ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE, CONCENTRATION GÉNIE LOGICIEL
M. Sc. A.

PAR
Samah KHORCHANI

PROCESSUS ET PRATIQUES DE L'INGÉNIERIE DE LA QUALITÉ DANS LA PHASE TEST DU MODÈLE GÉNÉRIQUE DE CYCLE DE VIE DES SYSTÈMES TI

MONTRÉAL, LE 13 MAI 2016

PRÉSENTATION DU JURY

CETTE MÉMOIRE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Witold Suryn, directeur de mémoire
Département de génie logiciel et TI à l'École de technologie supérieure

M. Michel Kadosh, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Alain April, membre du jury
Département de génie logiciel et TI à l'École de technologie supérieure

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 12 MAI 2016

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Mes remerciements s'adressent principalement à mon directeur de recherche, le professeur Witold Suryn, pour la précieuse opportunité de poursuivre mes études aux cycles supérieurs sous sa direction. J'ai pu compter à chaque étape du développement de ce mémoire sur sa grande disponibilité, sur ses importants conseils et sur sa vision globale de la recherche. Je le remercie aussi pour son temps, son dévouement et sa patience tout au long des deux années de mes études de maîtrise.

J'exprime mes vifs remerciements et mon respect au président de jury, le professeur Michel Kadosh, au membre de jury le professeur Alain April de m'avoir donné l'honneur d'évaluer mon travail de recherche.

Je tiens aussi à exprimer ma gratitude à l'égard de mes parents et mon mari pour leur encouragement et leur soutien avec quoi je suis arrivé l'aboutissement de ce travail. Et je dédie particulièrement ce mémoire à ma fille Malak.

PROCESSUS ET PRATIQUES DE L'INGÉNIERIE DE LA QUALITÉ DANS LA PHASE TEST DU MODÈLE GÉNÉRIQUE DE CYCLE DE VIE DES SYSTÈMES TI

Samah KHORCHANI

RÉSUMÉ

La phase test logiciel repose sur un procédé ramifié et complexe, et constitue une étape décisive du cycle de vie de développement informatique. Mais, dans la pratique, elle est souvent mal gérée ou considérée comme une activité élémentaire.

Afin de remédier à ces problématiques, nous avons pensé à introduire les pratiques de l'ingénierie de la qualité dans le test des systèmes TI. L'objectif principal de ce travail de recherche est donc la création d'un processus d'ingénierie de qualité QETP, intégré dans la phase de test logiciel et basé sur des normes internationaux. Ce processus a pour rôle de coordonner les différentes activités de test, unifier les efforts des équipes et départements au sein de l'organisation, se soucier de la satisfaction des exigences de qualité et non seulement des besoins fonctionnels, et accroitre la qualité de produit et de processus de développement.

Ce processus d'ingénierie de qualité de test QETP qui répond à ces intentions a été conçu. Le processus de test a été transformé pour s'adapter aux normes de qualité. Les livrables adéquats générés par le QETP proposé ont été déterminé, et l'ensemble de la documentation nécessaire pour son application a été identifié.

Mots clés : Test logiciel, ingénierie de qualité, qualité logicielle.

PROCESSES AND PRACTICES OF QUALITY ENGINEERING IN TESTING PHASE OF IT SYSTEMS' GENERIC LIFE CYCLE

Samah KHORCHANI

ABSTRACT

The software testing phase is based on a branched and complex process, and is a milestone of the software development lifecycle. But in practice, it is often badly handled or considered as a single activity. This mismanagement and false estimation affect both the quality of the development process and the realized software product significantly.

To overcome these issues, we decided to introduce the practice of quality engineering in the IT systems testing. The main objective of our research is the creation of a quality engineering process QETP integrated in the software testing phase and based on high-level standards. This process has the role of coordinating the various test activities, unifying the efforts of teams and departments within the organization, worrying about meeting the quality requirements and not only functional requirements, and increase product and development process quality.

This testing quality engineering process QETP that satisfy these intentions has been successfully completed, and the testing process was adjusted to meet the quality standards. We also determined the appropriate deliverables generated by the proposed QETP and identified all of the documentation necessary for its implementation.

Keywords: software testing, quality engineering, software quality.

TABLE DES MATIÈRES

Page

INTRODUCTION 1

CHAPITRE 1 REVUE DE LA LITTÉRATURE 5
1.1 Introduction 5
1.2 Synthèse de l'état de l'art sur le test logiciel 5
1.3 Synthèse de l'état de l'art sur l'ingénierie de qualité logiciel 10
1.4 Discussion des normes ISO 15
1.4.1 ISO 25000 15
1.4.2 ISO 25010 17
1.4.3 ISO 29119-1 19
1.4.4 ISO 29119-2 20
1.4.5 ISO 29119-3 21
1.4.6 ISO 29119-4 22
1.5 Conclusion 22

CHAPITRE 2 MÉTHODOLOGIE DE LA RECHERCHE 23
2.1 Introduction 23
2.2 Problématique 23
2.3 Objectifs de la recherche 24
2.4 Portée de la recherche 25
2.5 Conclusion 27

CHAPITRE 3 PROBLÈMES ET LACUNES AVEC LES TESTS LOGICIELS 29
3.1 Introduction 29
3.2 Le test logiciel : les problématiques de la pratique 29
3.2.1 L'absence du processus de test 30
3.2.2 L'absence de plan de test 30
3.2.3 Les tests retardés 31
3.2.4 Les attentes irréelles 32
3.2.5 La dépendance de l'équipe de tests 32
3.2.6 L'absence des mesures adéquates 33
3.2.7 L'absence d'historique d'incidents 33
3.2.8 Le manque de priorisation 34
3.3 Conclusion 35

CHAPITRE 4 LE PROCESSUS D'INGÉNIERIE QETP 36
4.1 Introduction 36
4.2 La collecte des modèles 36
4.2.1 Le processus de test STLC 36
4.2.2 La normalisation de STLC 38

4.2.3 Le STLC et la qualité 40
4.3 Le processus QETP 40
4.3.1 Le processus de test logiciel 41
4.3.1.1 Organisation de test 41
4.3.1.2 Gestion et planification de test 41
4.3.1.3 Implémentation de test 42
4.3.1.4 Environnement de test 42
4.3.1.5 Exécution de test 42
4.3.1.6 Clôture de test 43
4.3.2 Le processus d'ingénierie de qualité de la phase test 44
4.3.2.1 Organisation de test de la qualité 44
4.3.2.2 Planification de test de la qualité 45
4.3.2.3 Implémentation de test de la qualité 45
4.3.2.4 Opération de test de la qualité 45
4.3.2.5 Rapport de test de la qualité 45
4.3.3 Le processus d'ingénierie de qualité intégré 45
4.3.4 La documentation de QETP 47
4.3.4.1 Politique de test de la qualité 48
4.3.4.2 Plan de test de la qualité 48
4.3.4.3 Cas de test de la qualité 48
4.3.4.4 Rapport de test de la qualité 48
4.4 Les caractéristiques du QETP 50
4.4.1 Généralité 50
4.4.2 Adaptation 50
4.4.3 Facilité 50
4.4.4 Normalisation 50
4.4.5 Interaction 50
4.5 L'apport du QETP 51
4.6 Conclusion 51

CONCLUSION 53

BIBLIOGRAPHIE 55

Tableau 2.1 Récapitulatif de contexte du QETP 26

Tableau 4.1 Les normes utilisées à travers le processus QETP 47

LISTE DES FIGURES

Page

Figure 1.1 Les quatre composants de processus de développement logiciel 6
Figure 1.2 Le modèle consolidé de cycle de vie de qualité Suryn-Abran version 0.1 11
Figure 1.3 Processus d'ingénierie de la qualité de la conception 12
Figure 1.4 Organisation de la série des normes SQuaRE 16
Figure 1.5 Modèle de qualité d'utilisation 17
Figure 1.6 Modèle de qualité de produit 18
Figure 1.7 Les processus multicouches de test 20
Figure 1.8 Les modèles multicouches montrant tous les processus de test 21
Figure 2.1 Enchainement des objectifs du QETP 25
Figure 4.1 Cycle de vie général de test logiciel 37
Figure 4.2 Processus de la documentation de test logiciel ISO 29119-3 38
Figure 4.3 Processus de test logiciel 43
Figure 4.4 Processus d'ingénierie de qualité de la phase test 44
Figure 4.5 Processus d'ingénierie de qualité intégré de la phase test 46
Figure 4.6 Mouvement des documents dans le QETP 49

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CMMI	Capability Maturity Model Integration
CQL	Consolidated Quality Lifecycle
ISO	International Organization for Standardization
PDCA	Plan – Do – Check – Act
PDP	Program Design Phase
PME	Petites et Moyennes Entreprises
QA	Quality Assurance
QPDP	Quality Program Design Process
SDLC	Software Development Life Cycle
SQuaRE	Systems and software Quality Requirements and Evaluation
SRS	Software Requirements Specification
STLC	Software Testing Life Cycle
SWEBOK	Software Engineering Body of Knowledge
TMMI	Test Maturity Model integration
QETP	Quality Engineering Testing Process
V&V	Verification & Validation

INTRODUCTION

Dans le domaine du génie logiciel, comme tout autre domaine de production, la qualité des produits devient de plus en plus concurrentielle, aiguë et sophistiquée. Donc pour gagner sa place sur le marché, où pour simplement ne pas la perdre, il est impératif d'envisager d'améliorer la qualité et d'y investir.

Contrairement à un matériel, la qualité d'un produit logiciel est difficile à prédire et n'est pas facile à évaluer puisqu'un défaut peut ne pas paraître dans l'exécution du programme, alors qu'il est présent d'une manière latente (à long terme) et peut ne jamais survenir. La qualité d'un logiciel dépend principalement de sa construction et des processus utilisés pour son développement. Les tests constituent une des pratiques du contrôle qualité logiciel les plus importantes.

Le test logiciel et la qualité

Le test logiciel est le processus qui permet d'exécuter une application ou un système dans l'intention d'en révéler des erreurs. Avec le test il est impossible de prouver que le système ne contient aucun défaut, car ce n'est tout simplement pas possible, mais par contre il nous permet de révéler des défauts dans le système et donc les corriger et de valider de plus en plus le comportement du logiciel et par suite d'améliorer d'avantage sa qualité.

Le test constitue une phase entière du cycle de vie du produit logiciel dont il ne s'agit pas d'une tâche élémentaire, mais en revanche d'un processus complexe qui côtoie le projet logiciel dans presque toutes ses phases de développement. Et, selon les besoins et les différentes étapes où le projet est arrivé, les types de tests se diffèrent. Il y a principalement, le test unitaire, le test d'intégration, le test système et le test d'acceptation.

Afin de normaliser les pratiques de test et de créer un domaine d'expertise clair et commun pour les spécialistes de l'assurance qualité logicielle, plusieurs organismes internationaux ont conçu et publié des standards pour le test logiciel. Il y a, par exemple, la norme ISO 12207 qui décrit tous les processus de cycle de vie du génie logiciel, dont le processus de test. Et la nouvelle série des normes ISO 29119 qui contient le guide 29119-1 présentant les concepts et les définitions, l'ISO 29119-2 décrivant le processus de test, le guide 29119-3 qui décrit la documentation des tests, des gabarits et des exemples; et finalement l'ISO 29119-4 concernant les techniques de test.

Les tests logiciels ont été aussi couverts par les modèles de maturité CMMI et TMMI, accompagnés de plusieurs articles, revues et recherches de professionnels de la qualité logicielle, durant les dernières années, qui seront abordés dans la revue de la littérature.

Problèmes et propositions

Dans la plupart des cas, les problèmes les plus importants dans les tests sont reliés à la qualité du procédé de test lui-même, y compris la composition globale de l'équipe de test et si l'entreprise suit des processus bien intégrés.

Toutefois, il est possible de contribuer à un test logiciel réussi à l'aide de la conception d'un processus d'ingénierie de la qualité qui couvre la totalité du cycle de vie de test dans les moindres détails, et qui soit générique, prêt à appliquer et qui peut appuyer n'importe quel projet logiciel. Ce processus, qui est le sujet de cette recherche, va être le résultat de l'intégration des pratiques de l'ingénierie de la qualité dans le processus de test. En se basant fortement sur les normes internationales, ce processus de test aura comme objectif la résolution des problèmes majeurs rencontrés par les testeurs, et favoriser la qualité logiciel par les tests.

Structure du document

Ce mémoire se compose de quatre chapitres, à savoir :

- Le premier chapitre intitulé « REVUE DE LA LITÉRATURE » dans lequel les récents travaux de recherche au sujet de l'ingénierie de la qualité et de tests logiciels sont présentés, et les notions et concepts de base sont introduits;

- Le deuxième chapitre titré « MÉTHODOLOGIE DE LA RECHERCHE » dans lequel les objectifs de ce travail, ainsi que la conception de la nouvelle approche proposée seront présentés;

- Le troisième chapitre a pour titre « PROBLÈMES ET LACUNES AVEC LES TESTS LOGICIELS » où les problématiques de la phase de test qui vont être traitées à travers cette recherche sont discutées;

- Le quatrième chapitre intitulé « LE PROCESSUS D'INGÉNIERIE QETP » présente le processus complet de la méthode proposée de test et fait ressortir ses points forts et avantages.

