les tests internalisés ou externalisés, en France ou à l'étranger.

10.5 Motivation

- (K2) Donner des exemples de facteurs de motivation ou démotivation des testeurs

10.6 Communication

- (K2) Décrire en utilisant un exemple une communication professionnelle, objective et efficace selon la vue d'un testeur. Les notions de risques et opportunités peuvent être prises en compte.

0.5 Objectifs d'Apprentissage pour les Analystes de Test

Cette section fournit la liste détaillée d'objectifs d'apprentissage pour le module analyste de test. En général, toutes les parties de ce syllabus sont examinables au niveau K1. Cela signifie que le candidat doit reconnaître, se souvenir de et mémoriser un terme ou un concept. C'est pour cela que seuls les objectifs de niveau K2, K3 et K4 sont mentionnés dans la suite de cette section.

Introduction au syllabus pour l'analyste de test – [60 minutes]
(Intégrant une révision du syllabus niveau fondation de l'ISTQB)

Chapitre 1: Éléments de base du test logiciel – [30 minutes]

Chapitre 2:Processus de test – [180 minutes]
2.4 Analyse et conception des tests

- (K2) Expliquer les causes de la présence du test fonctionnel à des étapes spécifiques d'un cycle de vie d'application
- (K2) Donner des exemples de critères qui influencent la structure et le niveau du développement des conditions de test
- (K2) Expliquer comment l'analyse et la conception des tests sont des techniques de test statique qui peuvent être utilisées pour découvrir des défauts.
- (K2) Expliquer en donnant des exemples, le concept d'oracle de test et la façon avec laquelle un oracle de test peut être utilisé lors de la spécification des tests.

2.5 Implémentation et exécution des tests

- (K2) Décrire les pré-conditions à l'exécution des tests, y compris: le testware, l'environnement de test, la gestion de configuration et la gestion des tests.

2.6 Évaluer les critères de sortie et informer

- (K3) Déterminer, pour un ensemble donné de mesures si un critère de complétude des tests est satisfait.

Chapitre 3 : Gestion des tests – [120 minutes]
3.9.2 Gestion des risques

- (K3) Prioriser la sélection des cas de test, de la couverture de test et des données de test en se basant sur les risques et le documenter de façon appropriée dans un planning et une procédure de test.
- (K2) Présenter les activités d'une approche basée sur les risques pour la planification et l'exécution des activités de test.

Chapitre 4 : Techniques de test – [1080 minutes]
4.2 Techniques basées sur les spécifications

- (K2) Donner des exemples de défauts pouvant typiquement être identifiés par chacune des techniques basées sur les spécifications, fournir les critères de couverture associés.
- (K3) Écrire des cas de test à partir de modèles logiciels donnés en utilisant les techniques de conception suivantes (les tests doivent permettre d'atteindre un niveau de couverture donné)
  - Partitions d'équivalence
  - Analyse des valeurs limites
  - Tables de décision
  - Tests de transition d'état
  - Méthode de classification arborescente
  - Classification par paires
  - Cas d'utilisation
- (K4) analyser un système, ou la spécification de ses exigences, afin de déterminer quelle technique basée sur les spécifications appliquer pour des objectifs particuliers, et détailler une spécification de test basée sur l'IEEE 829, en se concentrant sur des cas de test et procédures de test fonctionnels et portant sur un domaine.

4.4 Techniques basées sur les défauts et sur l'expérience

- (K2) Décrire les principes et motifs des techniques basées sur les erreurs et différencier leur usage des techniques basées sur les spécifications et la structure

- (K2) Expliquer en donnant des exemples les taxonomies des défauts et leur utilisation
- (K2) Comprendre les principes et les motifs de l'utilisation des techniques basées sur l'expérience et déterminer quand les utiliser
- (K3) Dans le cadre des tests exploratoires, savoir spécifier les tests, les exécuter et communiquer
- (K2) Classifier les défauts à identifier selon les différents types d'attaques logicielles visant à les faire apparaître
- (K4) Analyser un système afin de déterminer, pour un objectif particulier, quelle(s) technique(s) appliquer parmi celles basées sur les spécifications, les défauts ou l'expérience.

Chapitre 5: Tester les caractéristiques du logiciel – [210 minutes]

5.2 Caractéristiques Qualité pour les tests par domaine

- (K4) Expliquer avec des exemples quelles seront les techniques, parmi celles présentées au chapitre 4, appropriées pour tester les caractéristiques d'exactitude, de pertinence, d'interopérabilité, de sécurité et d'accessibilité.
- (K3) Présenter, concevoir, spécifier et exécuter des tests d'utilisabilité en utilisant les techniques appropriées, couvrant des objectifs de test donnés et ciblant des défauts particuliers.

5.3 Caractéristiques Qualité pour les tests techniques

- (K2) Expliquer les raisons d'introduire dans une stratégie de test les tests de rendement, de fiabilité et de sécurité technique; fournir des exemples de défauts attendus.
- (K2) Caractériser les différents types de tests non fonctionnels possibles pour mener des tests techniques ciblant certains types de défauts (attaques), leur mise en œuvre dans le cycle de vie de l'application et les techniques de test requises pour la conception des tests.

Chapitre 6 : Revues – [180 minutes]

- (K3) Utiliser une check-list de revue pour vérifier le code et l'architecture avec une vision du niveau testeur.
- (K3) Utiliser une check-list de revue pour vérifier les exigences et les cas d'utilisation avec une vision du niveau testeur.
- (K2) Comparer les différents types de revue entre eux et montrer leurs forces, faiblesses et champs d'application relatifs.

Chapitre 7 : Gestion des incidents – [120 minutes]

- (K4) Analyser, classifier et décrire des défauts fonctionnels et non fonctionnels dans les rapports de défauts compréhensibles.

Chapitre 8 : Normes et processus d'amélioration des tests – [0 minutes]

Pas d'objectifs d'apprentissage pour l'analyste de test (d'aucun niveau de connaissance).

Chapitre 9: Outils de test et automatisation – [90 minutes]

9.2 Concepts des outils de test

- (K2) Comparer les différents éléments et aspects présents dans les concepts liés aux outils de test : "Bénéfices et risques", "Stratégie d'outillage des tests", "Intégration de l'outil", "Langages d'automatisation", "Oracles de test", "Déploiement des outils", "Outils Open Source", "Développement d'outils", et "Classification des outils"

9.3 Les catégories d'outils de tests

- (K2) En donnant des exemples, présenter les catégories d'outils de test en les classant par objectifs, usage prévu, forces, et risque.
- (K2) Associer les outils des différentes catégories aux différents niveaux et types de test.

Chapitre 10: Compétences – Composition de l'équipe – [30 minutes]

10.6 Communication

- (K2) Décrire, en s'appuyant sur des exemples réels, ce qu'est une communication objective et efficace selon la vue testeur. Considérer éventuellement les risques et opportunités.

0.6 Objectifs d'Apprentissage pour les Analystes Techniques de Test

Cette section fournit la liste détaillée d'objectifs d'apprentissage pour le module analyste technique de test. En général, toutes les parties de ce syllabus sont examinables au niveau K1. Cela signifie que le candidat doit reconnaître, se souvenir de et mémoriser un terme ou un concept. C'est pour cela que seuls les objectifs de niveau K2, K3 et K4 sont mentionnés dans la suite de cette section.

Introduction au syllabus pour l'analyste technique de test – [60 minutes]
(Intégrant une révision du syllabus niveau fondation de l'ISTQB)

Chapitre 1: Éléments de base du test logiciel – [30 minutes]

Chapitre 2:Processus de test – [180 minutes]
2.4 Analyse et conception des tests

- (K2) Expliquer à quels endroits dans le cycle de vie de l'application les tests non fonctionnels et les tests basés sur l'architecture peuvent être appliqués. Expliquer les raisons pour lesquelles les tests non fonctionnels ne peuvent être menés qu'à certaines étapes du cycle de vie.
- (K2) Donner des exemples de critères qui influencent la structure et le niveau du développement des conditions de test.
- (K2) Décrire de quelle manière l'analyse et la conception des tests sont des techniques de test statiques qui peuvent être utilisées pour découvrir des défauts.
- (K2) Expliquer en donnant des exemples le concept d'oracle de test ainsi que la façon d'utiliser un oracle de test dans la spécification des tests.

2.5 Implémentation et exécution des tests

- (K2) Décrire les pré-conditions à l'exécution des tests, en considérant : le testware, l'environnement de test, la gestion de configuration et la gestion des défauts.

2.6 Évaluer les critères de sortie et informer

- (K3) Déterminer, à partir d'un ensemble de mesures donné, si un critère de complétude des tests a été rempli.

Chapitre 3: Gestion des tests – [120 minutes]
3.9.2 Gestion des risques

- (K2) Présenter les activités de l'approche basée sur les risques pour la planification et l'exécution des tests techniques.

Chapitre 4: Techniques de test– [930 minutes]
4.2 Techniques basées sur les spécifications

- (K2) Donner des exemples de défauts classiques qui pourront être identifiés par chacune des techniques basées sur les spécifications.
- (K3) Écrire des cas de test pour un modèle logiciel réel en utilisant les techniques de conception suivantes (les tests doivent atteindre un modèle de couverture donné)
  - Partitions d'équivalence
  - Analyse des valeurs limites
  - Tables de décision
  - Tests de transition d'état
- (K4) Analyser un système ou ses exigences de spécification afin de déterminer quelles techniques basées sur les spécifications appliquer pour des objectifs particuliers, et expliquer une spécification de test basée sur l'IEEE829, en se concentrant sur les cas de test et procédures de test de composant et non-fonctionnels

4.3 Techniques basées sur la structure ou boite blanche

- (K2) Donner des exemples de défauts classiques qui pourront être identifiés par chacune des techniques basées sur la structure ou boite blanche.
- (K3) Écrire des cas de test réel en utilisant les techniques de conception suivantes (les tests doivent atteindre un modèle de couverture donné)
  - Test des instructions
  - Test des décisions
  - Test des conditions
  - Test des conditions multiples

- (K4) Analyser un système afin de déterminer quelles techniques basées sur la structure utiliser pour des répondre à des objectifs de test particuliers.
- (K2) Comprendre chaque technique basée sur la structure ainsi que les critères de couverture correspondant et savoir quand les utiliser.
- (K4) Être capable d'analyser et comparer les différentes techniques basées sur la structure pour choisir les plus adaptées à différentes situations.

4.4 Techniques basées sur les défauts et sur l'expérience

- (K2) Décrire les principes et les éléments favorisant l'utilisation des techniques basées sur les défauts et les différentier des techniques basées sur les spécifications et des techniques basées sur la structure.
- (K2) Expliquer avec des exemples les taxonomies des défauts et leurs utilisations.
- (K2) Comprendre les principes et les éléments favorisant l'utilisation des techniques basées sur l'expérience et savoir quand les utiliser.
- (K3) Spécifier, exécuter et communiquer les résultats des tests exploratoires
- (K3) Spécifier des tests utilisant en utilisant différents types d'attaques logicielles visant à détecter des défauts particuliers.
- (K4) Analyser un système afin de déterminer quelles techniques basées sur les spécifications, les défauts ou l'expérience, utiliser pour répondre à des objectifs particuliers.

4.5 Analyse statique

- (K3) Utiliser les algorithmes "Analyse de flot de contrôle" et "Analyse de flot de données" pour vérifier que le code ne contient par d'erreur de flot de contrôle ou de données.
- (K4) Interpréter les résultats sur les flots de contrôle et de données fournis par un outil afin de vérifier que le code ne contient par d'erreur de flot de contrôle ou de données.
- (K2) Expliquer l'utilisation des graphes d'appel pour l'évaluation de la qualité de l'architecture. Cela doit inclure les défauts à identifier, l'utilisation pour la conception et la planification des tests, la limitation du nombre de résultats.

4.6 Analyse dynamique

- (K2) Expliquer comment l'analyse dynamique du code peut être exécutée, résumer les défauts qui peuvent être identifiés avec cette technique, et présenter ses limites.

Chapitre 5: Tester les caractéristiques du logiciel – [240 minutes]

5.2 Caractéristiques Qualité pour les tests par domaine

- (K2) Décrire les différents types de tests non fonctionnels applicables au test par domaine et les classer selon le type de défauts recherchés (attaques), leur usage classique dans le cycle de vie de l'application et selon les techniques de test adaptées à la conception des tests.
- (K4) Spécifier des cas de test pour des types particuliers de tests non fonctionnels en couvrant des objectifs de tests précis et en visant des défauts précis.

5.3 Caractéristiques Qualité pour les tests techniques

- (K2) Décrire les types de test non fonctionnels utilisés pour les tests techniques, en les classant selon les défauts visés (attaques), selon leur position dans le cycle de vie de l'application, et en précisant les techniques de test adaptées à la conception de ces tests.
- (K2) Comprendre et expliquer les étapes du cycle de vie de l'application auxquelles des tests de sécurité, de fiabilité et d'efficacité peuvent être appliqués (en incluant les sous-attributs spécifiés dans ISO9126)
- (K2) Faire la distinction entre les types de fautes trouvées par les tests de sécurité, de fiabilité et d'efficacité (en incluant les sous-attributs spécifiés dans ISO9126)
- (K2) Décrire les approches de test pour les attributs liés à la sécurité, la fiabilité et l'efficacité ainsi que pour leurs sous-attributs décrits dans ISO 9126.
- (K3) Spécifier des cas de test pour les attributs liés à la sécurité, la fiabilité et l'efficacité ainsi que pour leurs sous-attributs décrits dans ISO 9126.
- (K2) Comprendre et expliquer les raisons d'inclure dans une stratégie de test des tests de maintenabilité, portabilité et accessibilité.
- (K3) Spécifier des cas de test non fonctionnels pour la maintenabilité et la portabilité.

Chapitre 6: Revues – [180 minutes]

- (K4) Mettre en place une check-list de revue destinée à trouver les défauts classiques lors d'une revue de code ou d'architecture.

- (K2) Comparer les différents types de revue entre eux et montrer leurs forces, faiblesses et champs d'application relatifs.

Chapitre 7: Gestion des incidents – [120 minutes]

- (K4) Analyser, classifier et décrire des défauts fonctionnels et non fonctionnels dans des rapports de défauts compréhensibles.

Chapitre 8:Normes et processus d'amélioration des tests – [0 minutes]

Pas d'objectifs d'apprentissage pour l'analyste technique de test (d'aucun niveau de connaissance).

Chapitre 9: Outils de test et automatisation – [210 minutes]

9.2 Concepts des outils de test

- (K2) Comparer les différents éléments et aspects présents dans les concepts liés aux outils de test : "Bénéfices et risques", "Stratégie d'outillage des tests", "Intégration de l'outil", "Langages d'automatisation", "Oracles de test", "Déploiement des outils", "Outils Open Source", "Développement d'outils", et "Classification des outils"

9.3 Les catégories d'outils de test

- (K2) Présenter, avec des exemples, les différentes catégories d'outils de test selon leurs objectifs, leur mode d'utilisation, leurs forces et les risques associés.
- (K2) Associer les outils des différentes catégories aux différents niveaux et types de test.

9.3.7 L'automatisation des tests déterminés par les mots-clés

- (K3) Créer des tables de mots-clés (ou mots-actions) qui seront utilisées par un outil d'exécution des tests en utilisant l'algorithme de sélection des mots clés
- (K3) Enregistrer des tests avec un outil de capture-rejeu afin de permettre des tests de régression de grande qualité, rapides à mettre en œuvre et à exécuter et permettant la couverture de nombreux cas de test.

9.3.8 Les outils de test de performance

- (K3) Concevoir le test de performance un utilisant des outils de test de performance incluant des fonctionnalités de planification et de mesure des caractéristiques du système.

Chapitre 10: Compétences – Composition de l'équipe – [30 minutes]

10.6 Communication

(K2) Décrire, en s'appuyant sur des exemples réels, ce qu'est une communication objective et efficace selon la vue testeur. Considérer éventuellement les risques et opportunités.

1. Éléments de base du test logiciel

Termes

Éthique, mesure, métrique, systèmes critiques du point de vue de la sécurité, système de systèmes, cycle de vie du logiciel.

1.1 Introduction

Ce chapitre introduit quelques thèmes fondamentaux du test, pertinents pour tous les professionnels du test, qu'ils soient Gestionnaires de Tests, Analystes de Test ou Analystes Technique de Test. Les organismes de formation expliqueront ces thèmes généraux dans le contexte du module enseigné, et donneront des exemples pertinents. Par exemple, dans le module « Analystes Technique de Test», le thème général de « métriques et mesures » (section 1.4) se basera sur des exemples de métriques techniques spécifiques, comme des mesures de performances.
Dans la section 1.2, le processus de test est considéré comme une partie du cycle de vie du développement logiciel. Ce thème est construit à partir des concepts de base introduits dans le Syllabus Niveau Fondation et porte une attention particulière à l'alignement du processus de test avec les modèles de cycle de vie de développement du logiciel et avec les autres processus des techniques de l'information.
Les systèmes peuvent prendre de multiples formes qui peuvent influencer de manière significative la façon de voir le test. Dans la section 1.3, deux types spécifiques de systèmes sont introduits que tous les testeurs doivent connaitre : les systèmes de systèmes (appelés parfois multi-systèmes) et les systèmes critiques.
Les testeurs expérimentés relèveront un grand nombre de challenges en appliquant les différents éléments de base du test décrits dans ce syllabus, dans le contexte de leurs propres organisations, équipes et tâches.

1.2 Tester dans le Cycle de Vie du Logiciel

Le test est une partie intégrante des différents modèles de développement de logiciel comme :

- Le cycle séquentiel (modèle en cascade, cycle en V, cycle en W)
- Le cycle itératif (Développement Rapide d'Application ou RAD et modèle en spirale)
- Le cycle incrémental (méthodes agiles ou évolutives)

L'approche cycle de vie à long terme pour tester doit être considérée et définie comme une partie de la stratégie de test. Cela inclut l'organisation, la définition de processus et la sélection de méthodes et d'outils.

Les processus de test ne doivent pas être menés de façon isolée mais plutôt interconnectés et liés à d'autres processus comme :

- La gestion et l'ingénierie des exigences
- La gestion de projet
- La gestion de configuration et du changement
- Le développement logiciel
- La maintenance logicielle
- Le support technique
- L'élaboration de documentations techniques

La planification très tôt des tests et leur exécution par la suite sont en relation dans les modèles de développement séquentiels. Les tâches de tests peuvent se recouvrir et/ou être simultanées.

La gestion de configuration et du changement sont des tâches de support importantes pour le test logiciel. Sans une gestion du changement appropriée, l'impact de changements sur le système ne peut pas être évalué. Sans la gestion de configuration, des évolutions simultanées peuvent être perdues ou mal gérées.

En fonction du contexte du projet, des niveaux de test additionnels peuvent être définis à ceux abordés dans le Syllabus Niveau Fondation, comme :

- Les tests d'intégration matériel-logiciel
- Les tests d'intégration système
- Les tests d'interaction entre fonctionnalités
- Les tests d'intégration de produit client

Chaque niveau de test a les caractéristiques suivantes :

- Objectifs du test
- Périmètre du test
- Traçabilité par rapport aux bases de test
- Critères d'entrée et de sortie
- Livrables de test incluant le reporting
- Techniques de test
- Mesures et métriques
- Outils de test
- Conformité avec les standards de l'organisation et autres standards applicables.

Selon le contexte, les buts et le périmètre de chaque niveau de test peuvent être considérés isolément ou au niveau projet (par exemple pour éviter la duplication non nécessaire des mêmes tests à des niveaux différents).

Les activités de test doivent être en phase avec le modèle de cycle de développement du logiciel choisi, qui par nature peut être séquentiel (par exemple en cascade, cycle en V, cycle en W), itératif (par exemple Développement Rapide d'Application, modèle en spirale) ou incrémental (par exemple méthodes agiles ou évolutives).

Par exemple, dans le cycle en V, le processus fondamental de test de l'ISTQB® appliqué au niveau système se décline de la façon suivante :

- La planification des tests système est réalisée en même temps que la planification du projet, et le contrôle des tests continue jusqu'à ce que l'exécution des tests et les activités de clôture des tests soient terminées.
- L'analyse des tests système et leur conception sont réalisées en parallèle de l'expression des exigences, des spécifications de la conception générale, des spécifications de conception détaillée.
- La mise en place de l'environnement de test système (par exemple les bancs de tests, les équipements de test) peut commencer pendant la conception système, quoique la charge correspondante apparait généralement en parallèle du codage et du test de niveau composant, avec des activités relatives aux activités de réalisation des tests système s'étalant souvent jusqu'à quelques jours avant le démarrage de l'exécution des tests système.
- L'exécution des tests système commence quand les critères d'entrée associés sont tous satisfaits (ou « dérogés »), ce qui veut dire habituellement qu'au minimum les tests au niveau composant et souvent aussi les tests d'intégration des composants ont été réalisés. L'exécution des tests système continue jusqu'à ce que les critères de sortie soient satisfaits.
- L'évaluation des critères de sortie et le bilan des résultats du test système peuvent avoir lieu pendant l'exécution des tests système, généralement de manière plus fréquente et plus urgente quand la fin du projet approche.
- Les activités de clôture des tests système interviennent après l'atteinte des critères de sortie associés et après que l'exécution des tests système ait été déclarée comme terminée, donc elles peuvent parfois être décalées jusqu'à la fin des tests d'acceptation et jusqu'à ce que toutes les activités du projet sont terminées.

Pour chaque niveau de test, et pour chaque combinaison sélectionnée de processus de test et de cycle de vie du logiciel, le Test Manager doit réaliser le phasage pendant la planification des tests et/ou pendant la planification du projet. Pour des cas particuliers de projets complexes, comme les projets de système de systèmes (fréquents dans le domaine militaire et dans les grosses organisations), les processus de test ne doivent pas seulement être cadrés, mais ils doivent aussi être modifiés en fonction du contexte des projets (par exemple, quand il est plus facile de détecter un défaut à un niveau de test élevé par rapport à un niveau de test plus bas).

1.3 Systèmes Spécifiques

1.3.1 Systèmes de Systèmes

Un système de systèmes est un ensemble de composants communicants (incluant du matériel, des applications logicielles et des modules de communication), interconnectés pour réaliser un but commun, sans une structure de gestion unique. Les caractéristiques et les risques associés aux systèmes de systèmes incluent :

- L'intégration progressive des systèmes communicants indépendants pour éviter de créer un système entier depuis la feuille blanche. Cela peut être réalisé par exemple, en intégrant des composants sur étagère avec des développements complémentaires limités.
- Une complexité technique et organisationnelle (par exemple à travers les différents intervenants) qui représente des risques pour une gestion efficace. Différentes approches de cycle de vie de développement peuvent être adoptés pour des systèmes collaboratifs qui peuvent engendrer des problèmes de communication dus aux différentes équipes impliquées (développement, test, production, ligne d'assemblage, utilisateurs, etc.). La gestion de projet globale de systèmes de systèmes doit être capable de considérer la complexité technique inhérente de la combinaison de différents systèmes communicants, et doit être capable de prendre en compte les différents choix d'organisations comme l'externalisation et l'offshore.
- La confidentialité et la protection de savoir-faire spécifique, les interfaces entre les différentes organisations (par exemple le secteur gouvernemental et le secteur privé) ou les décisions réglementaires (par exemple la prohibition d'un comportement de monopole) peuvent signifier qu'un système complexe doit être considéré comme un système de systèmes.
- Les systèmes de systèmes sont intrinsèquement moins fiables que les systèmes individuels, car chaque limitation sur un (sous-)système est automatiquement applicable à l'ensemble du système de systèmes.
- Le haut niveau d'interopérabilité fonctionnelle et technique requis pour les composants individuels dans un système de systèmes rend les tests d'intégration hautement critiques et requiert des interfaces très bien spécifiées et validées.

1.3.1.1 Gestion et Test des Systèmes de Systèmes

La grande complexité de la gestion de projet et de la gestion en configuration des composants sont des contraintes connues associées aux systèmes de systèmes. Une forte implication de l'Assurance qualité et la présence de processus définis sont souvent associés aux systèmes complexes et aux systèmes de systèmes. Un cycle de vie de développement formel, des jalons et des revues sont souvent associés aux systèmes de systèmes.

1.3.1.2 Caractéristiques du Cycle de Vie des Systèmes de Systèmes

Chaque niveau de test pour un système de systèmes comprend les caractéristiques suivantes complémentaires à celles décrites dans la section 1.2 Tester dans le Cycle de Vie du Logiciel.

- Multiples niveaux d'intégration et de gestion de version
- Projet à long terme
- Transfert formel d'information entre les différents membres du projet
- Évolution non concurrente des composants, et exigence sur les tests de non régression au niveau système de systèmes
- Test de maintenance dû au remplacement de composants individuels suite à une obsolescence ou à une mise à jour

A l'intérieur des systèmes de systèmes, chaque niveau de test doit être traité en fonction du point de vue avec lequel on aborde le système de système. Par exemple, le niveau de test « système » pour un élément peut être considéré comme un niveau de test « composant » si on se place à un plus haut niveau.

Habituellement, chaque système individuel (au sein d'un système de systèmes) passera par chaque niveau de test, puis sera intégré dans un système de systèmes avec des tests complémentaires appropriés.

Pour des choix spécifiques de gestion de projet de systèmes de systèmes, se référer à la section 3.11.2.

1.3.2 Systèmes à Sécurité Critique

Les Systèmes à Sécurité Critique ou Systèmes Critiques du point de vue de la sécurité » sont ceux qui, si leur exécution est arrêtée ou dégradée (par exemple suite à une manipulation incorrecte ou par inadvertance), peuvent provoquer des conséquences critiques ou catastrophiques. Le fournisseur de système critique peut être responsable des dommages ou des dédommagements, et les activités de test sont donc utilisées pour réduire cette responsabilité. Les activités de test apportent la preuve que le système a été correctement testé pour éviter les conséquences critiques ou catastrophiques.

On peut citer comme exemple de systèmes critiques les systèmes de contrôle des flottes aériennes, les systèmes de change automatiques, les systèmes de régulation des cœurs des centrales nucléaires, les systèmes médicaux, etc.

Les aspects suivants doivent être implémentés dans les systèmes critiques :

- La traçabilité vers les exigences réglementaires et les moyens de conformité
- Une approche rigoureuse du développement et du test
- Une analyse de sûreté de fonctionnement
- Une architecture certifiée et sa qualification
- Un focus sur la qualité
- Un haut niveau de documentation (sur le fond et sur la forme de la documentation)
- Un haut niveau d'auditabilité

La section 3.11.3 aborde les choix en termes de gestion de tests pour les systèmes critiques.

1.3.2.1 Conformité Réglementaire

Les systèmes critiques doivent souvent respecter des réglementations ou standards gouvernementaux, internationaux ou spécifiques à un secteur (voir aussi §8.2 Normes & Standards). Ceux-ci doivent être appliqués au processus de développement et à la structure organisationnelle, ou au produit en cours de développement.

Pour démontrer la conformité d'une structure organisationnelle ou d'un processus de développement, des audits et des schémas organisationnels peuvent suffire.

Pour démontrer la conformité à des règles spécifiques pour le système développé (produit), il est nécessaire de montrer que chaque exigence de cette réglementation a été couverte correctement. Dans ce cas, une traçabilité complète de l'exigence vers la preuve est nécessaire pour démontrer la conformité. Cela impacte la gestion de projet, le cycle de vie du développement, les activités de test et la qualification/certification (par une autorité reconnue) à travers le processus de développement.

1.3.2.2 Systèmes Critiques et Complexité

De nombreux systèmes complexes et systèmes de systèmes ont des composants critiques. Quelquefois, l'aspect sûreté de fonctionnement n'est pas évident au niveau système (ou sous-système) mais seulement à un niveau supérieur où les systèmes complexes sont implémentés (par exemple la définition de missions pour les flottes aériennes, les systèmes de contrôle de trafic aérien).

Par exemple, un routeur n'est pas un système critique en lui-même, mais il peut le devenir quand un système d'information critique en a besoin, comme dans les services de télémédecine.

La gestion du risque, qui réduit la probabilité et/ou l'impact d'un risque, est essentielle pour le développement et le test dans un contexte critique de sûreté de fonctionnement (se référer au chapitre 3). En complément, une analyse AMDEC «Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité» (voir la section 3.10) et une analyse des causes communes des erreurs logicielles sont généralement utilisées dans ce type de contexte (SCCFA Software Common Cause Failure Analysis).

1.4 Métriques & Mesures

Une variété de métriques (nombres) et de mesures (tendances, graphes, etc.) devrait être utilisée tout au long du cycle de vie du développement logiciel (par exemple, planification, couverture, charge de travail, etc.). Dans chaque cas, un état de référence doit être défini, et ensuite la progression doit être enregistrée en fonction de cet état de référence.

Les éléments possibles qui peuvent être couverts incluent :

1. l'ordonnancement planifié, la couverture et leur évolution dans le temps
2. les exigences, leur évolution et leur impact en termes d'ordonnancement, de ressources et de tâches
3. la charge de travail et l'utilisation des ressources, et leur évolution dans le temps
4. les jalons et portée, et leur évolution dans le temps
5. les coûts actuels et planifiés pour terminer les tâches
6. les risques et les actions pour les maîtriser, et leur évolution dans le temps
7. les anomalies détectées, corrigées et la durée de la correction

L'utilisation de métriques permet aux testeurs de fournir un reporting à leur direction d'une façon consistante, et permet un suivi cohérent de la progression dans le temps.

Trois points doivent être pris en compte :

- la définition de métriques : un ensemble limité de métriques intéressantes doit être défini. Une fois ces métriques définies, tous les acteurs doivent être d'accord sur leur interprétation, dans le but d'éviter les futures discussions quand les métriques évoluent. Les métriques peuvent être définies en fonction d'objectifs associés à un processus ou à une tâche, pour des composants ou des systèmes, pour des individus ou des équipes. Il y a souvent une tendance à définir trop de métriques au lieu de choisir les métriques les plus pertinentes.
- Le suivi des métriques : le reporting et la mise en forme de métriques doit être aussi automatique que possible pour réduire le temps consommé dans la production des valeurs. L'évolution des valeurs dans le temps pour une métrique donnée peut refléter d'autres informations que la simple interprétation validée lors de phase de définition de la métrique.
- Le reporting de métriques : l'objectif est de fournir une compréhension immédiate de l'information, à destination du management. Les présentations peuvent montrer une « image » des métriques à un certain temps ou montrer l'évolution des métriques dans le temps pour en évaluer les tendances.

1.5 Code d'Ethique

L'implication dans le test logiciel permet aux individus d'avoir accès à des informations confidentielles et privilégiées. Un code d'éthique est nécessaire, notamment pour assurer que les informations ne sont pas utilisées dans des cas non appropriés. En référence au code d'éthique d'ACM et de l'IEEE pour les ingénieurs, l'ISTQB® définit le code d'éthique suivant :

PUBLIC – les testeurs de logiciels certifiés doivent agir en fonction de l'intérêt public
CLIENT ET EMPLOYEUR – les testeurs de logiciels certifiés doivent agir pour l'intérêt de leur client et de leur employeur tout en respectant l'intérêt public
PRODUIT – les testeurs de logiciels certifiés doivent assurer que les fournitures qu'ils produisent (concernant les produits et les systèmes qu'ils testent) répondent le plus possible aux standards professionnels
JUGEMENT – les testeurs de logiciels certifiés doivent conserver leur intégrité et leur indépendance dans leur jugement professionnel
GESTION – les chefs de projet de test de logiciels certifiés et les responsables doivent respecter et promouvoir une approche morale dans la gestion de projets de test de logiciels
PROFESSION – les testeurs de logiciels certifiés doivent mettre en avant l'intégrité et la réputation du métier en cohérence avec l'intérêt public
COLLEGUES – les testeurs de logiciels certifiés doivent être loyaux, aider leurs collègues, et promouvoir le partenariat avec les développeurs de logiciels
PERSONNELLEMENT – les testeurs de logiciels certifiés doivent participer en permanence à de la formation pour leur métier et doivent promouvoir une approche morale concernant sa pratique.

2. Processus de test

Termes

BS 7925/2, critère de sortie, IEEE 829, cas de test, clôture des tests, condition de test, contrôle des tests, conception des tests, exécution des tests, implémentation des tests, planification des tests, procédure de test, script de test, rapport de synthèse des tests, registre de test.

2.1 Introduction

Dans le syllabus ISTQB® niveau fondation, le processus fondamental de test comprend les activités suivantes :

- Planification et contrôle
- Analyse et conception
- Implémentation et exécution
- Évaluer les critères de sortie et informer
- Activités de clôture des tests

Ces activités peuvent être implémentées séquentiellement ou certaines parallèlement, par exemple, analyse et conception pourraient être réalisées en parallèle de l'implémentation et de l'exécution, alors que les autres activités pourraient être mises en œuvre de façon séquentielle.

Depuis que la gestion des tests est fondamentalement liée au processus de test, les gestionnaires de tests doivent être en mesure d'appliquer l'ensemble du contenu de ce chapitre à la gestion d'un projet spécifique. Toutefois, pour les analystes de tests et les analystes de tests techniques, les connaissances acquises au niveau fondation sont largement suffisantes, à l'exception des tâches de développement des tests énumérées ci-dessus. Les connaissances requises pour ces tâches sont généralement couvertes dans ce chapitre et ensuite détaillées au chapitre 4 Techniques de conception des tests et au chapitre 5 Tester les Caractéristiques du logiciel.

2.2 Modèles de Processus de Test

Les modèles de processus sont des approximations et des abstractions. Les modèles de processus de test ne tiennent pas compte de l'ensemble des complexités, des nuances, et des activités que constituent le monde réel des projets ou de l'entreprise. Ces modèles doivent être considérés comme une aide à la compréhension et à l'organisation, et non comme immuables, la vérité révélée.

Bien que ce syllabus utilise, à titre d'exemple, les processus décrits dans le syllabus ISTQB® niveau fondation (voir ci-dessus), il existe d'autres modèles de processus de test importants, des exemples de trois d'entre eux sont énumérés ci-dessous. Ils sont tous des modèles de processus de test et des modèles d'amélioration des processus de test (le modèle « Practical Software Testing » inclut le « Test Maturity Model »), et sont définis en termes de niveaux de maturité qu'ils soutiennent. Les trois modèles de processus de test, de concert avec TPI®, sont examinés plus avant dans la section 8.3 8.3Amélioration des Processus de Test.

- Practical Software Testing – Test Maturity Model [Burnstein03]
- Critical Testing Processes [Black03]
- Systematic Test and Evaluation Process (STEP)

2.3 Planification & Contrôle des Tests

Ce chapitre se concentre sur le processus de planification et de contrôle des tests.

La planification des tests, la plupart du temps, intervient au début de l'effort de test et implique l'identification et la mise en œuvre de toutes les activités et des ressources nécessaires pour répondre à la mission et aux objectifs définis dans la stratégie de test.

Le test basé sur les risques (voir chapitre 3 Gestion des Tests) est utilisé pour adapter le processus de planification en ce qui concerne les actions nécessaires à la réduction des risques produits identifiés. Par exemple, s'il est identifié que de graves défauts sont habituellement issus des spécifications de conception, le processus de planification pourrait intégrer des tests statiques (revues) des spécifications de conception avant que celles-ci ne soient converties en code. Le test basé sur les risques permet ainsi d'adapter le processus de planification en tenant compte des priorités relatives des activités de test.