Finalement, la conclusion de ce travail de recherche présente quelques perspectives et recommandations futures.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

Ce premier chapitre survole un ensemble de publications et de travaux de recherche antérieurs en liaison avec ce mémoire afin de bien délimiter et circonscrire la portée globale du sujet et de renforcer la compréhension de la problématique.

1.2 Synthèse de l'état de l'art sur le test logiciel

Cette section présente une synthèse des travaux antérieurs portant explicitement sur le test logiciel, en ordre chronologique, à partir de l'année 1998 afin de démontrer 1'évolution de la recherche dans ce domaine. Cette revue littéraire contient des opinions individuelles ainsi que des opinions collectives d'auteurs.

Bill Hertzel (1998) souligne la relation entre le test et la qualité et stipule que la qualité d'un logiciel n'est pas tangible, et que le but de test est de rendre cette qualité visible, en indiquant que c'est une mesure de la qualité logicielle.

Hertzel présente aussi une liste de six principes de test, qui doivent être maîtrisés :

1) L'impossibilité d'un test complet.
2) L'activité de test est créative et difficile.
3) La raison de test est de prévenir les déficiences de se produire.
4) Le test est basé sur le risque.
5) Le test doit être planifié.
6) Le test nécessite l'indépendance.

William E. Perry (2000) présente le cycle de vie des quatre activités du processus de développement logiciel et indique que les tests indépendants comprennent uniquement l'activité de vérification du cycle Planifier-Faire-Vérifier-Agir (PDCA) et que l'équipe de

développement de logiciels est responsable des trois autres activités. Il explique que le travail des testeurs consiste à vérifier et déterminer que le logiciel répond aux besoins des clients et des utilisateurs. Les testeurs indépendants signalent les défauts à l'équipe de développement qui prend la décision si les défauts découverts doivent être corrigés ou non.

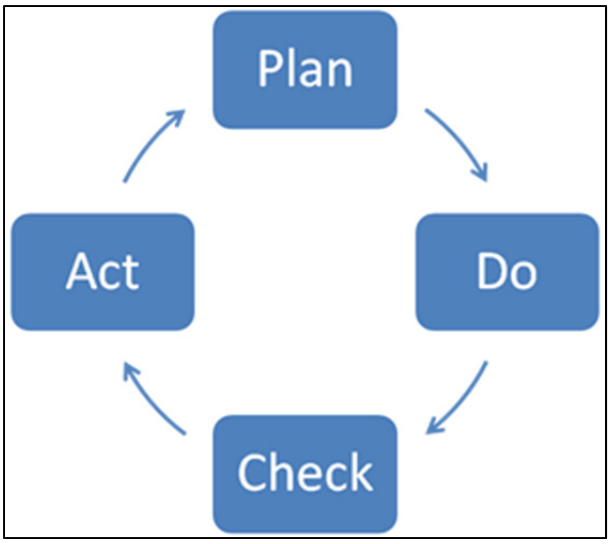

Figure 1.1 Les quatre composants de processus de développement logiciel
Adapté de (William E. Perry, 2000, p. 3)

L'auteur explique chacune des activités du PDCA, et considère que l'étape de vérification (check) du cycle correspond à l'activité de test indépendant. Il propose alors qu'il s'agit de vérifier les résultats et déterminer si les travaux avancent conformément au plan et si les résultats escomptés sont obtenus. Il est aussi nécessaire de vérifier aussi la performance des procédures établies, les changements dans les conditions ou les anomalies qui peuvent surgir et de comparer les résultats des travaux avec les objectifs.

William E. Lewis, en (2009), destine son ouvrage à l'amélioration de la qualité à l'aide des tests. Il présente le test comme étant l'activité qui comporte une série d'exécutions dynamiques des programmes logiciels. Le test vise à découvrir et corriger autant d'erreurs que possible avant la livraison du logiciel au client.

Il propose que le test puisse être considéré comme une technique de gestion de risque ou d'assurance qualité, mais il reste encore un « art » du point de vue qu'il n'est pas une science exacte et qu'il nécessite toujours des améliorations, car sa performance et son efficacité sont toujours relatives.

Lewis mentionne aussi que tout au long de l'histoire du développement logiciel, il y a eu de nombreuses définitions et avancées en ce qui concerne les tests logiciels.

Dans les années 1950, le test était défini comme ce que les programmes ont besoin pour trouver des défauts. Suite à la maturation du domaine du développement logiciel, entre les années 1960 et 1970, le test logiciel est défini comme ce qui est fait pour démontrer l'exactitude d'un programme, ou comme le processus visant à établir la confiance qu'un programme ou un système fait ce qu'il est censé faire. À la fin des années 1970, il est proposé que le test est un processus d'exécution d'un programme avec l'intention d'y trouver une erreur, et pas seulement de prouver qu'il fonctionne. Cette nouvelle définition propose qu'un bon cas de test soit celui qui a une forte probabilité de dévoiler une erreur pas encore découverte. Cette proposition vise exactement l'opposé de ce qui est populaire jusque-là.

Dans leur ouvrage publié en (2011), Claude Y. Laporte et Alain April ont dédié un chapitre entier sur les tests. Dans ce chapitre ils abordent les principales normes et les modèles qui s'intéressent au test, par exemple ISO 12207, ISO 29110 et la série ISO 29119 à côté du modèle CMMI et TMMI qui proposent une classification des tests selon cinq niveaux de maturité (initial, géré, défini géré et mesuré, optimisé). Ils expliquent comment il est possible de passer d'un niveau de maturité à un plus élevé. Ils y présentent aussi les différents types, stratégies et techniques de tests. Les auteurs catégorisent aussi les défauts logiciels trouvés lors de l'exécution du scénario de test où se trouvent des défauts au niveau des exigences, de la conception, de la programmation et même dans les tests.

Le chapitre, se termine par la présentation d'une classification des facteurs qui favorisent ou nuisent à la qualité des tests logiciels. Parmi ces facteurs de succès, ils citent la terminologie commune, la maturité du processus, la classification des défauts et une bonne documentation.

Glenford, Sandler et Badgett discutent dans leur ouvrage (2012) que le test logiciel est devenu, d'une part plus difficile et d'autre part plus facile en même temps.

Plus difficile, vu la présence grandissante de l'ordinateur dans presque tous les domaines de la vie. De la multitude des langages de programmation et de systèmes d'exploitation, et l'évolution des plates-formes matérielles. Ce qui augmente, de plus en plus, la valeur et l'importance du test logiciel. Par conséquent, le logiciel touche maintenant la vie quotidienne de millions de personnes, soit en leur permettant de faire leur travail avec efficacité, ou de leur causer des frustrations lors de problèmes.

D'autre part, le test est devenu plus facile, selon certains points de vue, car la gamme des logiciels et des systèmes d'exploitation est beaucoup plus sophistiquée. De plus l'utilisation des bibliothèques de langages de programmation toutes prêtes et déjà vérifiées, réduisent le besoin de tester certaines applications.

Ces auteurs n'ont pas manqué de proposer leur propre définition du test logiciel, où ils l'ont présenté comme étant un processus ou une série de processus, créés pour assurer que le programme fait ce qu'il est conçu pour faire et, inversement, qu'il ne fait rien d'imprévu.

Gererd O'Regan (2014) affirme que le test est une activité constructive, dont on vérifie l'exactitude des fonctionnalités du logiciel, comme il peut être considéré une activité destructive, en ce que l'objectif est de trouver des défauts dans le logiciel mis en œuvre. Alors, le test a pour rôle de vérifier si les exigences sont correctement implémentées, et confirmer la présence ou l'absence de défauts.

O'Regan souligne aussi que les cas de test doivent être nécessairement examinés par des experts indépendants afin de veiller à ce qu'ils soient suffisants pour vérifier l'exactitude du logiciel. Il stipule que l'efficacité de n'importe quel test logiciel est influencée par la maturité du processus de test employé.

Ayant comme but l'amélioration de la qualité à travers le test, il propose la mesure du « *cumulative test arrival rate »* en conjonction avec d'autres mesures, car elle donne une indication sur la stabilité du produit logiciel et aide à décider s'il est approprié de libérer le logiciel ou si d'autres tests doivent être effectués.

L'auteur se penche sur le fait que les défauts de test sont d'une grande valeur dans le sens qu'ils permettent à une organisation d'améliorer son processus de développement et d'empêcher des défauts de se reproduire à l'avenir. Une organisation de développement mature doit effectuer des revues internes des exigences, de conception et de code avant d'entamer les tests. Et l'efficacité du processus des revues internes et du processus de test peut être observée par le biais de la mesure « *containment phase »* décrite dans son ouvrage.

Il propose qu'une organisation mature vise à avoir 0 % de défauts signalés par le client, et que cet objectif nécessite des améliorations dans la méthodologie d'inspection du logiciel et dans la méthodologie de tests de logiciels. Les mesures fournissent un moyen de vérifier que les améliorations ont été couronnées de succès. Et chaque défaut est potentiellement utile, car il permet à l'organisation d'identifier les faiblesses dans le processus de développement logiciel et de cibler les améliorations.

En ce qui concerne les défauts qui ont échappé, ils offrent, selon l'auteur, une occasion d'améliorer le processus de test, car ils indiquent une faiblesse dans le processus de test. Ainsi les tests de logiciels jouent un rôle important dans l'amélioration de la qualité.

En abordant le thème des tests logiciels, il est incontournable de mentionner le corpus des connaissances du génie logiciel SWEBOK (2014). Dans sa troisième version, le SWEBOK regroupe quinze domaines de connaissances, le test logiciel inclus. Le chapitre concernant les tests comprend les principes fondamentaux de test, ses niveaux, ses techniques, ses mesures, le processus de test et ses outils. Cette norme présente le test comme une activité effectuée pour évaluer la qualité du produit avec l'objectif de l'améliorer en identifiant les défauts. Il implique la vérification dynamique du comportement d'un programme par rapport au comportement attendu sur un ensemble fini de cas de test. Ces cas de test sont choisis du domaine d'exécution.

1.3 Synthèse de l'état de l'art sur l'ingénierie de qualité logiciel

Les clients s'attendent, généralement, à un certain niveau de maturité des produits et services logiciels, ce niveau transforme la qualité des logiciels en un avantage concurrentiel.

Cette section survole quelques ouvrages et travaux de recherche qui ont mis l'accent sur l'identification et l'évolution de l'ingénierie de qualité logiciel.

En 2005, Witold Suryn et Alain Abran ont publié un article qui présente les résultats de l'analyse et la vérification applicative d'un modèle proposé pour l'intégration de l'ingénierie de la qualité dans le processus de développement logiciel (CQL) publié en 2003. Ils visent de permettre l'évolution du modèle sans s'éloigner de son utilité initiale.

Le résultat de l'analyse de 2005 contient des améliorations qui ont touché principalement les phases de conception architecturale, de l'intégration et de la transition.

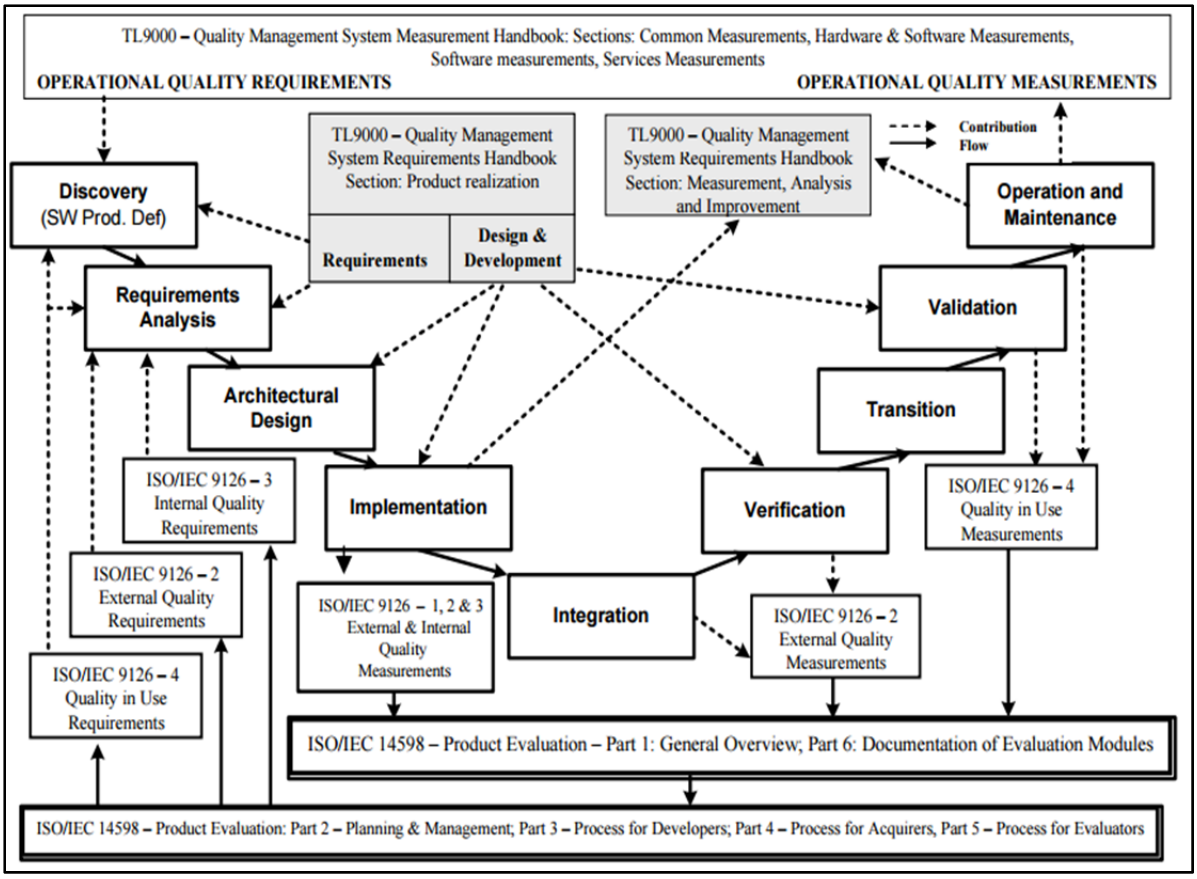

Figure 1.2 Le modèle consolidé de cycle de vie de qualité Suryn-Abran version 0.1
Tiré de (Witold Suryn, 2005, p. 2)

En se basant principalement sur le modèle Suryn-Abran CQL de 2005, Witold Suryn et Abdelileh Kahlaoui (2005) ont publié un article qui propose un processus de qualité pour la phase de conception du cycle de vie de produit logiciel.