Des relations complexes existent entre bases de test, conditions de test, cas de test et procédures de test si bien que des relations n – n peuvent exister entre ces différents éléments. Celles-ci doivent être prises en compte pour permettre la mise en œuvre effective de la planification et du contrôle des tests.

Le contrôle des tests est une activité permanente. Elle implique la comparaison entre l'avancement réel et l'avancement planifié et, la publication de rapport d'avancement incluant les écarts par rapport au plan initial. Le contrôle des tests permet de guider l'activité de test afin d'atteindre mission, stratégies et objectifs, et également d'être en mesure d'adapter la planification des activités de test en cas de besoin.

Le contrôle des tests doit répondre aux indicateurs générés par les tests ainsi qu'à l'évolution possible de la situation d'un projet ou d'une organisation. Par exemple, si des tests dynamiques mettent en évidence des groupes de défauts dans des zones qui avaient été jugées fiables, ou si la période d'exécution des tests est réduite en raison d'un retard du démarrage des tests, alors l'analyse de risque et la planification doivent être révisées. Cela peut se traduire par la redéfinition des priorités de test et la réattribution de la charge d'exécution de test restante.

Le contenu des documents de planification est traité au chapitre 3 Gestion des Tests.

Les métriques permettant de surveiller la planification et le contrôle des tests peuvent comprendre :

- Couverture des tests et des risques
- Détection et communication des anomalies
- Charge réelle de conception et d'exécution des tests par rapport à la charge prévue

2.4 Analyse & Conception des Tests

Durant la planification des tests, un ensemble d'objectifs de tests sont identifiés. Le processus d'analyse et de conception des tests utilise ces objectifs pour :

- Identifier les conditions de test
- Créer des cas de test qui exercent ces conditions de tests

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

2.4.1 Identification des Conditions de Test

Les conditions de test sont identifiées par l'analyse des bases et des objectifs de test pour déterminer quoi tester, en utilisant les techniques de test prévues par la stratégie de test et/ou le plan le test.

La définition du niveau et de la structuration des conditions de test peut se baser sur les caractères fonctionnel et non fonctionnel des articles de test en utilisant les éléments suivants :

1) Granularité des bases de test : des exigences de haut niveau devraient générer des conditions de test de haut niveau comme par exemple « Prouver que l'écran X fonctionne », à partir de laquelle pourrait

être définie une condition de tests de bas niveau comme « Prouver que l'écran X rejette un numéro de compte ne comportant pas le bon nombre de chiffres »
2) Risques produits relevés : par exemple, pour une fonctionnalité à risque élevé, détailler des conditions de test de bas niveau pourrait être un objectif défini
3) Exigences pour la gestion du reporting et la traçabilité de l'information
4) Savoir si la décision a été prise de ne travailler qu'à partir des conditions de test et de ne pas développer de cas de test par exemple en utilisant les conditions de tests pour se concentrer sur des tests non scriptés

2.4.2 Création des Cas de Test

Les cas de test sont conçus à partir des étapes d'élaboration et de raffinement des conditions de test en utilisant les techniques de test (voir chapitre 4) définies dans la stratégie de test. Ils doivent être reproductibles, vérifiables et leur traçabilité assurée vis-à-vis des exigences.

La conception de cas de test inclut l'identification :

- Des pré-conditions, telles que les exigences de mise à disposition de l'environnement de test et le planning associé
- Des exigences relatives aux données de test
- Des résultats attendus et des post-conditions

Un défi particulier est souvent la définition du résultat attendu d'un test; c'est-à-dire, l'identification d'un ou plusieurs oracles de test qui peuvent être utilisés pour ce test. Dans l'identification du résultat attendu, les testeurs sont concernés non seulement par les productions de l'écran, mais aussi par les données et les post-conditions environnementales.

Si la base de test est clairement identifiée, ce devrait être théoriquement plus simple. Toutefois, les bases de test sont souvent vagues, contradictoires, ne couvrent pas l'ensemble des domaines clés, ou sont tout simplement inexistantes. Dans ce cas, le testeur doit avoir l'expertise du domaine en question ou y avoir accès. Aussi, même si la base de test est bien spécifiée, la variété des interactions entre stimuli complexes et réponses peut rendre la définition des résultats attendus difficile, raison pour laquelle un oracle de test est essentiel. L'exécution de tests sans aucun moyen de déterminer l'exactitude des résultats a un très faible intérêt ou peu de valeur ajoutée, générant des rapports d'incident faux et une confiance erronée dans le système.

Les activités décrites ci-dessus peuvent être appliquées à tous les niveaux de test, bien que la base de test puisse varier. Par exemple, les tests d'acceptation utilisateurs peuvent être principalement basés sur les exigences, les cas d'utilisation et les processus métier existants, tandis que les tests de composants peuvent être essentiellement basés sur les spécifications de conception de bas niveaux.

Au cours du développement des conditions de test et des cas de test, de la documentation résultant des activités de test effectuées est généralement réalisée. Une norme pour cette documentation est l'IEEE 829. Cette norme traite des principaux types de documents applicables pour l'analyse et la conception des tests, la spécification de la conception des tests, la spécification des cas de test, ainsi que pour la mise en œuvre des tests. Dans la pratique, le niveau de documentation des activités de test varie considérablement. Cela peut être impacté par exemple par :

Les risques liés au projet (ce qui doit ou ne doit pas être documenté)

- La "valeur ajoutée" qu'apporte la documentation au projet
- Les normes à appliquer
- Le modèle de cycle de vie utilisé (par exemple, la méthode agile minimise la documentation en privilégiant d'étroites et fréquentes communications au sein de l'équipe)
- Les exigences de traçabilité avec la base de test, au travers de l'analyse et de la conception des tests

Selon la portée des tests, l'analyse et la conception des tests peuvent porter sur les caractéristiques qualité de l'objet du test. La norme ISO 9126 constitue une référence utile. Lorsque l'on teste des systèmes matériels/logiciels, d'autres types de caractéristiques peuvent s'appliquer.

Le processus d'analyse et de conception de test peut être renforcé en l'associant avec des revues et de l'analyse statique. Par exemple, la réalisation d'analyse et de conception de test basée sur les

spécifications d'exigences est une excellente façon de préparer une réunion de revue d'exigences. De la même manière, les livrables des activités de test tels les tests, l'analyse de risques et le plan de test devraient être soumis à des revues et à des analyses statiques.

Durant la conception des tests, les exigences détaillées de l'infrastructure de test nécessaire devraient être définies, bien qu'en pratique elles ne soient finalisées qu'au moment de leur mise en œuvre. Il faut comprendre que l'infrastructure de test inclut plus que les objets de test et les outils de test (bureaux, équipements, ressources humaines, logiciels, outils, périphériques, moyens de communications, gestion des autorisations pour les utilisateurs et tout autre moyen exigé pour exécuter les tests).

Les métriques pour suivre et contrôler l'analyse et la conception des tests doivent inclure :

- Le pourcentage d'exigences couvertes par des conditions de test
- Le pourcentage de conditions de test couvertes par les cas de test
- Le nombre d'anomalies trouvées pendant l'analyse et la conception des tests

2.5 Implémentation & Exécution des Tests

2.5.1 Implémentation des Tests

L'implémentation des tests comprend, l'organisation des cas de test en procédures de test (scripts de test), la mise à disposition de données de test et des environnements de test et la création d'un calendrier d'exécution des tests afin d'être en mesure de démarrer l'exécution. Cela inclut également la validation des critères d'entrée explicites et implicites pour le niveau de test requis.

Les procédures de test doivent être classées par ordre de priorité pour garantir que les objectifs identifiés dans la stratégie soient atteints le plus efficacement possible, par exemple exécuter les procédures de test les plus importantes en premier.

Le niveau de détail et la complexité associée du travail réalisé pendant l'implémentation des tests sont tributaires du détail des cas de test et des conditions de test. Parfois des dispositions réglementaires peuvent s'appliquer et les tests doivent alors fournir la preuve de leur conformité aux normes applicables telle que celle par exemple de la Direction Générale de l'Aviation Civile des États-Unis DO-178B/ED 12B.

Comme indiqué au 2.4 ci-dessus, les tests nécessitent des données de test et, dans certains cas, le volume de ces données peut être très important. Lors de l'implémentation, les testeurs doivent sélectionner ou créer ces données et les charger dans des bases de données ou via d'autres supports. Les testeurs doivent également créer des scripts et autres générateurs qui produiront les données à envoyer au système au cours de l'exécution des tests.

Lors de l'implémentation, les testeurs doivent finaliser et confirmer l'ordre dans lequel les tests manuels et automatisés seront exécutés. Quand l'automatisation est choisie, l'implémentation des tests inclut la création des harnais de test et des scripts de test. Les testeurs doivent soigneusement vérifier les contraintes liées à l'exécution de tests dans un ordre particulier. Les dépendances liées à l'environnement de test ou aux données de test doivent être connues et vérifiées.

L'implémentation des tests est aussi concernée par le ou les environnement(s) de test. Pendant cette étape il(s) doit(vent) être entièrement(s) disponible(s) et vérifié(s) avant l'exécution des tests. L'objectif étant de disposer d'un environnement de test opérationnel : l'environnement de test doit permettre, de mettre en évidence les défauts liés aux conditions de test, de fonctionner normalement en l'absence de défaillance et de reproduire en juste proportion ce qui est exigé, par exemple un environnement de production ou un environnement utilisateur final pour des tests de haut niveau.

Durant l'implémentation des tests, les testeurs doivent veiller à ce que les responsables de la création et l'administration de l'environnement de test soient identifiés et disponibles et que tous les articles de test, les outils de support aux tests et les processus associés soient prêts à être utilisés. Cela comprend aussi la gestion de la configuration, la gestion des incidents, l'inscription des tests et leur gestion. En outre, les testeurs doivent vérifier les procédures de recueil des données d'évaluation des critères de sortie et de reporting des résultats de tests.

Il est sage d'utiliser une approche équilibrée pour l'implémentation des tests. Par exemple, des stratégies fondées sur l'analyse de risque sont souvent associées à des stratégies de test dynamiques. Dans ce cas, un certain pourcentage de l'effort d'exécution des tests est alloué à des tests qui ne suivent pas de scripts prédéterminés.

Tester sans script ne doit pas être ad hoc ou sans but car la durée nécessaire serait imprévisible à moins de bien gérer le temps (voir SBTM). Au fil des ans, les testeurs ont développé un grand nombre de techniques basées sur l'expérience, telles que les tests intrusifs (voir chapitre 4.4 et [Whittaker03]), l'estimation d'erreurs [Myers79], et les tests exploratoires. L'analyse, la conception et l'implémentation des tests continuent à être mis en œuvre, mais elles le sont essentiellement au cours de l'exécution des tests. Lorsque de telles stratégies de test dynamiques sont suivies, le résultat de chaque test influence l'analyse, la conception et l'implémentation des tests ultérieurs. Bien que ces stratégies soient légères à mettre en œuvre et souvent efficaces pour trouver des anomalies, elles exigent de disposer de testeurs experts, peuvent ne plus être maîtrisées avec le temps, souvent ne fournissent pas une bonne couverture des informations, et peuvent être difficiles à reproduire sans l'assistance d'outils spécifiques pour les tests de régression.

2.5.2 Exécution des Tests

L'exécution des tests peut débuter une fois que l'objet de test est livré et que les critères d'entrée pour l'exécution sont satisfaits. Les tests doivent être exécutés suivant les procédures de test, quoiqu'une certaine latitude puisse être laissée au testeur pour lui permettre, d'ajouter des scénarios intéressants supplémentaires et, de s'adapter aux comportements observés pendant les tests (toute défaillance détectée durant une adaptation de ce type doit impliquer la description de la modification de la procédure de test écrite nécessaire à la reproduction de l'anomalie). Les tests automatisés suivent leurs instructions sans déviation possible.

Le cœur de l'activité d'exécution des tests est la comparaison des résultats obtenus avec les résultats attendus. Les testeurs doivent porter la plus grande attention et se concentrer sur ces tâches, sinon tout le travail de conception et d'implémentation des tests peut être perdu si des anomalies ne sont pas détectées (faux positif) ou si des comportements corrects sont considérés comme incorrects (faux négatif). Si les résultats attendus et les résultats observés ne correspondent pas, c'est qu'un incident a eu lieu. Les incidents doivent être soigneusement analysés pour établir la cause (qui peut ou ne pas être une anomalie dans l'objet de test) et recueillir les informations nécessaires à leur résolution. La gestion des incidents est examinée en détail au chapitre 7.

Quand une anomalie est décelée, la spécification de test doit être soigneusement vérifiée afin de s'assurer qu'elle est bien correcte. Un test peut être incorrect pour un certain nombre de raisons : des problèmes liés aux données de tests, des erreurs dans la documentation de test, ou une erreur dans la façon dont il est exécuté. S'il est incorrect, cela implique qu'il soit corrigé et ré-exécuté. Puisque des changements de la base de test et de l'objet de test peuvent rendre une spécification de test incorrecte même après qu'il ait été joué avec succès plusieurs fois, les testeurs doivent rester conscients de la possibilité que les résultats observés soient dus à un test incorrect.

Durant l'exécution des tests, les résultats de ceux-ci doivent être enregistrés convenablement. Les tests qui ont été exécutés mais pour lesquels les résultats n'ont pas été enregistrés doivent être rejoués pour identifier les résultats corrects, engendrant inefficacité et retards. (Il est à noter qu'un enregistrement adéquat permet de remédier aux problèmes de couverture et de capacité de reproductibilité associés à une stratégie de test dynamique.) Étant donné que l'objet de test, le processus test et les environnements de test sont susceptibles de tous évoluer, les enregistrements doivent permettre d'identifier précisément la ou les versions testées.

Les enregistrements des tests fournissent un compte rendu chronologique détaillé sur le déroulement de l'exécution des tests.

Les résultats enregistrés doivent concerner aussi bien les tests eux-mêmes que les événements. Chaque test doit être identifié de manière unique et son statut rattaché au produit de l'exécution du test. Tout événement affectant l'exécution d'un test doit être enregistré. Les informations nécessaires doivent être disponibles pour permettre la mesure de la couverture de test et, la documentation des raisons des retards

et des interruptions. En outre, les informations collectées doivent permettre le suivi des tests : rapports d'avancement des tests, mesure des critères de sortie et amélioration des processus de test.

Les résultats enregistrés varient selon le niveau de test et la stratégie. Par exemple, dans la mise en œuvre de l'automatisation, les tests automatisés recueillent eux-mêmes la plupart des résultats. S'il s'agit de tests d'acceptation manuels, c'est au responsable des tests de compiler ou de rassembler les résultats et logs. Dans certains cas, comme pour l'implémentation des tests, l'enregistrement des résultats est influencé par la réglementation ou les exigences d'audit.

L'IEEE 829 inclut une description des informations qui doivent être renseignées dans un rapport de test.

- Identification du rapport de test
- Description
- Activité et éléments en entrée

La norme BS-7925-2 contient également une description des informations qui doivent être renseignées

Dans certains cas, les utilisateurs ou les clients peuvent participer à l'exécution des tests. Cela peut être utile pour renforcer leur confiance dans le système, mais cela suppose aussi que ces tests ne mettent pas à jour trop d'anomalies. Une telle hypothèse est souvent invalide lors des premiers niveaux de test, mais peut-être valable au cours de tests d'acceptation.

Les métriques pour piloter l'implémentation et l'exécution des tests comprennent :

- Le pourcentage d'environnements de test configurés
- Le pourcentage de données de test chargées
- Le pourcentage de conditions et de cas de test exécutés
- Le pourcentage de cas de test automatisés

2.6 Evaluation des Critères de Sortie et Reporting

La documentation et l'information nécessaires au suivi et au contrôle de la progression des tests sont abordés dans la section 3.6. Du point de vue du processus de test, le suivi de la progression des tests implique de s'assurer que les informations collectées sont appropriées pour répondre aux exigences de reporting. Cela inclut la mesure de la progression jusqu'à l'achèvement.

Les métriques pour suivre la progression des tests et leur achèvement doivent montrer comment les critères de sorties (validés lors de la planification des tests) sont, ou ne sont pas encore, satisfaits. Elles peuvent indiquer :

- Le nombre de conditions de test, de cas de test ou de spécifications de test prévus et ceux qui ont été réellement exécutés ventilés par statut réussi ou échoué
- Le total des anomalies détectées, ventilées par gravité et priorité pour celles qui sont résolues et celles en cours de résolution
- Le nombre de modifications (évolutions) demandées, acceptées (développées) et testées
- Le coût réellement dépensé par rapport au coût prévu
- La durée réelle par rapport à la durée prévue
- Les risques identifiés ventilés par ceux ayant été réduits par les tests et ceux restant en suspens
- Le pourcentage de temps de test perdu à cause d'événements bloquants
- Les articles retestés
- La durée effective de test par rapport au temps total de test prévu

Pour le reporting des tests, l'IEEE 829 défini un rapport de synthèse des tests basé sur les items suivants:

- Identifiant
- Résumé
- Écarts
- Complétude des tests
- Résumé des résultats
- Évaluation
- Résumé des activités
- Approbations

Le reporting des tests peut être produit pour chaque niveau de test dès qu'il est réalisé, ou à l'issue du projet de test lui même.

2.7 Activités de Clôture des Tests

Une fois que l'exécution des tests est considérée comme complète, leurs résultats clés doivent être consignés et soit transmis à la personne concernée soit archivés. Cela correspond aux activités de clôture des tests. Elles se répartissent en quatre groupes principaux :

1) Veiller à ce que tout le travail lié aux tests soit effectivement terminé. Par exemple, l'ensemble des tests planifiés devra avoir été joué ou certains délibérément écartés, et toute anomalie connue devra soit avoir été résolue et sa correction testée, soit reportée pour une future version, soit identifiée comme restriction permanente.
2) Fournir des livrables utiles à ceux qui vont en avoir besoin. Par exemple, toute anomalie présente ou dont la correction est différée doit être communiquée à ceux qui vont utiliser le système et à ceux qui vont gérer le système, et les tests et les environnements de test transmis à ceux qui auront en charge les tests de maintenance. Une base de tests de non régression (automatisés ou manuels) peut être un livrable complémentaire.
3) Organiser ou participer à des réunions rétrospectives (retour d'expérience) peut être l'occasion de capitaliser les enseignements acquis (tant au niveau du projet de test qu'à celui du cycle de vie de développement logiciel complet) pour éviter de reproduire de mêmes erreurs dans l'avenir où adapter les futurs plans projet aux problèmes rencontrés. Par exemple,
   a) En raison de la découverte tardive d'un ensemble d'anomalies imprévues, l'équipe peut proposer qu'un panel plus large de représentants des d'utilisateurs participe aux réunions d'analyse de risques sur les projets futurs.
   b) Les estimations initialement réalisées peuvent s'être avérées significativement erronées et donc les activités d'estimation futures devront prendre en compte les raisons sous-jacentes de ces écarts, par exemple inefficacité des tests réalisés ou sous estimation.
   c) L'analyse de l'évolution, des causes et des effets des anomalies, en identifiant pourquoi et quand elles sont apparues et en faisant ressortir des tendances, par exemple si ce sont les demandes d'évolution tardives qui ont impacté la qualité de la conception et du développement, ou si ce sont les mauvaises pratiques : par exemple l'absence d'un niveau de test qui aurait permis la découverte d'anomalies plus tôt avec un coût de résolution inférieur, et un gain de temps. Aussi, faire le rapprochement entre l'évolution des anomalies et certains éléments comme un contexte de nouvelles technologies, des changements de personnel ou le manque de compétences.
4) Archiver les résultats, des logs, des rapports et tous documents et travaux produits dans la gestion de la configuration système, associée au système lui-même. Par exemple, le plan de test et le plan projet doivent être stockés dans l'archive prévue, et rattachés très clairement au système et à la version concernée.

Ces tâches sont importantes, souvent non réalisées et doivent être explicitement inclues dans le plan de test.

Il est assez commun de laisser de côté l'une ou l'autre de ces tâches, en raison souvent de la dissolution prématurée de l'équipe, de la pression du calendrier des projets suivants, ou de la démobilisation de l'équipe. Pour des projets réalisés au contrat, tout comme pour les activités de développement, le contrat doit spécifier les tâches exigées.

Les métriques pour piloter les activités de clôture des tests peuvent inclure :

- Le pourcentage des cas de test joués au cours de l'exécution des tests (couverture)
- Le pourcentage des cas de test référencés dans le référentiel de test réutilisable
- Le ratio des cas de test automatisés / à automatiser
- Le pourcentage des cas de test identifiés comme tests de régression
- Le pourcentage des rapports d'anomalie en suspens fermés (i.e. différés, sans actions supplémentaires, demandes d'évolution, etc.)
- Le pourcentage des livrables référencés et archivés.

3. Gestion des Tests

Termes

AMDE, plan de test de niveau, plan de test maître, risque du produit, risque du projet, test basé sur les risques, analyse de risque, identification des risques, niveau de risque, gestion des risques, réduction des risques, contrôle des risques, atténuation des risques, type de risque, contrôle des tests, gestion des tests basés sur la session, estimation des tests, niveau de test, gestion des tests, suivi des tests, plan de tests, politique de tests, analyse par point de test, planification des tests, stratégie de test, Delphi à large bande.

3.1 Introduction

L'ensemble de ce chapitre couvre les domaines de connaissance devant être maîtrisés par les Gestionnaires de Tests.

3.2 Documentation de la Gestion des Tests

Une documentation est souvent produite dans le cadre de la gestion des tests. Si les titres et la portée des documents de gestion des tests ont tendance à varier, les documents qui suivent sont des types de documents de gestion de tests trouvés dans des organisations et sur des projets :

- Politique de test, qui décrit la philosophie de l'organisation des tests (et éventuellement de l'assurance qualité).
- Stratégie de test (ou manuel de test), qui décrit l'organisation des méthodes de tests, notamment la gestion des risques de produit et de projet, la division des tests en étapes, les niveaux, ou phases, et les activités de haut niveau associés aux tests.
- Plan de test maître (ou plan de test d'un projet, ou approche des tests), qui décrit l'application de la stratégie de test pour un projet particulier, y compris les niveaux à effectuer et la relation entre ces niveaux.
- Plan de test par niveau (ou plan de test par phase), qui décrit les activités à mener au sein de chaque niveau de test, y compris des précisions complémentaires au plan de test maître pour le niveau à l'étude.

Dans certaines organisations et sur certains projets, ces types de documents peuvent être combinés en un seul document, le contenu de ces types de documents peut être trouvé dans d'autres documents, et une partie du contenu de ces types de documents peut être considéré comme intuitif, non écrit, ou pris en compte dans les méthodologies traditionnelles pour les tests. Des organisations et des projets plus grands et officiels ont tendance à avoir tous ces types de documents de travail, alors que des organisations et des projets plus petits et moins formels ont tendance à avoir moins de documents de la sorte. Ce syllabus décrit chacun de ces types de documents séparément, mais dans la pratique, le contexte du projet et de son organisation détermine la bonne utilisation de chaque type de document.

3.2.1 Politique de Tests

La politique de tests décrit la philosophie de l'organisation des tests (et éventuellement d'assurance qualité). Il est fixé, soit par écrit soit par la direction, établissant les objectifs généraux de tests que l'organisation souhaite atteindre. Cette politique peut être élaborée par les services de technologie de l'information, de la recherche et du développement, ou du développement du produit, et devrait refléter les valeurs organisationnelles et les objectifs du test.

Dans certains cas, la politique de tests sera complémentaire à une plus large politique de qualité ou en sera un composant. Cette politique de qualité décrit l'ensemble des valeurs et des objectifs de gestion liés à la qualité.

Lorsqu'une politique de tests écrite existe, cela peut être un document court et de haut niveau qui :

- Fournit une définition des tests, telle que l'instauration de la confiance que le système fonctionne comme prévu, et la détection des défauts.
- Établit un processus de test, par exemple, la planification et le contrôle des tests, l'analyse et la conception des tests, la mise en œuvre et l'exécution des tests, l'évaluation des critères de sortie de test et les rapports de tests et, les activités de fin de test.
- Explique comment évaluer l'efficacité des tests, par exemple, le pourcentage de défauts à détecter (Pourcentage de détection des défauts, ou DDP) et le coût relatif des défauts détectés pendant une phase de tests, au lieu d'une détection après mise en production.
- Définit les objectifs de qualité souhaités, comme la fiabilité (par exemple mesurée en termes de taux d'échec) ou l'utilisabilité.
- Précise les activités pour l'amélioration du processus de test, par exemple, l'application du 'Test Maturity Model' ou du modèle de 'Test Process Improvement', ou la mise en œuvre des recommandations à partir des rétrospectives du projet.

La politique de tests peut adresser les activités de test pour les nouveaux développements ainsi que pour la maintenance. Elle peut également référencer un standard pour la terminologie des tests à utiliser dans toute l'organisation.

3.2.2 Stratégie de Test

La stratégie de test décrit les méthodes de tests de l'organisation, notamment la gestion des risques des produits et des projets ; la répartition des tests en niveaux, ou phases ; et les activités de haut niveau associées aux tests. La stratégie de test, son processus et ses activités doivent être compatibles avec la politique de test. Ils devraient fournir des exigences de test génériques pour l'organisation ou pour un ou plusieurs projets.

Comme indiqué dans le syllabus de niveau fondation, les stratégies de test (aussi appelées approches de test) peuvent être classées en fonction du moment auquel commence la conception des tests :

- les stratégies de test préventives conçoivent les tests tôt pour prévenir les défauts,
- les stratégies de test réactives placent la conception des tests après mise en production du logiciel ou du système.

Les stratégies (ou approches) typiques sont les suivantes:

- les stratégies analytiques, comme les tests basés sur les risques
- les stratégies basées sur les modèles, comme le test du profil opérationnel
- les stratégies méthodiques, comme celles basées sur les caractéristiques qualité
- les stratégies conformes aux processus ou aux standards, comme celles basées sur IEEE 829
- les stratégies dynamiques et heuristiques, comme l'utilisation des attaques basées sur les bugs
- les stratégies consultatives, comme les tests dirigés par les utilisateurs
- les stratégies de tests de régression, comme l'automatisation large.

Il peut y avoir une combinaison de stratégies différentes. La stratégie choisie doit être adaptée aux besoins et aux moyens de l'organisation, et ces organisations peuvent faire des stratégies sur mesure pour s'adapter à certaines opérations et à certains projets.

Dans de nombreux cas, une stratégie de test explique les risques projet et produit, et décrit comment le processus de test gère ces risques. Dans ces cas, le lien entre les risques et les tests devrait être expressément expliqué, de même que les possibilités de réduction et de gestion de ces risques.

La stratégie de test peut décrire les niveaux de test à effectuer. Dans ce cas, la stratégie devrait donner les orientations de haut niveau sur les critères d'entrée et de sortie de chaque niveau, et les relations entre les niveaux (par exemple, les objectifs de répartition de la couverture de test).

La stratégie de test peut également décrire les éléments suivants:

- Les procédures d'intégration
- Les techniques de spécification des tests
- L'indépendance des tests (qui peut varier selon le niveau)
- Les normes obligatoires et facultatives
- Les environnements de test

- L'automatisation des tests
- La réutilisation des produits logiciels des produits de test
- Re-test et tests de régression
- Contrôle de test et élaboration de rapports
- Mesures et métriques de tests
- Gestion des incidents
- L'approche gestion de la configuration du Testware

Des stratégies de test à court terme mais aussi à long terme devraient être définies, dans un ou plusieurs documents. De stratégies de test différentes peuvent être appropriées à des organisations et projets différents. Par exemple, lorsque la sécurité ou des applications critiques sont en jeu, une stratégie plus intensive peut être plus appropriée que dans d'autres cas.

3.2.3 Plan de Test Maître

Le plan de test maître décrit l'application de la stratégie de tests pour un projet particulier, y compris les niveaux à effectuer et la relation entre ces niveaux. Le plan de test maître doit être conforme à la politique et la stratégie de test, et s'il y a des écarts, les expliquer. Le plan de test maître doit compléter le plan de gestion de projet ou le guide des opérations, dans la mesure où il doit décrire l'effort de test qui est partie intégrante du projet global ou de l'opération.

Bien que le contenu spécifique et la structure du plan de test maître varient en fonction de l'organisation, des normes en matière de documentation et de la formalité du projet, les sujets typiques pour un plan de test maître comprennent :

- Les points à tester et à ne pas tester
- Les attributs de qualité à tester et à ne pas tester
- Le calendrier des tests et le budget (qui devrait être aligné sur le budget du projet ou de l'opérationnel)
- Les cycles d'exécution des tests et leur lien avec le plan de mise en production du logiciel
- La justification commerciale pour les tests et leur valeur
- Les relations et les livrables des tests et d'autres personnes ou départements
- L'identification des éléments de test qui sont où ne sont pas dans le périmètre de chaque niveau décrit
- Les critères d'entrée, les critères de poursuite (suspension / reprise), et les critères de sortie pour chaque niveau et les relations entre ces niveaux
- Les risques du projet de test.

Sur des projets ou des opérations de moindre envergure, où seulement un niveau de test est formalisé, le plan de test maître et le plan de test pour ce niveau formalisé seront souvent regroupés en un seul document. Par exemple, si le test de système est le seul niveau formel, avec des tests informels de composant et d'intégration réalisés par des développeurs et des tests informels d'acceptation réalisés par des clients dans le cadre d'un processus de béta test, alors le plan de tests système peut inclure les éléments mentionnés dans la présente rubrique.

En outre, les tests dépendent généralement d'autres activités dans un projet. Si ces activités ne sont pas suffisamment documentées, notamment en ce qui concerne leurs influences et leurs relations avec les tests, des sujets liés à ces activités peuvent être couverts par le plan de test maître (ou dans le plan de test de niveau approprié). Par exemple, si le processus de gestion de la configuration n'est pas documenté, le plan de test devrait préciser comment les objets de test doivent être livrés à l'équipe de test.

3.2.4 Plan de Test de Niveau

Le plan de test de niveau décrit les activités à mener au sein de chaque niveau de test, le cas échéant vient compléter le plan de test maître pour le niveau objet de l'étude. Il prévoit le calendrier, les tâches, le détail des étapes pas nécessairement couvert par le plan de test maître. En outre, dans la mesure où des normes et des modèles différents s'appliquent à la spécification de tests à différents niveaux, ces détails seront couverts dans le plan de test de niveau.

Pour des projets ou des opérations moins formels, le plan de test de niveau est souvent le seul document de gestion des tests qui est rédigé. Dans de telles situations, certains des éléments d'information mentionnés dans les rubriques 3.2.1, 3.2.2 et 3.2.3 pourraient être abordés dans le présent document.

3.3 Modèles de Documentation des Plans de Test

Comme mentionné dans la rubrique 3.2, le contenu spécifique et la structure du plan de test maître varient en fonction de l'organisation, ses normes en matière de documentation, et la formalité du projet. De nombreuses organisations développent ou adaptent des modèles communs pour assurer l'uniformité et la lisibilité au travers des projets et des opérations, et les modèles sont disponibles pour la documentation du plan de test.

L'IEEE 829 « Standard for Software Testing Documentation » contient les modèles de documentation de test et des conseils pour leur application, y compris pour la préparation des plans de test. Il aborde également la question connexe de la diffusion de l'élément de test (c'est-à-dire, la livraison des items de test à l'équipe de test).

3.4 Estimation des Tests

L'estimation, en tant qu'activité de gestion, est la création d'un objectif approximatif pour les coûts et les dates d'achèvement associés aux activités impliquées dans une opération ou un projet. Les meilleures estimations:

- Représentent la sagesse collective des praticiens expérimentés et ont le soutien des participants impliqués
- Fournissent des catalogues spécifiques et détaillés des coûts, des ressources, des tâches et des personnes impliquées
- Présentent, pour chaque activité estimée, les coûts, l'effort et la durée les plus probables

L'estimation de l'ingénierie des logiciels et des systèmes a longtemps été connue pour être semée d'embûches, à la fois techniques et politiques, bien que les meilleures pratiques de la gestion de projet pour les estimations soient bien établies. L'estimation des tests est l'application de ces meilleures pratiques aux activités de test associées à un projet ou à une opération.

L'estimation de test devrait inclure toutes les activités impliquées dans le processus de test, c'est-à-dire la planification et le contrôle des tests, l'analyse et la conception des tests, la mise en œuvre et l'exécution des tests, l'évaluation et le rapport des tests, et les activités de fin de tests. Le coût estimé de l'effort et, surtout, de la durée d'exécution des tests est souvent le plus intéressant pour la direction, comme l'exécution des tests est en général sur le chemin critique du projet. Toutefois, les estimations d'exécution des tests ont tendance à être difficiles à produire et sont peu fiables lorsque la qualité générale du logiciel est faible ou inconnue. Une pratique courante est également d'estimer le nombre de cas de test requis. Les hypothèses faites au cours de l'estimation devraient toujours être documentées dans le cadre de cette estimation.

L'estimation des tests doit examiner tous les facteurs qui peuvent influer sur le coût, l'effort et la durée des tests. Ces facteurs comprennent les informations suivantes (liste non limitative) :

- Le niveau de qualité requis pour le système
- La taille du système à tester
- Les données historiques des projets de test précédents (cela peut également inclure des données de référence)
- Les facteurs du processus, notamment : la stratégie de test, le cycle de vie du développement ou de la maintenance et la maturité de ce processus, ainsi que la précision des estimations du projet.
- Les facteurs matériels, notamment : l'automatisation des tests et les outils, les environnements de test, les données de test, le(s) environnement(s) de développement; la documentation du projet (par exemple, les exigences, la conception, etc.), et les produits de tests réutilisables.
- Les facteurs de personnes, notamment : les managers et les leaders techniques ; l'engagement et les attentes de la direction ; les compétences, l'expérience et les attitudes de l'équipe du projet ; la stabilité de l'équipe du projet ; les relations dans l'équipe du projet ; l'assistance aux

environnements de test et de débogage ; la disponibilité d'entreprises et de consultants qualifiés ; et la connaissance du domaine.

Parmi les autres facteurs qui peuvent influer sur l'estimation des tests, il y a ce qui concerne la complexité du processus ; la technologie ; l'organisation ; le nombre de parties prenantes dans les tests ; de nombreuses équipes secondaires, en particulier des équipes géographiquement éloignés ; une montée en compétences difficile ; des besoins de formation et d'orientation ; l'assimilation ou le développement de nouveaux outils ; les techniques ; le matériel personnalisé ; le nombre d'articles de test; des exigences d'un haut degré de détail pour les spécifications de tests, en particulier avec un standard de documentation inconnu ; un planning complexe de l'arrivée des composants, en particulier pour les tests d'intégration et leur développement ; et des données fragiles de test (par exemple, des données qui sont sensibles dans le temps).

L'estimation peut se faire soit de façon ascendante ou descendante. Les techniques suivantes peuvent être utilisées dans les estimations de test :

- Intuition et supposition
- L'expérience du passé
- Structure de la répartition du travail (WBS Work-breakdown-structure)
- Des sessions d'estimation en équipe (par exemple, Wide Band Delphi)
- Estimation sur trois points
- Analyse par point de test (TPA) [Pol02]
- Les règles et normes de la société
- Des pourcentages de l'effort pour l'ensemble du projet ou du niveau des effectifs (par exemple, les ratios testeurs / développeurs)
- L'histoire et les métriques de l'organisation, y compris les modèles de métriques dérivés qui estiment le nombre de défauts, le nombre de cycles de tests, le nombre de cas de test, l'effort moyen de chaque test, et le nombre de cycles de régression impliqués
- Les modèles moyens et de prévision du métier, tels que les points de tests, les points de fonction, les lignes de code, l'effort estimé du développeur, ou d'autres paramètres du projet

Dans la plupart des cas, l'estimation, une fois établie, doit être livrée à la direction, avec une justification (voir 3.7 Valeur Commerciale du Test). Souvent, certaines négociations s'en suivent, résultant souvent en une refonte de l'estimation. Idéalement, l'estimation finale des tests représente le meilleur équilibre possible entre les objectifs du projet et ceux de l'organisation dans les domaines de la qualité, de la planification, du budget et des fonctionnalités.

3.5 Planification des Tests

En général, la planification à l'avance de tout un ensemble d'activités permet la découverte et la gestion des risques pour ces activités, une bonne coordination en temps voulu avec les autres acteurs, et un plan de haute qualité. Il en est de même de la planification des tests. Toutefois, dans le cas de la planification des tests, une planification avancée basée sur l'estimation des tests présente encore plus d'avantages, notamment :

- La détection et la gestion des risques du projet et de problèmes en dehors du cadre du test lui-même
- La détection et la gestion des risques du produit (de la qualité) et de problèmes qui précèdent l'exécution des tests
- La reconnaissance des problèmes dans le plan de projet ou dans d'autres livrables du projet
- Des opportunités d'augmenter le personnel alloué aux tests, le budget, l'effort et / ou la durée pour atteindre une meilleure qualité
- L'identification des éléments critiques (et donc la possibilité d'accélérer la livraison de ces composants)

La planification des tests se fait en coopération étroite avec le développement, puisque le démarrage des tests dépend fortement du calendrier de développement (de livraisons).

Lorsque toutes les informations requises pour construire un plan de test complet sont disponible, il est souvent trop tard pour tirer parti de ces avantages potentiels. Par conséquent, il est recommandé d'établir

un plan de tests sous forme de brouillon le plus tôt possible. Dès qu'arrivent de plus amples renseignements, l'auteur du plan de tests (en général un gestionnaire des tests) peut rajouter ces informations au plan. Cette approche itérative pour la création du plan de tests, sa diffusion, et sa revue permet également de se servir du plan de tests comme véhicule de promotion de consensus, de communication et de discussion sur les tests.

3.6 Contrôle & Suivi de l'Avancement des Tests

Il y a cinq principales dimensions sur lesquelles la progression des tests est suivie :

- Les risques produits
- Les défauts
- Les tests
- La couverture
- La confiance

Les risques produits, les incidents, les tests, et la couverture peuvent être et sont souvent mesurés et communiqués de manière spécifique dans le cadre du projet ou de l'opération. De préférence ces mesures sont liées aux critères de sortie définis dans le plan de test. La confiance, bien que mesurable par des enquêtes, est habituellement signalée subjectivement.

On peut citer comme métriques liées aux risques produits :

- Nombre d'autres risques (y compris le type et le niveau des risques)
- Nombre de risques contrôlés (y compris le type et le niveau des risques)

On peut citer comme métriques relatives à des défauts :

- Nombre total de défauts rapportés (identifiés) comparé au nombre total de défauts résolus (clôturés)
- Temps moyen entre défaillances ou le taux d'arrivée de défaillances
- Répartition du nombre de défauts liés : à certains éléments ou composants de tests ; aux causes racines ; aux sources ; aux versions de test ; à la phase d'introduction, de détection ou de résolution ; et, dans certains cas, au propriétaire
- Tendances du temps de latence entre la déclaration d'un défaut et sa résolution

On peut citer comme métriques relatives aux tests :

- Le nombre total de tests prévus, spécifiés (développés), exécutés, passés, en échec, bloqués, et ignorés
- Le statut des tests de régression et de confirmation
- Le nombre d'heures de tests prévu par jour comparé au nombre d'heures réel

On peut citer comme métriques liées à la couverture de test :

- La couverture des exigences et des éléments de conception
- La couverture des risques
- La couverture de l'environnement / la configuration

Ces mesures peuvent être signalées verbalement, numériquement dans des tableaux, ou avec des graphiques, et peuvent être utilisées pour un certain nombre de raisons, notamment :

- L'analyse, pour découvrir ce qui se passe avec le produit, le projet ou le processus à l'aide des résultats des tests
- Les rapports, pour communiquer les conclusions des tests aux acteurs du projet et aux différents intervenants
- Le contrôle, pour changer le cours des tests ou du projet dans son ensemble et suivre les résultats de cette correction de cap

La meilleure façon de recueillir, d'analyser et de faire le rapport de ces mesures des tests dépend des besoins d'information, des objectifs et des aptitudes des personnes qui utiliseront ces mesures.

Lorsque vous utilisez des résultats des tests pour influencer ou mesurer des efforts de contrôle sur le projet, les options suivantes doivent être envisagées :

- La revue de l'analyse des risques qualité, des priorités des tests et / ou des plans de tests

- Le rajout de ressources ou sinon l'augmentation d'effort de test
- Le report de la date de sortie
- L'allègement ou le renforcement des critères de sortie des tests

La mise en œuvre de ces options exige en général un consensus entre les acteurs du projet ou de l'opération et l'accord des responsables.

La manière dont un rapport de test est mis en place dépend en grande partie du public cible, par exemple un responsable projet ou un responsable métier. Pour un chef de projet, avoir des informations détaillées sur les défauts à priori intéressant ; pour un responsable métier, le statut des risques du produit pourrait être un point clé du rapport.

L'IEEE 829 "Standard for Software Testing Documentation" fournit un modèle de rapport de tests.

Sur la base des différences par rapport au plan de tests, comme indiqué dans le rapport de progression des tests, un contrôle de test doit être effectué. Le contrôle de test vise à réduire au minimum les différences par rapport au plan de tests. Les mesures de contrôle possibles comprennent :

- La revue de la priorisation des cas de test
- L'obtention de ressources supplémentaires
- Le report de la date de sortie
- Un changement de la portée (les fonctionnalités) du projet
- La revue des critères de complétude des tests (uniquement avec l'accord des parties prenantes).

3.7 Valeur Commerciale du Test

Alors que la plupart des organisations sont conscients de la valeur des tests d'une certaine manière, peu de chefs, y compris les gestionnaires de test, peuvent quantifier, décrire, ou articuler cette valeur. En outre, de nombreux gestionnaires de test, des leaders de tests et des testeurs se concentrent sur les détails tactiques des tests (les aspects spécifiques à la tâche ou au niveau), tout en ignorant la plus grande question stratégique (d'un niveau supérieur) relative aux tests qui intéresse d'autres acteurs du projet, en particulier les managers.

Le test fournit à l'organisation, au projet, et / ou à l'exploitation une valeur tant sur le plan des moyens quantitatifs que qualitatifs :

- Des valeurs quantitatives comprennent les défauts trouvés qui sont prévenus ou fixés avant la mise en production, les défauts trouvés avant la mise en production, la réduction des risques en exécutant des tests, et la fourniture d'un statut sur le projet, le processus et le produit.
- Des valeurs qualitatives comprennent une image de qualité renforcée, des versions plus lisses et plus prévisibles, une confiance accrue, l'instauration de la confiance, la protection contre la responsabilité juridique, et la réduction des risques de perte de l'ensemble des missions ou même des vies.

Les gestionnaires et les leaders de test devraient comprendre lesquelles de ces valeurs s'appliquent à leur organisation, projet, et / ou à l'opération, et être en mesure de communiquer sur les tests en fonction de ces valeurs.

Une méthode établie pour mesurer la valeur quantitative et l'efficacité des tests est appelé « le coût de la qualité » (ou, parfois, « le coût de la mauvaise qualité »). Le coût de la qualité implique la classification des projets ou des frais de fonctionnement en quatre catégories :

- Coût de la prévention
- Coût de détection
- Coût des défaillances en interne
- Coût des défaillances externe

Une partie du budget de test est un coût de détection, alors que le reste est un coût de défaillance interne. Le coût total de détection et des défaillances internes est généralement bien inférieur au coût des défaillances externes, ce qui fait des tests un excellent rapport qualité / prix. En déterminant les coûts dans ces quatre catégories, les gestionnaires et les leaders de tests créent une analyse commerciale de rentabilité convaincante pour les tests.

3.8 Tests Distribués, Externalisés et Internalisés

Dans de nombreux cas, tous les efforts de test sont réalisés par une équipe de test unique, composée de collègues de l'équipe du projet, à un seul et même endroit que le reste de l'équipe du projet. Si l'effort de test se produit à plusieurs endroits, cet effort de test peut être appelé distribué. Si l'effort de test est effectué à un ou plusieurs endroits par des personnes qui ne sont pas des collègues de l'équipe du projet et qui ne sont pas localisés avec l'équipe du projet, cet effort de test peut être appelé externalisé. Si l'effort de test est effectué par des personnes qui sont localisés avec l'équipe du projet mais qui ne sont pas des collègues, cet effort de test peut être appelé internalisé.

La nécessité de définir clairement les voies de communication et les attentes en termes de missions, des tâches et des livrables est commune à tous ces efforts de test. L'équipe de projet ne doit pas trop compter sur une communication informelle, comme des discussions à la pause café entre collègues d'un même bureau. La localisation, le fuseau horaire, les différences culturelles et linguistiques rendent ces questions de communication encore plus critiques.

Il est également nécessaire à tous les efforts de tests d'harmoniser les méthodologies. Si deux groupes de test utilisent des méthodes différentes, ou si le groupe de test utilise une méthodologie différente de l'équipe de développement ou de la direction de projet, cela se traduira par des problèmes importants, en particulier au cours de l'exécution des tests.

Pour les tests distribués, la division du travail de test à travers plusieurs sites devra être explicite et choisie intelligemment. En l'absence de telles orientations, le groupe le plus compétent ne peut pas faire le travail de test pour lequel il est hautement qualifié. En outre, le travail de test dans son ensemble va souffrir de lacunes (qui augmentent un risque résiduel de qualité à la livraison) et de chevauchements (qui réduisent l'efficacité).

Enfin, pour tous ces efforts de test, il est essentiel que l'ensemble de l'équipe projet développe et maintienne la confiance que chacune des équipes de test s'acquittera de son rôle correctement en dépit de défis d'organisation, de culture, de langues et de frontières géographiques. Le manque de confiance conduit à l'inefficacité et aux retards associés aux activités de vérification, à la détermination de faute, et au jeu de la politique d'organisation.

3.9 Tests Basés sur les Risques

3.9.1 Introduction aux Tests Basés sur les Risques

Le risque est la possibilité d'un résultat non souhaité. Les risques existent lorsqu'il pourrait survenir un problème qui diminuerait l'image de qualité du produit ou de réussite du projet pour le client, l'utilisateur ou les acteurs.

Lorsque l'effet principal du problème potentiel est sur la qualité du produit, les problèmes potentiels sont appelés les risques du produit (ou risques de qualité). On peut citer un possible défaut de fiabilité (bug) qui pourrait faire planter un système lors de son fonctionnement normal. Lorsque le principal effet du problème potentiel est sur la réussite du projet, ce problème potentiel est appelé un risque de projet (ou un risque de planning). Par exemple : pénurie d'effectifs, qui pourrait retarder un projet.

Tous les risques ne sont pas aussi préoccupants. Le niveau de risque est influencé par différents facteurs :

- la probabilité que le problème survienne
- l'impact du problème s'il survenait

Dans les tests basés sur les risques, le test est mené d'une manière qui répond à des risques de trois façons :

- Attribution de l'effort de test, choix de techniques, ordonnancement des activités de test, et réparation des défauts (bugs) doivent être faits d'une manière qui convienne au niveau de risque associé à chaque risque produit (qualité) significatif identifié.
- Planification et gestion du travail de test doivent être menées d'une manière qui atténue les risques et qui répond à chaque risque éventuel important du projet (planification).

- État d'avancement des résultats de tests et statut du projet doivent être définis en termes de risques résiduels, par exemple en se basant sur les tests qui n'ont pas encore été joués ou ont été ignorés, ou sur les défauts qui n'ont pas encore été fixés ou rejoués.

Ces trois types de réponses aux risques devraient avoir lieu dans l'ensemble du cycle de vie, pas simplement au début et en fin de projet. Plus précisément, au cours du projet, les testeurs devraient s'efforcer à :

- Réduire les risques en trouvant le nombre le plus important de défauts (pour les risques de produit) et en mettant en œuvre les activités appropriées de réduction et de contrôle telles qu'énoncées dans la stratégie de test et le plan de test.
- Faire une évaluation des risques, en augmentant ou en diminuant la probabilité ou l'impact des risques déjà identifiés et analysés en fonction des informations recueillies à mesure que le projet se déroule.

Dans les deux cas, les mesures prises influencent la manière dont les tests répondent aux risques.

Les tests basés sur les risques peuvent être considérés comme analogue à une assurance selon différentes vues. On achète une assurance quand on est préoccupé par certains risques potentiels, et on ne tient pas compte de risques qui n'inquiètent pas. L'analyse quantitative similaire à l'évaluation des risques faite par les professionnels de l'assurance peut être pertinente, mais en général les tests basés sur les risques reposent sur des analyses qualitatives.

Pour être en mesure de s'acquitter correctement des tests fondés sur les risques, les testeurs doivent être en mesure d'identifier, d'analyser et d'atténuer les risques typiques du produit et du projet liés à la sécurité, au métier et aux préoccupations d'ordre économique, à la sécurité du système et des données, et aux facteurs techniques et politiques.

3.9.2 Gestion des Risques

La gestion des risques peut être considérée comme composée de trois activités principales :

1. Identification des risques
2. Analyse des risques
3. Réduction des risques (aussi appelé contrôle des risques)

Ces activités sont séquentielles d'une certaine manière, mais la nécessité d'une gestion des risques continue mentionnée dans les rubriques précédentes et suivantes signifie que, pendant la plus grande partie du projet, les trois types d'activité de gestion des risques doivent être menées par itération.

Pour être plus efficace, la gestion des risques comprend tous les acteurs sur le projet, bien que parfois les réalités du projet fassent que quelques intervenants se substituent à d'autres parties prenantes. Par exemple, dans le développement de logiciels de marché de masse, un petit échantillon de clients potentiels peut être invité à aider à identifier les défauts potentiels qui auraient une incidence plus forte sur leur utilisation du logiciel – dans ce cas, l'échantillon de clients potentiels sert de substitut à l'ensemble de la clientèle éventuelle.

En raison de leur expertise particulière, les analystes de tests devraient être activement impliqués dans l'identification des risques et les processus d'analyse.

3.9.2.1 Identification des Risques

Pour les risques de produit et de projet, les testeurs peuvent identifier les risques par une ou plusieurs des techniques suivantes :

- Des entretiens avec des experts
- Des évaluations indépendantes
- En utilisant des modèles de risque
- Les leçons tirées (par exemple des sessions d'évaluation des projets)
- Les ateliers de risque (par ex AMDE)
- Les Brainstorming
- Les listes de contrôle (« check-lists »)
- L'expérience passée

En faisant appel à un échantillon d'acteurs le plus large possible, le processus d'identification des risques augmente ses chances de détecter le plus grand nombre possible de risques importants.

Dans certaines approches d'identification des risques, le processus s'arrête à l'identification du risque lui-même.

Certaines techniques, telle que l'Analyse des Modes de Défaillance et Effets (AMDE), exigent pour chaque mode de défaillance potentiel de procéder à l'identification des effets du mode de défaillance sur le reste du système (y compris les systèmes de niveau supérieur dans le cas des systèmes de systèmes), et sur les utilisateurs potentiels du système.

D'autres techniques, telle que l'analyse des dangers, exigent une anticipation de la source du risque.

Pour une description de l'utilisation de l'Analyse de Modes de Défaillance et Effets, l'analyse des dangers et l'analyse de criticité, voir la rubrique 3.10 et [Stamatis95], [Black02], [Craig02], [Gerrard02].

3.9.2.2 Analyse des Risques

Alors que l'identification des risques doit identifier autant de risques pertinents que possible, l'analyse des risques est l'étude de ces risques identifiés – en particulier, le classement de chaque risque et la détermination de la probabilité et l'impact associés à chacun des risques.

La classification des risques affecte un type approprié à chaque risque. Les types de risques de qualité typiques sont abordés dans la norme ISO 9126 avec les caractéristiques de qualité pour classer les risques. Certaines organisations ont leur propre ensemble de caractéristiques de qualité. Il est à noter que, lors de l'utilisation des listes de contrôle pour l'identification des risques, la sélection du type de risque se produit souvent au cours de l'identification.

La détermination du niveau de risque conduit généralement à évaluer, pour chaque point de risque, la probabilité de son occurrence et son impact. La probabilité d'occurrence est souvent interprétée comme la probabilité que le problème potentiel puisse exister dans le système testé. En d'autres termes, il émane des risques techniques.

Les facteurs qui influent sur les risques techniques comprennent:

- La complexité de la technologie et des équipes
- Le personnel et les questions de formation entre les analystes métier, les concepteurs et programmeurs
- La gestion de conflits au sein de l'équipe
- Les problèmes contractuels avec les fournisseurs
- La répartition géographique de l'organisation de développement
- Les anciens systèmes contre les nouvelles approches
- Les outils et les technologies
- Un mauvais pilotage de la direction ou technique
- Le temps, les ressources et la pression de la direction
- Manque d'une assurance qualité au début
- Fort taux de changement
- Fort taux de défauts très tôt
- Les questions d'interface et d'intégration

L'impact de la survenance du risque est souvent interprété comme la gravité de l'effet sur les utilisateurs, les clients, ou d'autres acteurs. En d'autres termes, il résulte des risques métier. Les facteurs influant sur les risques métier comprennent:

- La fréquence d'utilisation de la fonctionnalité touchée
- Les dégâts à l'image de marque
- La perte de commerce
- La possibilité de pertes financières, écologiques, sociales ou juridiques
- Des sanctions légales, civiles ou pénales
- La perte de licence(s)
- Le manque de contournements raisonnables
- La visibilité de l'échec conduisant à une contre publicité

Les testeurs peuvent avoir une approche quantitative ou qualitative pour établir le niveau de risque. Si la probabilité et l'impact peuvent être déterminés quantitativement, on peut multiplier les deux valeurs pour calculer le coût de l'exposition. C'est la perte associée à un risque particulier.

En règle générale, cependant, le niveau de risque ne peut être établi que qualitativement. C'est à dire qu'on peut parler d'un risque très élevé, élevé, moyen, faible ou très faible, mais on ne peut pas dire avec certitude si la probabilité est de 90%, 75%, 50%, 25%, ou 10%. Cette approche qualitative ne doit pas être considérée comme inférieure aux approches quantitatives – en effet, lorsque des approches quantitatives sont utilisées de façon inappropriée, elles induisent en erreur les acteurs sur la compréhension et la gestion du risque. Des approches informelles telles que celles décrites dans [vanVeenendaal02], [Craig02] et [Black07b] sont souvent qualitatives et moins rigoureuses.

À moins que l'analyse des risques soit fondée sur des données de risque vastes et statistiquement valables (comme c'est le cas dans le secteur des assurances), indépendamment du fait que l'analyse des risques soit de nature qualitative ou quantitative, l'analyse des risques sera fondée sur la probabilité perçue du risque et de son impact. Cela signifie que les perceptions personnelles et les opinions au sujet de ces facteurs auront une influence sur la détermination du niveau de risque. Les chefs de projet, les programmeurs, les utilisateurs, les analystes métier, les architectes et les testeurs ont généralement des perceptions différentes et donc peut-être des opinions différentes sur le niveau de risque pour chaque risque. Le processus d'analyse des risques devrait inclure un moyen de parvenir à un consensus ou, dans le pire des cas, de dicter un accord sur le niveau de risque. Sinon, les niveaux de risque ne pourront pas être utilisés comme un guide pour les activités de réduction des risques.

3.9.2.3 Réduction des Risques

Dès lors qu'un risque a été identifié et analysé, il y a quatre principaux moyens de le gérer :

1. Réduire le risque par des mesures préventives pour diminuer sa probabilité et / ou son impact.
2. Faire des plans d'urgence pour réduire son impact si le risque devient une réalité.
3. Transférer le risque à une autre partie pour le traiter.
4. Ignorer et accepter le risque.

Les options pour sélectionner une de ces fonctions dépendent des avantages et des opportunités créées par chaque option, ainsi que le coût et, potentiellement, d'autres risques associés à chaque option.

Stratégies de Réduction des Risques

Dans la plupart des stratégies de tests basées sur les risques, les activités d'identification des risques, d'analyse des risques, et de réduction des risques sont à la base du plan de tests maître et des autres plans de tests. Le niveau de risque associé à chacun des points de risque détermine l'étendue des efforts de test (c'est-à-dire, les mesures de réduction) pris pour faire face à chaque risque. Certaines normes liées à la sécurité (par exemple, les FAA DO-178B/ED 12B, IEC 61508), prescrivent des techniques de test et un degré de couverture en fonction du niveau de risque.

Réduction des Risques sur le Projet

Si des risques de projet sont identifiés, ils pourront être communiqués et pris en charge par le chef de projet. La réduction de ces risques n'est pas toujours de la compétence de l'organisation des tests. Toutefois, certains risques liés aux projets peuvent et doivent être contrôlés avec succès par le gestionnaire de test, tels que:

- La préparation des environnements et des outils de test
- La disponibilité et la formation du personnel de test
- La faible qualité des entrées de test
- Les changements trop importants pour les entrées de test
- L'absence de normes, de règles et de techniques pour l'effort de test

Les approches de réduction des risques comprennent la préparation préliminaire des objets de tests, le pré-test des équipements de test, les pré-tests des versions antérieures du produit, des critères d'entrée plus sévères pour les tests, des exigences pour la testabilité, la participation à l'examen des résultats des projets antérieurs, la participation aux problèmes et à la gestion du changement, et le suivi de l'avancement du projet et de sa qualité.

Réduction des Risques sur le Produit

Quand on parle d'un risque (qualité) produit, alors le test est une forme de réduction de ces risques. Dans le cas où l'on trouve des défauts, les testeurs réduisent les risques en mettant en évidence des défauts et des possibilités de les traiter avant l'intégration dans la version. Dans le cas où les testeurs ne trouvent pas les défauts, les tests réduisent le risque en prouvant que, sous certaines conditions (c'est-à-dire, les conditions du test), le système fonctionne correctement.

Les risques (qualité) produit peuvent être atténués par d'autres activités que le test. Par exemple, si l'on identifie que les exigences ne sont pas écrites correctement, une solution efficace de réduction de risque serait de mettre en place des revues méthodiques, plutôt que d'écrire et prioriser des tests qui ne seront lancés qu'après que le logiciel mal conçu soit transformé en algorithmes et en code exécutable.

Le niveau de risque est également utilisé pour prioriser les tests. Dans certains cas, tous les tests de hauts risques sont exécutés avant tout test de moindre risque, et les tests sont exécutés strictement dans le ordre des risques (souvent appelé « parcours en profondeur »). Dans d'autres cas, une méthode d'échantillonnage est utilisée pour sélectionner un échantillon de tests pour tous les risques mis en évidence, en utilisant le poids des risques pour couvrir la sélection et en même temps de façon à couvrir tous les risques au moins une fois (souvent appelé « parcours en largeur »).

D'autres mesures de réduction des risques peuvent être utilisées par les testeurs :

- Le choix d'une technique de conception des tests appropriée
- Les revues et les inspections
- Les revues de conception des tests
- Le niveau d'indépendance
- La personne la plus expérimentée
- La manière dont on effectue le re-test
- Le test de régression

Dans [Gerrard02], la notion de l'efficacité des tests est introduite pour indiquer (en pourcentage) comment l'efficacité des tests doit aussi être une mesure de réduction des risques. La tendance serait à ne pas tester pour réduire les risques là où il y a un faible niveau d'efficacité des tests.

Que les tests basés sur les risques appliquent le « parcours en profondeur » ou le « parcours en largeur », il est possible que le temps alloué aux tests soit consommé sans que tous les tests soient exécutés. Les tests basés sur les risques permettent aux testeurs de faire un rapport à la direction qui rappelle le niveau de risque existant, et permet à la direction de décider s'il y a lieu d'intensifier les tests ou de transférer les risques restant sur les utilisateurs, les clients, le service d'assistance / de support technique, et/ou sur le personnel d'exploitation.

Ajustement du Test pour Cycles de Tests Futurs

S'il y a du temps pour faire d'autres tests, tout prochain cycle de test doit être adapté à une nouvelle analyse des risques. Les principaux facteurs sont ici des risques de produit qui sont totalement nouveaux, ou qui ont beaucoup changé ; des zones instables ou sujettes à défauts sont découvertes pendant les tests ; des risques provenant de défauts fixés ; la tentative de concentrer les tests sur les différents types de défauts constatés au cours des tests ; et, les domaines éventuellement sous-testés (faible couverture de test). Tout cycle de test nouveau ou supplémentaire devrait être planifié en utilisant une analyse de ces risques en tant qu'entrée. Il est également fortement recommandé d'avoir une fiche de risque qui est mise à jour à chaque étape du projet.

3.9.3 Gestion des Risques dans le Cycle de Vie

Idéalement, la gestion des risques se produit sur l'ensemble du cycle de vie. Si une organisation a un document de politique de test et/ou une stratégie de test, celles-ci doivent décrire le processus général par lequel les risques projet et produit sont gérés dans le test, et montrer comment la gestion des risques y est intégrée et affecte tous les niveaux.

L'identification des risques et des activités d'analyse de risque peuvent commencer au cours de la phase initiale du projet, quel que soit le modèle du cycle de développement logiciel. Les activités de réduction des risques peuvent être planifiées et mises en place dans le cadre de l'ensemble du processus de planification des tests. Dans plan de test maître et/ou les plans de test de niveau, les risques du projet et du produit peuvent être abordés. Le type et le niveau de risque auront une influence sur les niveaux de test dans

lesquels les risques seront abordés, sur l'ampleur de l'effort à utiliser pour atténuer le risque, sur le test et d'autres techniques à appliquer pour réduire le risque, et sur les critères permettant de juger de l'adéquation de l'atténuation des risques.

Après que la planification est terminée, la gestion des risques, y compris l'identification, l'analyse et la réduction doivent être en cours sur le projet. Ceci comprend l'identification des nouveaux risques, la réévaluation du niveau des risques existants, et l'évaluation de l'efficacité des activités de réduction de risques. Par exemple, si l'identification et l'analyse d'un risque ont eu lieu sur la base des spécifications des exigences au cours de la phase des exigences, alors, une fois la spécification de la conception finalisée, une réévaluation des risques devrait survenir. Pour prendre un autre exemple, si au cours des tests on découvre qu'un composant contient beaucoup plus de défauts que prévus, on peut conclure que la probabilité des défauts dans ce domaine est plus élevée que prévue, et donc revoir la probabilité et le niveau général de risque à la hausse. Il pourrait en résulter une augmentation du nombre des tests à effectuer sur ce composant.

Une fois que l'identification initiale des risques et des activités d'analyse est terminée, et que les activités de réduction de risques sont menées, on peut mesurer à quel point les risques ont été réduits. Cela peut être fait en traçant les cas de test et les défauts découverts à leurs risques associés. Quand les tests sont exécutés et les défauts constatés, les testeurs peuvent examiner le niveau de risque résiduel. Cela confirme que l'utilisation des tests basés sur le risque permet de déterminer le bon moment pour la livraison. Pour un exemple d'un rapport des résultats de tests basés sur la couverture des risques, voir [Black03].

Les rapports de test devraient inclure des informations sur les risques couverts et toujours ouverts, ainsi que les avantages obtenus et ceux qui n'ont pas encore été obtenus.

3.10 Analyse des Modes de Défaillance et Effets

L'analyse des modes de défaillance et effets (AMDE) et une variante ajoutant l'analyse de criticité (AMDEC) sont des activités itératives, destinées à analyser les effets et la criticité des modes de défaillance dans un système. L'application de ces analyses à des logiciels est parfois appelée AMDEL où le L signifie Logiciel. Dans les rubriques suivantes, seul AMDE est utilisée, mais l'information est applicable aux trois autres méthodes aussi.

Les testeurs doivent être en mesure de contribuer à la réalisation du document AMDE. Cela comprend la compréhension du but et l'application de ces documents ainsi que la possibilité d'appliquer leurs connaissances pour aider à déterminer les facteurs de risque.

3.10.1 Domaines d'Application

L'AMDE doit être appliquée :

- lorsque la criticité du logiciel ou du système en cours d'examen doit être analysée, afin de réduire le risque de défaillance (par exemple les systèmes critiques en termes de sécurité, comme les systèmes de contrôle de vol d'un avion)
- lorsque des exigences réglementaires s'appliquent (voir 1.3.2 Systèmes à Sécurité Critique)
- pour éliminer les défauts à un stade précoce
- pour définir des considérations spécifiques au test, des contraintes opérationnelles, la conception des décisions en matière de sécurité des systèmes critiques

3.10.2 Étapes de Mise en Oeuvre

L'AMDE devrait être planifiée dès que des informations préliminaires sont disponibles à un niveau élevé, puis étendue à des niveaux inférieurs une fois qu'il y a plus de détails disponibles. L'AMDE peut être appliquée à tous les niveaux d'un système ou d'un logiciel selon les informations disponibles et les exigences d'un programme.

Pour chaque fonction, module ou composant critique, par itération:

- Sélectionner une fonction et déterminer ses modes de défaillance possibles, c'est-à-dire comment la fonction peut être mise en échec
- Définir les causes possibles de ces échecs, leurs effets et concevoir des mécanismes pour réduire ou atténuer ces échecs

3.10.3 Coûts & Avantages

L'AMDE offre les avantages suivants :

- Des défaillances attendues, qui sont la faute d'une défaillance logicielle ou d'une erreur d'utilisation, peuvent être décelées
- L'utilisation systématique peut contribuer à une analyse globale au niveau système
- Les résultats peuvent être utilisés pour des décisions de conception et/ou des justifications de conception
- Les résultats peuvent être utilisés pour orienter les tests sur une zone spécifique (critique) d'un logiciel

Les considérations suivantes doivent être prises en compte lors de l'application de l'AMDE :

- Les séquences d'échecs sont rarement prises en compte
- La création de tableaux AMDE peut prendre beaucoup de temps
- Les fonctions indépendantes peuvent être difficiles à définir
- La propagation d'une erreur peut être difficile à identifier

3.11 Préoccupations de la Gestion des Tests

3.11.1 Préoccupations de la Gestion des Tests sur les Tests Exploratoires

La gestion des tests basée sur session (SBTM : Session Based Test Management) est un concept dans la gestion des tests exploratoires. Une session est l'unité fondamentale de test, sans interruption, et porte sur un objet de test avec un objectif de test spécifique (la charte de test). À la fin d'une session, un rapport, habituellement appelé une fiche de session, est rédigé sur les activités menées. SBTM fonctionne dans une structure de processus documentée et produit des enregistrements qui complètent la documentation de vérification.

Une session de tests peut être réparti sur trois étapes :

- Étape d'installation : Installation de l'environnement de test et amélioration de la compréhension du produit.
- Conception et exécution de tests : Scanning de l'objet du test et recherche de problèmes
- Enquête et rapports sur défauts : Commence quand le testeur trouve quelque chose qui ressemble à un échec.

La fiche de session SBTM contient les éléments suivants:

- Agrément de tests pour la session
- Nom(s) du (des) testeur(s)
- Date et heure de début
- Ventilation des tâches (sessions)
- Fichiers de données
- Remarques sur les tests
- Problèmes
- Défauts

À la fin de chaque session, le gestionnaire de tests tient une réunion de débriefing avec l'équipe. Au cours de ce débriefing le gestionnaire passe en revue les fiches de session, améliore les chartes de test, reçoit les commentaires des testeurs et estime et planifie de futures sessions.

L'ordre du jour de la session de restitution peut être résumé par l'acronyme PROOF (preuve):

- **P**assé : Que s'est-il passé au cours de la session ?
- **R**ésultats : Qu'est-ce qui a été réalisé pendant la session ?

- **O**bjectifs : Qu'est-ce qu'il reste à faire ?
- **O**bstacles : Qu'est-ce qui a empêché le bon déroulement du test ?
- e**F**fets : Qu'est-ce que le testeur ressent de tout cela, quelles sont ses sensations ?

3.11.2 *Préoccupations Relatives aux Systèmes de Systèmes*

Les enjeux suivants sont associés à la gestion de tests pour les systèmes de systèmes:

- La gestion de tests est plus complexe parce que les tests des différents systèmes qui composent les systèmes de systèmes peuvent être menés à différents endroits, par les différentes organisations et en utilisant différents modèles de cycle de vie. Pour ces raisons, en général le plan de test maître pour les systèmes de systèmes met en œuvre un modèle de cycle de vie, l'accent étant mis sur les enjeux de gestion tels que les jalons et la qualité. Il y a souvent un processus officiel d'assurance qualité qui peut être défini dans un plan qualité séparé.
- Le support des processus tels que la gestion de la configuration, la gestion du changement et la gestion de la mise en production doit être défini formellement et un accord convenu pour les interfaces avec la gestion de tests. Ces processus sont essentiels pour assurer que les livraisons du logiciel sont contrôlées, que les changements sont bien gérés et que les lignes de base du logiciel testé sont définies.
- La mise en place et la gestion des environnements de test représentatifs, y compris les données de test, peut être l'un des principaux défis techniques et organisationnels.
- La stratégie de tests d'intégration peut exiger la mise en place de simulateurs. Bien que cela pourrait être relativement simple et peu coûteux pour les niveaux de test d'intégration, la mise en place de simulateurs pour l'ensemble des systèmes peut être complexe et coûteuse aux niveaux de test les plus élevés pour les systèmes de systèmes. La planification, l'estimation et le développement de simulateurs sont fréquemment gérés en tant que projets distincts.
- Les dépendances entre les différentes parties lors du test des systèmes de systèmes génèrent des contraintes supplémentaires sur le système et pour les tests d'acceptation. Il y a également besoin de mettre l'accent sur le test d'intégration des systèmes et la documentation de test qui s'y rapporte, par exemple des spécifications d'interface.

3.11.3 *Préoccupations Relatives aux Systèmes Critiques*

Les enjeux suivants sont associés à la gestion de tests de systèmes critiques:

- En général des normes spécifiques à un métier (domaine) s'appliquent (par exemple le secteur des transports, les industries médicales et militaires). Ces normes peuvent s'appliquer au processus de développement et à la structure organisationnelle, ou au produit en cours d'élaboration. Référez-vous au chapitre 6 pour plus de détails.
- En raison de la responsabilité juridique liée aux systèmes de sécurité critiques, les aspects formels, tels que la traçabilité des exigences, les niveaux de couverture de test à atteindre, les critères d'acceptation à atteindre, et la documentation de test exigé, peuvent s'appliquer afin de démontrer la conformité.
- Pour démontrer la conformité de la structure organisationnelle et du processus de développement, des audits et des organigrammes peuvent suffire.
- En fonction de la norme applicable, un cycle de développement prédéfini est suivi. Ces cycles de vie sont généralement de nature séquentielle.
- Si un système a été classé comme « critique » par une organisation, les attributs non-fonctionnels suivants doivent être adressés par la stratégie de test et/ou le plan de tests :
  - La fiabilité (**R**eliability)
  - La disponibilité (**A**vailability)
  - La maintenabilité (**M**aintainability)
  - La sécurité et la sûreté (**S**afety and security)
- En raison de ces caractéristiques, ces systèmes sont parfois appelés des systèmes « RAMS » en anglais.