Dans cet article, ils analysent les standards disponibles et adoptés par le modèle de 2005, pour identifier les attributs de qualité qui peuvent être utilisés lors de la phase de conception (PDP). Ils proposent un processus de qualité de la phase de conception (QPDP) qui peut être intégré et effectué en parallèle avec la phase de conception usuelle.

Dans cette proposition, chaque activité de la conception est mappée à une activité du processus de qualité proposé, ce qui permet concrètement d'intégrer l'ingénierie de la qualité dans le processus de développement. L'article propose aussi un processus de construction des documents de qualité de la conception élaboré et structuré en sept étapes.

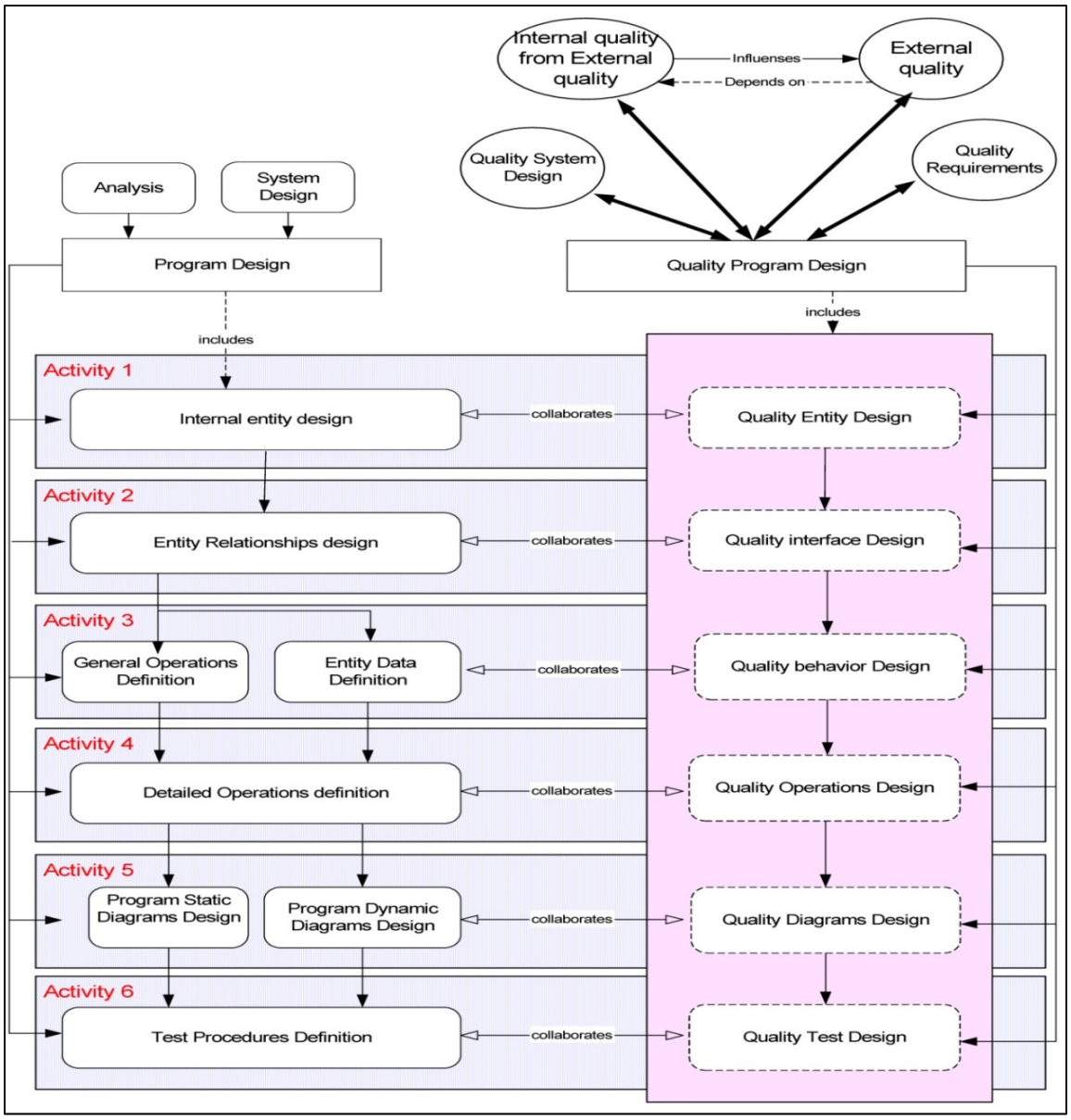

Figure 1.3 Processus d'ingénierie de la qualité de la conception

Tiré de (Witold Suryn, 2005, p. 6)

La même année, en 2005, et concernant le même sujet, Jeff Tian précise que le processus d'ingénierie de la qualité fait partie intégrante du processus d'ingénierie logicielle globale où d'autres préoccupations, telles que le coût et le calendrier, sont également considérées et gérées. Il propose que pour gérer les activités de l'assurance qualité, il faille effectuer la planification de la qualité avant que les activités spécifiques d'assurance qualité soient exécutées. Il faut aussi améliorer la quantification de la qualité à l'aide de mesures, l'analyse, les commentaires et les activités de suivi et cela après le début d'exécution des activités d'assurance qualité spécifiques.

Plus tard en 2008, G. Gordon Schulmeyer a couvert un ensemble de standards commerciaux et gouvernementaux qui sont en relation avec l'assurance qualité logiciel. Et sa vision des standards sera adoptée au cours de ce travail de recherche.

Schulmeyer présente au cours de son analyse, la série des normes ISO 25000 (SQuaRE) qui comporte cinq guides: gestion de la qualité, modèle de la qualité, mesure de la qualité, exigences de la qualité et évaluation de la qualité. Il précise qu'une organisation peut se conformer à la norme ISO 25001 en satisfaisant ses exigences et en expliquant toute exclusion, ou en fournissant ses propres recommandations pour la planification et la gestion des exigences de qualité du produit logiciel et son évaluation, et de s'assurer de faire correspondre ces recommandations aux exigences de la norme.

L'auteur évoque aussi le rôle spécial de la norme ISO 12207, qui vise à définir un vocabulaire et une architecture de plusieurs processus de cycles de vies logiciels ainsi qu'un processus d'adaptation. Par exemple, elle peut être utilisée dans des situations où un accord contraignant de deux parties à acquérir des produits ou des services logiciels est appliqué. L'accord contraignant pourrait être sous la forme d'un contrat entre deux organisations différentes, ou il pourrait être un accord informel entre différentes parties de la même organisation.

Plus récemment, Witold Suryn (2013) présente dans son livre une approche pour les praticiens de l'ingénierie de qualité logicielle. Selon Suryn la qualité logicielle peut se définir selon différentes perspectives dépendamment des intérêts des consommateurs et des fournisseurs. Il signale aussi qu'il y a toujours une corrélation entre la maturité des processus, des organisations, et la qualité. Mais malgré l'évolution continue des normes et modèles de maturité, et de la publication du SWEBOK, (qui a pour objectif de fournir une caractérisation consensuelle validée des limites de la discipline de génie logiciel et de fournir un accès à l'ensemble de connaissances qui soutiennent cette discipline) il persiste toujours de imperfections dans le logiciel qui doivent être gérées, notamment, les tests de la qualité sont abordés seulement en référence aux processus de vérification alors qu'en réalité l'évaluation réelle du produit logiciel est effectuée tout au long de son cycle de vie.

Son livre présente des modèles de qualité qui présentent une approche pour lier ensemble différents attributs de qualité avec les objectifs de base pour, aider à comprendre comment les différentes facettes de la qualité contribuent à l'ensemble, souligner que la qualité du logiciel est beaucoup plus que des simples défauts et échecs, aider à naviguer dans le plan des caractéristiques de qualité, sous-caractéristiques et mesures adéquates, et aider à définir le profil d'évaluation nécessaire.

En 2013, Stefan Wagner introduit le concept du contrôle continu de la qualité en expliquant son applicabilité. Pour cela, et en s'inspirant du cycle PDCA, il propose le circuit du contrôle qualité qui est, selon lui, est une contre-mesure dynamique qui peut adresser la dégradation de la qualité au cours de l'évolution des logiciels. En partant des objectifs généraux du produit, il est possible d'utiliser un modèle de qualité et une base de connaissances de la qualité du produit pour fixer les exigences qualité. Ensuite, elles sont remises à l'équipe de développement qui construit une version du produit logiciel qui sera validé par l'équipe d'assurance qualité. Celle-ci appliquera différentes techniques d'assurance de qualité, basées sur ce que le modèle de qualité propose de mesurer et les exigences de qualité spécifiées. Cela inclut les avis, les tests et les autres analyses. Les résultats sont utilisés par le modèle de

qualité et comparés aux exigences qualité. Les écarts et les objectifs de produits modifiés conduisent à modifier les demandes reçues par l'équipe de développement qui produira une nouvelle version du produit. Ce cycle se poursuivra jusqu'à l'atteinte d'objectifs préfixés.

En se fixant sur les pratiques d'assurance qualité chacune à part, les tests dans ce cas, il est possible de partir de ce modèle pour en générer des autres processus encore plus applicables en pratiques et plus détaillés.

Gerard O'Regan (2014) révèle qu'un de défis du génie logiciel est de livrer des logiciels de haute qualité dans les délais et le budget fixés par les clients. Afin de tenter de résoudre ce problème à trois dimensions; qualité, délai et budget, le gestionnaire de projet doit déterminer la qualité de son processus d'estimation et d'y apporter les améliorations nécessaires. L'utilisation de mesures est un moyen de le faire. L'amélioration de l'estimation sera observée une fois que l'écart est réduit entre l'effort estimé et réel. Le gestionnaire de projet détermine l'effort et l'échéancier réels par rapport à ceux estimés pour le projet.

Une estimation précise est cruciale afin de permettre d'atteindre une haute qualité, qui est selon l'auteur un facteur à ne pas négliger. Parallèlement, la qualité logicielle doit être soigneusement examinée lors de la conception et le développement du logiciel, car l'effet d'une défaillance du logiciel peut engendrer d'importants coûts pour le corriger, peut mener à une perte de crédibilité de l'entreprise, ou même dans certains cas à la perte de la vie.

1.4 Discussion des normes ISO

1.4.1 ISO 25000

La série des normes ISO 25000, aussi connu sous le nom de SQuaRE (System and Software Quality Requirements and Evaluation) a pour objectif de créer un cadre pour l'évaluation de la qualité du produit logiciel. ISO 25000 est le résultat de l'évolution de plusieurs autres

normes; l'ISO 9126, qui définit un modèle de qualité pour l'évaluation d'un produit logiciel, et l'ISO 14598, qui définit le processus d'évaluation du produit logiciel. La série de normes ISO 25000 se divise en cinq divisions.

Le guide de gestion de qualité est la norme ISO 25000 (Software product Quality Requirements and Evaluation). Le but de ce guide est de donner un aperçu général du contenu de SQuaRE, des modèles de référence communs et des définitions, ainsi que les relations entre les documents. Ceci permet aux utilisateurs de ce guide d'avoir une bonne compréhension d'ensemble des différents guides en fonction de leur objectif d'utilisation. Ce document contient aussi une explication du processus de transition entre l'ancienne norme ISO 9126 et ISO 14598 et la nouvelle version SQuaRE.