3.11.4 *Préoccupations Relatives aux Tests Non Fonctionnels*

L'absence de tests non-fonctionnels peut mettre en péril la réussite d'un logiciel de manière considérable. De nombreux types de tests non-fonctionnels sont, toutefois, associés à des coûts élevés, qui doivent être équilibrés avec les risques.

Il existe de nombreux types de tests non-fonctionnels, dont tous ne sont pas forcément adaptés à une application donnée.

Les facteurs suivants peuvent influer la planification et l'exécution des tests non-fonctionnels :

- Exigences des acteurs
- Outils nécessaires
- Matériel nécessaire
- Facteurs organisationnels
- Communication
- Sécurité des données

3.11.4.1 Exigences des Acteurs

Les exigences non-fonctionnelles sont souvent mal spécifiées ou même inexistantes. Au stade de la planification, les testeurs doivent être en mesure d'obtenir des niveaux d'attente des acteurs et d'évaluer les risques représentés par leurs attentes.

Il est conseillé d'obtenir de multiples points de vue lors du recueil des exigences. Les exigences doivent provenir d'acteurs tels que les clients, les utilisateurs, et le personnel opérationnel et de maintenance, sinon un certain nombre d'exigences sont susceptibles d'être oubliées.

Les éléments essentiels suivants doivent être pris en compte pour améliorer l'aptitude au test des exigences non-fonctionnelles :

- Les exigences sont lues très souvent. Un investissement dans la rédaction d'exigences testables est presque toujours rentable. Utiliser un langage simple, cohérent et concis (c'est-à-dire utiliser le langage défini dans le glossaire projet). En particulier, prendre soin en utilisant les verbes tels que « il doit » (c'est-à-dire qu'il est obligatoire) et « il devrait » (c'est-à-dire qu'il est souhaitable).
- Les lecteurs des exigences viennent d'horizons divers.
- Les exigences doivent être rédigées de façon claire et concise pour éviter des interprétations multiples. Un format standard pour chaque exigence devrait être utilisé.
- Spécifier les exigences quantitativement dans la mesure du possible. Choisir le meilleur métrique pour exprimer un attribut (par exemple mesuré de la performance en millisecondes) et spécifier une plage au sein de laquelle les résultats peuvent être évalués et acceptés ou rejetés. Pour certains attributs non-fonctionnels (p.ex. utilisabilité) ce n'est pas toujours facile.

3.11.4.2 Outils Nécessaires

Des outils commerciaux ou des simulateurs sont particulièrement pertinents pour la performance, le rendement et certains tests de sécurité. La planification des tests devrait inclure une estimation des coûts et des délais pour l'outillage. Lorsque des outils de spécialiste doivent être utilisés, la planification doit tenir compte des montées en compétences pour les nouveaux outils, ou le coût de l'embauche de spécialistes d'outils externes.

Le développement d'un simulateur complexe peut représenter un projet de développement à part et devrait être planifié en tant que tel. Notamment, la planification des simulateurs à utiliser dans des applications de sécurité critiques devrait prendre en compte les tests d'acceptation et, éventuellement, la certification du simulateur par un organisme indépendant.

3.11.4.3 Matériel Nécessaire

Beaucoup de tests non-fonctionnels exigent un environnement de test semblable à celui de production afin de fournir des mesures réalistes. Selon la taille et la complexité du système à tester, cela peut avoir un impact significatif sur la planification et le financement des tests. Le coût de l'exécution des tests non-fonctionnels peut être tellement élevé que seul un temps limité est disponible pour l'exécution des tests.

Par exemple, la vérification des exigences d'évolutivité pour un site Internet avec un très grand trafic peut exiger la simulation de centaines de milliers d'utilisateurs virtuels. Cela peut avoir une influence significative sur les coûts de matériel et d'outillage. Comme ces coûts sont généralement réduits au minimum par la location (par exemple: "top-up") de licences pour les outils de performance, le temps disponible pour ces tests est limité.

Les tests d'utilisabilité peuvent nécessiter la mise en place de laboratoires dédiés ou la conduite de questionnaires généralisés. En règle générale ces tests sont effectués seulement dans un cycle de développement.

Plusieurs autres types de tests non-fonctionnels (par exemple les tests de sécurité ou de performance) exigent un environnement d'exécution semblable à celui de production. Étant donné que le coût de tels environnements peut être élevé, l'utilisation de l'environnement de production lui-même peut être la seule possibilité envisageable. Le calendrier de ces exécutions de tests doit être planifié avec soin et il est très probable que ces tests ne puissent être exécutés qu'à des moments précis (par exemple pendant la nuit).

Des ordinateurs et de la bande passante devraient être anticipés pour l'exécution de tests d'efficacité (par exemple les tests de performance et de charge). Les besoins dépendent essentiellement du nombre d'utilisateurs virtuels à simuler et du trafic réseau susceptible d'être généré. Ne pas tenir compte de ceci peut donner des mesures de performance non représentatives.

3.11.4.4 Facteurs Organisationnels

Les tests non-fonctionnels peuvent impliquer la mesure du comportement de plusieurs composants dans un système complet (serveurs, bases de données, réseaux, etc.). Si ces éléments sont répartis sur un certain nombre de sites et d'organismes, l'effort nécessaire pour planifier et coordonner les tests peut être important. Par exemple, certains composants logiciels peuvent être disponibles pour les tests système uniquement à certains moments de la journée ou de l'année ; ou bien des organismes pourraient fournir du support seulement pour un nombre de jours limité. L'impossibilité de confirmer la disponibilité « à la demande » des composants système, et du personnel des autres organismes, peut entraîner de graves perturbations sur les tests planifiés.

3.11.4.5 Communication

La capacité à définir et à exécuter certains types de tests non-fonctionnels (en particulier les tests de rendement) peut dépendre de la capacité de modifier des protocoles de communication à des fins de test. Au stade de la planification il convient de s'assurer que cela est possible (par exemple, qu'il existe des outils avec la compatibilité nécessaire).

3.11.4.6 Sécurité des Données sécurité des données

Des mesures de sécurité spécifiques mises en œuvre pour un système devraient être prises en compte lors de la phase de planification des tests pour assurer que toutes les activités de test sont possibles. Par exemple, l'encryption des données peut rendre difficile la création de données de test et la vérification des résultats.

Des politiques et lois de protection des données peuvent faire obstacle à la création d'utilisateurs virtuels sur la base de données de production. Le fait de créer des données de test anonymes peut exiger un travail conséquent qui doit être planifié dans le cadre de la mise en œuvre des tests.

4. Techniques de conception des tests

Termes

BS 9725-2, analyse des valeurs limites, tests des branches, graphe de cause à effet, méthode de classification arborescente, test de conditions, tests de détermination des conditions, analyse des flux de contrôle, D-D path, analyse du flux de données, test par tables de décisions, tests des décisions , techniques basées sur les défauts, classification des défauts, analyse dynamique, estimation d'erreur, partition d'équivalence, tests exploratoires, techniques basées sur l'expérience, PLCS, fuite mémoire, tests des conditions multiples, test en binôme, tests des chemins, test basés sur les exigences, attaques logicielles, techniques basées sur les spécifications, analyse statique, tests des instructions, test de transition d'état, technique basée sur la structure, agrément de tests, tests des cas d'utilisation, pointeur non défini

4.1 Introduction

Les techniques de conception de test abordées dans ce chapitre traitent des domaines suivants:

- Les tests basés sur les spécifications (tests de comportement ou tests boite noire)
- Les tests basés sur la structure (ou boite blanche)
- Les tests basés sur les défauts
- Les tests basés sur l'expérience

Ces techniques sont complémentaires et doivent être utilisées à bon escient suivant l'activité de test donnée, indépendamment du niveau de test atteint. Bien que chacune de ces techniques puisse être utilisée aussi bien par des analystes de test que des techniciens de test, pour l'objectif de ce syllabus, le découpage suivant a été choisi suivant l'utilisation la plus commune:

- Les tests basés sur les spécifications → Analyste de test et technicien de test
- Les tests basés sur la structure (ou boite blanche) → Technicien de test
- Les tests basés sur les défauts → Analyste de test et technicien de test
- Les tests basés sur l'expérience → Analyste de test et technicien de test

En complément à ces parties, ce chapitre appréhende d'autres techniques, comme les attaques, l'analyse statique et l'analyse dynamique. Ces techniques sont généralement mises en œuvre par des techniciens de test.

A noter que les techniques basées sur les spécifications peuvent aussi bien être fonctionnelles que non-fonctionnelles. Les techniques non-fonctionnelles sont abordées dans le chapitre suivant.

4.2 Techniques Basées sur les Spécifications

Les techniques de test basées sur les spécifications sont un moyen d'obtenir et de choisir des conditions de test ou des cas de test basés sur une analyse de la documentation de test d'un composant ou d'un système sans référence à sa structure interne.

Ci-dessous, les principales caractéristiques des techniques basées sur les spécifications :

- Les modèles, aussi bien formels qu'informels, sont utilisés pour la spécification du problème à résoudre
- Les cas de test peuvent être dérivés systématiquement de ces modèles

Des techniques fournissent également le critère de couverture, qui peut être utilisé pour mesurer la tâche de conception de test. Satisfaire complètement au critère de couverture ne signifie pas que tous les tests possibles ont été exécutés mais plutôt que cette technique ne permet pas d'obtenir de nouveaux tests utiles.

Les tests basés sur les spécifications sont souvent des tests basés sur les exigences. Puisque les spécifications des exigences doivent préciser comment le système se comporte, en particulier dans le domaine fonctionnel, dériver les tests des exigences fait souvent partie des tests de comportement du

système. Les techniques évoquées précédemment dans cette sous-section peuvent être directement appliquées aux spécifications des exigences, en créant un modèle pour le comportement du système directement à partir des spécifications graphiques ou textuelles des exigences pour ensuite obtenir les tests comme défini précédemment.

Les techniques de conception de tests basées sur les spécifications suivantes sont abordées dans le présent syllabus:

Nom	Technique	Critère de couverture
Partition d'équivalence	Voir section 4.3.1 du Syllabus niveau fondation ISTQB®.	Nombre de partitions couvertes / Nombre total de partitions
Analyse des valeurs limites	Voir section 4.3.2 du Syllabus niveau fondation ISTQB®. A noter que cette analyse peut être appliquée en utilisant aussi bien 2 ou 3 valeurs. La décision de ce choix est la plupart du temps basée sur les risques.	Nombre de valeurs limites couvertes / Nombre total de valeurs limites
Test par tables de décisions et graphe de cause à effet	Voir section 4.3.3 du Syllabus niveau fondation ISTQB® pour une description des tests par tables de décision. Avec les tests par table de décision, toutes les combinaisons de conditions, relations ou contraintes sont testées. En complément aux tables de décision, une technique graphique utilisant une notation logique, appelée graphe de cause à effet, peut être utilisée. A noter que, en complément du test de toutes les combinaisons de conditions possibles, l'utilisation de tables d'erreurs est possible. Le choix d'utiliser des tables de décision complètes ou des tables d'erreur sera en général basé sur le risque. [Copeland03]	Nombre de combinaisons de conditions couvertes / Nombre maximum de combinaisons de conditions
Test de transition d'état	Voir section 4.3.4 du Syllabus niveau fondation ISTQB®.	Pour des transitions simples, le métrique de couverture est le pourcentage de toutes les transitions valides exécutées pendant les tests. Appelé couverture d'aiguillage-0. Pour n transitions, la mesure de couverture est le pourcentage de toutes les séquences valides de n transitions exécutées pendant un test. Celui-ci est appelé la couverture d'aiguillage n-1
Méthode de classification arborescente, tableaux orthogonaux et tables de toutes les paires	Les facteurs ou les variables intéressantes et les options ou valeurs pouvant être prises sont identifiés, Ensuite, les singletons, paires, triplets ou combinaisons supérieures de ces options ou de ces valeurs sont identifiés. [Copeland03] La méthode de classification arborescente utilise une notation graphique pour montrer les conditions de test (classes) et les combinaisons adressées par les cas de test. [Grochtmann94]	Dépend de la technique appliquée. Par exemple la classification par paires est distincte de la classification arborescente.

Nom	Technique	Critère de couverture
Test des cas d'utilisation	Voir section 4.3.5 du Syllabus niveau fondation ISTQB®.	Aucun critère formel n'est valable

Les techniques sont parfois combinées pour créer des tests. Par exemple, les conditions identifiées dans une table de décision peuvent aussi faire l'objet d'une partition d'équivalence pour découvrir les différents chemins satisfaisant une condition. Les tests ne couvriront pas simplement les combinaisons de conditions car en plus, pour les conditions qui ont été partitionnées, d'autres tests pourront être générés pour couvrir les partitions d'équivalence.

4.3 Techniques Basées sur la Structure ou Boite Blanche

La technique de test basée sur la structure, aussi connue sous les noms tests boite blanche ou tests basés sur le code est celle dans laquelle le code, les données, l'architecture ou le flot d'exécution du système sont utilisés comme base de conception des tests, avec des tests systématiquement dérivés de la structure. Cette technique détermine les éléments de la structure qui doivent être pris en compte. Elle donne aussi le critère de couverture qui détermine à quel moment la création des tests doit s'arrêter.

Ces critères ne garantissent en rien l'exhaustivité des tests, mais bien qu'aucun test supplémentaire n'est utile ou nécessaire pour tester la structure analysée.

Chaque critère doit être mesuré et associé à un objectif défini et propre à chaque projet ou entreprise.

Les principales caractéristiques des techniques basées sur la structure :

- Des informations sur la manière dont le logiciel est architecturé et sur la manière de générer des cas de test, par exemple le code et la conception
- Le degré de couverture d'un logiciel peut être mesuré à partir de cas de test existants. Mais des cas de test supplémentaires pourront être générés automatiquement et augmenteront la couverture.

Dans le syllabus niveau avancé, on aborde, comme technique de conception de tests basés sur la structure :

Nom	Technique	Critère de Couverture
Tests des instructions	Identification des instructions exécutables (hors commentaires, hors espaces)	Nb. d'instructions exécutées / Nb. total d'instructions
Tests des décisions	Identification des instructions de décision telles que : if/else, switch/case, for, et while	Nb. de résultats de décisions exécutés / Nb. total de résultats de décisions
Tests des branches	Identification des branches telles que if/else, switch/case, for, et while. A noter que les tests des décisions et les tests des branches sont les mêmes pour une couverture de 100%, mais peuvent être différents pour une couverture moindre.	Nb. de branches exécutées / Nb. total de branches
Tests des conditions	Identification des conditions true/false (et instruction « case »)	Nb. d'opérandes booléennes exécutées / Nb. total d'opérandes booléennes exécutables
Tests des conditions multiples	Identification de toutes les combinaisons possibles de conditions true/false.	Nb. de combinaisons d'opérandes booléennes exécutées / Nb. total de combinaisons d'opérandes booléennes exécutables
Test de détermination des conditions	Identification des possibles combinaisons de conditions true/false qui peuvent impacter les décisions (branches)	Nb. d'opérandes booléennes identifiées comme impactant

Nom Technique	**Critère de Couverture**
	les décisions / Nb. total d'opérandes booléennes
PLCS (test des boucles)	
Identification des différentes conditions contrôlant l'itération d'une boucle. Une Portion Linéaire de Code et Saut (PLCS) est utilisée pour tester une section spécifique du code (Une séquence linéaire de code exécutable) qui commence au début d'un contrôle de flux ou d'un saut jusqu'à la fin du programme. Ces fragments de code sont aussi appelés chemins-DD (chemin de décision à décision). Cette technique est utilisée pour définir des cas de test spécifiques et les données de test associées nécessaires à l'exécution du flot de contrôle voulu. Concevoir ces tests nécessite un accès à un modèle du code source qui définit le flux des sauts. PLCS peut être utilisée comme base pour la mesure de la couverture de code.	Nb. de PLCSs exécutés / Nb. total de PLCSs
Test des chemins	
Identification des séquences uniques d'instructions (chemins)	Nb. de chemins exécutés / Nb. total de chemins

Une utilisation classique du critère de couverture structurel est de mesurer l'exhaustivité ou la non-exhaustivité d'un ensemble de tests basés conçus à partir des spécifications, et/ou à partir des techniques basées sur l'expérience. Les outils de tests, appelés outils de couverture de code, sont utilisés pour analyser la couverture structurelle atteinte par ces tests. Si des déficiences importantes sont observées dans la couverture de la structure (ou que la préconisation de la couverture par l'application de standards n'est pas atteinte), alors des tests manipulant la structure ou d'autres techniques seront ajoutés pour remédier à ces manques.

Analyse de la Couverture

Des tests dynamiques basés sur la structure ou d'autres techniques peuvent être menés pour déterminer si la couverture de code souhaitée est atteinte ou non. Ceci est appliqué pour apporter des garanties dans les systèmes critiques où un taux de couverture doit être atteint (voir aussi la section 1.3.2 Systèmes à Sécurité Critique). Les résultats peuvent indiquer que certaines portions d'un code ne sont pas exécutées, et dans ce cas, amènent à définir d'autres cas de test pour exécuter ces portions.

4.4 Techniques Basées sur les Défauts et sur l'Expérience

4.4.1 Techniques Basées sur les Défauts

Dans la technique de conception des tests basée sur les défauts, le type de défaut recherché est utilisé comme base pour la conception des tests, et les tests sont systématiquement dérivés de notre connaissance du défaut.

Cette technique fournit aussi des critères de couverture qui sont utilisés pour déterminer quand la production de tests peut s'arrêter. De façon pratique, le critère de couverture pour les techniques basées sur les défauts tend à être moins systématique que pour les techniques basées sur la structure ou le comportement, dans la mesure où les règles générales de couverture sont données et la définition exacte du niveau de couverture utile est discrétionnaire pendant la phase de conception des tests ou d'exécution. Le critère de couverture ne garantit en rien l'exhaustivité des tests, mais bien qu'aucun test supplémentaire n'est utile ou nécessaire pour les défauts considérés.

Dans le syllabus niveau avancé, on aborde une technique de conception basée sur les défauts:

Nom	Technique	Critère de Couverture
Taxonomies (catégories et listes des défauts potentiels)	L'utilisateur qui utilise des échantillons de taxonomie d'une liste, choisit un problème potentiel comme base d'analyse. Les taxonomies peuvent lister les causes originelles, les défauts et les défaillances. Les taxonomies de défauts listent les principaux défauts d'une application testée. Cette liste est utilisée pour concevoir les cas de test	Les données de défauts appropriées et les types de défauts.

4.4.2 Techniques Basées sur l'Expérience

Il existe d'autres techniques de conception, qui prennent en compte l'historique du défaut mais qui n'ont pas systématiquement de critères de couverture. Celles-ci sont classées comme des techniques basées sur l'expérience.

Les techniques basées sur l'expérience font appel aux compétences, à l'intuition des testeurs, ainsi qu'à l'expérience acquise sur des applications ou technologies similaires. Ces tests sont efficaces pour trouver des défauts mais pas aussi appropriées que d'autres techniques pour atteindre des niveaux de couverture de tests spécifiques ou pour produire des procédures de test réutilisables.

Lorsque des approches dynamiques ou heuristiques sont utilisées, les testeurs s'orientent davantage vers des tests basés sur l'expérience, où l'activité de test est plus réactive aux évènements que les approches pré-planifiées. En plus, l'exécution et l'évaluation sont des tâches concurrentes. Les approches de tests basées sur l'expérience ne sont pas entièrement dynamiques ; c'est-à-dire que les tests ne sont pas complètement crées alors que l'exécution des tests a déjà commencé.

Noter que bien que des idées sur la couverture soient présentées dans le tableau suivant, les techniques basées sur l'expérience n'ont pas de critère formel de couverture.

Dans ce syllabus on présente les techniques suivantes :

Nom	Technique	Critère de Couverture
Estimation d'erreur	Le testeur utilise son expérience pour deviner les erreurs potentielles et détermine les méthodes pour découvrir les défauts. L'estimation d'erreur est aussi utile pendant une analyse de risques pour identifier les potentiels modes de défaillances. [Myers97]	Lors de l'utilisation d'une taxonomie, les critères sont les données sur les défauts et les types de défauts. Sans une taxonomie, la couverture est limitée par l'expérience du testeur et le temps disponible.

Nom	Technique	Critère de Couverture
Tests basées sur les listes de vérifications	Le testeur expérimenté utilise une liste haut-niveau d'éléments à noter, à vérifier, ou à se rappeler ou encore un ensemble de règles ou critères sur lesquels un produit doit être vérifié. Ces listes de vérification sont créées à partir d'un ensemble de standards, de l'expérience, ou bien d'autres considérations. Une liste de vérification de standards IHM employée comme base de test pour une application est un exemple de test basée sur les listes de vérification.	Chaque élément de la liste de vérification a été couvert.
Les tests exploratoires	Le testeur apprend simultanément aussi bien l'application que ses défauts, planifie l'effort de test à faire, conçoit et exécute les tests, et remonte les résultats. De bons tests exploratoires sont planifiés, interactifs et créatifs. Le testeur ajuste dynamiquement les objectifs du test pendant l'exécution et prépare seulement une documentation succincte. [Veenendaal02]	Des chartes peuvent être créées pour définir les tâches, les objectifs et les livrables. Une session de tests exploratoires est prévue qui définit ce qui doit être effectué, sur quoi se concentrer, ce qui est dans et hors du périmètre, quelles ressources devraient être allouées. De plus, un ensemble d'heuristiques sur les défauts et la qualité peuvent être pris en compte.
Les attaques	Le testeur établit une évaluation dirigée et ciblée de la qualité du système en tentant de forcer des défaillances spécifiques à apparaître. Le principe des attaques de logiciel, comme décrit dans [Whittaker03], est basé sur les interactions du logiciel testé avec son environnement. Cela inclut l'interface utilisateur, le système d'exploitation avec le noyau, les APIs et le système de fichiers. Ces interactions sont basées sur des échanges précis de données. Toutes déviations d'une (ou plusieurs) interaction(s) peut être la cause d'une défaillance	Les différentes interfaces de l'application testée. Les principales interfaces sont les IHM, le système d'exploitation, le noyau, les APIs et le système de fichiers.

Les techniques basées sur les défauts et celles basées sur l'expérience appliquent les connaissances sur les défauts et les autres expériences pour augmenter la détection des défauts. Elles partent de « tests rapides » dans lesquels le testeur n'a pas d'activités de planification formelles à effectuer, puis des sessions pré-planifiées et enfin des sessions scriptées. Elles sont presque toujours utiles mais sont particulièrement intéressantes dans les circonstances suivantes :

- Aucune spécification n'est disponible
- La documentation du système sous test est insuffisante
- Le temps alloué à la conception et à la création des procédures de test est insuffisant
- Les testeurs sont expérimentés dans le domaine et/ou la technologie
- Recherche de diversité par rapport aux tests basés sur les spécifications et sur la structure
- Lors de l'analyse des défaillances opérationnelles

Les techniques basées sur l'expérience et les défauts sont aussi très utiles lorsqu'elles sont combinées avec des techniques basées sur le comportement et des techniques basées sur la structure car elles comblent les manques de couverture de tests qui résultent des faiblesses systématiques de ces techniques.

4.5 Analyse Statique

L'analyse statique vise à tester sans exécuter le logiciel sous test et peut s'appliquer au code ou à l'architecture du système.

4.5.1 Analyse Statique de Code

L'analyse statique est n'importe quel type de test ou d'examen qui peut être réalisé sans exécuter le code. Il existe de nombreuses techniques d'analyses statiques qui sont abordées dans cette section.

4.5.1.1 Analyse du Flot de Contrôle

L'analyse de flot de contrôle fournit des informations sur les points de décision logiques du système logiciel et la complexité de leur structure. L'analyse de flot de contrôle est décrite dans le syllabus niveau fondation ISTQB® et dans [Beizer95]

4.5.1.2 Analyse du Flot de Données

L'analyse de flot de données est une technique de test structuré qui teste les chemins qui vont de la définition d'une variable jusqu'aux endroits ou elle est utilisée. Ces chemins sont appelés définition-utilisation (DU-pairs) ou affectation-utilisation (Set-Use pairs). Dans cette méthode, les tests sont générés pour atteindre 100% de couverture (si possible) pour chaque ensemble de chemins.

Cette technique bien qu'appelée analyse de flot de données, prend aussi en compte le flot de contrôle de l'application testée car elle poursuit la série d'affectation/utilisation de chaque variable et peut avoir à traverser le flot de contrôle du logiciel. Voir aussi [Beizer95]

4.5.1.3 Conformité aux Règles de Codage

Pendant la phase d'analyse statique, la conformité aux standards de codage peut également être évaluée. Les standards de codage couvrent aussi bien les aspects architecturaux que l'utilisation (ou l'interdiction) de certaines structures de programmation.

La conformité aux standards de codage permet à l'application d'être plus maintenable et testable. Des exigences spécifiques à un langage peuvent également être vérifiées en utilisant des tests statiques.

4.5.1.4 Génération de Métriques sur le Code

Des métriques de code peuvent être générées durant l'analyse statique, ce qui contribue à un meilleur niveau de maintenabilité et de fiabilité du code.

- Complexité cyclomatique
- Taille
- Fréquence des commentaires
- Nombre de niveaux imbriqués
- Nombre d'appels de fonctions

4.5.2 Analyse Statique de l'Architecture

4.5.2.1 Analyse Statique de Site Web

L'architecture d'un site web peut aussi être évaluée en utilisant des outils d'analyse statique. L'objectif est ici de vérifier si l'arborescence du site est bien répartie ou s'il y a un déséquilibre qui entraîne :

- Des tâches de test plus difficiles
- Une augmentation de la charge de maintenance
- Des difficultés de navigation pour l'utilisateur

Des outils de test spécifiques incluant un moteur d'exploration web peuvent aussi fournir, via l'analyse statique, des informations sur la taille des pages, sur le temps de téléchargement nécessaire, sur la présence ou non d'une page (http error 404). Cela fournit des informations utiles aux développeurs, aux webmestres et aux testeurs.

4.5.2.2 Graphes d'Appels

Les graphes d'appels peuvent être utilisés pour plusieurs objectifs :

- Concevoir des tests qui appellent un module spécifique
- Établir le nombre d'emplacements au sein de l'application à partir desquels le module est appelé
- Fournir une suggestion sur l'ordre d'intégration (intégration de proche en proche et par paires [Jorgensen02])
- Évaluer la structure de l'ensemble du code ainsi que son architecture

Les informations sur les modules appelant peuvent aussi être obtenues pendant une analyse dynamique. L'information obtenue indique le nombre de fois où un module est appelé pendant une exécution. En fusionnant les informations, obtenues à partir des graphes d'appels pendant une analyse statique, avec les informations, obtenues pendant une analyse dynamique, le testeur peut se concentrer sur les modules qui sont appelés souvent et/ou de manière répétée. Si de tels modules sont complexes (Voir 1.3.2.2 Systèmes Critiques et Complexité), ils sont les candidats privilégiés pour des tests détaillés et intensifs.

4.6 Analyse Dynamique

Le principe de l'analyse dynamique est d'analyser une application pendant son exécution. Cela requiert souvent une instrumentation particulière, insérée dans le code aussi bien manuellement qu'automatiquement.

4.6.1 Présentation

Les défauts qui ne sont pas immédiatement reproductibles, peuvent avoir des conséquences significatives sur l'effort de test, sur la capacité de livrer une application, ou de l'utiliser en production. De tels défauts peuvent être la cause de fuites mémoire, d'une utilisation incorrecte des pointeurs, ou d'autres corruptions (exemple de la pile système) [Kaner02]. En raison de la nature de ces défauts qui peuvent aggraver progressivement les performances du système ou même planter le système, les stratégies de test doivent prendre en compte les risques associés à de tels défauts et, le cas échéant, exécuter des analyses dynamiques pour les réduire (typiquement en utilisant des outils).

L'analyse dynamique est réalisée pendant que l'application est en cours d'exécution. Elle est appliquée pour :

- Empêcher les défaillances d'apparaître en détectant les pointeurs non définis et les pertes mémoire
- Analyser les défaillances systèmes qui ne peuvent pas facilement être reproduites
- Évaluer le comportement du réseau
- Améliorer les performances du système en fournissant les informations sur les temps d'exécution du système.

L'analyse dynamique peut être réalisée à chaque niveau de test mais est particulièrement utile lors des tests de composants ou d'intégration. Des compétences techniques ainsi que sur le système sont requises pour spécifier les objectifs de test de l'analyse dynamique et, en particulier, pour analyser les résultats.

4.6.2 Détection des Fuites Mémoire

Une fuite mémoire apparait lorsque la mémoire (RAM) disponible pour un programme est allouée par ce programme mais, suite à des erreurs de programmation, n'est pas libérée. Le programme perd alors la capacité d'accéder à cette partie de la mémoire et peut parfois manquer de mémoire utilisable.

Les fuites mémoires posent des problèmes qui apparaissent avec le temps et qui ne sont pas immédiatement évidents. Cela peut être le cas si, par exemple, le logiciel a été récemment installé ou le système redémarré, ce qui arrive souvent durant le test.

Pour ces raisons, les effets négatifs de fuites mémoires ne sont souvent détectés que lorsque le programme est en production.

Les signes avant-coureurs d'une fuite mémoire sont des aggravations progressives du temps de réponse qui amènent au final à une défaillance système. Même si de telles défaillances peuvent être résolues par des redémarrages du système, cela n'est pas toujours pratique ou simplement possible.

Des outils identifient les zones du code où les fuites mémoire apparaissent afin d'être corrigées. De simples moniteurs de mémoire peuvent aussi être utilisés pour obtenir une impression globale sur la diminution progressive de mémoire disponible, bien qu'un suivi d'analyse soit toujours nécessaire si le cas se présente.

Il existe d'autres types de fuites qui peuvent être prises en compte comme par exemple les gestionnaires de fichiers, les sémaphores, et les pools de connexion aux ressources.

4.6.3 Détection des Pointeurs Sauvages (pointeurs non définis)

Les pointeurs « sauvages » au sein de d'un programme sont des pointeurs qui dans certains cas sont inutilisables. Par exemple, ils peuvent avoir « perdu » l'objet ou la fonction sur lesquels ils devaient pointer, ou ils ne pointent pas sur la zone mémoire attendue (par exemple en dehors des limites allouées d'un tableau). Quand un programme utilise des pointeurs non définis, des conséquences diverses peuvent advenir:

1. Le programme s'exécute comme attendu. Cela peut être le cas quand un pointeur non défini accède à de la mémoire qui n'est pas actuellement utilisée par le programme et est considérée comme « libre »
2. Le programme peut s'arrêter brutalement. Dans ce cas, le pointeur non défini peut avoir pris une partie de la mémoire à utiliser qui est critique au fonctionnement du programme (par exemple. le système d'exploitation).
3. Le programme ne fonctionne pas correctement parce que les objets requis par le programme ne sont pas accessibles. Dans ces conditions le programme peut continuer à fonctionner, bien qu'un message d'erreur puisse être affiché.
4. Les données peuvent être corrompues par le pointeur et des valeurs incorrectes sont alors utilisées

A noter que tout changement effectué dans l'utilisation de la mémoire du programme (par exemple. une nouvelle livraison suite à un changement dans l'application) peuvent déclencher une des quatre conséquences décrites plus haut. Cela est particulièrement critique lorsque le programme s'exécute initialement comme attendu malgré l'utilisation de pointeurs non définis, et que soudainement il s'arrête d'une manière inexpliquée (peut être même en production) suite à une modification dans l'application.

Il est important de signaler que de tels défaillances sont les symptômes d'une erreur cachée (par exemple. le pointeur non défini) mais non pas l'erreur elle-même. (Se référer à [Kaner02], « Lesson 74 »).

Les outils doivent identifier les pointeurs non définis comme ils sont utilisés par le programme, sans tenir compte de leurs conséquences sur l'exécution du programme.

4.6.4 Analyse des Performances

L'analyse dynamique peut être menée pour analyser les performances d'un programme. Les outils aident à identifier les goulets d'étranglement de performances et fournissent un large panel de métriques de performance, qui peuvent être utilisées par le développeur pour « régler » le système. Ceci est aussi appelé « profilage des performances ».

5. Tester les Caractéristiques du logiciel

Termes

Tests d'accessibilité, tests d'exactitude, tests de rendement, évaluation heuristique, tests d'interopérabilité, tests de maintenabilité, tests d'acceptation opérationnelle, profil opérationnel, tests de portabilité, tests de récupérabilité, modèle de croissance de fiabilité, tests de fiabilité, tests de sécurité, tests de pertinence, SUMI, tests d'utilisabilité, tests de scalabilité.

5.1 Introduction

Alors que le chapitre précédent décrivait des techniques spécifiquement utilisables par les testeurs, ce chapitre considère l'application de ces techniques en évaluant les critères principaux utilisés pour décrire la qualité d'applications ou de systèmes logiciels.

Dans ce syllabus les critères qualité qui peuvent être évalués par un analyste de test ou un analyste technique de test sont présentés dans des sections différentes. La description des critères qualité fournis dans ISO 9126 est utilisée comme un guide de description de ces caractéristiques.

La compréhension des divers critères qualité est l'objectif d'apprentissage de base de ces trois modules.

La compréhension de chaque critère qualité est approfondie, dans le module « analyse de test technique», ou dans le module « analyse de test », de façon à ce que les risques puissent être identifiés, que les stratégies de test appropriées soient développées et les cas de test spécifiés.

5.2 Caractéristiques Qualité pour les Tests par Domaine

Le test fonctionnel concerne ce « Que » le produit est censé faire. Les bases du test pour le test fonctionnel sont généralement les exigences ou les documents de spécifications, un domaine d'expertise spécifique ou encore la compréhension du besoin d'origine.

Les tests fonctionnels évoluent en fonction du niveau de test ou de la phase dans laquelle ils sont conduits. Par exemple, un test fonctionnel exécuté pendant les tests d'intégration va permettre de tester le fonctionnement des interfaces de modules qui exécutent une fonction définie simple.

Au niveau des tests systèmes, les tests fonctionnels vérifient le fonctionnement de l'application dans sa globalité. Pour les systèmes de systèmes, le test fonctionnel va tout d'abord se focaliser sur les tests de bout en bout au travers de systèmes intégrés.

Une large variété de techniques de test est employée pendant les tests fonctionnels (voir section 4). Le test fonctionnel peut être réalisé par un testeur dédié, un expert du domaine, ou un développeur (habituellement pendant les tests unitaires).

Les critères ou attributs qualité suivants sont pris en compte :

- Exactitude
- Pertinence
- Interopérabilité
- Sécurité fonctionnelle
- Utilisabilité
- Accessibilité

5.2.1 Tests d'Exactitude

Les tests d'exactitude impliquent de vérifier la concordance avec les exigences ou la spécification des besoins et peuvent inclure l'exactitude des calculs. Les tests d'exactitude utilisent une grande partie des techniques de test expliquées dans le chapitre 4.

5.2.2 Tests de Pertinence

Les tests de pertinence sont chargés d'évaluer et de valider l'adéquation d'un ensemble de fonctions pour des tâches spécifiées. Ce type de test peut être basé sur des cas d'utilisation ou des procédures.

5.2.3 Tests d'Interopérabilité

Les tests d'interopérabilité testent si une application donnée est peut fonctionner correctement dans tous les environnements cibles souhaités. (Matériel, logiciel, intergiciels, EAI, systèmes d'exploitation, etc.)

La spécification des tests d'interopérabilité exige que les combinaisons d'environnements souhaités soient identifiées, configurées et disponibles pour l'équipe de test.

Ces environnements sont alors testés en utilisant une sélection de cas de test fonctionnels qui testent les divers composants présents dans l'environnement.

L'interopérabilité aborde la manière avec laquelle les différents systèmes logiciels agissent les uns avec les autres. Les logiciels ayant de bonnes caractéristiques d'interopérabilité peuvent être facilement intégrés dans un certain nombre d'autres systèmes sans nécessiter de modifications majeures.

Le nombre de modifications et l'effort nécessaire pour les réaliser peuvent être utilisés comme mesure d'interopérabilité.

Tester l'interopérabilité peut par exemple se concentrer sur les caractéristiques de conception suivantes:

- L'utilisation de normes développées par l'industrie de la communication, telle que XML.
- La capacité des logiciels à détecter automatiquement le format de l'interface de communication avec un autre système et à s'y adapter.

Les tests d'interopérabilité sont particulièrement pertinents pour

- Les organisations qui développent des outils et des composants sur étagère (COTS : "Commercial Of The Shell")
- Le développement de systèmes de systèmes.

Cette forme de test est surtout utilisée pendant les tests d'intégration.

5.2.4 Tests Fonctionnels de Sécurité

Les tests fonctionnels de sécurité (tests d'intrusion) vérifient la capacité du logiciel à empêcher les accès non autorisés vers les fonctionnalités ou les données, qu'ils soient accidentels ou délibérés.

Les droits des utilisateurs, les accès et les privilèges font partie de ces tests.

Ces informations devraient être présentes dans les spécifications du système.

Les tests de sécurité incluent également de nombreux aspects qui relèvent plus des analystes de tests techniques et sont discutés dans la section 5.3 ci-dessous.

5.2.5 Tests d'Utilisabilité

Il est important de comprendre pourquoi les utilisateurs peuvent avoir des difficultés à utiliser le système logiciel proposé. Pour se faire, il faut en premier lieu apprécier ce que le mot « utilisateur » veut dire pour différentes personnes allant de l'expert en technologies de l'information, à l'enfant, jusqu'aux personnes handicapées.

Certaines institutions (ex : L'Institut Royal Britannique pour les aveugles) recommandent que les pages internet soient accessibles pour les personnes handicapées, aveugles, à mobilité réduite paralysées partiellement, sourdes, ou aux sensations réduites.

Vérifier que les applications et les sites internet sont utilisables pour toutes ces personnes, augmenterait l'utilisabilité pout tout le monde.

Les tests d'utilisabilité permettent de chiffrer dans quelle mesure le logiciel convient aux utilisateurs et visent à mesurer les facteurs suivants grâce auxquels des utilisateurs spécifiques pourront atteindre des buts précis dans des environnements ou contextes d'usage particuliers. :.

- Efficacité: La capacité du logiciel à répondre aux besoins spécifiques des utilisateurs avec exactitude et perfection et dans un contexte d'utilisation précis.
- Rendement: Ensemble de critères portant sur le rapport existant entre le niveau de service d'un logiciel et la quantité de ressources utilisées, dans des conditions déterminées.
- Satisfaction: La capacité du produit logiciel à satisfaire les utilisateurs dans un certain contexte d'utilisation.

Les critères pouvant être mesurés sont :

- Compréhension: Caractéristique du logiciel qui affecte l'effort de l'utilisateur pour reconnaitre le concept logique et son application.
- Apprentissage: Caractéristique du logiciel qui affecte l'effort de l'utilisateur pour apprendre le fonctionnement de l'application
- Opérabilité: Caractéristique du logiciel qui affecte l'effort de l'utilisateur pour exécution des fonctions avec efficacité et rendement
- Attrait: la capacité du logiciel à être apprécié par l'utilisateur

L'évaluation de l'utilisabilité a deux objectifs:

- Supprimer les défauts d'utilisabilité (quelquefois dite « évaluation formative »)
- Tester les exigences d'utilisabilité, (quelquefois dite « évaluation sommative »)

Les compétences des testeurs devraient inclure l'expertise ou la connaissance des domaines suivants:

- Sociologie
- Psychologie
- Conformité aux standards nationaux (par exemple, American Disabilities Act)
- Ergonomie

La validation d'une implémentation devrait être faite dans des conditions aussi proches que possible de celles dans lesquelles le système sera utilisé.

Cela peut signifier la mise en place d'un laboratoire d'essais avec caméra vidéo, salles de démonstration, des panels d'utilisateurs, etc., de façon à ce que l'équipe de développement puisse observer les effets du nouveau système sur des personnes réelles.

Beaucoup des tests d'utilisabilité peuvent être exécutés comme parties d'autres tests, par exemple pendant des tests fonctionnels du système. Pour détecter et informer sur les fautes d'utilisabilité aux différentes étapes du cycle de vie, et ce avec une approche consistante, des guides de tests d'utilisabilité peuvent être utiles.

5.2.5.1 Spécification des Tests d'Utilisabilité

Les principales techniques pour les tests d'utilisabilité sont

- L'inspection, l'évaluation ou la revue
- Réaliser la vérification et la validation de l'implémentation effective du système
- Réaliser des enquêtes et mettre en place des questionnaires

Inspection, **Evaluation ou Revue**

L'inspection ou la revue des spécifications et de la conception, dans une optique d'utilisabilité pour accroitre le niveau d'utilisation du système par l'utilisateur final, peut également agir sur les coûts en découvrant les problèmes au plus tôt.

L'évaluation heuristique (inspection systématique de la conception d'une IHM pour utilisabilité) peut être utilisée pour découvrir les problèmes d'utilisabilité de l'IHM. Elle peut être alors considérée comme faisant une partie d'un processus de conception itératif.

Cela implique d'avoir un petit groupe d'évaluateurs qui examinent l'IHM et jugent sa conformité avec des principes reconnus d'utilisabilité (les «heuristiques »)

Validation de l'Implémentation Effective

Pour réaliser la validation de l'implémentation effective du système, les tests fonctionnels spécifiés pour le système peuvent être développés comme des scénarios de tests d'utilisabilité. Ces scénarios de test mesurent alors des critères spécifiques d'utilisabilité comme la vitesse d'apprentissage ou l'opérabilité, plutôt que les résultats fonctionnels attendus.

Les scenarios de test d'utilisabilité peuvent être développés afin de tester spécifiquement la syntaxe et la sémantique.

- Syntaxe: La structure ou la grammaire de l'interface (ex : ce qui peut être entré dans un champ de saisie)
- Sémantique: signification et objectif (ex: messages système et d'information fournis à l'utilisateur, de taille raisonnable et significatifs)

Les techniques utilisées pour développer ces scenarios de test peuvent inclure:

- Les méthodes boites noires comme décrites, par exemple, dans la norme BS-7925-2
- Les cas d'utilisation, soit sous forme de texte, soit définis en UML (Unified Modeling Language)

Les scénarios de test d'utilisabilité incluent les instructions utilisateur, l'allocation de temps pour les pré et post interviews de test pour donner des instructions et recevoir le compte rendu, ainsi qu'un protocole d'accord pour exécuter les sessions de test.

Ce protocole inclue une description de « comment » le test sera effectué, les temps, la prise de note et le démarrage d'une session, ainsi que les méthodes d'interview et d'enquête à utiliser.

Enquêtes et Questionnaires

Les enquêtes et les questionnaires techniques peuvent être utilisés pour collecter des observations sur le comportement de l'utilisateur avec le système dans un laboratoire de tests d'utilisabilité.

Des questionnaires de suivi publics et standardisés comme SUMI (Software Usability Measurement Inventory) et WAMMI (Website Analysis and MeasureMent Inventory) permettent de faire un comparatif avec une base de données de mesures d'utilisabilité collectées.

De plus, puisque SUMI fournit des mesures concrètes d'utilisabilité, c'est une bonne opportunité de les utiliser comme critères d'achèvement et d'acceptation.

5.2.6 Tests d'Accessibilité

Il est important de prendre en compte les critères d'accessibilité du logiciel avec des exigences particulières ou en pensant à des restrictions d'utilisation. Cela inclut les exigences concernant les handicapés. Il faudra utiliser des standards comme le « Web Content Accessibility Guidelines », et la législation, comme la « Disability Discrimination Acts (UK, Australia) and Section 508 (US).

5.3 Caractéristiques Qualité pour les Tests Techniques

Les critères qualité pour les analystes techniques de test concernent "Comment le produit fonctionne », plutôt que les aspects liés à « Que fait le produit ». Ces tests peuvent se dérouler à tous niveaux, mais revêtent une signification particulière pour:

Les tests de composants (spécialement les systèmes temps réels ou embarqués)

- Les comparatifs de performance
- L'utilisation des ressources

Les tests système et les tests d'acceptation opérationnelle

- Incluent n'importe quel critère et sous-critère mentionné ci-dessous, en fonction des risques et des ressources disponibles.
- Les tests orientés « technique » à ce niveau sont destinés à tester un système spécifique, par exemple les combinaisons matériels-logiciels, incluant les serveurs, stations de travail clientes, bases de données, réseaux et autres ressources.

Fréquemment, les tests continuent d'être exécutés une fois en production, souvent par une équipe ou une organisation séparée. Les mesures des critères qualité collectés pendant les tests de pré-production peuvent donner les bases des contrats de service entre le fournisseur et l'exploitant du système logiciel.

Les critères qualité suivants sont pris en compte:

- Sécurité technique
- Fiabilité
- Rendement
- Maintenabilité
- Portabilité

5.3.1 Tests de Sécurité Techniques

Les tests de sécurité diffèrent d'autres domaines d'activité ou des tests techniques par deux éléments essentiels:

1. Les techniques standards pour sélectionner les données d'entrée des tests peuvent ne pas prendre en compte d'importants éléments de sécurité.

2. Les symptômes des fautes de sécurité sont très différents de ceux révélés avec d'autres types de test.

Beaucoup de failles de sécurité existent lorsque le logiciel fonctionne non seulement comme spécifié, mais exécute des actions supplémentaires non prévues. Ces effets de bord sont une des plus grandes menaces pour la sécurité logicielle. Par exemple, un media player qui joue correctement l'audio mais qui le fait en écrivant des fichiers temporaires non cryptés, met en évidence un effet de bord qui peut être exploité par des pirates.

Le principal élément en matière de sécurité est d'empêcher des utilisations non prévues par des personnes non autorisées. Les tests de sécurité essayent de compromettre la sécurité du système en évaluant sa vulnérabilité face aux menaces comme

- Les accès non autorisés aux applications ou aux données (ex : piratage)
- Le contrôle des accès non autorisés (ex : possibilité d'effectuer des tâches pour lesquelles l'utilisateur n'a pas les droits)
- Les débordements de pile qui peuvent être causés en entrant des chaines de caractères extrêmement longues dans un champ de saisie d'une interface utilisateur. Une fois que le débordement de pile a été causé, une possibilité d'exécuter du code malicieux peut exister.
- Déni de services, qui empêche les utilisateurs d'interagir avec une application (ex : en surchargeant un serveur avec des requêtes nuisantes).
- Des écoutes indiscrètes pendant les transferts de données via les réseaux pour obtenir des informations sensibles. (ex : les transactions bancaires)
- Casser les codes de cryptage utilisés pour protéger les données sensibles.
- Les bombes logiques (appelées parfois « Œufs de pâques » aux USA), qui peuvent être malicieusement insérées dans du code et qui s'activent seulement sous certaines conditions (ex : à une date spécifique). Quand les bombes logiques s'activent, elles peuvent exécuter des actes malicieux comme l'effacement de fichiers ou encore le formatage de disques.

En particulier les sujets concernés par la sécurité peuvent être regroupés comme suit:

- Relatifs à l'interface utilisateur
  - Accès non autorisés
  - Entrées malicieuses
- Relatifs au système de fichier
  - Accès à des données sensibles stockées dans des fichiers ou des bases de stockage
- Relatifs au système d'exploitation
  - Mise en mémoire de données sensibles comme les mots de passe une fois décryptés. Faire tomber un système par des entrées malicieuses peut faire afficher ces informations
- Relatifs à des logiciels externes

- Les interactions qui peuvent se créer entre des composants externes que le système utilise. Cela peut se dérouler au niveau réseau (ex: paquets incorrects ou des messages passés) ou au niveau d'un composant logiciel (ex: défaillance d'un composant logiciel sur lequel le logiciel s'appui.

Il est à noter que les améliorations faites à la sécurité d'un système peuvent affecter ses performances. Après des améliorations de la sécurité, il est conseillé de répéter des tests de performance.

5.3.1.1 Spécification des Tests de Sécurité

L'approche suivante peut être suivie pour développer des tests de sécurité :

- Recueillir des informations
  Un profil ou une carte réseau du système est construit en utilisant les outils du marché. Ces informations peuvent inclure le nom des employés, leur adresse physique, des détails concernant les réseaux internes, les adresses IP, la liste des matériels et des logiciels utilisés ainsi que les versions de systèmes d'exploitation.
- Pratiquer un scan de vulnérabilité.
  Le système est scanné par des outils du marché. Ces outils ne sont pas utilisés pour directement compromettre le système, mais pour identifier les vulnérabilités qui sont, ou qui peuvent créer une faille dans la politique de sécurité. Des vulnérabilités peuvent également être identifiées en utilisant des check-lists comme celles fournies sur le site www.testingstandards.co.uk
- Développer des « plans d'attaque » (ex un plan d'actions de tests destinées à compromettre une politique de sécurité système particulière) en utilisant les informations recueillies. Plusieurs intrusions à travers différentes interfaces (ex interface utilisateur, système de fichier) doivent être spécifiées en plans d'attaque pour détecter les failles de sécurité les plus sévères.
- Les différentes « attaques » décrites dans [Whittaker04] sont une source de techniques spécifiquement développées pour les tests de sécurité. Pour plus d'information sur les « attaques », se référer à la section 4.4.

5.3.2 Tests de Fiabilité

Un des objectifs des tests de fiabilité est de suivre dans le temps des mesures statistiques de la maturité logicielle et de les comparer avec les objectifs attendus. Ces mesures peuvent prendre la forme de Temps Moyen Entre Défaillances (Mean Time Between Failures MTBF), Temps Moyen de Réparation (Mean Time To Repair MTTR) ou toute autre forme de mesure de l'intensité des défaillances (ex: nombre de défaillance d'une certaine gravité par semaine).

L'évolution des valeurs suivies dans le temps est appelée « modèle de croissance de fiabilité » Ces tests peuvent prendre un temps considérable (de l'ordre de la journée). L'analyse des mesures de maturité logicielle dans le temps peut être utilisée comme critère de sortie (ex : pour une mise en production).

Les mesures spécifiques comme le MTBF et le MTTR peuvent faire partie d'un contrat de service (SLA) et suivies en production. La méthode Software Reliability Engineering and Testing (SRET) est une approche standard de tests de fiabilité.

5.3.2.1 Tests de Robustesse

Alors que les tests fonctionnels évaluent la tolérance aux fautes du logiciel à travers l'injection de données d'entrée inattendues (également appelés tests négatifs), les tests orientés technique évaluent la tolérance aux erreurs du logiciel qui arrivent en dehors de l'application testée. Ces erreurs sont typiquement retournées par les systèmes d'exploitation (ex disque plein, processus ou programme non accessible, fichier non trouvé, mémoire non disponible). Des outils sont spécialisés dans les tests de tolérance aux erreurs au niveau système.

5.3.2.2 Tests de Récupérabilité

Des formes supplémentaires de tests de fiabilité évaluent la capacité du système logiciel à se remettre, de façon contrôlée, dans un état permettant un fonctionnement normal à la suite de défaillances logicielles ou matérielles. Les tests de récupérabilité incluent la tolérance aux pannes et les tests de sauvegarde et de restauration.

Les tests de tolérance aux pannes sont exécutés là ou les conséquences d'une défaillance logicielle sont si fortes que des mesures spécifiques matérielles et logicielles sont mises en œuvre pour assurer le fonctionnement du système même en cas de défaillance. Les tests de tolérance de pannes peuvent être appliqués, par exemple, quand le risque de pertes financières est extrême ou quand des risques critiques de sécurité existent. Lorsque les défaillances résultent d'un évènement catastrophique, cette forme de test de récupérabilité est également connue sous le nom de « Tests de récupérabilité de désastres »

Les solutions matérielles habituelles incluent la balance de charge entre plusieurs processeurs, la clusterisation de serveurs, processeurs ou disques de façon à ce qu'un autre remplace un autre immédiatement en cas de en panne. (Ex. RAID: Redundant Array of Inexpensive Disks).

Une solution habituelle peut être la mise en œuvre de plus d'une instance d'un système logiciel. (ex : Le tableau de pilotage d'un avion), appelés redondance de systèmes non identiques. Les systèmes redondants sont typiquement une combinaison de solutions matérielles et logicielles et sont parfois appelés Duplex, Triplex ou encore systèmes Quadriplex, suivant le nombre d'instances indépendantes (respectivement 2,3 ou 4).

La non similarité du logiciel est assurée lorsque les mêmes exigences logicielles sont fournies à deux (ou plus) équipes de développement indépendantes et sans aucun lien, avec pour objectif d'avoir les mêmes fonctions fournies par différents logiciels. Cela protège les systèmes redondants non similaires car une même donnée erronée en entrée aura moins de chance de donner le même résultat en sortie. Ces mesures prises pour améliorer la récupérabilité d'un système influencent directement sa fiabilité et donc doivent être prises en compte pour réaliser ces tests de fiabilité. Le test concernant la tolérance de pannes est conçu pour tester explicitement de tels systèmes en simulant les modes de défaillances ou en les réalisant dans un environnement contrôlé.

Suivant la défaillance, le mécanisme de tolérance de pannes est testé pour s'assurer que les données ne sont pas perdues ou corrompues et que les niveaux de service contractuels sont maintenus.

(ex: disponibilité, temps de réponse). Pour plus d'informations sur le test de tolérance de pannes, voir www.testingstandards.co.uk.

Les tests de sauvegarde et de restauration se focalisent sur les mesures procédurales mises en place pour minimiser les effets d'une défaillance. De tels tests évaluent les procédures (habituellement documentées dans un manuel) des différentes formes de sauvegardes et restaurations prévues, suivant que les données soient perdues ou corrompues. Des cas de test sont conçus pour s'assurer que les chemins critiques sont couverts au travers des procédures. Des revues techniques sont réalisées pour « jouer à blanc » ces scénarios et valider les manuels par rapport aux procédures d'installation mises en place. Les tests d'acceptation opérationnelle exécutent les scénarios dans un environnement de production ou proche de la production afin de valider leur fonctionnement réel.

Les mesures prises pour les tests de sauvegardes et de restaurations peuvent inclure les éléments suivants :

- Le temps passé à réaliser différentes sortes de sauvegarde (ex : complète et incrémentale)
- Le temps de restaurations
- Les niveaux garantis des sauvegardes de données (ex : restauration de toutes les données âgées de moins de 24 heures, restauration d'une transaction âgée de moins d'une heure).

5.3.2.3 Spécification des Tests de Fiabilité

Les tests de fiabilité sont le plus souvent basés sur des modèles d'utilisation (quelquefois appelés "Profils opérationnels ou profils métiers") et peuvent être réalisés littéralement ou bien en tenant compte du risque. Les données de test peuvent être générées en utilisant des méthodes aléatoires ou pseudo aléatoires.

Le choix de la courbe de croissance de fiabilité doit être justifié et des outils peuvent être utilisés pour analyser un ensemble d'erreurs et déterminer la courbe de fiabilité qui correspond le mieux avec les données disponibles. Les tests de fiabilité peuvent rechercher les fuites mémoire. La spécification de tels tests requiert que des actions intensives sur la mémoire soient exécutées de manière répétée pour s'assurer que la mémoire allouée est correctement libérée.

5.3.3 Tests de Rendement

Le critère qualité de rendement est évalué en conduisant des tests basés sur le temps et le comportement des ressources. Le test de rendement lié au comportement dans le temps est abordé dans cette section avec les aspects performance, charge, stress, et scalabilité.

5.3.3.1 Tests de Performance

Les tests de performance peuvent être généralement classés en différents types de test en fonction de l'exigence non fonctionnelle que l'on teste. Ces types de tests incluent la performance, la charge, le stress et la scalabilité.

Des tests spécifiques de performance se concentrent sur la capacité d'un composant ou d'un système à répondre aux entrées utilisateur ou système dans un temps donné et sous certaines conditions (voir aussi charge et stress ci-dessous). La mesure des performances varie en fonction des objectifs de test.

Pour des composants logiciels indépendants, la performance peut être mesurée en temps CPU, alors que la performance de systèmes clients peut être mesurée en fonction du temps de réponse à une demande utilisateur. Pour les systèmes dont l'architecture est constituée de plusieurs composants, (ex clients, serveurs, bases de données) les mesures de performance sont prises entre les composants individuels de façon à ce que les goulets d'étranglement soient identifiés.

5.3.3.2 Tests de Charge

Les tests de charge évaluent la capacité d'un système à accepter des montées en charge réalistes et prévues résultant de transactions générées par de multiples utilisateurs en parallèle. Les temps de réponse moyen de l'exécution de différents scénarios métier (profils opérationnels) peuvent être mesurés et analysés. (Voir également [Splaine01])

Il existe deux sous-types de tests de charge, « multiutilisateurs » (avec un nombre réaliste d'utilisateurs) et « volume » (avec un grand nombre d'utilisateurs). Les tests de charge recherchent aussi bien les réponses en temps qu'en utilisation (charge) réseau.

5.3.3.3 Tests de Stress

Les tests de stress considèrent la capacité d'un système à accepter les pics de charge au delà ou à la capacité maximum. La performance du système doit se dégrader lentement, de façon prévisible et sans panne lorsque ces niveaux de stress augmentent. En particulier l'intégrité fonctionnelle du système devrait être testée alors que le système est sous stress de façon à détecter les défauts possible de fonctionnement ou des données incohérentes. Un des objectifs des tests de stress est de découvrir les limites auxquelles le système tombe en panne de façon à ce que le « maillon faible » soit identifié.

Les tests de stress permettent également l'ajout de composants au système de façon opportune (ex mémoire, capacité CPU, base de données).

Dans le test de pic de charge, les combinaisons de conditions qui peuvent résulter d'une charge soudaine et extrême sur le système sont simulées. Des tests de rebond (tests de diagnostic) génèrent des pics sur le système avec des périodes de faible utilisation entre les pics. Ces tests déterminent la capacité du système à appréhender les changements de charge et si en fonction d'eux, le système est alors capable de demander ou de libérer les ressources. Voir également [Splaine01].

5.3.3.4 Tests de Scalabilité

Les tests de scalabilité vérifient les capacités du système à satisfaire de futures exigences de rendement pouvant aller plus loin que celles en vigueur. L'objectif de ces tests est de juger de la capacité du système à croître (ex : avec plus d'utilisateurs, des bases de données plus importantes) sans dépasser les limites prévues ou défaillir. Une fois que ces limites sont connues, des valeurs seuil sont alors données et surveillées en production afin envoyer une alerte en cas de problème imminent.

5.3.3.5 Test d'Utilisation des Ressources

Les tests de rendement relatifs à l'utilisation des ressources évaluent l'utilisation des ressources système. (ex: mémoire, espace disque, bande passante). Leur fonctionnement normal est comparé à celui sous stress avec un grand nombre de transactions et de données. Par exemple, l'utilisation de la mémoire dans

les systèmes embarqués temps réel (quelquefois appelés « empreinte mémoire ») joue un rôle important dans les tests de performance.

5.3.3.6 Spécification des Tests de Rendement

Les spécifications des tests de rendement comme la performance, la charge et le stress, sont basées sur la définition de profils opérationnels (métiers). Ce sont des formes distinctes d'utilisation de l'application par des groupes d'utilisateurs. Il peut y avoir plusieurs profils métier pour une application. Le nombre d'utilisateurs par métier peut être obtenu en utilisant un outil de surveillance (quand une application comparable est en utilisation) ou bien des prévisions.

Ces prévisions peuvent être basées sur des algorithmes ou fournies par l'organisation métier. Elles sont particulièrement importantes pour spécifier les profils métiers des tests de scalabilité. Les profils opérationnels sont la base des cas de test et sont générés habituellement par des outils de tests. Dans ce cas, le terme d'utilisateur virtuel est utilisé pour représenter un utilisateur simulé du profil opérationnel.

5.3.4 Tests de Maintenabilité

Les tests de maintenabilité sont relatifs à la facilité avec laquelle le logiciel peut être analysé, modifié et testé. Les techniques appropriées pour les tests de maintenabilité incluent les méthodes d'analyse statique et les check-lists.

5.3.4.1 Tests Dynamiques de Maintenabilité

Les tests dynamiques de maintenabilité concernent les procédures documentées développées pour la maintenance d'une application particulière. (ex : pour réaliser des mises à jour logicielles). Une sélection de scénarios de maintenance est utilisée comme cas de test pour s'assurer que les niveaux de service sont atteignables avec les procédures documentées.

Cette forme de test est particulièrement utile lorsque l'infrastructure est complexe et que les procédures font appel à de multiples départements ou organisations. Cette forme de test peut faire partie des tests d'acceptation opérationnelle . [www.testingstandards.co.uk]

5.3.4.2 Analysabilité (maintenance corrective)

Cette forme de tests de maintenabilité pointe sur le temps passé à diagnostiquer et résoudre les problèmes identifiés dans un système. Une mesure simple peut être le temps moyen pris pour diagnostiquer et réparer une faute identifiée (voir ISO 9126 : attributs : analysabilité)..

5.3.4.3 Variabilité, Stabilité et Testabilité (maintenance adaptive)

La maintenabilité d'un système peut également être mesurée en termes d'efforts requis pour réaliser les changements sur ce système (ex modifications de code). Puisque l'effort requis est dépendant d'un bon nombre de facteurs comme la méthodologie de conception logicielle (ex : conception objet), les standards de programmation, etc., cette forme de tests de maintenabilité peut également être réalisée par analyse ou revue. La testabilité est liée à l'effort demandé pour tester le changement fait. La stabilité est relative à la réponse au changement du système. Les systèmes avec une stabilité faible montrent un grand nombre de problèmes induits (également connus comme « effet de vague ») à chaque fois qu'un changement est réalisé. [ISO9126] [www.testingstandards.co.uk]

5.3.5 Tests de Portabilité

Les tests de portabilité sont en général relatifs à la facilité avec laquelle un logiciel peut être transféré dans son environnement déjà créé ou pas. Les tests de portabilité incluent les tests d'installation, de compatibilité/co-existence, d'adaptation et de remplacement d'un autre composant logiciel.

5.3.5.1 Tests d'Installabilité

Les tests d'installabilité sont conduits sur le logiciel d'installation du logiciel vers son environnement cible. Cela inclut, par exemple, le logiciel développé pour installer un système d'exploitation sur un processeur, un wizard d'installation utilisé pour installer un produit sur un PC. Les objectifs habituels concernant les tests d'installabilité incluent :

- La validation de la bonne installation du logiciel effectuée en suivant les instructions d'un manuel d'installation (y compris l'exécution de scripts), ou en utilisant un wizard d'installation. Cela signifie essayer les options d'installation pour différentes configurations logicielles et matérielles et différent niveaux d'installation. (ex complète ou partielle)
- Tester si les pannes qui arrivent pendant l'installation (ex : Erreur de chargement d'une DLL) sont pris en charge par le programme d'installation sans le quitter dans un état indéfini (ex : logiciel partiellement installé ou configuration incorrecte)
- Tester si une installation partielle ou une désinstallation est possible
- Tester si un wizard d'installation peut identifier des plateformes matérielles ou des configurations de système d'exploitation incorrectes
- Mesurer si le processus d'installation peut être réalisé dans un temps défini ou sans dépasser un certain nombre d'étapes
- Valider que le logiciel peut être téléchargé ou désinstallé.

Le test fonctionnel est habituellement conduit après les tests d'installation pour détecter toutes les anomalies pouvant avoir été introduites par l'installation (ex : configurations incorrectes, fonctions non disponibles). Les tests d'utilisabilité sont habituellement conduits en parallèle des tests d'installabilité (ex : pour valider que les utilisateurs, pendant l'installation, reçoivent des instructions et des messages de confirmation ou d'erreur compréhensibles).

5.3.5.2 Tests de Co-Existence

Les systèmes informatiques non reliés les uns aux autres sont dits compatibles lorsqu'ils peuvent fonctionner dans le même environnement (ex : sur le même matériel) sans qu'il n'y ait d'effet sur le comportement de chacun (ex conflits de ressources). Des tests de compatibilité peuvent être faits, par exemple, quand un logiciel nouveau ou mis à jour est déployé dans des environnements (ex serveurs) qui contiennent déjà des applications. Les problèmes de compatibilité peuvent survenir lorsque l'application est testée dans un environnement vierge (ou les problèmes de compatibilité ne sont pas détectables) puis déployée sur un autre environnement (ex : la production) qui fait tourner d'autres applications.

Les principaux objectifs des tests de compatibilité sont :

- L'évaluation d'un impact négatif sur une fonctionnalité quand des applications sont chargées sur le même environnement (ex conflits de ressources quand un serveur fait tourner plusieurs applications)
- L'évaluation de l'impact sur toute application résultant du déploiement des mises à jour et des correctifs d'un système d'exploitation.

Les tests de compatibilité sont normalement réalisés quand les tests d'acceptation système et utilisateur se sont bien déroulés.

5.3.5.3 Tests d'Adaptabilité

Les tests d'adaptabilité vérifient si une application peut fonctionner correctement dans tous les environnements cibles (matériels, logiciels, intergiciels, systèmes d'exploitation, etc.). Spécifier des tests d'adaptabilité requiert que les combinaisons des environnements cibles soient identifiées, configurées et disponibles pour l'équipe de tests. Ces environnements sont alors testés avec une sélection de cas de test fonctionnels qui essayent les divers composants présents dans l'environnement.

L'adaptabilité peut être relative à la capacité du logiciel à être porté sur des environnements divers en exécutant une procédure prédéfinie. Les tests peuvent évaluer cette procédure.

Les tests d'adaptabilité peuvent être réalisés en conjonction des tests d'installation et sont habituellement suivis par les tests fonctionnels pour détecter toutes les anomalies qui auraient été introduites en adaptant le logiciel à un environnement différent.

5.3.5.4 Tests de Remplaçabilité

La remplaçabilité est la capacité de composants logiciels dans un système à être échangés avec d'autres. Cela est particulièrement significatif pour les systèmes qui utilisent des composants sur étagère (COTS). Les tests de remplacement peuvent être réalisés en parallèle des tests fonctionnels d'intégration quand plus d'un composant alternatif est disponible pour l'intégration dans un système complet.

La remplaçabilité peut être évaluée par des revues techniques ou des inspections dans lesquelles l'accent est mis sur la définition claire des interfaces avec d'éventuels composants de remplacement.

6. Revues

Termes

Audit, IEEE 1028, revue informelle, inspection, modérateur, revue de gestion, revue, réviseur, revue technique, walkthrough.

6.1 Introduction

Un processus de revue réussi exige planification, participation et suivi. Les organismes de formation ont besoin de s'assurer que les responsables de test comprennent les responsabilités qu'ils ont pour les activités de planification et de suivi. Les testeurs doivent être des participants actifs dans le processus de revue, fournissant leur unique point de vue. Ils devront posséder une formation de revue formelle pour mieux comprendre leurs rôles respectifs dans n'importe quel processus de revue technique. Tous les participants de la revue doivent être convaincus des avantages d'une revue technique bien dirigée. Quand elles sont réalisées proprement, les inspections sont le plus grand et le plus économique atout pour la qualité globale délivrée. Une norme internationale sur les revues est IEEE 1028.

6.2 Principes des Revues

Une revue est un type de tests statiques. Les revues ont fréquemment comme objectif majeur la détection des défauts. Les réviseurs trouvent les défauts en examinant directement les documents.

Les types fondamentaux de revues sont décrits dans le chapitre 3.2 du Syllabus niveau Fondation (version 2005) et sont listés ci-après dans le chapitre 6.3.

Tous les types de revue sont le mieux exécutés dès que les documents sources pertinents (documents qui décrivent les exigences du projet) et les normes (celles auxquelles le projet doit adhérer) sont disponibles. Si l'un des documents ou normes est manquant, alors les erreurs et les divergences à travers toute la documentation ne peuvent être découvertes, seules celles contenues dans un document peuvent être découvertes. Les réviseurs doivent être munis du document à revoir suffisamment tôt pour pouvoir se familiariser avec son contenu.

Tous les types de documents peuvent être soumis à une revue, par exemple code source, spécifications des exigences, concepts, plan de tests, documents de tests, etc. Le Test Dynamique suit normalement une revue de code source; il est conçu pour trouver tous les défauts qui ne peuvent pas être trouvés par un examen statique.

Une revue peut conduire aux trois résultats possibles:

- Le document peut être utilisé inchangé ou avec des changements mineurs
- Le document doit être modifié mais une revue supplémentaire n'est pas nécessaire
- Le document doit être énormément modifié et une revue supplémentaire est nécessaire

Les rôles et responsabilités des personnes impliquées dans une revue formelle classique sont couverts dans le Syllabus Niveau Fondation, par exemple responsable, modérateur, auteur, réviseurs et scribe. D'autres rôles peuvent être impliqués dans les revues, comme les décideurs ou parties-prenantes, et les représentants des clients ou des utilisateurs. Un rôle supplémentaire optionnel parfois utilisé dans les inspections est celui de lecteur, qui est destiné à paraphraser les chapitres du travail produit pendant la réunion. Outre les rôles de revues, des réviseurs individuels peuvent être chacun affectés à un rôle basé sur les anomalies pour regarder des types particuliers de défauts.

Plus d'un type de revues peut-être employé sur un même produit. Par exemple, une équipe peut tenir une revue technique pour décider quelles fonctionnalités sont à mettre en œuvre dans le prochain cycle. Une inspection pourra ensuite être réalisée sur les spécifications de ces fonctionnalités.

6.3 Types de Revues

Le Syllabus Fondation introduit les types de revues suivants:

- Revue informelle
- Walkthrough
- Revue technique
- Inspection

Des hybrides de ces types de revues peuvent aussi apparaître en pratique, tel qu'une revue technique utilisant des règles fixées.