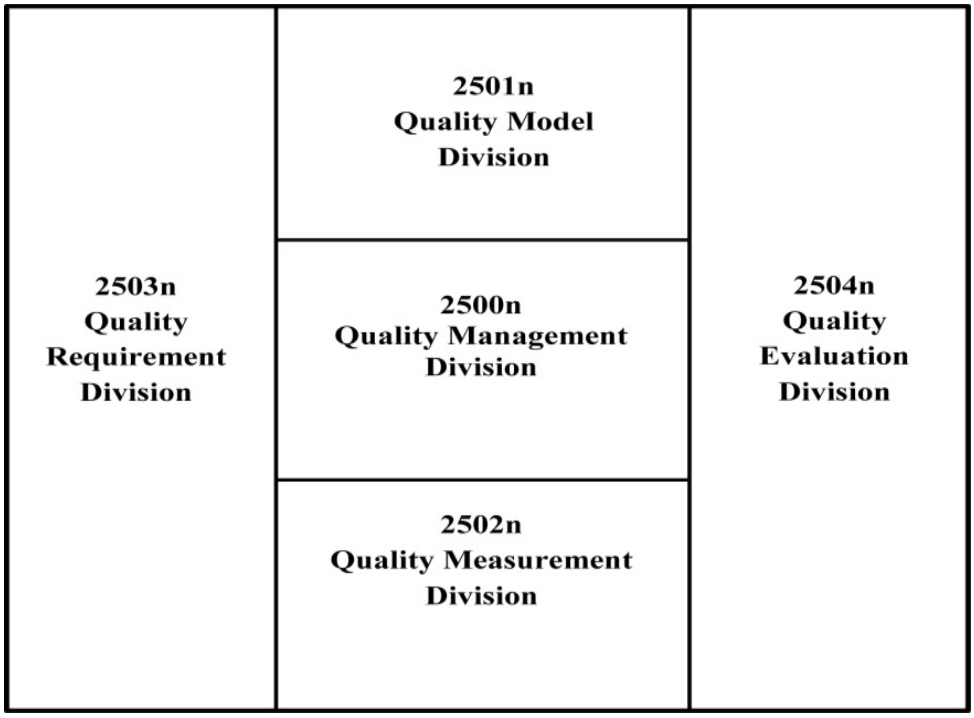

Figure 1.4 Organisation de la série des normes SQuaRE
Tiré de (ISO 25010, 2010, p. vii)

1.4.2 ISO 25010

Cette norme définit, premièrement, un modèle de qualité d'utilisation composé de cinq caractéristiques (dont certaines sont subdivisées en sous-caractéristiques) qui concernent le résultat de l'interaction du produit quand il est utilisé dans un contexte particulier. Ce modèle est applicable au système personne-machine complet, y compris les systèmes informatiques et les logiciels.

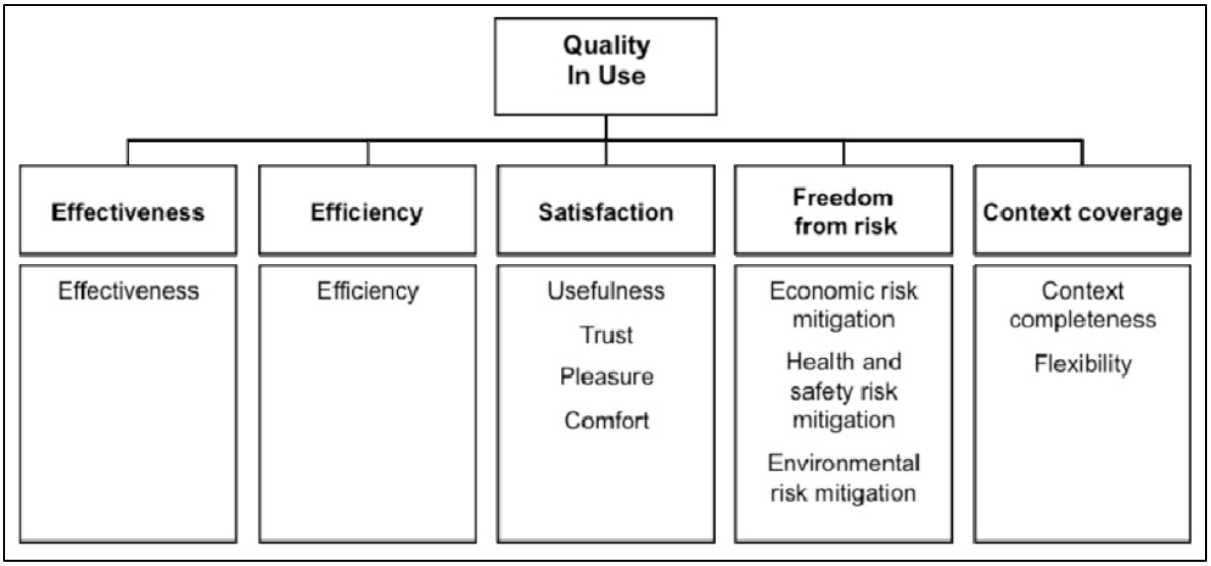

Figure 1.5 Modèle de qualité d'utilisation
Tiré de (ISO 25010, 2010, p. 3)

Deuxièmement, la norme définit un modèle de qualité du produit composé de huit caractéristiques (qui sont subdivisées en sous-caractéristiques) qui concernent les propriétés statiques de logiciels et les propriétés dynamiques du système.

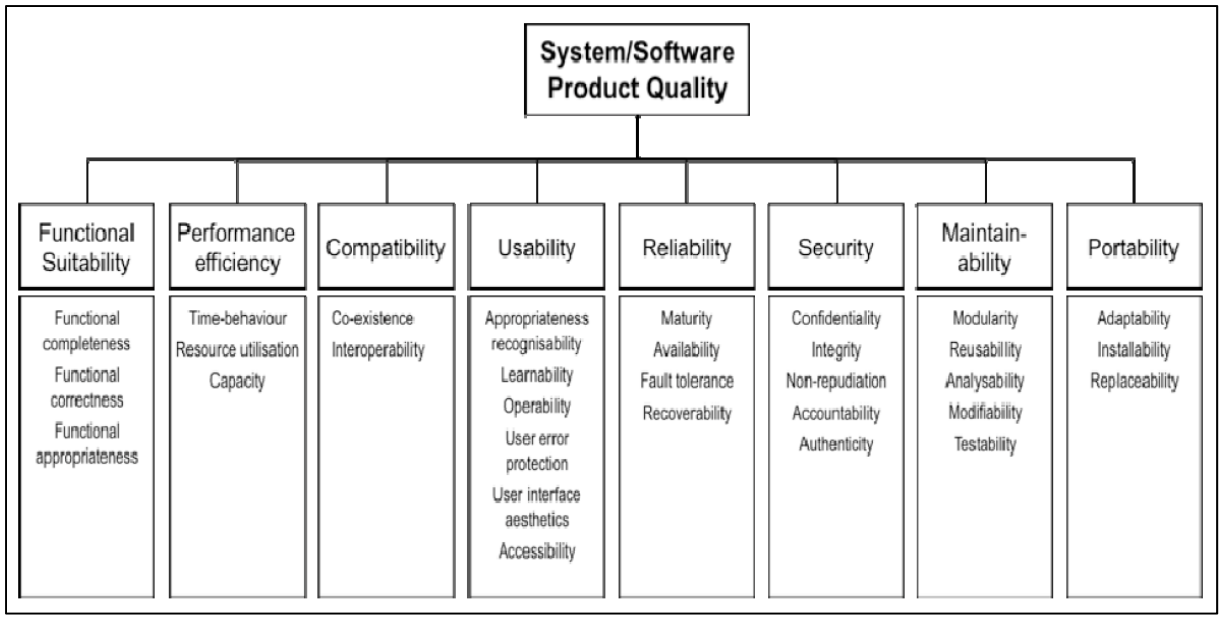

Figure 1.6 Modèle de qualité de produit
Adaptée de (ISO 25010, 2010, p. 4)

Les caractéristiques définies par ces deux modèles sont pertinentes pour tous les types de logiciels et de systèmes informatiques. Les caractéristiques et sous-caractéristiques fournissent une terminologie cohérente pour spécifier, mesurer et évaluer la qualité des systèmes et produits logiciels. Ils fournissent également un ensemble de caractéristiques de qualité qui peut être comparée aux exigences qualité initiales aux fins d'exhaustivité.

Le modèle de la figure 1.6 possède huit caractéristiques de qualité: aptitude fonctionnelle, fiabilité, opérabilité, efficacité de la performance, sécurité, compatibilité, maintenabilité et transférabilité. Dans ce modèle, la sécurité et la compatibilité ont été ajoutées à titre de caractéristiques principales. Certaines sous-caractéristiques ont aussi été ajoutées au modèle et un certain nombre d'entre elles ont été renommées avec des termes plus précis.

1.4.3 ISO 29119-1

L'ISO 29119 Software Testing, est un ensemble de normes pour les tests logiciels. Ces normes peuvent être utilisées dans n'importe quelle organisation ou cycle de vie de développement de logiciel. Cette série de standards est destinée à être indépendante du domaine du logiciel, de son environnement et de l'organisation, tout en soutenant une variété de cycles de vie et des méthodes logicielles.

La première partie de la norme ISO 29119 présente les définitions de base de test logiciel et les concepts qui prennent en charge les autres guides. Elle fournit un aperçu des approches de test, des méthodes et des notions de base.

Un principe clé de l'ISO 29119 est le concept de tests basé sur le risque. D'autres principes de tests sont aussi reconnus tels que le test basé sur le calcul, le test basé sur le modèle et le test basé sur l'expérience.

En outre, les concepts généraux de test logiciel y sont présentés, y compris : le rôle du test logiciel dans une organisation, la considération du contexte du projet, le test selon différents cycles de vie et la planification des tests.

Cette première norme facilite l'utilisation des autres normes de la série en introduisant les concepts et le vocabulaire sur lequel ces normes sont construites, tout en fournissant des exemples d'applications pratiques.

1.4.4 ISO 29119-2

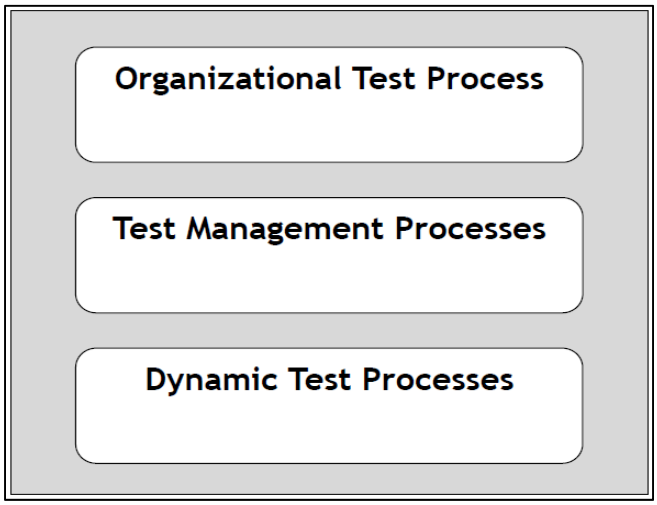

Figure 1.7 Les processus multicouches de test
Tiré de (ISO 29119-2, 2012, p. 10)

Cette deuxième partie de la norme est le cœur de la série, elle couvre le processus de test logiciel au niveau de l'organisation, la gestion et l'exécution de test. Cette section de la norme comprend des descriptions de processus de test au niveau organisationnel, au niveau de gestion de test et au niveau des tests dynamiques. Elle décrit les tests dynamiques, les tests fonctionnels et non fonctionnels, les manuels et les tests automatisés, et finalement les tests scénarisés et les tests non scénarisés. Les processus présentés dans cette norme peuvent être utilisés en conjonction avec tout modèle de cycle de vie de développement de logiciel.

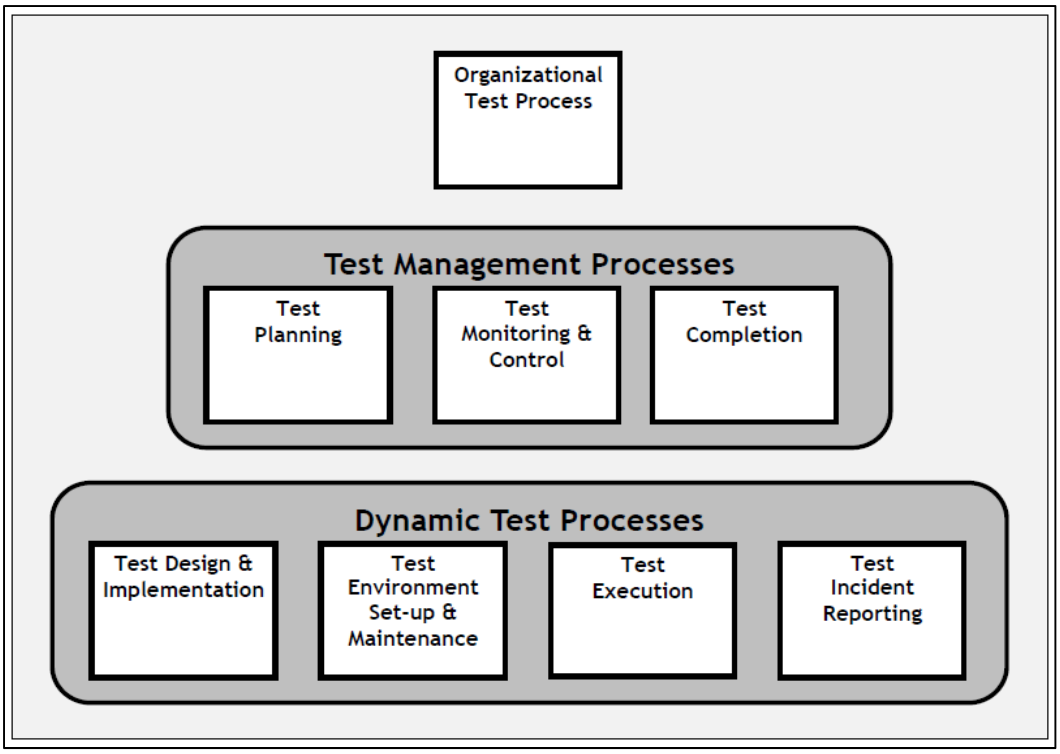

Figure 1.8 Les modèles multicouches montrant tous les processus de test
Tiré de (ISO 29119-2, 2012, p. 11)

Une approche fondée sur les risques est utilisée dans cette norme. Les tests basés sur le risque sont une approche pratique pour l'élaboration des stratégies et la gestion de tests, car elle permet au test d'être priorisé et axé sur les caractéristiques les plus importantes et les attributs de qualité de chaque système en cours de test.