6.3.1 Revue de Gestion et Audit

Outre les types évoqués dans le Syllabus Fondation, IEEE 1028 décrit aussi les types de revues suivants:

- Revue de gestion
- Audit

Les caractéristiques clés d'une revue de gestion sont:

- Objectifs principaux: surveiller l'avancement, évaluer les états, et prendre les décisions à propos des futures actions
- Mis en place par ou pour les responsables ayant une responsabilité directe sur le projet ou système
- Mis en place par ou pour un intervenant ou décideur, par exemple un haut niveau de responsable ou directeur
- Vérifie la cohérence et les divergences des plans, ou l'adéquation de la gestion des procédures
- Inclue l'évaluation des risques du projet
- Le résultat inclus la liste des actions et des problèmes à résoudre
- Les participants doivent préparer, les décisions sont documentées

Noter que les responsables de tests devront y participer et seront les instigateurs des revues de gestion et de l'avancement des tests.

Les audits sont extrêmement formels, et sont généralement réalisés pour démontrer la conformité à certaines attentes, comme une norme applicable ou une obligation contractuelle. Comme tels, les audits sont les moins efficaces pour révéler les défauts.

Les caractéristiques clés d'un audit sont:

- Objectifs principaux: fournir une évaluation indépendante de la conformité des processus, règlements, normes etc.
- Un chef auditeur est responsable pour l'audit et les actes comme le modérateur
- Les auditeurs rassemblent les preuves de conformité à travers les entretiens, les témoignages et l'étude des documents
- Le résultat inclus les observations, recommandations, actions correctives et une évaluation réussie/échouée

6.3.2 Revues de Livrables Particuliers

Ces revues peuvent être décrites en termes de travaux ou activités sont susceptibles d'être revus, tels que:

- Revue contractuelle
- Revue des exigences
- Revue de conception
  - revue de conception préliminaire
  - revue de conception critique
- Revue d'acceptation / revue de qualification
- Revue de préparation opérationnelle

Une revue contractuelle peut être associée à un jalon de contrat, et pourrait être typiquement une revue de gestion pour un système à sécurité critique ou un système lié à la sécurité. Elle impliquerait responsables, clients et équipes techniques.

Une revue des exigences peut-être une relecture technique, une revue technique ou une inspection, et peut considérer la sûreté et la véracité des exigences aussi bien que les exigences fonctionnelles et non-fonctionnelles. Une revue des exigences peut inclure critère d'acceptation et conditions de tests.

Les revues de conception sont typiquement des revues techniques ou inspections, et impliquent les équipes techniques et clients ou décideurs. Le Revue de Conception Préliminaire propose l'approche initiale à quelques concepts techniques et tests; La Revue de Conception Critique couvre toutes les solutions des concepts proposés, incluant les cas de test et les procédures.

La revue d'acceptation à pour objectif d'obtenir l'approbation de la direction, d'un système. Cela est aussi évoqué dans la Revue de Qualification, et est normalement une revue de gestion ou un audit.

6.3.3 Réaliser une Revue Formelle

Le Syllabus Fondation décrit les six phases d'une revue formelle: planification, lancement, préparation individuelle, réunion de revue, correction, suivi. Les éléments à revoir devraient être appropriés pour la qualification ou le réviseur, par exemple un Plan de Tests pour le gestionnaire de Tests, des exigences fonctionnelles ou une conception de test pour un Testeur, ou une spécification fonctionnelle, cas de test ou scripts de tests pour un Testeur Technique.

6.4 Introduction des Revues

Pour que les revues soient introduites avec succès dans une organisation, les étapes suivantes doivent apparaître (par nécessairement dans cet ordre):

- Sécurisation des supports de gestion
- Sensibilisation des responsables aux coûts, bénéfices et aux possibilités de mise en œuvre
- Sélection et documentation des procédures de revues, formulaires et infrastructures (par exemple revue des indicateurs de la base de données)
- Formation aux techniques et procédures de revue
- Obtention du support de ceux qui feront les revues et auront leurs travaux revus
- Exécution de revues pilotes
- Démonstration de l'avantage des revues à travers l'économie des coûts
- Application des revues aux documents les plus importants, par exemple exigences, contrats, plans etc.

Des métriques telles que la réduction ou l'élimination du coût de réparation des défauts et/ou leur conséquence peuvent être utilisées pour évaluer la réussite de l'introduction des revues. Les économies peuvent être aussi mesurées avec le temps gagné grâce à la découverte et à la résolution de défauts plus tôt.

Les processus de revues devraient être continuellement surveillés et améliorés avec le temps. Les responsables devraient être conscients qu'une nouvelle technique de revue est un investissement – les avantages ne sont pas immédiats mais grandiront significativement avec le temps.

6.5 Facteurs de Succès pour les Revues

Plusieurs facteurs contribuent au succès des revues. Les revues ne doivent pas être difficiles à réaliser, mais elles peuvent mal se dérouler en différents cas si certains facteurs ne sont pas pris en compte :

Facteurs Techniques

- S'assurer que le processus défini pour le type de revue est suivi correctement, particulièrement pour les types de revues les plus formels tel que les inspections
- Noter les coûts des revues (incluant le temps dépensé) et les avantages réalisés
- Passer en revue au plus tôt les brouillons et les documents partiels pour identifier les modèles de défauts avant qu'ils ne soient introduits dans le document entier.

- S'assurer que les documents ou les documents partiels sont prêts à être revus avant de commencer le processus de revue (par exemple définir un critère d'entrée)
- Utiliser des check-lists des défauts les plus fréquents, dans l'organisation
- Utiliser plus d'un type de revue, dépendant des objectifs, tel que la relecture de document, l'amélioration technique, le transfert d'information, ou la gestion de l'avancement
- Relire ou inspecter les documents sur lesquels des décisions importantes vont être prises, par exemple, inspecter une proposition, un contrat ou une exigence de haut niveau avant une revue de gestion autorisant une dépense majeure sur le projet
- Prélever une partie limitée d'un document pour évaluer la relecture
- Encourager à trouver les plus importants défauts: se concentrer sur le contenu pas sur la forme
- Continuellement améliorer le processus de revue

Facteurs Organisationnels

- S'assurer que les responsables accordent le temps nécessaire aux activités de revues, même sous la pression des échéances
- Se souvenir que le temps et l'argent dépensés ne sont pas proportionnels aux erreurs trouvées
- Dégager le temps nécessaire pour retravailler les défauts identifiés par les revues
- Ne jamais utiliser les indicateurs des revues pour une estimation de performance individuelle
- S'assurer que les bonnes personnes sont impliquées dans les différents types de revues
- Dispenser des formations sur les revues, particulièrement pour les types de revues les plus formels
- Maintenir un forum du chef de revue pour partager expériences et idées
- S'assurer que chacun participe aux revues et que les documents de chacun sont relus
- Appliquer les techniques de revues les plus solides aux les documents les plus importants
- S'assurer du bon équilibre de la composition de l'équipe de revue avec différentes compétences et expériences
- Contribuer aux actions d'amélioration du processus pour traiter les problèmes systémiques
- Identifier les améliorations obtenues à travers le processus de revue

Facteurs Humains

- Sensibiliser les parties prenantes à l'intérêt des revues pour trouver des défauts et à la nécessité de consacrer du temps pour retravailler et répéter les revues
- S'assurer que la revue est une expérience positive pour l'auteur
- Accueillir l'identification des défauts dans une ambiance sereine
- S'assurer que les commentaires sont constructifs, utiles et objectifs, non subjectifs
- Ne pas relire si l'auteur n'est pas d'accord ou n'est pas prêt
- Motiver chacun à penser profondément aux aspects les plus importants des documents revus

Pour de plus informations sur les revues et inspections voir [Gilb93] et [Weigers02].

7. Gestion des Incidents

Termes

IEEE 829, IEEE 1044, IEEE 1044.1, anomalie, comité de contrôle de configuration, défaut, erreur, défaillance, incident, inscription des incidents, priorité, analyse des causes racines, sévérité

7.1 Introduction

Les gestionnaires de test et les testeurs doivent être familiers avec le processus de gestion des défauts. Les gestionnaires de test portent leur attention sur le processus lui-même, incluant les méthodes de reconnaissance, de suivi et de suppression des défauts. Les testeurs sont concernés au premier chef par l'enregistrement précis des défauts trouvés dans leur périmètre de test. Pour chacune des étapes du cycle de vie, les analystes de test et les analystes techniques de test adopteront une approche différente. Les analystes de test évalueront le comportement en termes d'exigences métier ou utilisateur, par exemple, les utilisateurs sauront-ils comment réagir face à tel message ou comportement. Les analystes techniques de test évalueront le comportement du logiciel lui-même et feront plus volontiers une analyse technique du problème, par exemple, tester la même défaillance sur différentes plateformes ou avec différentes configurations de mémoire.

7.2 Quand peut-on Détecter un Défaut ?

Un incident est un événement inattendu qui nécessite une analyse plus approfondie. Un incident est la détection d'une défaillance causée par un défaut. Un incident peut ou non donner suite à la production d'un rapport d'anomalie. Un défaut est un problème réel qui nécessite d'être résolu en modifiant le composant.

Un défaut peut être détecté au moyen de tests statiques. Une défaillance ne peut être détectée que par les tests dynamiques. Chaque phase du cycle de vie du logiciel devrait pouvoir fournir une méthode permettant de détecter et d'éliminer les défaillances potentielles. Par exemple, durant la phase de développement, des revues du code et du design devraient être employées pour détecter les défauts. Lors des tests dynamiques, des cas de test sont employés pour détecter les défaillances. Plus le défaut est détecté et corrigé tôt, moins il aura d'impact sur le coût de la qualité du système dans son ensemble.

Il faut aussi garder à l'esprit que des défauts peuvent aussi être présents dans le testware de même que dans les objets de test.

7.3 Cycle de Vie des Défauts

Tous les défauts ont un cycle de vie, cependant certaines étapes peuvent parfois être sautées. Ce cycle de vie (décrit par IEEE 1044-1993) se compose de quatre étapes:

- Étape 1: Détection
- Étape 2: Analyse
- Étape 3: Action
- Étape 4: Clôture

Pour chacune des étapes, on distingue trois activités de collecte d'informations

- Enregistrement
- Classification
- Identification de l'impact

7.3.1 Étape 1 Détection

L'étape de détection a lieu lorsqu'un défaut potentiel (incident) est découvert. Ceci peut se produire à n'importe quelle étape du cycle de vie du logiciel. Au moment de la découverte, les données permettant d'identifier le défaut sont enregistrées. Celles-ci incluent des informations sur l'environnement où le défaut

a été découvert, l'auteur, sa description, le moment de la découverte et le fournisseur (s'il y a lieu). La détection est caractérisée au moyen de certains attributs du défaut potentiel incluant l'activité et la phase du projet, la cause suspectée, la répétabilité, les symptômes et le statut du produit résultant de l'anomalie. Avec cette information, l'impact est évalué au moyen d'un niveau de sévérité, de l'impact sur le planning du projet et de l'impact sur les coûts du projet.

7.3.2 Étape 2 Analyse

Après la détection, chaque défaut potentiel est analysé. Cette étape est tout d'abord utilisée pour identifier tout problème lié à celui-ci et proposer des solutions pouvant inclure l'absence d'action (par exemple le défaut potentiel n'est plus considéré comme un vrai défaut). Des informations supplémentaires sont enregistrées et l'on procède à une réévaluation de la caractérisation et des informations qui ont permis d'estimer l'impact à l'étape précédente.

7.3.3 Étape 3 Action

En s'appuyant sur les résultats de l'analyse, l'étape d'action commence. Les actions incluent toutes les mesures requises pour résoudre le défaut ainsi que toutes celles qui s'imposent afin de réviser/améliorer les processus et les politiques destinées à éviter de futurs défauts semblables. Des tests de régression, un re-test ainsi que des tests de progression doivent être exécutés pour chaque changement. Des informations supplémentaires sont enregistrées, la caractérisation est revue de même que les informations concernant l'impact obtenues à l'étape précédente.

7.3.4 Étape 4 Conclusion

L'anomalie passe ensuite par l'étape de conclusion où d'autres informations sont enregistrées et où la conclusion prend l'un des statuts suivants : fermé, reporté, fusionné ou se rapportant à un autre projet.

7.4 Champs des Défauts

IEEE 1044-1993 spécifie un ensemble de champs obligatoires qui sont utilisés à différents moments au cours du cycle de vie du défaut. Selon IEEE 1044.1, lorsqu'on implémente IEEE 1044-1993, il est acceptable de créer une correspondance entre les termes décrits dans IEEE pour nommer les champs relatifs aux défauts et les termes en usage dans l'entreprise. Ceci permet la conformité avec IEEE 1044-1993 sans pour autant adhérer strictement aux conventions de nommage. La conformité à IEEE permet une comparaison des informations de défauts entre différentes entreprises et organisations.

Indépendamment du fait que la conformité à IEEE soit un but ou non, les champs fournis pour caractériser le défaut sont destinés à fournir suffisamment d'informations pour que le défaut puisse être traité. Un rapport d'anomalie permettant de traiter le défaut est :

- Complet
- Concis
- Précis
- Objectif

De plus, afin de résoudre le défaut, il est encore nécessaire de fournir des informations permettant sa caractérisation, l'analyse des risques et l'amélioration du processus.

7.5 Métriques et Gestion des Défauts

L'information relative au défaut doit impérativement comporter suffisamment d'informations pour compléter le suivi de l'avancement des tests, l'analyse de la densité des défauts, les métriques de type « découvert vs. résolu » et les métriques de convergence (ouvert vs. fermé). De plus, l'information relative au défaut doit participer aux initiatives d'amélioration de processus en contribuant à la maîtrise des phases, à l'analyse des causes racines, à l'identification des tendances des défauts afin de servir d'apport aux ajustements stratégiques de réduction des risques.

7.6 Communiquer les Incidents

La gestion des incidents comprend une communication efficace, non polémique, qui soutient la collecte et l'interprétation d'une information objective. La précision du rapport d'anomalie, une caractérisation appropriée et une objectivité démontrée sont cruciales pour maintenir des relations professionnelles entre les personnes qui identifient les défauts et celles qui les résolvent.

Les testeurs peuvent être consultés quant à l'importance relative d'un défaut et doivent fournir l'information objective disponible.

Des réunions destinées à trier les défauts peuvent être conduites afin de permettre l'assignation des priorités. Un outil de suivi des anomalies ne se substitue pas à une bonne communication. De même, les réunions destinées à trier les défauts ne se substituent pas à l'emploi d'un bon outil de suivi des anomalies. Une bonne communication et le soutien adéquat des outils sont nécessaires à un processus de gestion des anomalies efficace.

8. Normes & Processus d'Amélioration des Tests

Termes

Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI), Test Maturity Model (TMM), Test Maturity Model integration (TMMi), Test Process Improvement (TPI).

8.1 Introduction

Le support pour l'établissement et l'amélioration des processus de tests peut venir d'une multitude d'origines. Ce chapitre considère premièrement les normes comme une source d'information utile (parfois obligatoire) pour un nombre de sujets reliés au test. Savoir quelles normes sont disponibles et où elles peuvent être appliquées est considéré comme un objectif d'apprentissage pour les responsable de tests et les testeurs. Les organismes de formation devront souligner ces normes spécifiques qui sont particulièrement pertinentes pour le module enseigné.

Une fois établi, un processus de test devra subir une amélioration continue. Dans la section 8.3 les aspects relatifs à l'amélioration sont traités, suivis par l'introduction de quelques modèles spécifiques pouvant être utilisés pour l'amélioration du processus de test. Alors qu'il est demandé aux gestionnaires de test de maîtriser l'ensemble de cette section, il est aussi important que les analystes de tests et les analystes de tests techniques, acteurs clés dans la réalisation des améliorations, connaissent les modèles d'amélioration disponibles.

8.2 Normes & Standards

Ici, et dans le Syllabus Niveau Fondation, quelques standards sont mentionnés. Il existe des standards pour différents sujets associés au logiciel tels que:

- Cycles de vie de développement logiciel
- Méthodologies et test logiciel
- Gestion de configuration logicielle
- Maintenance logicielle
- Assurance Qualité
- Gestion de projet
- Exigences
- Langages de programmation
- Interfaces logicielles
- Gestion des anomalies

Ce n'est pas le but de ce syllabus de lister ou recommander des normes spécifiques. Les testeurs devraient savoir comment les normes sont créées, et comment elles devraient être utilisées dans l'environnement de l'utilisateur.

Les standards peuvent provenir de différentes sources:

- Internationale ou avec objectifs internationaux
- Nationale, telle que la mise en œuvre nationale de normes internationales
- Un domaine spécifique, quand les normes internationales ou nationales sont adaptées à un domaine particulier, ou développées pour des domaines spécifiques

Quelques considérations s'appliquent lorsque l'on utilise ces normes. Elles sont décrites dans la suite de ce chapitre.

8.2.1 Aspects Généraux sur les Standards

8.2.1.1 Origines des Standards

Les normes sont créées par des groupes de professionnels et reflètent une sagesse collective. Il y a deux sources majeures de normes internationales: IEEE et ISO. Les normes nationales et spécifiques à un domaine sont aussi importantes et disponibles.

Les testeurs devront être informés des normes à appliquer dans leur environnement et contexte, que ce soient des normes formelles (internationales, nationales ou spécifiques) ou des normes et pratiques internes. Comme les normes évoluent et changent il est nécessaire de spécifier la version (date de publication) de la norme pour assurer que la conformité est obtenue. Quand une référence à une norme est spécifiée sans la date ou version, c'est la dernière version qui s'applique.

8.2.1.2 Utilité des Standards

Les normes peuvent être utilisées comme un instrument pour promouvoir une assurance qualité constructive, qui se concentre sur la minimisation de l'introduction de défauts introduits plutôt que sur leur découverte à travers les tests (assurance qualité analytique). Toutes les normes ne sont pas applicables à tous les projets; l'information déclarée dans une norme peut être utile pour un projet, ou peut le bloquer. Suivre une norme avec comme seule motivation d'en suivre une n'aidera pas le testeur à trouver plus de défauts dans un composant [Kaner02]. Cependant les normes peuvent fournir quelques structures de référence, et fournir une base sur laquelle définir des solutions de tests.

8.2.1.3 Cohérence et Conflits

Certains standards peuvent manquer de cohérence avec d'autres, ou même fournir des définitions conflictuelles. C'est aux utilisateurs des différents standards de déterminer leurs utilités dans leurs environnements et contextes respectifs.

8.2.2 Standards Internationaux

8.2.2.1 ISO

L'ISO [www.iso.org] est l'International Standards Organization (récemment modifié en International Organization for Standardization) et est constitué de membres représentant, pour leur pays, le comité national le plus représentatif de la normalisation. Ce comité international a publié différentes normes utiles pour les testeurs logiciels, telles que:

- ISO 9126:1998, qui est maintenant divisée dans les normes et rapports techniques suivants:
  - ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model
  - ISO/IEC TR 9126-2:2003 Software engineering – Product quality – Part 2: External metrics
  - ISO/IEC TR 9126-3:2003 Software engineering – Product quality – Part 3: Internal metrics
  - ISO/IEC TR 9126-4:2003 Software engineering – Product quality – Part 4: Quality in use metrics
- ISO 12207:1995/Amd 2:2004 Systems and Software Engineering – Software Lifecycle Processes
- ISO/IEC 155042-2:2003 Information technology – Process assessment – Part 2: Performing an assessment

Cette liste n'est pas exhaustive; d'autres normes ISO peuvent être applicables au contexte et à l'environnement du testeur.

8.2.2.2 IEEE

IEEE [www.ieee.org] est l'Institute of Electrical and Electronics Engineer, une organisation professionnelle basée aux États-Unis. Les représentants nationaux sont disponibles dans plus d'une centaine de pays. Cette organisation a publié plusieurs standards utiles aux testeurs logiciels comme:

[2] ISO 15504 est aussi connu comme SPICE, et dérivé du projet SPICE.

- IEEE 610:1991 IEEE standard computer dictionary. Une compilation des glossaires informatiques des normes IEEE
- IEEE 829:1998 IEEE standard for software test documentation
- IEEE 1028:1997 IEEE standard for software reviews
- IEEE 1044:1995 IEEE guide to classification for software anomalies

Cette liste n'est pas exhaustive; d'autres normes IEEE peuvent être applicables dans votre contexte ou votre environnement.

8.2.3 Standards Nationaux

Beaucoup de pays possèdent leurs propres normes spécifiques. Certaines sont applicables et/ou utiles pour le test logiciel. L'une d'elle, une norme Britannique - BS-7925-2:1998 « Software testing. Software component testing » - fournit des informations concernant beaucoup de techniques de tests décrites dans ce syllabus, incluant:

- Partitions d'équivalence
- Analyse des valeurs limites
- Tests de transition d'état
- Diagrammes de Causes-Effet
- Tests des instructions
- Tests des branches/décisions
- Tests des conditions
- Tests aléatoires

BS-7925-2 fournit aussi une description du processus de Test de composant.

8.2.4 Standards Spécifiques à des Domaines

Les normes sont aussi présentes dans différents domaines techniques. Quelques industries adoptent d'autres normes pour leur domaine technique spécifique. Ici sont aussi présents les aspects d'intérêt dans le test logiciel, qualité logicielle et développement logiciel.

8.2.4.1 Système Avioniques

RTCA DO-178B/ED 12B « Software Considerations in Airborne Systems and Equipment Certification » est applicable au logiciel utilisé dans les avions civils. Cette norme s'applique aussi au logiciel utilisé pour créer (ou vérifier) de tels logiciels utilisés dans les avions. Pour le logiciel avionique, l'administration fédérale de l'aviation des États-Unis (Federal Aviation Administration) et les autorités internationales de l'aviation (International Joint Aviation Authorities) prescrivent certains critères de couverture structurelle basés sur le niveau de criticité du logiciel testé:

Niveau de criticité	Impact potentiel de la panne	Couverture structurelle exigée
A Catastrophique	Prévenir la continuité de la sureté du vol et de l'atterrissage	Détermination des conditions, Décision, et Déclaration
B Hasardeux / Majeur-Sévère	Large réductions des marges de sécurité ou capacités fonctionnelles L'équipage n'est pas assuré de pouvoir réaliser ses tâches correctement et complètement Dommages sérieux ou fatals pour un petit nombre de passagers	Décision et Déclaration
C Majeur	Réductions significatives des marges de sécurité Augmentations significatives de la charge de travail de l'équipage Désagrément pour les passagers incluant des dommages	Déclaration
D Mineur	Réduction légère des marges de sécurité de l'avion ou de ses capacités fonctionnelles Augmentation légère de la charge de travail de l'équipage Quelques incommodités aux passagers	Aucune
E Pas d'effet	Pas d'impact sur les capacités de l'avion Pas d'augmentation de la charge de travail de l'équipage	Aucune

Le niveau approprié de couverture structurelle doit être réalisé selon le niveau de criticité du logiciel qui sera certifié pour l'utilisation dans l'aviation civil.

8.2.4.2 Industrie Spatiale

Quelques industries adoptent d'autres normes pour leur domaine spécifique. C'est le cas pour l'industrie spatiale avec le ECSS (European Cooperation on Space Standardization) [www.ecss.org]. Selon la criticité du logiciel, l'ECSS conseille les méthodes et techniques qui sont en accord avec les Syllabus Fondation et Avancé de l'ISTQB® incluant:

- SFMECA – Software Failure Modes, Effects and Criticality Analysis (AMDEC : Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité)
- SFTA – Software Fault Tree Analysis (Analyse par arbre de panne)
- HSIA – Hardware Software Interaction Analysis (Analyse des interactions matérielles/logicielles)
- SCCFA – Software Common Cause Failure Analysis (Analyse des causes communes de défaillance logicielle)

8.2.4.3 Administration Agro-Alimentaire et Médicale

- Pour les systèmes médicaux concernés par le Titre 21 CFR Part 820, la Food and Drug Administration (FDA) des États-Unis conseille certaines techniques de tests fonctionnels et structurels.

La FDA conseille aussi les principes et stratégies de tests qui sont en accord avec les Syllabus Fondation et Avancé de l'ISTQB®.

8.2.5 Autres Normes & Standards

Un grand nombre de normes et standards sont disponibles dans les différentes industries. Certaines sont des adaptations à des domaines ou industries spécifiques, d'autres sont applicables pour des tâches spécifiques ou fournissent des explications sur la façon d'appliquer une norme.

C'est au testeur d'être au courant des différentes normes (y compris les normes internes, bonnes pratiques, etc.) qui sont applicables dans son domaine, industrie ou contexte. Parfois les normes applicables sont spécifiées, avec une applicabilité hiérarchique pour les contrats spécifiques. C'est au responsable de tests d'être au courant des normes qui doivent être respectées, et de s'assurer que la conformité adéquate est maintenue.

8.3 Amélioration des Processus de Test

De même que le test est utilisé pour améliorer le logiciel, les processus de qualité logicielle sont sélectionnés et utilisés pour améliorer le processus de développement logiciel (et les livrables logiciels qui en découlent). L'amélioration du processus peut aussi être appliquée aux processus de tests. Différentes manières et méthodes sont disponibles pour améliorer le test de logiciel ou de systèmes contenant du logiciel. Ces méthodes ont pour but l'amélioration du processus (et donc des livrables) en fournissant des directives et domaines de progression.

Le test compte souvent pour une part majeure des coûts du projet total. Cependant une attention limitée est donnée au processus de test dans divers modèles d'amélioration du processus logiciel, comme le CMMI (voir ci-dessous pour les détails).

Les modèles d'amélioration des tests comme le Test Maturity Model (TMM), le Systematic Test and Evaluation Process (STEP), le Critical Testing Processes (CTP) et le Test Process Improvement (TPI) ont été développés pour couvrir ce manque. TPI et TMM fournissent des métriques transverses à l'organisation pouvant être utilisées pour faire des comparaisons. Il y a beaucoup de modèles d'amélioration disponibles dans l'industrie de nos jours. Outre ceux couverts dans ce chapitre, Test Organization Maturity (TOM), Test Improvement Model (TIM) et Software Quality Rank (SQR) peuvent également être pris en compte. Il y a aussi un grand nombre de modèles locaux qui sont utilisés aujourd'hui. Les professionnels du test devront rechercher tous les modèles disponibles pour déterminer celui qui convient le mieux à leur situation.

Les modèles présentés dans ce chapitre n'ont pas objectif d'être une recommandation d'usage mais sont montrés ici pour fournir une vue représentative de comment ces modèles fonctionnent et ce qu'ils incluent.

8.3.1 Introduction sur l'Amélioration des Processus

L'amélioration des processus est pertinente pour les processus de développement logiciel aussi bien que pour le processus de test. Apprendre de ses propres erreurs permet d'améliorer les processus utilisés par l'organisation pour le développement et le test logiciel. Le cycle d'amélioration de Deming: Planifier, Faire, Vérifier, Jouer, a été utilisé pendant des décennies, et reste pertinent quand les testeurs ont besoin d'améliorer le processus courant.

Un principe de l'amélioration du processus est de penser que la qualité d'un système est hautement influencée par la qualité des processus utilisés pour développer le logiciel. Une meilleure qualité dans l'industrie du logiciel réduit le besoin en ressources pour maintenir le logiciel et donc donne plus de temps pour créer dans le futur davantage de solutions et de meilleure qualité.

Les modèles de processus fournissent un point de départ pour l'amélioration, en mesurant la maturité des processus de l'organisation avec le modèle. Le modèle délivre également une architecture pour améliorer les processus des organisations en se basant sur les résultats et les évaluations.

Une évaluation de processus conduit à une Détermination de l'Aptitude du Processus, qui motive son amélioration. Ceci peut impliquer une nouvelle évaluation du processus, plus tard, pour mesurer les effets de l'amélioration.

8.3.2 Types de Processus d'Amélioration

Il y a deux types de modèles: les modèles de référence de processus et les modèles de référence de contenu.

- Le modèle de référence de processus est utilisé comme cadre quand une évaluation est faite, afin d'évaluer l'aptitude de l'organisation comparée avec le modèle, et d'évaluer l'organisation dans le cadre.
- Le modèle de référence de contenu est utilisé pour améliorer le processus une fois que l'évaluation est faite.

Quelques modèles peuvent être construit avec les deux modèles alors que d'autres n'en auront qu'un.

8.4 Amélioration des Processus de Test

L'industrie informatique a commencé à travailler avec les modèles d'amélioration des processus de tests quand elle a cherché à atteindre un haut niveau de maturité et de professionnalisme. Les modèles de normes industrielles aident à développer des indicateurs et métriques interprofessionnels qui peuvent être utilisés pour comparaison. Les modèles étagés, comme TMMi et CMMI fournissent des normes permettant de comparer différentes entreprises et organisations. Les modèles continus permettent à une organisation d'aborder les problématiques qui lui sont prioritaires avec plus de liberté dans l'ordre d'exécution. Le besoin d'amélioration du processus dans l'industrie du test a engendré la création de plusieurs normes comme STEP, TMMi, TPI et CTP. Celles-ci sont abordées dans la suite de ce chapitre.

Ces quatre modèles d'évaluation du processus de test permettent à une organisation de déterminer à quel niveau se situe son processus de test courant. Une fois l'évaluation réalisée, TMMi et TPI fournissent un programme prescriptif pour améliorer le processus de test. STEP et CTP fournissent plutôt à l'organisation un moyen de déterminer d'où viendra son plus grand retour sur investissement et lui laisse le soin de planifier les actions appropriées.

Une fois qu'il a été convenu que le processus de test doit être revu et amélioré, les étapes du processus à suivre seront:

- **I**nitier
- **M**esurer
- **P**rioriser et Planifier
- Définir et **R**edéfinir
- Faire f**O**nctionner
- **V**alider
- **É**voluer

Initier

Dans cette activité la confirmation des parties prenantes, des objectifs, des buts, de l'étendue et de la couverture du processus d'amélioration est acceptée. Le choix du modèle de processus sur lequel les améliorations seront identifiées est aussi réalisé durant cette activité. Le modèle pourra être sélectionné parmi ceux publiés ou défini individuellement.

Avant que le processus d'amélioration débute, les critères de succès devront être définis et une méthode de mesure devra être mise en place sur l'ensemble de l'activité d'amélioration.

Mesurer

La méthode de mesure choisie doit permettre de créer une liste d'améliorations possibles du processus.

Prioriser & Planifier

La liste des améliorations possibles est classée par priorités. L'ordre pourra être basé sur le retour sur investissements, les risques, l'alignement sur la stratégie de l'organisation, les avantages mesurables quantitatifs ou qualitatifs.

Une fois l'ordre de priorité établi, un plan pour la livraison des améliorations sera développé et réalisé.

Définir & Redéfinir

Sur la base des exigences d'amélioration du processus, de nouveaux processus seront définis là où c'est nécessaire, des processus existants pourront être mis à jour et leur déploiement sera préparé.

Faire fonctionner

Une fois développées, les améliorations sont déployées. Cela pourra inclure toute formation ou surveillance nécessaire, le pilotage des processus, et, finalement leur déploiement complet.

Valider

Après avoir déployé les améliorations de processus, il est primordial de vérifier que tous les bénéfices escomptés avant la mise en œuvre des améliorations sont obtenus, par exemple la réalisation d'un bénéfice Il est aussi important que tout critère de succès de l'activité de processus d'améliorations ait été satisfait.

Évoluer

Selon le modèle de processus utilisé, cette étape du processus est menée quand la surveillance du prochain niveau de maturité débute et qu'une décision est prise soit pour recommencer le processus d'amélioration, soit pour arrêter l'activité.

L'utilisation de modèles d'évaluation est une méthode courante qui assure une approche normalisée pour améliorer les processus de tests utilisant des pratiques éprouvées et reconnues. Néanmoins, l'amélioration du processus de test peut aussi être réalisée sans modèle en utilisant, par exemple, des approches analytiques et des réunions rétrospectives.

8.5 Améliorer les Processus de Test avec TMM

Le Test Maturity Model est composé de cinq niveaux et a la vocation de compléter le CMM. Chaque niveau contient des domaines de processus qui doivent être complètement accomplis avant que l'organisation ne puisse avancer au prochain niveau (représentation étagée). TMM fournit non seulement un modèle de référence du processus mais aussi un modèle de référence de son contenu.

Les niveaux de TMM sont :

Niveau 1: Initial

Le niveau initial représente un état où le processus de test n'est ni documenté ni structuré. Les tests sont typiquement développés de façon ad hoc après le codage, et le test est vu comme le débogage. Le test est perçu comme une activité destinée à prouver que le logiciel fonctionne.

Niveau 2: Définition

Le second niveau peut être atteint en mettant en place une politique de tests et des objectifs, en introduisant un processus de planification des tests, et en mettant en œuvre des techniques et méthodes de tests élémentaires.

Niveau 3: Intégration

Le troisième niveau est atteint quand un processus de tests est intégré dans le cycle de vie de développement logiciel, et documenté en normes, procédures, et méthodes formalisées. Il y aura une fonction de test logiciel distincte pouvant être contrôlée et surveillée.

Niveau 4: Gestion & Mesure

Le niveau quatre est réalisé lorsque le processus peut être mesuré efficacement, contrôlé, et adapté aux projets spécifiques.

Niveau 5: Optimisation

Le niveau final représente un état de maturité du processus de tests, où les données du processus de tests peuvent être utilisées pour aider à prévenir des défauts, et où l'on se concentre sur l'optimisation du processus en place.

Pour atteindre un niveau particulier un ensemble prédéfini d'objectifs de maturité et sous-objectifs doit être réalisé. Ces objectifs sont définis en termes d'activités, tâches et responsabilités et évalués selon des «vues» spécifiées pour responsable, développeur/testeur et client/utilisateur. Pour plus d'informations sur TMM, voir [Burnstein03].

La fondation TMMi [voir www.tmmifoundation.org pour les détails] a définit le successeur de TMM: TMMi. TMMi est un modèle détaillé pour l'amélioration du processus de test. Il se base sur TMM développé par l'Illinois Institute of Technology et sur les expériences pratiques d'utilisation de TMM et est positionné comme étant complémentaire au CMMI.

L'architecture de TMMi est largement basée sur l'architecture de CMMI (par exemple, domaines de processus, objectifs globaux, pratiques génériques, objectifs spécifiques, pratiques spécifiques).

8.6 Améliorer les Processus de Test avec TPI

TPI (Test Process Improvement) utilise une représentation continue plutôt que la représentation étagée de TMM.

Le plan d'optimisation du processus de test comme souligné dans [Koomen99] implique un ensemble de domaines clés qui sont placés dans les quatre pierres angulaires du Cycle de vie, Organisation, Infrastructure et outils, et Techniques. Les domaines clés peuvent être évalués en niveaux entre A et D, A étant le plus bas. Il est aussi possible pour les domaines clés très immatures de ne pas atteindre le niveau initial A. Quelques domaines clés peuvent être estimés à A ou B (tel que Estimation et Planification) alors que d'autres (tel que Indicateurs) peuvent être évalués à A, B, C ou D.

Le niveau atteint pour un domaine clé donné est évalué en examinant les points de vérification définis dans le modèle TPI. Si, par exemple, tous les points de vérifications des niveaux A et B du domaine clé «documentation» sont répondus positivement, alors le niveau B est atteint dans ce domaine clé.

Le modèle TPI définit les dépendances entre les différents domaines clés et niveaux. Ces dépendances assurent que le processus de tests est développé équitablement. Il n'est pas possible, par exemple, de compléter le niveau A dans le domaine clé «métriques» sans aussi compléter le niveau A dans le domaine clé «documentation» (quel serait l'intérêt d'utiliser des métriques si elles ne sont pas documentées). L'utilisation des dépendances est optionnelle dans le modèle TPI.

TPI est tout d'abord un modèle de référence de processus.

Une Matrice de Maturité de Test (Test Maturity Matrix) est fournie, elle établit, pour chaque domaine clé, une correspondance entre les niveaux (A, B, C ou D) et un niveau global de maturité du processus de tests. Les différents niveaux globaux sont:

- Contrôlé
- Efficace
- Optimisé

Pendant une évaluation TPI, les métriques quantitatives et les entretiens qualitatifs sont utilisés pour établir le niveau de maturité du processus de test.

8.7 Améliorer les Processus de Test avec CTP

Comme décrit dans [Black02], le principe général de l'évaluation du modèle Critical Testing Process est que certains processus de test sont critiques. Ces processus critiques, s'ils sont bien menés, soutiendront avec succès les équipes de tests. Inversement, si ces activités sont mal exécutées, même des testeurs et responsables de tests talentueux ont peu de chance de réussir. Le modèle identifie douze processus de tests critiques.

CTP est tout d'abord un modèle de référence de contenu.

Le modèle de processus critiques est une approche sensible au contexte qui permet d'adapter le modèle incluant:

- Identification des défis spécifiques
- Reconnaissance des attributs des bons processus
- Sélection de l'ordre et de l'importance de la mise en œuvre des améliorations du processus

Le modèle de processus de tests critique est adaptable dans le contexte de tous les modèles de cycle de vie du développement logiciel.

Le processus d'améliorations utilisant CTP commence par l'évaluation des processus de tests existants. L'évaluation identifie quels processus sont solides et lesquels sont faibles, et fournit des recommandations priorisées pour une amélioration basée sur les besoins de l'organisation. Alors que les évaluations varient selon les contextes dans lesquels elles sont réalisées, les métriques quantitatives suivantes sont systématiquement examinées pendant une évaluation CTP:

- Pourcentage de détection des défauts
- Retour sur investissement des tests
- Couverture des exigences et couverture du risque
- Gestion des versions de test
- Taux de rejet des défauts

Les facteurs qualitatifs suivants sont généralement évalués pendant une évaluation CTP:

- Efficacité et rôle de l'équipe de tests
- Utilité du plan de tests
- Compétences de l'équipe de tests en tests, connaissance des domaines, et technologie
- Utilité du rapport de défauts
- Utilité du rapport des résultats de tests
- Utilité et équilibre de la gestion du changement

Une fois qu'une évaluation a identifié les domaines faibles, des plans d'amélioration sont mis en place. Les plans d'améliorations génériques sont fournis par le modèle pour chacun des processus de tests critiques, mais l'équipe d'évaluation doit les adapter fortement.

8.8 Améliorer les Processus de Test avec STEP

STEP (Systematic Test and Evaluation Process), comme CTP et à la différence de TMMi et TPI, n'exige pas que les améliorations apparaissent dans un ordre spécifique.

Les principes généraux de la méthodologie incluent:

- Une stratégie de test basée sur les exigences
- Les tests commencent au début du cycle de vie
- Les tests sont utilisés comme exigences et modèles d'usage
- La conception du testware conduit la conception logicielle
- Les défauts sont détectés au plus tôt ou évités complètement
- Les défauts sont systématiquement analysés
- Les testeurs et développeurs travaillent ensemble

STEP est tout d'abord un modèle de référence de contenu.

La méthodologie STEP est basée sur l'idée que le test est une activité du cycle de vie qui débute pendant la formulation des exigences et continue jusqu'à la retraite du système. La méthodologie STEP insiste sur «tester puis coder» en utilisant une stratégie de test basée sur les exigences pour assurer que la création des cas de test au plus tôt valide la spécification des exigences avant de concevoir ou de coder. La méthodologie identifie et se concentre sur l'amélioration de trois phases majeures du test:

- Planification
- Acquisition
- Mesure

Durant une évaluation STEP, les indicateurs quantitatifs et les entretiens qualitatifs sont utilisés. Les métriques quantitatives incluent :

- Statut des tests au cours du temps
- Exigences des tests ou couverture du risque
- Évolution des défauts selon la détection, la sévérité, et leur regroupement
- Densité des défauts
- Efficacité de la suppression des défauts
- Pourcentage de détection des défauts
- Introduction, détection, et phases de suppression des défauts
- Coût des tests en termes de temps, effort, et budget

Les facteurs quantitatifs incluent :

- Utilisation du processus de tests défini
- Satisfaction client.

Dans quelques cas le modèle d'évaluation STEP est fusionné avec le modèle de maturité TPI.

8.9 Capability Maturity Model Integration, CMMI

Le CMMI peut être mis en œuvre via deux approches ou représentations: la représentation étagée ou la représentation continue. Dans la représentation étagée il y a cinq «niveaux de maturité», chaque niveau est basé sur tous les domaines de processus établis dans les niveaux précédents. Dans la représentation continue l'organisation a la possibilité de concentrer ses efforts d'amélioration sur des domaines de processus particuliers en fonction des besoins sans regarder les niveaux prédécesseurs.

La représentation étagée est essentiellement inclue dans le CMMI pour assurer la normalisation avec CMM, alors que la représentation continue est généralement considérée plus flexible.

Dans CMMI les domaines de processus Validation et Vérification référencent les processus de tests statiques et de tests dynamiques.

9. Outils de Test & Automatisation

Termes

Outil de débogage, outil d'analyse dynamique, émulateur, outil de génération de fautes, outil de tests d'hyperliens, tests déterminés par les mots-clés, outil de test de performance, simulateur, analyseur statique, outil d'exécution des tests, outil de gestion des tests, oracle de test

9.1 Introduction

Cette section étend le syllabus Niveau Fondation en couvrant d'abord un certain nombre de concepts généraux puis en détaillant quelques outils spécifiques.

Même si certains concepts couverts sont plus pertinents pour les gestionnaires de test, les analystes de test ou les analystes techniques de test, une compréhension de base de ces concepts est exigée par tous les professionnels du test. Cette compréhension de base pourra alors être élargie selon le contexte.

Les outils peuvent être regroupés de différentes façons, incluant le regroupement en fonction de leur utilisateur réel, comme les gestionnaires de test, les analystes de test ou les analystes techniques de test. Ce regroupement, qui est reflété dans les modules de ce syllabus, est utilisé dans les autres sections de ce chapitre. En général, les outils discutés dans ces sections sont principalement pertinents pour un module spécifique, bien que certains outils (par exemple, les outils de gestion de test) puissent avoir une pertinence plus large. Dans ce cas, des exemples des applications de ces outils dans des contextes spécifiques seront donnés par le formateur.

9.2 Concepts des Outils de Test

Les outils de tests peuvent grandement améliorer l'efficacité et la précision de l'effort de test, mais seulement si les bons outils sont utilisés de la bonne manière. Les outils de tests doivent être gérés au même titre que les autres éléments d'une organisation de tests bien dirigée. L'automatisation d'un test est souvent associée à son exécution, mais la plupart des tâches manuelles peut être automatisée de différentes manières, ce qui veut dire que la plupart des tâches liées à un test pourrait être automatisée à un certain degré si les bons outils étaient présents.

N'importe quelle version d'un outil de test, script de test ou session de test devrait être, comme n'importe quelle base de test, gérée en configuration et liée à la version du logiciel pour laquelle elle a été utilisée. N'importe quel outil de tests est une partie importante du testware et devrait être géré convenablement :

- En créant une architecture avant de créer l'outil de test
- En assurant une correcte gestion de la configuration des scripts et des versions des outils, des patches etc. incluant la documentation des versions
- En créant et en maintenant des librairies (réutilisabilité de concepts similaires dans les cas de test), en documentant l'implémentation des outils de tests (par exemple, des processus décrivant comment l'outil est utilisé dans l'organisation)
- En anticipant pour le futur en structurant les cas de test pour des prochains développements, par exemple en les rendant plus extensibles et maintenables.

9.2.1 Coûts, Bénéfices et Risques des Outils de Test et de l'Automatisation

Une analyse des coûts-bénéfices devrait toujours être réalisée et montrer un retour sur investissement significatif. Les caractéristiques principales d'une analyse coûts-bénéfices devraient, en comparant le coût actuel des tests manuels (non-outillés) par rapport à celui des tests outillés, le tout en termes de coûts (heures traduites en coût, coûts directs, coûts récurrents et non-récurrents), prendre en compte les postes de coût suivants :

- Coûts initiaux
  - Montée en compétence (courbe d'apprentissage de l'outil)

  - Évaluation (comparaisons d'outils) si appropriée
  - Intégration avec d'autres outils
  - Considération des coûts initiaux pour l'achat, l'adaptation ou le développement de l'outil
- Coûts récurrents
  - Coûts d'acquisition de l'outil (maintenance, coût des licences, coût du support, soutenance de niveaux de connaissance)
  - Portabilité
  - Disponibilité et dépendances (s'il en manque)
  - Évaluation continue du coût
  - Amélioration de la qualité, pour assurer une utilisation optimale des outils sélectionnés.

Les analyses de rentabilité basées sur des projets d'automatisation pilotes oublient souvent les coûts tels que la maintenance, les mises à jour et l'évolutivité des scripts de test quand le système change. La longévité d'un cas de test correspond au temps pendant lequel il restera valide sans avoir à être modifié. Le temps requis pour l'implémentation de la première version de scripts de test automatisés est souvent bien supérieur au temps nécessaire pour passer ces tests manuellement, mais il permettra de créer beaucoup de scripts de test similaires plus rapidement et d'étendre le nombre de bons cas de test dans le temps. De plus, des améliorations significatives du taux de couverture et de l'efficacité des tests seront constatées dans les futures utilisations des automates après la période d'implémentation. L'analyse de rentabilité de l'implémentation d'un outil doit être réalisée sur le long terme.

Pour un niveau de test donné, chaque cas de test doit être examiné pour voir s'il mérite d'être automatisé. Beaucoup de projets d'automatisation sont basés sur une implémentation de cas de test manuels aisément disponibles sans avoir revu le bénéfice réel d'automatiser chaque cas en particulier. Il est probable qu'un ensemble de cas de test donné (une suite) puisse contenir des tests manuels, semi-automatisés ou complètement automatisés.

En plus des sujets couverts dans le syllabus niveau Fondation, les aspects suivants devraient être pris en considération :

Bénéfices additionnels :

- Le temps d'exécution d'un test automatisé devient plus prévisible
- Les tests de régression et la validation des corrections d'anomalies sont plus rapides et sûrs dans le projet quand les cas de test sont automatisés
- L'utilisation d'outils automatisés peut améliorer le statut et la montée en compétence technique du testeur ou de l'équipe de test
- L'automatisation peut être développée en parallèle, en itératif ou en incrémental pour offrir de meilleurs tests de régression pour chaque livraison
- La couverture de certains types de test qui ne peuvent pas être couverts manuellement (par exemple : les tests de performance et de fiabilité)

Risques additionnels :

- Un test manuel incomplet ou incorrect pourra être automatisé tel quel
- Le matériel de test est difficile à maintenir et requiert de multiples changements quand le logiciel à tester est modifié
- La perte de l'implication directe des testeurs pendant l'exécution peut réduire la détection d'anomalies quand seuls des tests automatisés, scriptés, sont effectués.

9.2.2 Stratégies des Outils de Test

Les outils de test sont généralement partagés par plusieurs projets. Selon l'investissement et la durée du projet, un outil de test peut ne pas donner un bon retour sur investissement sur le projet, mais sur les versions suivantes du logiciel. Par exemple, comme la phase de maintenance donne souvent lieu à des tests intensifs (pour chaque correction une grande suite de tests de régression doit être exécutée), il peut être plus rentable d'automatiser un système en phase de maintenance, si le cycle de vie du système le permet d'un point de vue économique. Autre exemple : il est facile pour un humain de faire une erreur dans un test manuel (comme une faute de frappe), ainsi le bénéfice de l'automatisation de la fourniture de données en entrée et de la comparaison avec des données en sortie par un oracle peut être important (par exemple, comparer les résultats de tests aux résultats attendus).

Pour des entreprises qui utilisent et qui sont dépendantes de nombreux outils de tests (dans des phases différentes et pour des objectifs différents) une stratégie d'outillage des tests à long terme est souhaitable pour l'aide à la décision dans le choix des différentes versions d'un outil et de son support. Pour de plus grandes entreprises avec un domaine d'outillage intensif, il peut être souhaitable de fournir des directives générales pour l'acquisition d'un outil, des stratégies, des paradigmes d'outils ou les langages de scripts à utiliser.

9.2.3 Intégration & Échanges d'Informations entre Outils

Généralement plus d'un outil de tests est utilisé dans le processus de test (et de développement). Prenons l'exemple d'une entreprise qui utilise en même temps un outil d'analyse statique, un outil de gestion de test et de pilotage, un outil de gestion de la configuration, un outil de gestion des anomalies et un outil d'exécution des tests. Il est important d'étudier si les outils peuvent s'intégrer et échanger des informations les uns avec les autres de manière profitable. Par exemple, il est bénéfique que tous les statuts d'exécution des tests soient intégrés directement dans le système de gestion de tests pour offrir une mise à jour immédiate de l'avancement et une traçabilité directe entre les exigences et les cas de test associés. Stocker les scripts de test à la fois dans la base de données de gestion de test et dans le système de gestion de configuration demande plus d'effort et est sujet à plus d'erreurs. Si un testeur veut émettre un rapport d'incident au milieu de l'exécution d'un cas de test, l'outil de gestion des défauts et le système de gestion des tests doivent s'intégrer. Même si les outils d'analyse statique peuvent être séparés des autres outils, il est plus pratique qu'un outil puisse notifier des incidents, des alarmes et des remarques directement au système de gestion des tests.

Acheter une suite d'outils de test chez un même éditeur ne veut pas signifier que tous les outils fonctionneront ensemble pour autant, mais c'est une exigence raisonnable. Le coût de tous ces aspects liés à l'automatisation des échanges d'information devrait être considéré, par rapport à la prise de risque de s'en passer et de perdre l'information d'un travail purement manuel, en supposant que l'organisation ait le temps que cela va nécessiter.

De nouveaux concepts comme les environnements de développement intégrés, tel Eclipse, ont pour objectif de faciliter l'intégration et l'utilisation d'outils différents en offrant une interface commune pour les outils de développement et de test. Un revendeur d'outils peut devenir conforme à Eclipse en créant un plug-in pour le framework Eclipse qui lui donne la même apparence que n'importe quel autre outil. Cela avantage l'utilisateur. Remarque : même si les interfaces utilisateurs sont similaires, cela ne signifie pas automatiquement que les outils vont pouvoir s'intégrer ensemble et échanger des informations.

9.2.4 Langages d'Automatisation : Scripts, Langages de Script

Les scripts et les langages de scripts sont quelques fois utilisés pour mieux implémenter et étendre les conditions de tests et les cas de test. Par exemple pour les tests des applications Web, un script peut être utilisé pour contourner l'interface utilisateur et ainsi tester de manière plus adéquate l'API (interface de programmation de l'application) elle-même. Un autre exemple serait le cas de test d'une interface qui est automatisé pour tester toutes les combinaisons possibles d'entrées, ce qui serait infaisable avec des tests manuels.

Les capacités des langages de script varient beaucoup. Notez que les langages de script s'étendent des langages de programmation normaux à des notations spécifiques et normées, par exemple les signalisations pour des protocoles comme TTCN-3.

9.2.5 Concept d'Oracle de Test

Les oracles de test sont généralement utilisés pour déterminer des résultats attendus. Ainsi ils réalisent les mêmes fonctionnalités que les logiciels à tester et sont par conséquent rarement disponibles. Ils peuvent toutefois être utilisés dans des situations où un ancien système est en train d'être remplacé par un nouveau système avec les mêmes fonctionnalités, ainsi l'ancien système peut être utilisé comme oracle. Les oracles peuvent aussi être utilisés là où les performances sont un objectif pour le système à livrer. Un oracle peu performant peut être conçu ou utilisé pour générer des résultats attendus pour les tests fonctionnels du logiciel performant à livrer.

9.2.6 Déploiement d'un Outil de Test

Chaque outil de test est un logiciel à part entière et peut avoir des dépendances matérielles ou logicielles. Un outil devrait être documenté et testé lui-même en regard de s'il a été acheté tel quel, adapté ou développé en interne. Certains outils s'intègrent plus à l'environnement, d'autres outils fonctionnent mieux de manière indépendante.

Quand le système testé fonctionne sur un matériel propriétaire, un système d'exploitation, un logiciel embarqué ou utilise des configurations non standard, il peut être nécessaire de créer (développer) ou adapter un outil pour correspondre à l'environnement spécifique. Il est toujours conseillé de faire une étude des coûts et profits qui inclut l'implémentation initiale aussi bien que la maintenance à long terme.

Pendant le déploiement d'un outil d'automatisation des tests, il n'est pas conseillé d'automatiser les cas de test manuels tels quels, mais de redéfinir les cas de test pour une meilleure automatisation. Cela inclut le formatage des cas de test, la prise en compte de la réutilisabilité des motifs, l'élargissement des entrées en utilisant des variables à la place de valeurs codées en dur, et l'utilisation des avantages des outils de tests, qui ont la capacité de parcourir, répéter et changer leur séquence avec de meilleures possibilités d'analyse et de reporting. Pour beaucoup d'outils d'automatisation des tests, des compétences en développement sont nécessaires pour créer des programmes de test (scripts) et des suites de tests efficients et efficaces. Les grandes suites de tests sont généralement difficiles à mettre à jour et à gérer si elles n'ont pas été conçues avec attention. Une formation appropriée sur les outils de tests et les techniques de développement et de conception est fondamentale pour s'assurer de la prise en main complète des avantages des outils.

Même quand des tests manuels ont été automatisés, il est important d'exécuter manuellement les cas de test périodiquement pour conserver la connaissance du fonctionnement des tests et pour vérifier la justesse des opérations.

Quand un outil est utilisé et que le nombre de scripts de test augmente, il peut y avoir un besoin d'ajouter des fonctionnalités offertes par d'autres outils. Ceci n'est pas toujours possible puisque les outils n'ont pas tous des interfaces ouvertes et utilisent parfois des langages de scripts non pas standardisés mais propriétaires. Il est conseillé d'utiliser des outils qui ont des plug-ins pour ouvrir leur framework ou leur API (interface de programmation de l'application). Cela garantira une meilleure anticipation des développements des scripts de tests en testware.

Pour chaque outil de test, sans faire attention à la phase d'utilisation, considérer les caractéristiques listées ci-dessous. Ces caractéristiques peuvent être utilisées à la fois pour les évaluations des outils et pour leur conception. Dans chacun de ces domaines, un outil peut être faible ou puissant. Une liste comme celle-ci est utile pour comparer des outils similaires :

- L'analyse (compréhension des concepts, entrées, informations fournies manuellement ou automatiquement)
- Conception (manuelle, génération automatique)
- Sélection (manuelle, sélection automatique selon un panel de critères comme le taux de couverture)
- Exécution (manuelle, automatique, pilotage, redémarrage etc.)
- Évaluation (par exemple un oracle de tests) et présentation. Cela se réfère souvent aux fonctions de reporting et de journalisation (manuelles, automatiques comme par exemple comparatives, par rapport à un formulaire, standardisées, générées selon un critère).

9.2.7 Utilisation des Outils Open Source

Les outils utilisés dans des systèmes à sûreté critique doivent être certifiés pour se conformer aux buts attendus par rapport aux standards correspondants. Il n'est pas conseillé d'utiliser des outils open source dans des systèmes à sûreté critique, à moins qu'ils aient obtenus les niveaux de certification appropriés.

La qualité d'un logiciel open source dépend de son exposition, de son historique et de l'utilisation du logiciel en question et ne devrait pas être considéré comme étant plus (ou moins) précis qu'un outil commercial.

La qualité devrait toujours être évaluée pour n'importe quel outil de tests pour juger de sa justesse. Pour certains types d'outils, il est facile de confondre une évaluation positive avec une mauvaise exécution de

l'outil (par exemple s'il omet des exécutions et ne le fait pas apparaître dans les rapports). Une attention particulière doit être portée sur le prix des licences. Le partage des modifications de code pour améliorer un outil peut aussi être escompté.

9.2.8 Développer son Propre Outil de Test

Beaucoup d'outils de tests sont développés sans prendre en compte le besoin d'un testeur ou d'un concepteur d'accélérer son propre travail. D'autres raisons pour développer ses propres outils de tests sont le manque d'outils appropriés ou l'utilisation d'environnements de tests ou de matériels propriétaires. Ces outils sont souvent efficaces pour accomplir les tâches pour lesquelles ils sont prévus, mais sont souvent dépendant des personnes qui les ont conçus. Ces outils devraient être documentés de façon à pouvoir être maintenus par d'autres. Il est aussi important de revoir les buts, les objectifs, les avantages et inconvénients éventuels avant de les déployer dans toute l'organisation. Ces outils doivent souvent prendre en compte de nouvelles exigences et être étendus bien au-delà de leur utilisation initiale, ce qui peut être un inconvénient.

9.2.9 Classification des Outils de Tests

En plus des outils classés selon les activités qu'ils assistent (concept utilisé dans le niveau Fondation), il y a d'autres types de classification d'outils, incluant :

- Les outils regroupés selon le niveau de tests qu'ils réalisent (composant, intégration, système, acceptation)
- Les outils regroupés selon les fautes qu'ils traitent et le support qu'ils effectuent
- Les outils basés sur une approche de test ou des techniques de test (voir les explications ci-dessous)
- Les outils pour tester différents objectifs, comme par exemple les mesures, les pilotes, la journalisation, les comparaisons
- Les outils spécifiques à certains domaines, comme par exemple la simulation et la signalisation de trafic, les réseaux, les protocoles, les transactions, les écrans tv, les systèmes experts
- Les outils de soutien des différents domaines de test, comme par exemple les données en entrée, les environnements, la configuration, ou d'autres domaines conceptuels
- Les outils basés sur la façon dont ils seront appliqués : sur étagère, framework (pour l'adaptation), l'adaptation de plug-ins (i.e. Eclipse), les standards ou les suites de tests de certification, les développements internes d'outils

Et enfin, les outils peuvent être regroupés selon leurs utilisateurs, comme les gestionnaires de test, les analystes de test et les analystes techniques de test. Ce regroupement, qui est reflété dans les modules de ce syllabus, est utilisé dans les autres sections de ce chapitre. Le Syllabus Niveau Fondation inclut une section qui concerne les outils. Les sections qui suivent sont complémentaires.

9.3 Catégories d'Outils de Tests

Les objectifs de cette section sont les suivants :

- Fournir des informations supplémentaires sur les catégories d'outils déjà introduites dans la section 6 du Syllabus Niveau Fondation, comme les outils de gestion de tests, les outils d'exécution de tests et les outils de tests de performance
- Introduire de nouvelles catégories d'outils

Veuillez vous référer au chapitre 6 du Syllabus niveau Fondation de l'STQB® pour des informations générales concernant les autres catégories d'outils qui ne sont pas inclues dans ce chapitre

9.3.1 Outils de Gestion de Tests

Veuillez vous référer au chapitre 6.1.2 du Syllabus Niveau Fondation pour des informations générales sur les outils de gestion de tests.

Les outils de gestion de tests devraient permettre de tracer les informations suivantes

- La traçabilité des artefacts de tests
- La capture des données d'environnement de tests dans le cas des environnements complexes

- Des données concernant l'exécution concurrente de suites de tests sur des environnements de tests différents dans la même session de test sur plusieurs sites (organisations de tests)
- Des métriques telles que :
  - Les conditions de test
  - Les cas de test
  - Les temps d'exécution (comme par exemple ceux d'une suite de tests, d'un cas de test, d'une suite de régression) et d'autres mesures de temps, incluant les moyennes qui peuvent contribuer à la prise de décision
  - Le nombre de cas de test, d'artefacts de test et d'environnements de tests
  - Le taux de réussites/échecs
  - Le nombre de cas de test en attente (et les raisons pour lesquelles ils n'ont pas été exécutés)
  - Les tendances
  - Les exigences
  - Les relations et la traçabilité entre les artefacts de test
- Les concepts soutenus par les outils de gestion de tests, comme :
  - L'organisation des artefacts de test, des répertoires et des conducteurs de cas de test
  - Les conditions et les environnements de tests
  - Les suites de régression, les sessions de test
  - La journalisation et la gestion des erreurs
  - Le redémarrage de l'environnement (et la réinitialisation)
  - Les métriques de tests sur les artefacts de tests (documentation de test) pour documenter l'avancement des tests

Les outils de gestion de tests sont utilisés par les gestionnaires de tests, les analystes de test et les analystes techniques de test.

9.3.2 Outils d'Exécution des Tests

Veuillez vous référer au chapitre 6.1.5 du Syllabus niveau Fondation pour des informations générales sur les outils d'exécution de tests.

Les outils d'exécution de tests sont principalement utilisés par les analystes de test et les analystes techniques de test à tous les niveaux de tests, pour exécuter et vérifier le résultat des tests. L'utilisation d'un outil d'exécution de tests a typiquement pour objectif au moins un des points suivants :

- Réduire les coûts (effort et temps)
- Exécuter plus de tests
- Rendre les tests plus répétables.

Les outils d'exécution de tests sont le plus souvent utilisés pour automatiser les tests de régression.

Les outils d'exécution de tests fonctionnent en exécutant un ensemble d'instructions écrites dans un langage de programmation, souvent appelé langage de script. Les instructions de l'outil sont souvent à un niveau très détaillé qui spécifie l'appui sur un bouton, la frappe d'une touche, et les mouvements de souris. Cela rend les scripts de test très sensibles aux changements de l'application à tester, particulièrement aux changements de l'interface utilisateur graphique.

Le point de départ pour un script peut être l'enregistrement (fait en capture rejeu) ou le développement ou la modification de scripts à partir de scripts existants, de modèles ou de mots-clés. Un script est un programme et fonctionne exactement comme n'importe quel autre logiciel. Capturer (ou enregistrer) peut être utilisé pour enregistrer une piste de vérification pour les tests non systématiques. La plupart des outils d'exécution de tests incluent un comparateur pour comparer un résultat de test à un résultat attendu. La tendance au niveau du test (comme au niveau du développement) est de passer d'instructions bas-niveau détaillées à des langages plus « haut-niveau », là encore des librairies, macros et sous-programmes sont utilisés. Une série d'instructions est étiquetée par un nom – dans les tests appelés tests dirigés par les mots-clés ou tests dirigés par mots-actions. L'avantage principal est la séparation des instructions et des données. C'est le même concept que l'utilisation de templates pour scripter afin de réduire les efforts.

La principale raison pour laquelle certains outils de test échouent provient des faibles compétences en programmation et dans la compréhension qu'un outil de test résout seulement une partie des problèmes dans l'automatisation de l'exécution de tests. Il est important de noter que toute automatisation d'exécution

de tests nécessite de la gestion, de la charge, des compétences et de l'attention, par exemple l'architecture de test et la gestion de configuration. Cela veut aussi dire que les scripts de test peuvent comporter des anomalies. L'utilisation d'une architecture de testware permet d'être indépendant vis-à-vis d'un éditeur spécifique d'outils. Lorsqu'un outil est fourni on a tendance à penser que les standards de l'outil doivent être suivis, par exemple la structure et les conventions de nommage des scripts. Cependant la mise en place de l'automatisation de tests peut constituer une passerelle entre vos propres bonnes pratiques d'organisation de tests et celles devant être mises en œuvre car nécessaires à l'outil.

9.3.3 Outils de Débogage et de Dépannage

Le dépannage peut employer des outils pour cerner le domaine où se produit une faute. Cela peut également être nécessaire pour un système où l'identification de la faute qui a causée la défaillance constatée n'est pas évidente. Les outils de dépannage incluent des traces et des environnements simulés utilisés pour interagir avec le logiciel ou extraire des informations du système pour cerner la faute.

Les développeurs reproduisent les fautes et analysent l'état des programmes en utilisant des outils de débogage et de traces. Les débogueurs et les traces permettent aux développeurs :

- D'exécuter les programmes ligne par ligne
- D'arrêter le programme sur n'importe quelle instruction
- De régler et contrôler les variables du programme.

Il est important de souligner que le débogage et les outils de débogage sont en lien avec le test mais ne sont pas du test (ni des outils de tests). Les outils de débogage et de traces peuvent être utilisés dans un objectif de dépannage par des testeurs pour mieux cerner l'origine d'une faute et pour aider à identifier le destinataire d'un rapport d'anomalie. Les outils de débogage, de traces et de dépannage sont principalement utilisés par les analystes techniques de test.

9.3.4 Outils de Génération de Défauts & d'Injection de Défauts

La génération de fautes et l'injection de fautes sont deux techniques différentes utilisables dans le test. La génération de fautes utilisera un outil similaire à un compilateur pour créer des types de fautes dans le code, unique ou limitées, et ce de manière systématique. Ces outils sont aussi souvent utilisés conjointement avec la technique de test de mutation et sont parfois appelés outils de test de mutation.

L'injection de fautes a pour but de modifier les interfaces réelles pour tester les composants (quand le code source n'est pas disponible) mais peut aussi être utilisée pour (ré-)injecter une faute particulière 1) pour vérifier si le logiciel peut y faire face (tolérance aux fautes) ou 2) pour déterminer si un test parmi une suite de tests peut trouver la faute délibérément insérée. Les outils de génération de fautes et les outils d'injection de fautes sont principalement utilisés au niveau du code par des analystes techniques de test mais il est aussi possible pour un analyste de test de manipuler des données dans une base de données ou d'injecter des fautes dans un flux de données pour tester le comportement d'un système.

9.3.5 Outils de Simulation & Emulation

Les simulateurs sont utilisés dans le cadre de tests où le code ou d'autres systèmes sont indisponibles, chers ou impraticables (comme par exemple pour tester des logiciels en rapport avec des fusions nucléaires). Certains simulateurs et outils de harnais de tests peuvent aussi supporter ou imiter un comportement de faute, pour vérifier un état ou un scénario d'erreur. Le principal risque à l'utilisation de ces outils est que les erreurs liées aux ressources comme les problématiques de temps peuvent ne pas être trouvées, ce qui est très important pour certains types de systèmes.

Les émulateurs appartiennent à une catégorie particulière des simulateurs et sont des logiciels développés pour imiter le matériel. L'avantage à utiliser un émulateur est que des tests plus élaborés peuvent être réalisés. En particulier le fait de pouvoir recréer des traces, des débogages et des causes de dépendances au temps, ce qui peut être impossible sur un système réel. Les émulateurs sont couteux à la création, mais l'avantage de l'analyse pour des systèmes qui peuvent tourner « au ralenti » n'est pas valable pour des systèmes complexes, en parallèle ou dépendants du temps.

Les analystes de test et les analystes techniques de test, moyennant le type d'émulation exigé, utilisent ces outils.

9.3.6 Outils d'Analyse Statique et Dynamique

Veuillez vous référer au chapitres 6.1.3 « Outils d'aide aux tests statiques » et 6.1.6 « Outils de support des performances et de surveillance » du syllabus niveau Fondation de l'ISTQB® pour des informations générales sur les tests statiques et les outils d'analyse dynamique.

9.3.6.1 Outils d'Analyse Statique

Les outils d'analyse statique peuvent être utilisés à tout moment dans le cycle de vie du logiciel ainsi qu'à tous les niveaux/phases de développement du logiciel, moyennant les mesures offertes par les outils.

Les outils d'analyse statiques informent de leurs découvertes en termes d'alertes. Les alertes qui sont injustifiées sont appelées des faux positifs. Les vrais positifs sont des fautes réelles qui peuvent conduire à des défaillances pendant l'exécution. Il peut être difficile et consommateur de temps de discerner les faux des vrais positifs puisque cela nécessite du dépannage. Les outils d'analyse statique plus récents peuvent utiliser l'information pendant la liaison dynamique lors de la compilation et sont donc plus puissants pour découvrir de vraies fautes et moins de faux positifs. Les analystes techniques de test utilisent ces outils.

9.3.6.2 Outils d'Analyse Dynamique

Les outils d'analyse dynamique offrent des informations d'exécution sur l'état du logiciel en cours d'utilisation. Ces outils sont principalement utilisés pour identifier des pointeurs non assignés, pour vérifier des pointeurs arithmétiques, surveiller les allocations, utilisations et désallocations de mémoire pour pointer des fuites de mémoire ou mettre en valeur d'autres erreurs difficiles à trouver « statiquement ». Les outils d'analyse de la mémoire devraient être réutilisés à plus d'un niveau dans des systèmes complexes importants puisque les problèmes de mémoire sont créés dynamiquement. Veuillez noter que les différents outils de tests commerciaux peuvent être implémentés différemment et ainsi cibler et informer sur différents types de problèmes de mémoire ou de ressource (pile). La conclusion est que des outils d'analyse de la mémoire différents peuvent identifier différents problèmes. Les outils d'analyse de la mémoire sont particulièrement utiles pour certains langages de programmation (C, C++) où la gestion de la mémoire est laissée au développeur. Les analystes techniques de test utilisent ces outils.

9.3.7 Automatisation des Tests par les Mots-Clés

Les mots-clés (parfois notés « mots-actions ») sont principalement (mais pas exclusivement) utilisés pour représenter des interactions haut-niveau avec le système (comme par exemple un « ordre d'annulation »). Chaque mot-clé est typiquement utilisé pour représenter un nombre d'instructions détaillées avec le système à tester. Les séquences de mots-clés (incluant les données de tests pertinentes) sont utilisées pour spécifier les cas de test. [Buwalda01]

Lors de l'automatisation des tests, un mot-clé est implémenté dans un ou plusieurs scripts de test. Les outils lisent les cas de test écrits avec des mots-clés et appellent les scripts de tests appropriés qui les implémentent. Les scripts sont implémentés de façon très modulaire pour faciliter la liaison avec les mots-clés. Des compétences en développement sont requises pour implémenter ces scripts modulaires.

Les principaux avantages de l'automatisation des tests déterminés par les mots-clés sont :

- Les mots-clés peuvent être définis par les experts d'une application ou d'un domaine métier en particulier. Ce qui peut rendre la conception des tests plus efficace.
- Une personne inexpérimentée dans un domaine peut profiter de l'exécution automatique du cas de test (une fois que les mots-clés ont été implémentés dans un script).
- Les cas de test écrits en utilisant des mots-clés sont plus facilement maintenables car ils sont moins susceptibles d'évoluer si un détail dans le logiciel à tester est modifié.
- Les spécifications des cas de test sont indépendantes de leur implémentation. Les mots-clés peuvent être implémentés en utilisant une grande variété d'outils et de langages de scripting.

L'automatisation des tests basée sur les mots-clés est principalement utilisée par des experts de domaines et des analystes de test.

9.3.8 Outils de Test de Performance

Veuillez vous référer au chapitre 6.1.6 « Outils de support des performances et de surveillance » du Syllabus Niveau Fondation de l'ISTQB® pour des informations générales sur les outils de test de performances.