1.4.5 ISO 29119-3

L'objectif de cette partie de la norme est de proposer des gabarits pour la documentation de test qui couvrent entièrement le cycle de vie du test logiciel. Chacun de ces gabarits peut être adapté afin de répondre aux besoins uniques de chaque organisation. En plus, tous ces modèles de documentation sont cohérents avec la partie ISO 29119-2 et peuvent être générés en appliquant les processus qui sont définis dans cette norme.

Cette troisième partie de la série des normes ISO 29119 a été conçue en se basant principalement sur la norme ISO 829, et donc elle la remplace.

1.4.6 ISO 29119-4

Cette partie de la norme a comme objectif de définir un standard international couvrant les techniques de conception de test logiciel qui peuvent être utilisés lors de la conception, par exemple lors du processus d'implémentation avec tout modèle de cycle de vie de développement logiciel. Les techniques de conception de tests qui sont présentées dans cette partie peuvent être utilisées pour concevoir les cas de tests qui vont servir à recueillir des preuves que les exigences de chaque système ont été respectées.

L'ISO 29119-4 couvre une variété de techniques de tests dynamiques couramment utilisés et fournit également des définitions d'une variété de types de tests liés à la qualité. Finalement, elle fournit des exemples de la façon dont les techniques de conception de cas de test peuvent être appliquées.

1.5 Conclusion

En conclusion de ce premier chapitre, il est à signaler que l'étude de ces publications a été utile car cela permet d'adopter certaines approches proposées par les auteurs et d'en rejeter d'autres pour préciser les objectifs de recherche. Ceci permet de partir d'une base solide en s'appuyant sur des travaux effectifs depuis vingt ans, qui ont permis d'apporter des innovations et pousser encore plus les limites de l'ingénierie logicielle. Les ouvrages qui serviront plus à cette recherche sont les normes ISO 29119 et ISO 25000.

La méthodologie de cette recherche, sa présentation et ses objectifs, vont être abordés dans le chapitre suivant.

CHAPITRE 2

MÉTHODOLOGIE DE LA RECHERCHE

2.1 Introduction

Dans ce deuxième chapitre, l'accent est mis sur le sujet de recherche en présentant sa problématique, ses objectifs et son contexte d'une manière détaillée.

2.2 Problématique

Étant donné que la qualité d'un produit (matériel ou immatériel) est devenue de plus en plus exigeante, le domaine de l'ingénierie de qualité s'élargit considérablement pour atteindre tous les secteurs de production. En informatique, et en particulier en génie logiciel, la qualité logicielle porte sur l'ensemble du produit. Un logiciel est un produit dont la fiabilité est difficile à prédire, contrairement au matériel, d'où la complexité du processus d'assurance qualité logicielle.

À partir de cette constatation, un certain nombre de méthodes et techniques, autre que les tests logiciels, ont été reconnus efficaces pour répondre à cette problématique. Ces méthodes ont fait leur preuve dans divers domaines d'application. Et, étant donné la difficulté en ce qui concerne les tests logiciels et l'importance de mener à bien les processus des tests pour le client comme pour l'organisation de développement elle-même, ce travail de recherche y sera consacré.

Ce projet de recherche porte sur l'amélioration du processus de l'ingénierie de la qualité, pendant la phase test du cycle de vie de développement des logiciels, et la détection des imperfections présentes dans cette phase. L'objectif est donc l'amélioration de la qualité globale du produit logiciel grâce à l'amélioration du processus de développement lui-même. Et plus précisément, l'enrichissement du processus de test logiciel.

L'intention principale de cette recherche est d'intégrer un processus d'ingénierie de la qualité dans la phase test du cycle de développement logiciel. Pour y arriver, il est nécessaire d'étudier, tout d'abord, les processus des tests et y localiser les lacunes. Ensuite il sera nécessaire de proposer de nouvelles pratiques en proposant un processus de test personnalisé, qui adopte les notions de l'ingénierie de qualité et les intégrera dans ce nouveau processus.

Pour atteindre ce but, un ensemble des normes et standards internationaux vont guider la direction de cette recherche. À titre d'exemple, la série des normes ISO 25000, peut contribuer au projet au niveau de la gestion de qualité et les modèles de qualité. De plus, la série de normes ISO 29119 peuvent aussi contribuer concernant les tests logiciels (définitions, concepts, types, méthodes, techniques). Finalement, les normes ISO 12207 et ISO 15288 présentent les différents processus de cycle de vie logiciel et les processus de cycle de vie des systèmes.

La retombée prévue de cette recherche sur le domaine vise l'application de processus d'ingénierie de qualité normalisés, approuvés et fiables pendant les tests d'un logiciel et cela contribuera à une qualité logicielle plus élevée.

2.3 Objectifs de la recherche

Le processus d'ingénierie de qualité test (QETP) proposé et qui doit être intégré dans le cycle de développement possède deux buts principaux :

Un but immédiat, qui concerne l'amélioration de la qualité du processus de test logiciel lui-même, et la proposition de solutions aux problématiques que peuvent rencontrer le testeur et empêcher le bon déroulement de son travail.

Suivi d'un but indirect, qui vise le perfectionnement de la qualité globale du produit testé, et qui se confirmera lorsque le test logiciel sera jugé réussi.

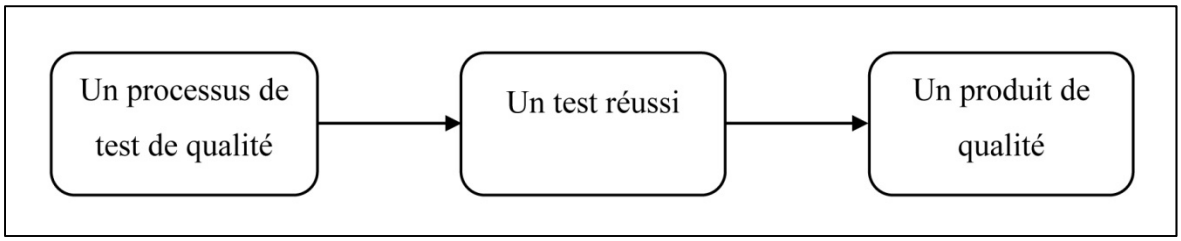

Figure 2.1 Enchainement des objectifs du QETP

La figure ci-haut représente la vision globale des objectifs du QETP, et explique qu'en partant d'un processus de test de qualité, bien élaboré et qui respecte les normes de haut niveau, mène à un test logiciel réussi et efficace. Et cela influence énormément la qualité du produit final de sorte qu'un nombre maximal des défauts qu'il contient ont été couverts et corrigés selon un ordre d'importance et un calendrier étudié.

2.4 Portée de la recherche

Le processus d'ingénierie de qualité de test proposé touche plusieurs volets pratiques tout au long la phase test du SDLC, et même lors des activités qui la précèdent et qui la suivent.

Le QETP côtoie le processus du test dans toutes ses étapes et commence tous les deux au démarrage du projet avec l'expression des besoins, et s'étale jusqu'au retrait du produit. Ce processus concerne essentiellement les testeurs et les responsables qualité, mais sa documentation doit être à la portée des développeurs et du chef de projet aussi pour assurer la bonne communication au sein de l'organisation. Cette documentation se résume dans la politique de qualité de test, le plan de qualité de test, les cas de qualité de test et le rapport de qualité de test qui sont accumulés avec la progression et l'avancement dans le processus.

La réalisation du QETP se base sur un ensemble des normes et standards lui donnants une certaine légitimité et un aspect universel afin qu'il soit utilisable avec tous types de projets et

dans tous les secteurs de développement. Parmi ces normes il y a l'ISO 29119-2, l'ISO 29119-3, l'ISO 29119-4, l'ISO 25010 et l'ISO 25030.

Tableau 2.1 Récapitulatif de contexte du QETP

Timing et calendrier	Phase test du SDLC intégré au processus de test Au démarrage du projet
Personnels	Chefs de projet Testeurs Ingénieurs qualité développeurs
Plans et livrables	Politique de qualité de test Plan de qualité de test Cas de qualité de test Rapport de qualité de test
Références normatives	ISO 29119-2 ISO 29119-3 ISO 29119-4 ISO 25010 ISO 25030
Portée et impact	Qualité du processus de test Qualité de produit Organisation des responsabilités Coopération entre les départements

L'adoption de ce processus d'ingénierie a un impact appréciatif sur la qualité du processus de test comme sur la qualité globale du logiciel. Il facilite aussi l'attribution des taches et des responsabilités et assure la bonne communication entre les départements.

2.5 Conclusion

Dans ce chapitre, le sujet de cette recherche a été mis en contexte en éclaircissant sa problématique et ses objectifs. Le prochain chapitre va mettre le point sur les lacunes de tests logiciels qui vont être résolues à l'aide du processus d'ingénierie proposé.

CHAPITRE 3

PROBLÈMES ET LACUNES AVEC LES TESTS LOGICIELS

3.1 Introduction

Ce chapitre est consacré à identifier et discuter les problèmes rencontrés pendant la phase de test du cycle de vie de développement logiciel, dont cette recherche s'intéresse à résoudre. Ces problèmes sont abordés par ordre d'importance.

3.2 Le test logiciel : les problématiques de la pratique

Bien que les tests ne représentent auparavant qu'une simple tâche de vérification du produit logiciel juste avant sa libération, il constitue actuellement une phase cruciale et critique du cycle de développement. De nos jours les activités de test suscitent de plus en plus l'attention des praticiens du domaine et sollicite la recherche visant à résoudre ses maintes problématiques.

Plusieurs chercheurs ont consacré leurs recherches à étudier les problématiques du test logiciel, à les catégoriser selon différents critères et à présenter des solutions et des recommandations pour les résoudre. Scott Tilley (2014) et Donald Firesmith (2013) ont essayé de couvrir l'ensemble des problématiques de leurs causes et de leurs manifestations, ainsi que la manière de les éliminer.

Ce mémoire énumère des problèmes de test logiciel auxquels il est possible de suggérer des solutions à l'aide de l'ingénierie de la qualité, ou simplement proposer des techniques pour éviter ces problèmes.

3.2.1 L’absence du processus de test

Afin de bien diriger un test logiciel, un processus de test est nécessaire. C’est un modèle à suivre qui offre une vue globale du cheminement du travail qu’il faut faire pour tester un produit logiciel. Le processus de test représente l’outil principal qui garantit le bon déroulement du test, et sans lequel il n’y a que des activités de test primitives et éparpillées qu’on ne peut pas évaluer la performance ni les résultats. L’adoption d’un bon processus de test nous permet alors d’administrer, de gérer et de mettre en œuvre le test logiciel. Ayant cela comme objectif, l’organisation internationale de normalisation ISO a créé la norme ISO 29119 qui propose un modèle, entre autres, de processus de test générique et détaillé. Cette norme couvre bien la procédure intégrale de la phase de test et offre une flexibilité qui permet de l’adapter quasiment à tous les types de produits logiciels et tous les types de tests.

3.2.2 L’absence de plan de test

Un plan de test logiciel permet l’exécution des tests selon les exigences et définit les critères d’entrée et de sortie pour chaque étape du test. La bonne planification des tests constitue l’un des facteurs de succès les plus importants, mais elle est souvent négligée ou mal conçue. La pression pour entamer la programmation est souvent la cause de l’absence de la planification de test. Mais sans une planification détaillée, des problématiques vont forcément survenir. Et la négligence d’adresser ces dernières peut créer de dépassements de délais et de budgets et aussi influencer la qualité attendue du produit.

Quand un projet n’a pas un plan pour décrire le processus de test, ce projet aura tendance à ne pas avoir ce processus. Parmi les soucis dus à l’absence du document de plan de test figure principalement : la difficulté de définition des rôles et des responsabilités, l’imprécision des objectifs de test et la fragilité de la boucle de rétroaction.

Le test est une des phases les plus importantes du SDLC, et représente une grande partie du budget du projet. Un plan de test détaillé influence grandement le succès du projet. Autrement dit, la planification détaillée des tests permet d'assurer que les problèmes éventuels seront traités au fur et à mesure et qu'ils seront découverts.

3.2.3 Les tests retardés

Lorsque le test est prévu d'être réalisé tard dans le cycle de développement, ou qu'un temps insuffisant de planification du projet est alloué à l'achèvement de toutes les activités de test, il pourrait y avoir des conséquences néfastes pour ce projet. Il est possible ici citer à titre d'exemple, la forte probabilité que le produit soit livré avec un retard significatif ou des défauts résiduels. Cela peut engendrer, également, la baisse de productivité des testeurs vue qu'ils ont dû travailler excessivement afin de respecter les délais prévus. Et à cause de l'intégration, qui précède le test, de la plupart des composants du système, il est très difficile de trouver et localiser les défauts et encore plus difficile et couteux de les corriger.

Dans le but d'éviter ces problèmes et ces retards, des mesures de préventives doivent être prises. Une de ces mesures est que le test doit être planifié pour être exécuté d'une manière itérative, incrémentale, parallèle et tôt dans le cycle de développement. Il faut aussi s'assurer qu'une marge de temps suffisante et adéquate est allouée aux activités de test et incluse dans le calendrier principal pendant le déroulement du projet. Même dans les circonstances spéciales, le test n'est jamais à négliger. Pour des résultats satisfaisants, toutes les parties prenantes du projet doivent communiquer souvent avec le département de test et lui transmettre les informations le plus tôt possible dans le cycle de développement. Comme il faut procurer aux testeurs tous les outils nécessaires et l'environnement favorable.

3.2.4 Les attentes irréelles

L'un des facteurs majeurs qui diminuent l'efficacité des tests est le faux espoir et l'attente des résultats irréels à accomplir à travers les tests. Certains testeurs croient que le test peut détecter toutes les anomalies, et donc prouver l'absence des défauts et que le système fonctionne comme prévu. Ils pensent donc que le test seul peut faire tout le travail de la vérification, et ils négligent les autres activités de certification et d'inspection.

Quelques organisations et chefs de projets ne sont pas encore convaincus que le rôle d'un testeur nécessite une expertise spécialisée. Certains considèrent aussi que le faible nombre d'erreurs détecté signifie le succès des tests, alors qu'on réalité cela ne signifie que la fragilité de test réalisé et sa faiblesse, parce qu'un test utile et réussi est capable de trouver le plus des défauts possibles.

Et afin d'éviter l'échec, chaque organisation doit s'assurer que son équipe de tests a reçu la formation et l'encadrement nécessaire, et que ses testeurs ont conscience des points forts et faibles de chaque type et méthode de test, que la quantité minime des erreurs détectées ne prouve pas la réussite du travail et qu'on ne doit pas se contenter des tests pour ce qui est vérification et ignorer les autres activités qui sont d'une importance majeure.

3.2.5 La dépendance de l'équipe de tests

Ce qui représente un souci commun dans de nombreuses organisations de développement logiciel, est la dépendance du département de test aux autres départements et surtout au choix et à l'orientation adoptée par le chef de projet. Ce dernier a souvent de la difficulté à obtenir un consensus avec l'équipe de testeurs ce qui leur donne un minimum d'indépendance et de liberté. Ils n'hésitent même pas à interférer dans leur travail et changer le calendrier dédié aux tests au profit des autres phases de développement. Cette attitude nuit au bon déroulement du travail de test. Le testa besoin d'une stratégie et des politiques

organisationnelles qui fixent dès le début du projet, le rôle de chacun des intervenants, et qui garantissent à chaque département le temps et les outils nécessaires à la réussite de ses travaux, dont le département de test. Il est nécessaire aussi de reconnaître l'importance de la contribution des testeurs, et par la suite leur fournir la formation adéquate.

3.2.6 L'absence des mesures adéquates

La collecte, l'analyse et la documentation d'un ensemble des mesures de test adéquates permettent aux gestionnaires de projets, et aux développeurs d'évaluer le logiciel, de quantifier la qualité des tests et la productivité des testeurs. En outre, l'étude de l'ensemble de ces métriques collectées aide à évaluer l'amélioration du processus de test.

L'absence de la mesure engendre une fausse évaluation du projet. Ce qui veut dire que sans une politique précise des mesures, les testeurs seront focalisés sur la quantité des erreurs détectées, au lieu de favoriser la découverte et la correction des défauts critiques.

Pour obtenir un meilleur déroulement de test, maximiser la qualité du produit et exploiter les efforts fournis par le personnel, il faut incorporer, dans le plan de test, un programme des mesures efficace et robuste, qui est capable de prioriser les défauts à trouver selon la criticité et permettre d'en trouver le maximum possible. Il est aussi important que tous les intervenants dans les activités de test aient un minimum d'acquis concernant les mesures et en aient reçu la formation nécessaire.

3.2.7 L'absence d'historique d'incidents

Lorsque l'organisation ne dispose pas d'un mécanisme pour conserver la traçabilité des tests antérieurs et l'historique des leçons apprises concernant les tests via les différents projets réalisés, les problèmes et les difficultés déjà rencontrées, par les testeurs, risquent de se reproduire. Toute l'équipe perdra ainsi du temps à les résoudre et à refaire un travail pour

lequel ils y ont investi de l'effort auparavant. L'absence de rapports d'incidents liés aux projets achevés, ou le manque d'étude de ces rapports, au début des nouveaux projets, fait perdre à l'organisation l'opportunité d'amélioration du processus de test.
Il est important à archiver les leçons apprises de chaque projet de développement, mais aussi il important de commencer avec le début de chaque nouveau projet. Car la procrastination de la mesure seulement à la fin du cycle résultera en un historique faible en détails et incapable d'expliquer les incidents futurs.

Afin d'éviter ces problématiques, les gestionnaires de l'organisation peuvent imposer la documentation des acquis retenus des problèmes rencontrés, et l'incorporer explicitement dans le processus de test. Ils peuvent aussi rendre la révision de ces rapports une étape principale au moment de démarrage du projet.

3.2.8 Le manque de priorisation

Pour être efficace, un test logiciel doit éliminer, en premier, les défauts les plus importants en termes de conséquences de défaillance, pour ensuite éliminer ceux de faible impact. Il est donc important d'avoir une priorisation des tests, sans laquelle il est possible de se retrouver dans des situations critiques et face à un calendrier serré avec des défauts majeurs non encore détectés. Il est à noter ainsi que les ressources limitées ont été mal gérées et n'ont pas contribué aux résultats souhaités. Dans d'autres cas, même si les tests sont bien planifiés et listés par ordre de priorité, les testeurs auront tendance à se détacher de la documentation soit par manque d'expertise, soit à cause de la dépendance aux composants du programme non encore prêts et donc l'impossibilité de suivre la planification.

Pour résoudre ces problématiques, il est clair qu'il faut ordonnancer les défauts recherchés par ordre d'importance et les inclut dans le plan de test. Cela peut aider les testeurs à être plus organisés et plus productifs. Mais pour que toutes les parties prenantes puissent travailler en cohérence, cette priorisation doit être effectuée pour commencer par les parties

de système qui ont plus de liaisons avec ces défauts. De cette façon, pas seulement les testeurs seront concernés, mais aussi les différents départements de développement. Et il n'y aura pas de décalage entre les équipes, comme il aura une concordance et une exploitation optimisée des ressources.

3.3 Conclusion

Ce chapitre a souligné les problématiques de tests logiciels qui vont être résolues par l'application du processus proposé dans cette recherche. Ce processus d'ingénierie QETP va être décrit dans le chapitre qui suit.

CHAPITRE 4

LE PROCESSUS D'INGÉNIERIE QETP

4.1 Introduction

Ce chapitre est dédié à présenter le processus d'ingénierie de qualité de test proposé, expliquer sa composition et son fonctionnement, ainsi que souligner ses avantages et son apport potentiel en matière de qualité.

4.2 La collecte des modèles

Afin de parvenir à la création d'un modèle d'ingénierie de qualité intégrable dans la phase test du SDLC, nous devons choisir le modèle de STLC sur lequel le processus va être bâti, ainsi que les différents modèles de qualité.

4.2.1 Le processus de test STLC

Le cycle de vie de test logiciel fait référence à un processus de test permettant de garantir que les fonctionnalités et les objectifs de qualité ont été respectés. Dans le procédé STLC, chaque activité est réalisée d'une manière planifiée et systématique. Chaque phase a des objectifs différents et des livrables. Différentes organisations ont différents STLC, bien que le fondement reste toujours le même.

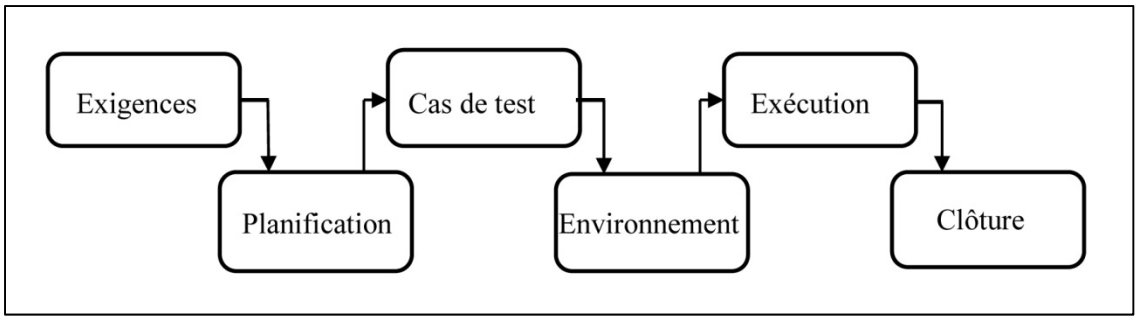

Figure 4.1 Cycle de vie général de test logiciel

La première phase d'analyse des exigences permet de déterminer la portée du test. Si une caractéristique n'est pas testable, il faut la communiquer au cours de cette phase de sorte que la stratégie d'atténuation peut être planifiée.

Dans les scénarios pratiques, la planification des tests est la première étape du processus. Dans cette phase, nous identifions les activités et les ressources qui permettent d'atteindre les objectifs de test. Lors de la planification, nous essayons également d'identifier les mesures et leur méthode de collecte et de suivi.

La tâche principale dans le STLC est la phase de création des cas de test détaillés. Avant de les finaliser, il est important de les examiner pour assurer leur exactitude. Si le projet implique l'automatisation, il faut identifier les cas de test candidat pour l'automatisation et procéder à les rédiger les.

Comme son nom l'indique, la phase d'exécution est où le test proprement dit se déroule. Mais avant de commencer l'exécution, il faut s'assurer que les critères d'entrée sont remplis. Et lorsqu'on exécute les cas de test, il faut souligner les défauts dans les cas de divergence, ainsi que remplir les mesures de traçabilité pour suivre les progrès.

Les activités de clôture de test consistent généralement à vérifier que tous les cas de test sont exécutés et de créer le document des leçons apprises et y inclure ce qui a bien passé et ce qui peut être amélioré.

4.2.2 La normalisation de STLC

Afin de standardiser et normaliser ce cycle de vie de test logiciel, la norme ISO 29119-2 est venue le présenter dans un ensemble des processus imbriqués qui expliquent toutes les phases et les étapes de test.

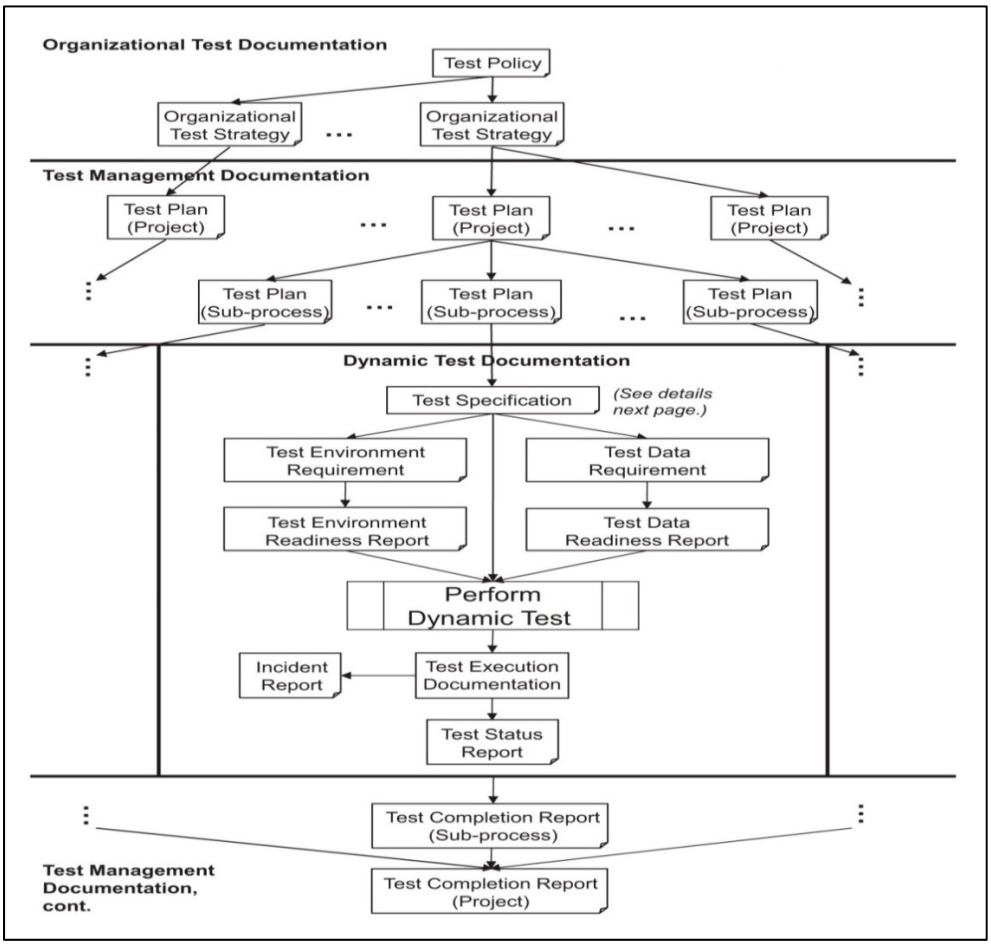

Figure 4.2 Processus de la documentation de test logiciel ISO 29119-3
Tirée de (ISO 29119-3, 2013, p. 2)

La norme ISO 29119-3 de la documentation des tests est venu ensuite compléter et renforcer celle des processus, et puisqu'elles sont toutes les deux complémentaires et en parfaite cohérence, elles sont prêtes à l'exploitation et peuvent être intégrées aisément au sein de l'organisation de développement.
La documentation de test adopte donc aussi le concept de multicouches, et comme la figure l'indique, elle peut être classée en documentation organisationnelle de test, documentation de gestion de test et documentation de test dynamique.