Les outils de test de performance ont deux principales fonctionnalités:

- La génération de charge
- La mesure et l'analyse des réponses d'un système à une charge donnée

La génération de charge est utilisée pour implémenter un profil opérationnel prédéfini (voir section 5.3.3) tel qu'un script. Le script peut être capturé initialement pour un unique utilisateur (possiblement en utilisant un outil de capture/rejeu) et ensuite être implémenté selon le profil opérationnel spécifié en utilisant l'outil de test de performance. Cette implémentation peut prendre en compte la variation des données par transaction (ou ensemble de transactions).

Les outils de performance génèrent une charge en simulant un grand nombre d'utilisateurs (utilisateurs « virtuels ») avec des volumes spécifiques de données. En comparaison avec les outils de capture/rejeu, beaucoup de script de test de performance reproduisent les interactions de l'utilisateur avec le système au niveau des protocoles de communication et non pas en simulant les interactions des utilisateurs via l'interface utilisateur graphique. Peu de générateurs de charge peuvent générer la charge en pilotant l'application via l'interface graphique.

Un large panel de mesures sont prises par un outil de test de performances pour permettre l'analyse pendant ou après l'exécution du test. Des métriques types sont prises et des rapports sont fournis en incluant :

- Le nombre d'utilisateurs simulés
- Le nombre et le type de transactions générées par les utilisateurs simulés
- Les temps de réponse de transactions particulières faites par les utilisateurs
- Les rapports basés sur les registres de tests et les graphiques des temps de réponse en fonction de la charge

On peut citer comme facteurs significatifs à considérer dans l'implémentation des outils de test de performance :

- Le matériel et la bande passante exigés pour générer la charge
- La compatibilité de l'outil avec les protocoles de communication utilisés par les systèmes à tester
- La flexibilité de l'outil pour faciliter l'implémentation de différents profils opérationnels
- Les capacités de surveillance, d'analyse et de reporting

Les outils de test de performance sont en général acquis du fait de l'effort exigé pour les développer. Cependant il peut être approprié de développer un outil de test de performance spécifique si les restrictions techniques empêchent l'utilisation d'un produit, ou si les profils de charge et les compétences à mettre en œuvre sont simples. Les outils de test de performance sont typiquement utilisés par les analystes techniques de test.

Note : Les défauts relatifs à la performance ont souvent de lourds impacts sur l'application à tester. Quand les exigences de performance sont impératives il est souvent plus utile de faire des tests de performance sur les composants critiques (via des pilotes et des bouchons) plutôt que d'attendre les tests systèmes.

9.3.9 Outils de Test des Sites Internet

Les outils de tests d'hyperliens sont utilisés pour scanner et vérifier qu'il n'y a aucun hyperlien cassé ou manquant sur un site Internet. Certains outils offrent aussi des informations additionnelles comme des graphiques d'architecture (arborescence du site), la vitesse et la taille des téléchargements (par URL), les atteintes (hits) et les volumes. Ces outils sont aussi utiles pour vérifier le respect des contrats de niveau de service (Service Level Agreement). Les Analystes de Test et les Analystes Technique de Test utilisent ces outils.

10. Compétences – Composition de l'Equipe

Termes

Indépendance du test

10.1 Introduction

Tous les professionnels du test devraient connaitre les compétences individuelles requises pour mener à bien leurs activités respectives. Ce chapitre se focalise tout d'abord sur ces compétences individuelles pour ensuite aborder un certain nombre de sujets spécifiques aux gestionnaires de test, comme par exemple la dynamique d'équipe, l'organisation, la motivation et la communication.

10.2 Compétences Individuelles

La capacité d'un individu à tester un logiciel peut être obtenue par l'expérience ou par la formation dans différents domaines d'activité. Chacune des activités suivantes peut contribuer à l'acquisition des compétences de base du testeur :

- Utilisation des applications logicielles
- Connaissance du domaine fonctionnel ou du métier
- Activités durant les différentes phases du processus de développement logiciel comme l'analyse, le développement et le support technique
- Activités dans le domaine du test logiciel

Les utilisateurs d'applications logicielles ont une bonne vision du point de vue « utilisateur » du système et ont une bonne connaissance de la façon dont il est utilisé. Ils savent quelles défaillances auraient le plus gros impact et connaissent les comportements attendus du système. Les utilisateurs ayant un domaine d'expertise donné savent quelles parties du système sont de la plus haute importance pour le métier et comment elles affectent la capacité du métier à répondre à ses exigences. Cette connaissance peut aider à prioriser les activités de test, à élaborer des données de test réalistes et à vérifier ou fournir des cas d'utilisation.

La connaissance du processus de développement logiciel (analyse des exigences, conception et codage) donne un aperçu de la manière dont les erreurs peuvent être introduites, où elles peuvent être détectées et comment prévenir leur introduction. L'expérience dans le support technique fournit une connaissance de l'expérience, des attentes et des exigences d'utilisabilité de l'utilisateur. L'expérience en développement logiciel est importante pour l'utilisation d'outils d'automatisation des tests haut de gamme qui requièrent une expertise en programmation et conception.

Les compétences spécifiques au test logiciel comprennent : la capacité à analyser des spécifications, à participer à une analyse de risques, à concevoir des cas de test, à exécuter des tests et à enregistrer leurs résultats.

Pour les gestionnaires de test, avoir des connaissances, des compétences et de l'expérience en gestion de projet est important puisque la gestion des tests se déroule comme un projet, il faut par exemple, faire un plan, suivre l'avancement et rendre compte aux responsables.

Les compétences relationnelles comme émettre et recevoir des critiques, influencer et négocier sont tout aussi importantes dans les métiers du test. Un testeur techniquement compétent est susceptible d'échouer dans ce métier à défaut de disposer et d'utiliser les compétences relationnelles requises. En plus de travailler efficacement avec les autres, le professionnel du test doit également être bien organisé, attentif aux détails et posséder de fortes compétences pour la communication écrite et orale afin de réussir.

10.3 Dynamique de l'Equipe de Test

La sélection de l'équipe est l'une des fonctions les plus importantes d'un responsable dans l'organisation. Il y a beaucoup de points à prendre en compte en plus des compétences individuelles spécifiques requises pour le poste. Au moment de choisir un individu pour rejoindre l'équipe, la dynamique de l'équipe doit également être prise en compte. Cette personne complètera-t-elle les compétences et les différentes personnalités déjà présentes dans l'équipe de tests ? Il est important de prendre en compte l'intérêt qu'il y a à disposer de différents types de personnalité tout autant que d'une variété de compétences techniques. Une équipe de tests forte est capable de traiter de multiples projets de complexité variée tout en traitant avec succès les interactions relationnelles avec les autres membres de l'équipe du projet.

Les nouveaux membres de l'équipe doivent être intégrés rapidement dans l'équipe et disposer d'un tutorat adéquat. Un rôle précis doit être attribué à chaque personne de l'équipe. Cela peut être basé sur un processus d'évaluation individuelle. L'objectif est d'amener chaque individu à sa réussite individuelle tout en contribuant à la réussite complète de l'équipe.

Cela est largement fait en faisant coïncider types de personnalité et rôles dans l'équipe et en construisant l'équipe autant sur les compétences innées de chaque individu qu'en développant leur portefeuille de compétences.

Un point important à rappeler est que la personne parfaite sera rarement disponible, mais qu'une équipe forte peut être construite en équilibrant les forces et les faiblesses de chaque individu. La formation mutuelle au sein de l'équipe est nécessaire pour maintenir et construire les connaissances de l'équipe et accroitre la flexibilité.

10.4 Introduire le Test dans une Organisation

L'adaptation du test à la structure organisationnelle varie largement selon les organisations. Alors que la qualité est du ressort de tous tout au long du cycle de développement du logiciel, une équipe de test indépendante peut largement contribuer à un produit de qualité. L'indépendance de la fonction du test varie largement dans la pratique comme le montre la liste suivante, classée du moins au plus d'indépendance :

- Pas de testeurs indépendants
  - Dans ce cas, il n'y a pas d'indépendance et le développeur teste son propre code
  - Le développeur, s'il arrive à dégager du temps pour les tests, va décider que le code fonctionne comme il le voulait, ce qui peut correspondre ou non aux exigences réelles
  - Le développeur peut corriger rapidement toute anomalie rencontrée
- Réalisation des tests par un développeur différent de celui qui a écrit le code
  - Peu d'indépendance entre le développeur et le testeur
  - Un développeur testant le code d'un autre développeur peut être réticent à signaler les anomalies
  - L'état d'esprit d'un développeur envers le test est de se concentrer souvent sur les cas de test positifs
- Réalisation des tests par un testeur (ou une équipe de test) faisant part de l'équipe de développement
  - Le testeur (ou l'équipe de tests) rendra compte à la gestion de projet
  - L'état d'esprit d'un testeur est plus de se concentrer sur la vérification de l'adhérence aux exigences
  - Comme le testeur est un membre de l'équipe de développement, il peut avoir des responsabilités sur le développement en plus de celles du test
- Testeurs issus de l'organisation métier, de la communauté des utilisateurs ou d'une autre organisation technique non liée au développement
  - Reporting indépendant aux responsables
  - La qualité est le centre d'intérêt principal de cette équipe
  - La formation et le développement des compétences sont concentrés sur le domaine du test
- Spécialistes de test externes réalisant les tests sur des cibles de test spécifiques
  - Les cibles de tests pourraient être l'utilisabilité, la sécurité ou la performance
  - La qualité devrait être le centre d'intérêt de ces individus, mais cela peut dépendre de la structure de reporting

- Réalisation des tests par une organisation externe à l'entreprise
  - Le maximum d'indépendance est atteint
  - Le transfert de compétence peut ne pas être suffisant
  - Des exigences claires et une structure de communication bien définies vont être nécessaires
  - La qualité de l'organisation externe doit être régulièrement auditée

Il y a différents degrés d'indépendance entre le développement et les organisations de tests. Il est important de comprendre qu'il peut y avoir un compromis car plus d'indépendance peut induire plus d'isolement et moins de transfert de compétences. Un niveau d'indépendance plus bas peut accroitre la connaissance mais peut également introduire des objectifs contradictoires. Le niveau d'indépendance sera aussi déterminé par le modèle de développement du logiciel utilisé, par exemple dans le cadre d'un développement agile, les testeurs font le plus souvent partie de l'équipe de développement.

Chacune des options citées ci-dessus peut être mélangée dans une organisation. Il peut y avoir des tests réalisés au sein de l'organisation en charge du développement tout autant qu'au sein d'une organisation de tests indépendante et il peut y avoir une certification finale par une organisation externe. Il est important de comprendre les responsabilités et les attentes pour chaque phase de test et de couvrir ces exigences pour maximiser la qualité du produit fini tout en respectant les contraintes de planning et de budget.

La sous-traitance est l'une des formes de mise en place d'une organisation externe. Le sous-traitant peut être une autre société qui fournit des services dans le domaine du test et qui peut être localisée dans vos locaux ou non, dans votre pays ou à l'étranger (on parle parfois d'off-shore). La sous-traitance entraine des défis, notamment quand elle se fait à l'étranger. Dans ce cas, voici une liste d'éléments à prendre en compte :

- les différences culturelles
- le suivi des ressources sous-traitées
- le transfert d'informations, la communication
- la protection de la propriété intellectuelle
- la matrice des compétences, le développement des compétences et la formation
- le turn-over des employés
- l'évaluation précise des coûts
- la qualité

10.5 Motivation

Il y a beaucoup de façons de motiver une personne qui travaille dans le domaine des tests :

- la reconnaissance du travail accompli
- l'approbation par la hiérarchie
- le respect au sein de l'équipe projet et de la communauté des pairs
- la récompense en fonction du travail fourni (en incluant salaire, augmentations au mérite et primes)

Il y a des contraintes projet qui peuvent rendre ces outils de motivation difficiles à utiliser. Par exemple, un testeur peut travailler très dur sur un projet qui a une date de fin impossible à respecter. Le testeur peut faire tout ce qui est en son pouvoir pour faire de la qualité le centre d'intérêt de l'équipe, faire des heures supplémentaires et des efforts et finalement, le produit peut être livré avant en raison d'influences externes. Il peut en résulter une faible qualité du produit malgré les plus grands efforts du testeur. Cela peut être démotivant si la contribution du testeur n'est pas comprise ni mesurée indépendamment du succès ou de l'échec du produit final.

L'équipe de tests doit s'assurer qu'elle suit les métriques appropriées pour prouver qu'un bon travail a été fait pour réaliser les tests, limiter les risques et enregistrer les résultats avec précision. A défaut de recueillir et de publier ces données, il est facile pour une équipe de se démotiver si ses membres ne reçoivent pas la reconnaissance qu'ils pensent due pour un travail bien fait.

La reconnaissance n'est pas seulement déterminée par des aspects immatériels comme le respect ou l'approbation, elle est aussi visible au travers d'opportunités de promotion, de hausse de salaire et de plan de carrière. Si l'équipe de test n'est pas respectée, ces opportunités peuvent ne pas être possibles.

La reconnaissance et le respect sont acquis quand il est clair que le testeur contribue à la valeur du projet. Dans un projet individuel, cela est plus facilement atteint en impliquant le testeur dans la conception du

projet et en gardant cette implication tout au long du cycle de vie. Finalement les testeurs vont gagner la reconnaissance et le respect par leur contribution au développement positif du projet, mais cette contribution devrait également être quantifiée en termes de couts des réductions de qualité et d'atténuation des risques.

10.6 Communication

La communication de l'équipe de test intervient principalement à 3 niveaux :

- documentation des produits de tests : stratégie de tests, plan de tests, cas de test, rapport de synthèse de tests, rapports d'anomalies, etc.
- retours sur des documents passés en revue : exigences, spécifications fonctionnelles, cas d'utilisation, documentation de tests de composant, etc.
- Recueil et partage d'information : interaction avec les développeurs, les autres membres de l'équipe de tests, la hiérarchie, etc.

Toute communication doit être professionnelle, objective et effective pour construire et maintenir le respect pour toute l'équipe de tests. La diplomatie et l'objectivité sont requises pour fournir des retours, particulièrement des retours constructifs, sur les produits du travail des autres.

Toute la communication devrait être centrée sur l'atteinte des objectifs de test et sur l'amélioration de la qualité tant sur les produits que sur les processus mis en œuvre pour produire les systèmes logiciels. Les testeurs communiquent avec une large audience, incluant les utilisateurs, les membres de l'équipe projet, la hiérarchie, des groupes de test externes et des clients. La communication doit être efficace pour le public ciblé. Par exemple un rapport de tendance d'anomalies à destination de l'équipe de développement peut être trop détaillé pour être approprié à une réunion de management exécutif.

11. Références

11.1 Standards

Cette section liste les standards mentionnés dans ce syllabus.

Par chapitre

Les chapitres suivants font référence aux standards indiqués

- Chapitre 2
  BS-7925-2, IEEE 829, DO-178B/ED-12B.
- Chapitre 3
  IEEE829, D0-178B/ED-12B.
- Chapitre 4
  BS 7925-2.
- Chapitre 5
  ISO 9126.
- Chapitre 06
  IEEE 1028.
- Chapitre 7
  IEEE 829, IEEE 1044, IEEE 1044.1.

Par ordre alphabétique

Les standards suivants sont mentionnés dans les chapitres indiqués

- [BS-7925-2] BS 7925-2 (1998) Software Component Testing
  Chapitres 02 et 04
- [IEEE 829] IEEE Std 829™ (1998/2005) IEEE Standard for Software Test Documentation (currently under revision)
  Chapitres 02 et 03
- [IEEE 1028] IEEE Std 1028™ (1997) IEEE Standard for Software Reviews
  Chapitre 06
- [IEEE 1044] IEEE Std 1044™ (1993) IEEE Standard Classification for Software Anomalies
  Chapitre 07
- [ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality
  Chapitre 05
- [ISTQB] ISTQB Glossary of terms used in Software Testing, Version 2.0, 2007
- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne systems and Equipment certification, RTCA/EUROCAE ED12B.1992.
  Chapitres 02 et 03

11.2 Livres

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

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

[Black03]: Rex Black, “Critical Testing Processes”, Addison-Wesley, 2003, ISBN 0-201-74868-1

[Black07]: Rex Black, “Pragmatic Software Testing”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Burnstein03]: Ilene Burnstein, “Practical Software Testing”, Springer, 2003, ISBN 0-387-95131-8

[Buwalda01]: Hans Buwalda, “Integrated Test Design and Automation” Addison-Wesley Longman, 2001, ISBN 0-201-73725-6

[Copeland03]: Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X

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

[Gerrard02]: Paul Gerrard, Neil Thompson, “Risk-based e-business testing”, Artech House, 2002, ISBN 1-580-53314-0

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

[Graham07]: Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black “Foundations of Software Testing”, Thomson Learning, 2007, ISBN 978-1-84480-355-2

[Grochmann94]: M. Grochmann (1994), Test case design using Classification Trees, in: conference proceeedings STAR 1994

[Jorgensen02]: Paul C.Jorgensen, “Software Testing, a Craftsman’s Approach second edition”, CRC press, 2002, ISBN 0-8493-0809-7

[Kaner02]: Cem Kaner, James Bach, Bret Pettichord; “Lessons Learned in Software Testing”; Wiley, 2002, ISBN: 0-471-08112-4

[Koomen99]: Tim Koomen, Martin Pol, “Test Process Improvement”, Addison-Wesley, 1999, ISBN 0-201-59624-5.

[Myers79]: Glenford J.Myers, “The Art of Software Testing”, John Wiley & Sons, 1979, ISBN 0-471-46912-2

[Pol02]: Martin Pol, Ruud Teunissen, Erik van Veenendaal, “Software Testing: A Guide to the Tmap Approach”, Addison-Wesley, 2002, ISBN 0-201-74571-2

[Splaine01]: Steven Splaine, Stefan P.,Jaskiel, “The Web-Testing Handbook”, STQE Publishing, 2001, ISBN 0-970-43630-0

[Stamatis95]: D.H. Stamatis, “Failure Mode and Effect Analysis”, ASQC Quality Press, 1995, ISBN 0-873-89300

[vanVeenendaal02]: van Veenendaal Erik, “The Testing Practitioner”, UTN Publsihing, 2002, ISBN 90-72194-65-9

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

[Whittaker04]: James Whittaker and Herbert Thompson, “How to Break Software Security”, Peason / Addison-Wesley, 2004, ISBN 0-321-19433-0

11.3 Autres références

Les références suivantes pointent vers des informations disponibles sur Internet.

Malgré une vérification de ces références à la date de la publication du Syllabus Avancé, l'ISTQB ne peut pas être tenu responsable si elles ne sont plus disponibles

- Chapitre 05
  - www.testingstandards.co.uk
- Chapitre 06
  - Bug Taxonomy: www.testingeducation.org/a/bsct2.pdf
  - Sample Bug Taxonomy based on Boris Beizer's work: inet.uni2.dk/~vinter/bugtaxst.doc
  - Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf
  - Heuristic Risk-Based Testing By James BachJames Bach, Interview on What IsTesting.com. www.whatistesting.com/interviews/jbach.htm
  - www.satisfice.com/articles/et-article.pdf
  - extrait de "Exploratory & Risk-Based Testing (2004) www.testingeducation.org"
  - Exploring Exploratory Testing , Cem Kaner and Andy Tikam , 2003
  - Pettichord, Bret, "An Exploratory Testing Workshop Report", www.testingcraft.com/exploratorypettichord
- Chapitre 09
  - www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview06.pdf
  - TMMi www.tmmifoundation.org/

12. Annexe A – Informations sur le Syllabus

Objectifs de la qualification Certificat Avancé

- Acquérir une reconnaissance du test comme une spécialisation professionnelle et essentielle de l'ingénierie logicielle.
- Fournir un cadre standard pour le développement des carrières des testeurs.
- Permettre une reconnaissance des testeurs qualifiés professionnellement par les employeurs, clients et leurs pairs, et accroître le profil des testeurs.
- Promouvoir de bonnes pratiques, régulières, dans toutes les disciplines d'ingénierie logicielle.
- Identifier des thèmes de tests qui sont pertinents et porteurs de valeur ajoutée pour l'industrie.
- Permettre aux fournisseurs de logiciels d'embaucher des testeurs certifiés et ainsi acquérir un avantage commercial sur la leur compétition en publiant leur politique de recrutement de testeurs.
- Fournir une opportunité aux testeurs et aux personnes intéressées par les tests d'acquérir une qualification reconnue internationalement sur le sujet.

Conditions d'entrée pour cette qualification

Les critères d'entrée pour passer l'examen du certificat de Testeur de logiciels certifié CFTL Niveau Avancé (ISTQB Advanced Level Certificate in Software Testing) sont:

- Posséder un certificat de testeur de logiciels Niveau Fondation délivré par un organisme d'examen ou par un comité national agréé par l'ISTQB.
- Posséder un nombre d'années d'expérience en test de logiciels requis par l'organisme d'examen ou par le comité national délivrant la certification Avancée
- Adhérer au Code d'Éthique défini dans ce Syllabus

Il est également recommandé au candidat de suivre une formation auprès d'un organisme accrédité par un comité national de l'ISTQB. Cependant, l'examen peut être passé sans formation préalable.

Un Certificat en Tests de Logiciels niveau Avancé existant (provenant d'un comité national ou d'un organisme d'examen reconnu par l'ISTQB) délivré avant la publication de ce Certificat International, sera considéré comme équivalent au Certificat International. Le Certificat niveau Avancé n'expire pas et ne doit pas être renouvelé. La date d'attribution est indiquée sur le Certificat

Dans chaque pays participant, les aspects locaux sont contrôlés par un Comité National des Tests Logiciels reconnu par l'ISTQB. Les devoirs des comités nationaux sont spécifiés par l'ISTQB, mais implémentés dans chaque pays. Les devoirs des comités nationaux incluent l'accréditation des organismes de formation et l'organisation d'examens, directement ou par l'intermédiaire d'un ou plusieurs comités d'examens accrédités.

13. Annexe B – Notes aux lecteurs

13.1 Comités d'Examen

Ce Syllabus niveau Avancé requiert la connaissance du contenu du Syllabus niveau Fondation du CFTL/ISTQB, version 2005, avec le niveau de connaissance spécifié dans le Syllabus niveau Fondation.

Des comités d'examen, reconnus par l'ISTQB/CFTL, peuvent générer des questions sur tout sujet mentionné dans ce syllabus.

Les questions auront un poids différent en fonction de l'objectif d'apprentissage des sujets correspondants. Par exemple, une question relative à un niveau d'apprentissage K1 se verra attribuer moins de points qu'une question relative à un niveau K3, et encore plus de points seront attribués à une question sur un niveau K4.

13.2 Candidats & Organismes de Formation

Pour recevoir la certification niveau Avancé, les candidats doivent détenir le certificat niveau Fondation et être en mesure de justifier, auprès du comité d'examen, de suffisamment d'années d'expérience dans le domaine du test logiciel. Il faut s'adresser à un comité d'examen pour prendre connaissance de ses critères en matière d'expérience. L'ISTQB suggère un minimum de 5 années d'expérience pratique dans le domaine de l'ingénierie logicielle comme pré-requis à l'obtention du niveau avancé, ou 3 ans d'expérience pour les titulaires d'un diplôme d'ingénieur en informatique ou d'un diplôme universitaire équivalent.

Atteindre le niveau de professionnalisme requis par le niveau avancé va au delà de la connaissance du contenu de ce syllabus. Il est vivement conseillé aux candidats et organismes de formation de consacrer à la lecture et à la recherche d'information davantage de temps que celui indiqué dans ce syllabus.

Le syllabus fournit une liste de références, livres et standards que les candidats et organismes de formation peuvent étudier afin de comprendre en profondeur certains sujets.

14. Annexe C - Notes pour les organismes de formation

14.1 Modularité

Ce syllabus fournit le contenu des trois modules suivants :

- Gestionnaire de Tests Niveau Avancé
- Analyste de Test Niveau Avancé
- Analyste Technique de Test Niveau Avancé

En passant les trois niveaux, le candidat peut avoir la certification « Testeur professionnel niveau avancé complet ».

14.2 Durée de Formation

14.2.1 Formation par Module

La durée recommandée de formation nécessaire pour former aux 3 différents rôles est comme suit :

- Gestionnaire de Tests Niveau Avancé 5 jours
- Analyste de Test Niveau Avancé 5 jours
- Analyste Technique de Test Niveau Avancé 5 jours

Cette durée est basée sur le nombre de chapitres par module et sur les objectifs spécifiques d'apprentissage pour chaque chapitre. La durée minimale spécifique est indiquée par chapitre et par rôle.

Les organismes de formation peuvent passer plus de temps qu'indiqué et les candidats peuvent aussi passer plus de temps en lectures et en recherches personnelles. Le plan de la formation ne doit pas obligatoirement suivre l'ordre indiqué dans le syllabus.

Les formations ne doivent pas forcément se dérouler sur des jours consécutifs. Les organismes de formation peuvent choisir d'organiser leurs cours de manière différente, comme, par exemple, 3+2 jours pour les gestionnaires de tests, ou 2 jours en commun suivis de 3 jours pour les analystes de tests et pour les analystes de tests techniques.

14.2.2 Aspects Communs

Les organismes de formation peuvent choisir d'enseigner les modules communs en une seule fois, et ainsi réduire le temps global et éviter les redites. Il est néanmoins rappelé aux organismes de formation que certains sujets peuvent être vus sous différents angles en fonction du module dans lequel ils se trouvent.

14.2.3 Sources

Le syllabus contient des références vers des standards qui peuvent être utilisés dans la préparation des supports de cours. Chaque standard utilisé doit correspondre à la version citée dans ce syllabus. D'autres publications, modèles ou standards non référencés dans ce syllabus peuvent être utilisés et référencés, mais ne feront pas objet de questions dans les examens.

14.3 Exercices Pratiques

Des travaux pratiques (exercices courts) devraient être inclus pour tous les aspects où les candidats sont amenés à appliquer leurs connaissances (objectif d'enseignement de niveau K3 ou supérieur). Les cours et les exercices doivent être basés sur les Objectifs d'Apprentissage et sur la description des sujets présents dans ce syllabus.

15. Annexe D – Recommandations

Comme ces recommandations s'appliquent à plusieurs chapitres du syllabus, elles ont été regroupées dans cette annexe. Les éléments listés peuvent faire l'objet de questions lors de l'examen.

Cette liste apporte un grand nombre de recommandations utiles pour relever les défis du test et s'appuie sur les sept principes de base du test introduits dans le syllabus niveau fondation. Elle ne se veut pas exhaustive mais fournit un ensemble de «leçons tirées » qu'il est intéressant de connaître. Les organismes de formation choisiront les points les plus pertinents selon le module enseigné.

15.1 Recommandations pour l'industrialisation

Un grand nombre de défis doivent être relevés lors de la mise en œuvre des tests de façon industrialisée.

Les testeurs avancés doivent être capables de mettre en œuvre les différentes recommandations décrites dans ce syllabus en fonction du contexte de leurs organisations, équipes, tâches et composants logiciels. La liste suivante mentionne des éléments qui ont un effet négatif manifeste sur la performance de l'effort de test. Veuillez noter que cette liste n'est pas exhaustive

- Créer des plans de test favorisant le test fonctionnel
  Les défauts ne se limitent pas aux aspects fonctionnels, ou à un seul utilisateur. Les interactions entre plusieurs utilisateurs peuvent avoir un impact sur le logiciel testé.
- Ne pas avoir suffisamment de configurations de tests
  Si plusieurs types de processeurs, de systèmes d'exploitation, de machines virtuelles, de navigateurs et de périphériques peuvent être combinés selon différentes configurations, des tests limités à simplement une partie des configurations possibles ne vont probablement pas permettre de découvrir tous les défauts
- Repousser les tests de stress et de charge à la dernière minute
  Les résultats des tests de stress et de charge peuvent conduire à des changements majeurs dans le logiciel (jusqu'à l'architecture de base elle-même). L'implémentation de ces changements pouvant nécessiter des ressources considérables, conduire ces tests juste avant la mise en production peut avoir des conséquences très négatives pour le projet.
- Ne pas tester la documentation
  Le logiciel est livré aux utilisateurs avec sa documentation. Si la documentation n'est pas adaptée au logiciel, l'utilisateur ne sera pas en mesure d'exploiter pleinement le logiciel, et peut même renoncer à l'utiliser
- Ne pas tester les procédures d'installation
  Les procédures d'installation, de même que les procédures de backup et de restauration ne sont faites qu'un petit nombre de fois. Cependant, elles sont plus critiques que le logiciel car si le logiciel ne peut pas être installé il ne sera pas utilisé du tout.
- Persister à achever entièrement une tâche de test avant de passer à la suivante
  Même si certains cycles de vie de développement logiciel suggèrent une exécution séquentielle des tâches, en pratique, il est souvent nécessaire de conduire plusieurs tâches (au moins partiellement) en parallèle.
- Ne pas parvenir à identifier correctement les parties à risque
  Il est possible d'identifier certaines parties à risque afin de les tester ensuite de façon plus approfondie. Cependant, les parties laissées avec un minimum de test ou pas de test pourront par conséquent voir leur niveau de risque augmenter par rapport à l'estimation initiale.
- Être trop précis pour les entrées et les procédures de test
  En ne laissant pas aux testeurs suffisamment de liberté pour leurs propres initiatives dans la création des entrées et procédures de tests, on ne les encourage pas à examiner des parties qui pourraient être prometteuses (en terme de défauts cachés)
- Ne pas tracer et investiguer les évènements curieux ou non expliqués
  Les observation ou les résultats bizarres qui paraissent sans importance sont souvent des indicateurs de défauts qui (comme pour les icebergs) sont cachés sous la surface

- Vérifier que le produit fait ce qu'il doit faire mais ne pas vérifier qu'il ne fait pas ce qu'il ne doit pas faire
  En se limitant à ce que le produit doit faire, on risque de ne pas détecter des choses que le logiciel n'est pas censé faire (fonctions supplémentaires non désirées par exemple)
- Avoir des suites de test compréhensibles uniquement par leurs propriétaires
  Les testeurs peuvent changer de domaine de responsabilité. D'autres testeurs vont alors avoir besoin de lire et comprendre des tests spécifiés préalablement par d'autres personnes. Ne pas fournir des spécifications de test lisibles et compréhensibles peut avoir un impact négatif car les objectifs de test peuvent ne pas être compris et ces tests peuvent même être tous supprimés.
- Tester uniquement au travers des interfaces visibles pour l'utilisateur
  Les interfaces avec le logiciel ne se limitent pas aux interfaces utilisateurs. Les communications interprocessus, l'exécution de batchs et autres programmes interagissent aussi avec le logiciel et peuvent générer des défauts
- Utiliser un reporting insuffisant pour les bugs et une gestion de configuration insuffisante
  La gestion et le reporting des incidents, de même que la gestion de configuration, sont extrêmement important pour assurer le succès du projet dans son ensemble (incluant aussi bien le développement et le test). On considère qu'un testeur efficace est quelqu'un qui permet de résoudre les défauts, plutôt que quelqu'un qui découvre beaucoup de défauts mais de parvient pas à les documenter et à les gérer suffisamment bien pour qu'ils puissent être corrigés
- Ajouter uniquement des tests de régression
  Les suites de test ne doivent pas évoluer dans le temps uniquement pour vérifier l'absence de défauts en régression. Le code évolue dans le temps et il est nécessaire d'implémenter des tests supplémentaires pour couvrir les nouvelles fonctionnalités, de même que pour vérifier l'absence de régression dans les autres parties du logiciel.
- Ne pas documenter les tests pour les prochaines versions
  Les tâches de tests ne se terminent pas quand le logiciel est fourni aux utilisateurs ou distribué en magasin. Une nouvelle version ou livraison du logiciel sera probablement produite, c'est pourquoi la connaissance et l'expérience du test doivent être enregistrées et transférées aux testeurs responsables du prochain effort de test.
- Vouloir automatiser tous les tests.
  L'automatisation peut sembler intéressante mais l'automatisation est un projet de développement à part entière. Il ne faut pas chercher à automatiser tous les tests : pour certains il sera plus rapide de procéder manuellement que d'automatise l'exécution.
- S'attendre à rejouer tous les tests manuels
  Il est souvent irréaliste de s'attendre à ce que tous les tests manuels soient rejoués. La durée d'attention des testeurs va diminuer et ils vont avoir tendance à se concentrer sur certains domaines du logiciel, consciemment ou non.
- Utiliser des outils d'automatisation basés sur l'IHM pour réduire le coût de création des tests.
  Un outil de capture/rejeu IHM représente un investissement initial important et doit être utilisé pour servir une stratégie précise, avec une bonne compréhension et une évaluation de tous les coûts associés.
- S'attendre à ce que les tests de régression trouvent une importante proportion de bugs
  En général, les tests de régression trouvent peu de défauts, principalement car il s'agit de tests ayant déjà été exécutés (par exemple pour une version précédente du même logiciel) et que les défauts doivent avoir été détectés lors de ces précédentes exécutions. Cela ne signifie pas que les tests de régression doivent être tous éliminés mais seulement que leur efficacité (capacité à détecter de nouveaux défauts) est inférieure à celle des autres tests.
- Adopter une mesure de couverture de code en pensant que seuls les chiffres seront utiles
  La couverture de code et les métriques peuvent sembler très intéressantes d'un point de vue management, basées sur les chiffres et graphiques, mais des chiffres ne peuvent pas refléter l'efficacité ou la pertinence d'un test. Exemple: 100% est un objectif séduisant pour la couverture de code, mais est-ce un objectif réaliste, est-ce le bon objectif (couverture des instructions, des conditions, ou couverture MCDC)?
- Enlever des tests d'une suite de test de régression uniquement parce qu'ils ne permettent pas d'augmenter la couverture.
  Dans une suite de test de régression, certains tests peuvent, ou doivent, être supprimés et d'autres

ajoutés. La suppression d'un test ne doit pas se baser uniquement sur l'augmentation de la couverture par ce test. La pertinence d'un cas de test (le type de défauts recherchés) peut être sans effet sur la couverture. Exemple: la couverture de code n'est pas le seul type de couverture; des tests peuvent avoir été créés pour d'autres raisons que celle visant à améliorer la couverture (comme des valeurs spécifiques à utiliser ou des séquences d'évènements).

- Utiliser la couverture pour mesurer la performance des testeurs
  La couverture est une mesure de complétude, pas de performance ou d'efficacité personnelle. D'autres métriques peuvent être définies pour évaluer l'efficacité des testeurs dans le contexte de leur organisation. L'utilisation de telles métriques doit être menée avec beaucoup de précautions afin d'éviter tout effet indésiré.
- Abandonner totalement la mesure de couverture
  Différent types de couverture sont disponibles (par exemple les instructions, les conditions, les modules et les fonctions pour le code), et la charge de travail pour obtenir les métriques adéquates peut être importante. Cependant, ce n'est pas une raison pour abandonner les métriques de couverture qui peuvent être très utiles.

Beaucoup de ces points sont traités dans les différentes sections de ce syllabus.

Lors de l'introduction de mesures et pratiques de test dans votre organisation, il est utile de considérer non seulement la liste ci-dessus mais aussi d'autres sources d'information telles que :

- Les syllabus (Fondation et Avancé) du CFTL
- Les ouvrages référencés dans ce syllabus
- Les modèles d'amélioration tels que ceux présentés dans la section 8.3

16. Index