La documentation de l'organisation de test comporte principalement deux fichiers

- La politique de test
- La stratégie organisationnelle de test

Alors que le deuxième lot de documents comprend

- Le plan de test
- Le rapport de statut de test
- Le rapport de la complétion de test

Et le dernier ensemble des livrables est composé de

- La spécification de la conception de test
- La spécification des cas de test
- La spécification de la procédure de test
- Les exigences des données de test
- Les exigences de l'environnement de test
- Le rapport de l'état de préparation de l'environnement de test
- Les résultats actuels
- Document d'exécution de test
- Rapport d'incidents de test

4.2.3 Le STLC et la qualité

De nombreux modèles ont été conçus pour la représentation de la qualité du produit logiciel, et parmi les plus connus il y a le modèle McCall, Boehm, Dromey et ISO/IEC 9126. Mais c'est le modèle de qualité ISO/IEC 25010 qu'on va adopter pour la réalisation du processus de qualité pour la phase test.
On va opter aussi pour l'utilisation de la norme internationale ISO 25001 de planification et de gestion qui vise à clarifier les concepts de gestion de l'évaluation et les exigences qui doivent être déterminées par l'organisation en vue d'assurer le succès de la spécification des exigences de qualité et l'exécution de l'évaluation.

On utilisera aussi dans le modèle, la norme ISO 25040 qui établit la relation entre le modèle de référence d'évaluation et les documents de la série SQuaRE et montre comment chacun devrait être utilisé pendant les activités du processus d'évaluation. Ainsi que la norme ISO 25041 spécifique aux développeurs, acquéreurs et évaluateurs indépendants et ayant pour rôle la présentation des exigences et des recommandations pour l'évaluation de la qualité des produits.

4.3 Le processus QETP

Dans cette partie du chapitre, la construction de processus de test va être décrite, de processus de qualité de test et de leur interrelation, pour aboutir à la fin au processus de l'ingénierie de qualité de test intégré.

On va aussi présenter l'ensemble des documents et des livrables qui interfèrent lors du processus de qualité de test.

4.3.1 Le processus de test logiciel

Cette section présente un processus de test logiciel inspiré de la norme ISO 29119-2. Comme pour le modèle de la norme, le processus de test proposé respecte la présentation en trois couches, à savoir la couche organisationnelle, la couche de gestion et la couche de test dynamique, et garde le même flux de relations entre les activités que dans le processus de l'ISO 29119-2. Ce processus est discuté étape par étape de sorte à montrer comment le QETP proposé peut y être intégré.

4.3.1.1 Organisation de test

Le rôle de cette phase est de définir un processus pour la création et la maintenance des spécifications de test organisationnelles, telles que les politiques organisationnelles de test, les stratégies, les processus et les procédures.

4.3.1.2 Gestion et planification de test

La gestion de test s'applique séparément aux différents types et phases de test sur la base d'un plan de test. Ce dernier est le résultat d'une planification de test qui se fait pour le test de projet entier, comme pour chaque phase de test.

Le plan de test peut avoir besoin d'être modifié en réponse aux résultats de sa mise en œuvre et de nouvelles informations qui sont disponibles. Pour cela il y a un lien récursif sur la figure.

Cette phase doit être aussi alignée avec les sorties de la phase qui la précède et peut produire une rétroaction sur elle en cas de besoin de modification de la stratégie par exemple, ce qui explique les liens avec la phase de l'organisation de test sur la figure. Et le plan de test qu'elle génère est transmis au processus suivant.

4.3.1.3 Implémentation de test

Dans le cadre de cette phase, les caractéristiques sont regroupées en ensembles de fonctionnalités, les conditions, les cas et les procédures de test sont dérivées et les jeux de tests sont assemblés.

À la sortie de l'implémentation de test, les exigences en données et les exigences d'environnement sont respectivement fournies pour entamer les phases d'exécution et de mise en œuvre d'environnement de test.

4.3.1.4 Environnement de test

Les exigences d'un environnement de test sont d'abord décrites dans le plan de test, mais leur composition détaillée s'éclaircit une fois que l'implémentation de test est établie.

Le but de cette phase est d'établir et de maintenir l'environnement de test requis et de communiquer son statut à toutes les parties prenantes concernées. Et la réalisation de ce but en soi constitue une entrée pour l'exécution de test.

4.3.1.5 Exécution de test

Cette phase a pour rôle d'exécuter les procédures de test générées à la suite de la phase d'implémentation sur l'environnement de test mis en place, et c'est la traduction des relations d'entrée représentées sur la figure.

Un bon déroulement de l'exécution de test signifie que les procédures de test sont exécutées et les résultats réels sont enregistrés et comparées aux celles attendues.

4.3.1.6 Clôture de test

À ce niveau du processus de test, la déduction et déclaration des incidents se font suite à l'identification des échecs et des cas où quelque chose d'inhabituel ou inattendu a eu lieu lors de l'exécution de test. Et la rédaction de ces rapports d'incidents est considérablement bénéfique à la gestion et planification des autres tests ou même des autres projets.

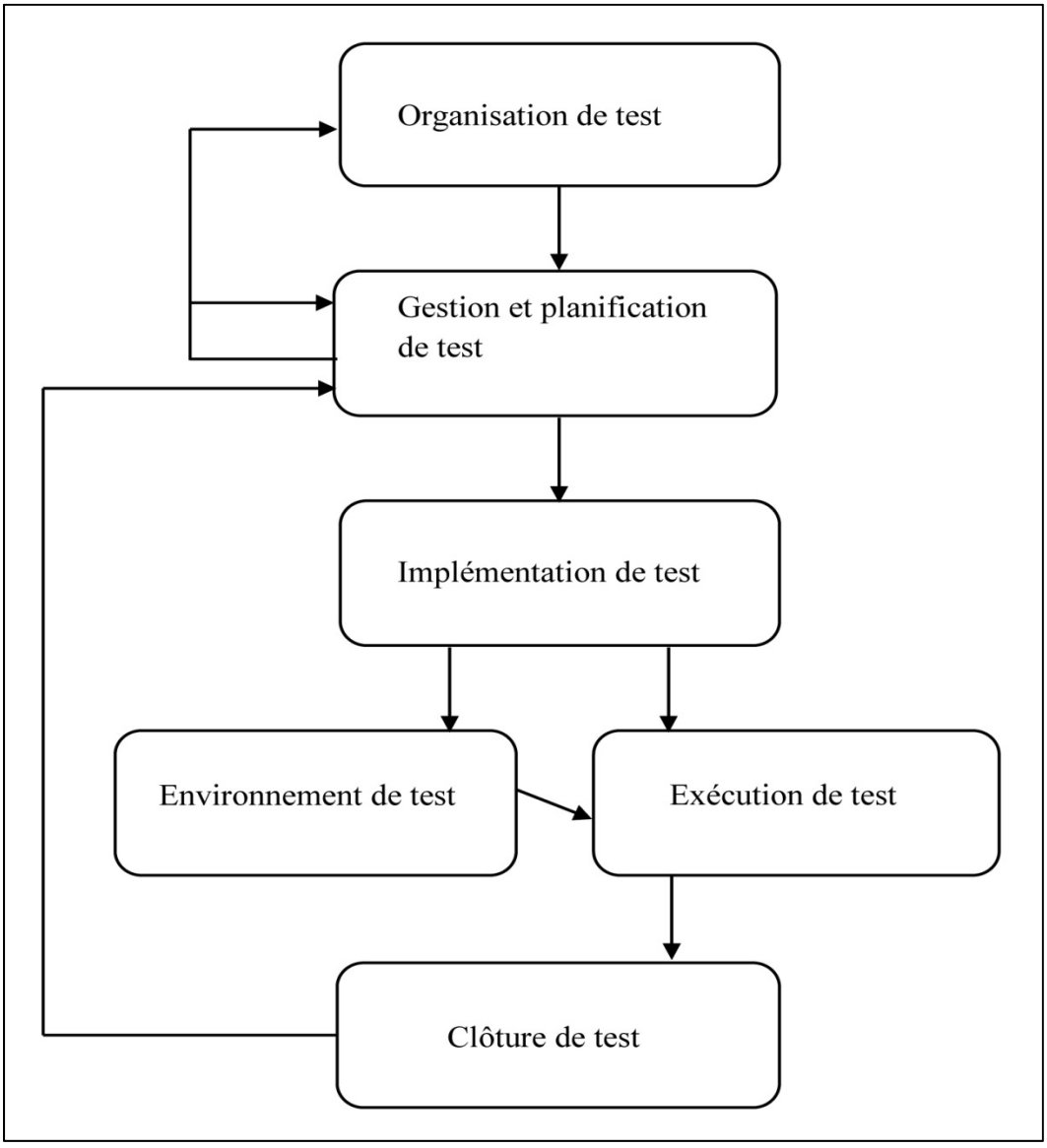

Figure 4.3 Processus de test logiciel

4.3.2 Le processus d'ingénierie de qualité de la phase test

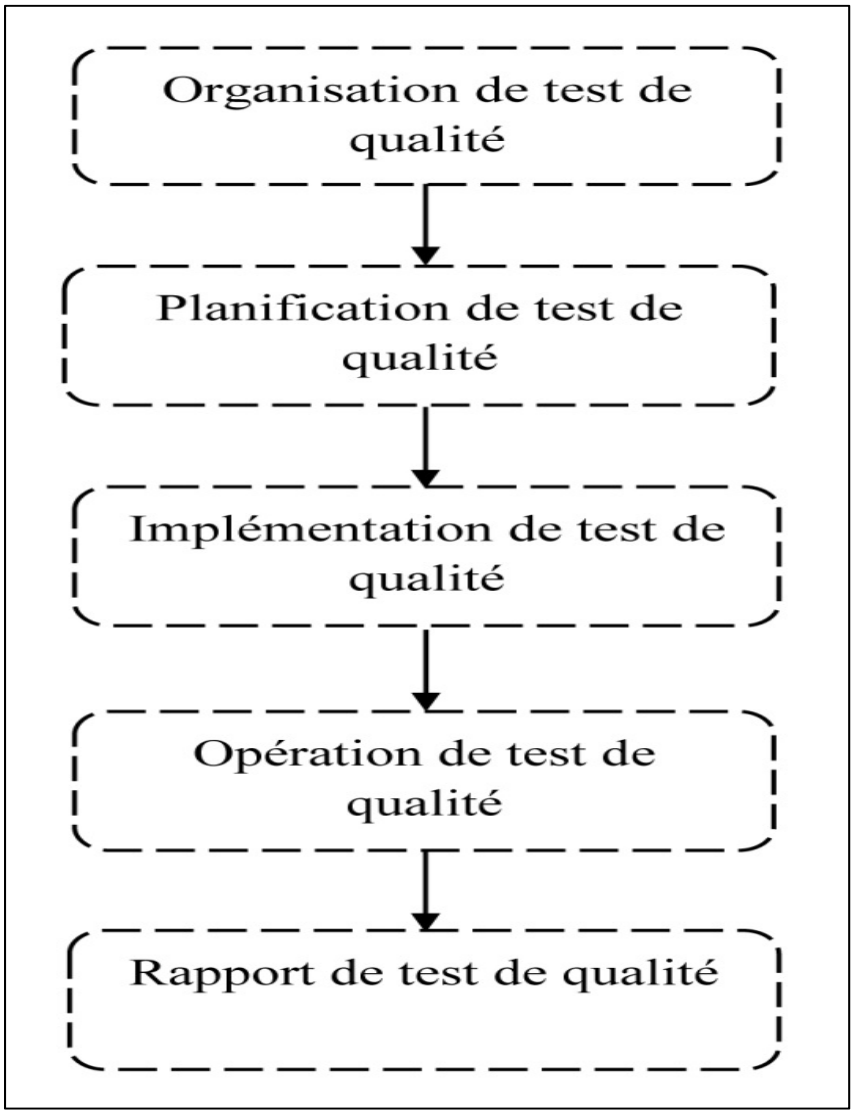

Figure 4.4 Processus d'ingénierie de qualité de la phase test

4.3.2.1 Organisation de test de la qualité

Cette activité consiste au fait que l'ingénieur responsable de la qualité doit intervenir lors de la détermination des stratégies et politiques de test, et vérifier s'ils sont en faveur des caractéristiques de qualité, sinon les discuter.

4.3.2.2 Planification de test de la qualité

À cette étape, l'ingénieur qualité doit accompagner la création de plan de test par le groupe de développement et collecter les caractéristiques de qualité et leurs mesures adéquates pour créer un plan de qualité de test.

4.3.2.3 Implémentation de test de la qualité

Parallèlement avec la création des cas de test pour les fonctionnalités, l'équipe responsable de la qualité doit implémenter les scénarios des jeux de test pour la vérification et la validation des exigences de la qualité.

4.3.2.4 Opération de test de la qualité

Dans cette activité, l'ingénieur qualité est responsable de l'exécution des procédures de test concernant les caractéristiques de qualité qui ont été issues de la phase de l'implémentation de qualité. Comme il doit observer et enregistrer les résultats réels et les comparer à celles attendues.

4.3.2.5 Rapport de test de la qualité

En arrivant au niveau de la clôture de test, le responsable qualité a pour rôle de générer un rapport de qualité de test, dans lequel il a empilé les résultats d'exécution de plan de qualité, les caractéristiques de qualité satisfaites, celles corrigées ou modifiées et celles qui n'ont pas pu être réalisées; ainsi que tous les incidents rencontrés.

4.3.3 Le processus d'ingénierie de qualité intégré

Il est parfaitement clair que pour garantir une bonne qualité de produit logiciel, il faut d'abord valoriser les efforts de l'ingénieur qualité et lui donner la place et les ressources qui

le permettent de mener au bien son travail. Comme il faut s'assurer qu'il travaille en coopération et en cohérence avec l'équipe de développement.

De ce constat, il est possible d'affirmer la nécessité de l'intégration du processus de l'ingénierie de qualité dans la phase test de cycle de vie de développement logiciel. Et l'apport du parallélisme et de la fusion de deux processus est reconnu.

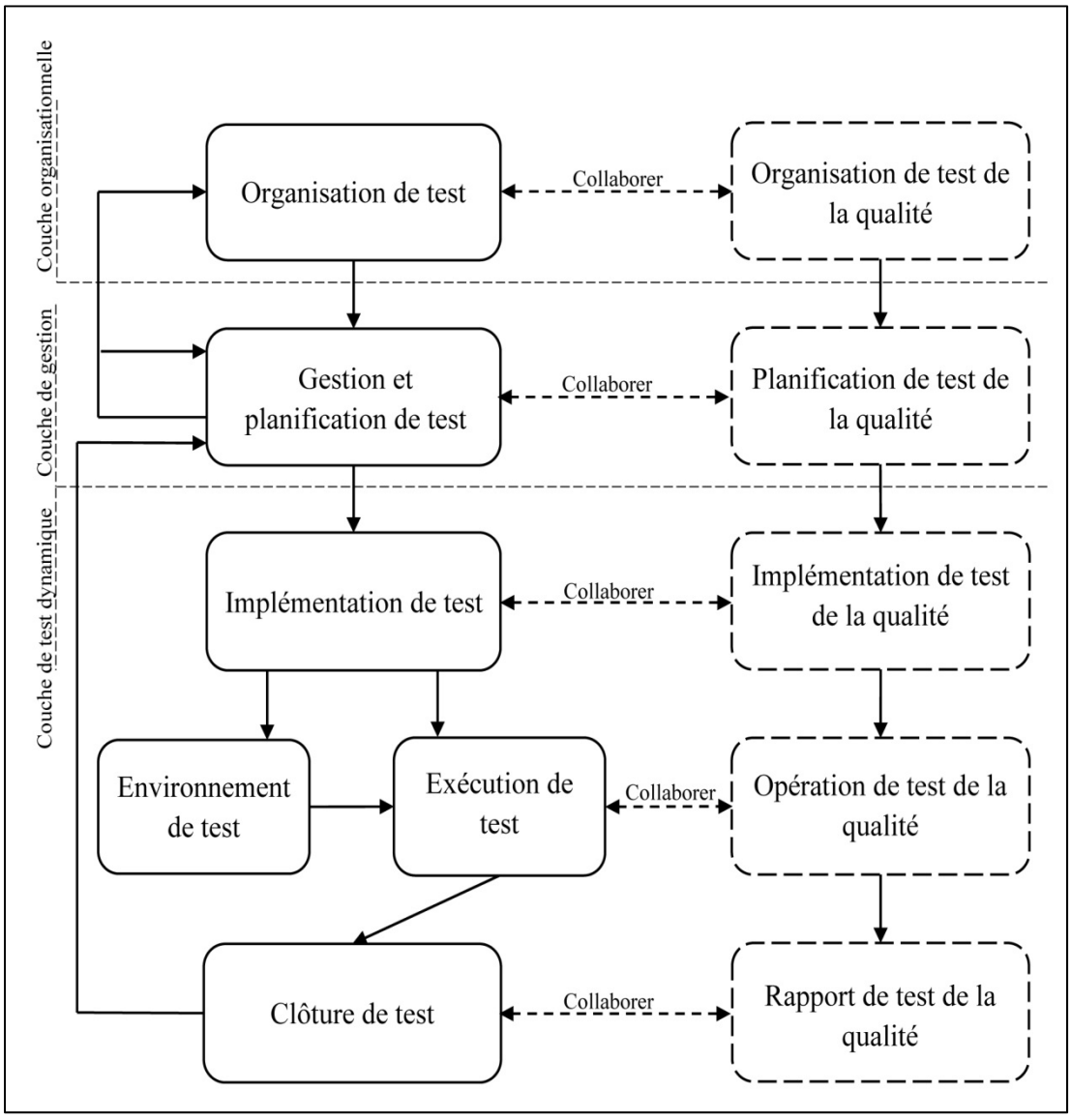

Figure 4.5 Processus d'ingénierie de qualité intégré de la phase test

On remarque que ce processus respecte bel et bien la présentation à trois couches de la norme ISO 29119-2 dont il est inspiré, comme il assure parfaitement l'intégration de l'ingénieur qualité au sein de l'organisation et permet aux équipes de développement et d'assurance qualité de collaborer dans la quasi-totalité des tâches de la phase test afin d'atteindre une qualité de produit maximale et un processus de test de haut niveau.

4.3.4 La documentation de QETP

On doit signaler que tout au long le déroulement de test, et à côté des rapports et documents de test accumulés et des normes et standards utilisés, il y a un ensemble de documents concernant le processus de qualité et générés en parallèle.

Tableau 4.1 Les normes utilisées à travers le processus QETP

ISO 25001
ISO 25010
ISO 25030
ISO 25040
ISO 25041
ISO 29119-3
ISO 29119-4

La figure 5.6 récapitule la relation de processus de qualité de test avec les documents fournis par le processus de test et l'ensemble des normes et standards nécessaire l'exécution de ses

différentes activités. Comme il présente les livrables générés au long de ce processus et qui sont :

4.3.4.1 Politique de test de la qualité

C'est un document qui négocie les politiques de test et précise si elles sont au profit de processus et de caractéristiques de qualité. Sinon, il présente des propositions qui peuvent être prises en considération par l'équipe de développement afin d'arriver à un consensus qui arrange les deux parties.

4.3.4.2 Plan de test de la qualité

Ce document contient la planification de test pour la vérification des caractéristiques de qualité, et les techniques et mesures qui permettent l'arrivée à cette fin.

4.3.4.3 Cas de test de la qualité

Le document des cas de test de qualité consiste à l'ensemble des jeux de test qui à travers leur exécution il est possible de déterminer les caractéristiques de qualité qui ont été satisfaites.

4.3.4.4 Rapport de test de la qualité

C'est le livrable qui rassemble les résultats de déroulement du processus de qualité. Et qui contient les objectifs atteints et ceux manquants. Ce document doit aussi contenir les commentaires concernant l'amélioration du processus et des résultats.

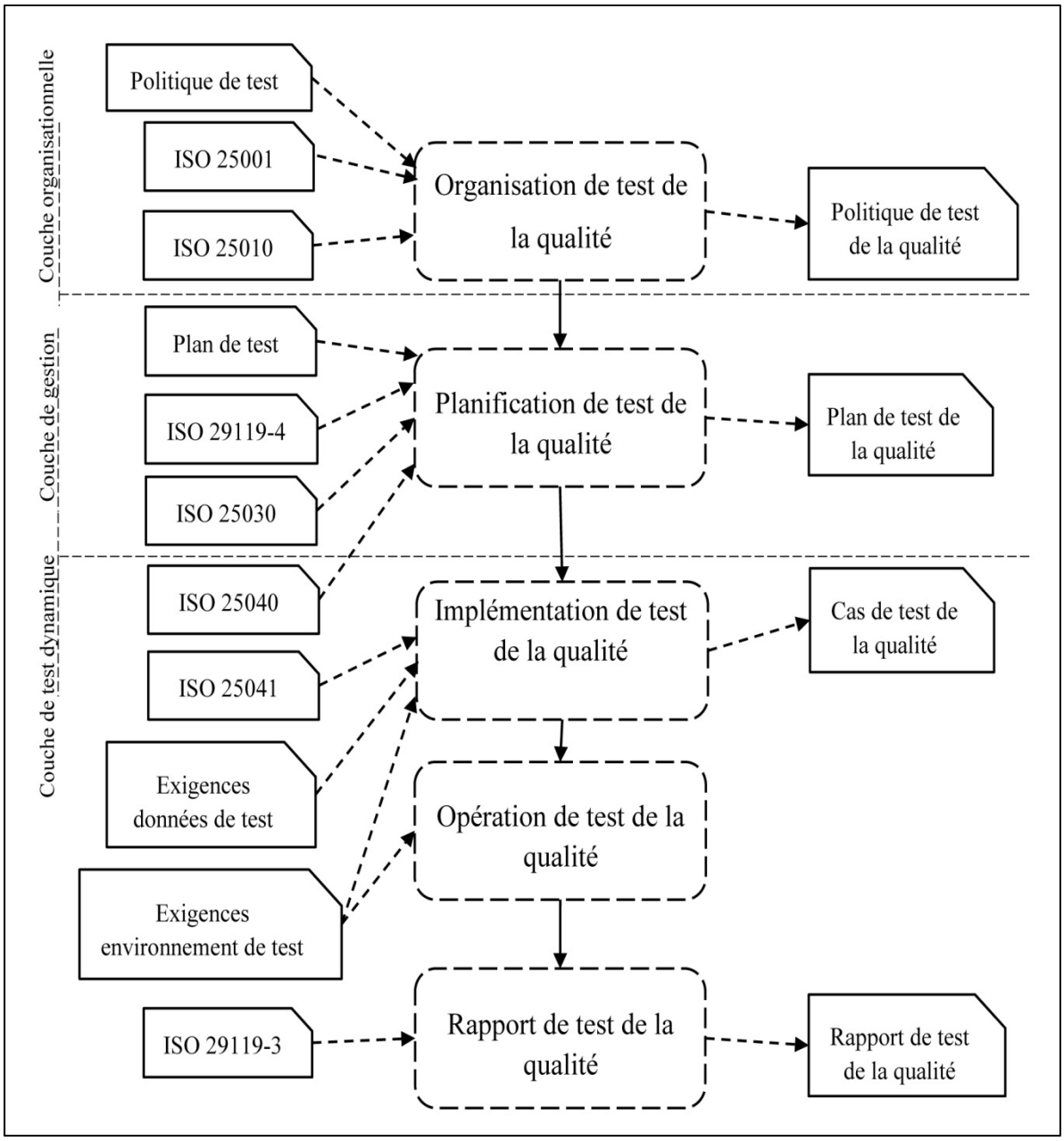

Figure 4.6 Mouvement des documents dans le QETP

4.4 Les caractéristiques du QETP

Le processus d'ingénierie de qualité de test proposé dispose d'un certain nombre des caractéristiques:

4.4.1 Généralité

Le QETP est générique de sorte qu'il peut être appliqué pour n'importe quel type de projet de développement et tous les domaines de production logicielle.

4.4.2 Adaptation

Il est aussi adaptable indépendamment de la politique interne de l'organisation. Il peut alors être intégré dans les PME, comme pour les entreprises de grande envergure.

4.4.3 Facilité

Ce processus est facile à utiliser, car il ne nécessite pas une formation spéciale et il est compréhensible par toute personne qui possède les acquis de base de développement et de test logiciel.

4.4.4 Normalisation

La haute qualité de ce processus est assurée puisqu'il est basé sur les plus récentes normes internationales dans le domaine, et les mieux appréciées.

4.4.5 Interaction

Ce processus de qualité est en échange constant avec le processus de test, et en interaction avec la quasi-totalité des processus de développement du projet en cours de réalisation.

4.5 L'apport du QETP

Le processus d'ingénierie de la phase test QETP proposé, décrit précédemment, met clairement en valeur le rôle crucial de l'ingénierie de la qualité dans la phase des tests du cycle de vie de développement logiciel. Comme il valide l'hypothèse de cette recherche et présente les solutions nécessaires à résoudre les majeurs soucis du test logiciel.

Ce processus de test rénové insiste sur l'importance d'avoir une planification claire et détaillée pour le test, et opte pour l'utilisation d'un plan de qualité à côté du plan de test. Il s'oppose aux conséquences néfastes des tests retardés en abordant les activités de test très tôt dans SDLC. Comme il résout le problème de dépendance de testeurs aux équipes de développement en instaurant une nouvelle relation de coopération et de complicité. Le processus QETP permet également d'éviter un ensemble des problématiques grâces à l'utilisation des normes et des standards déjà vérifiés et validés, ainsi qu'il permet la bonne gestion des outils et des ressources.

4.6 Conclusion

À travers ce dernier chapitre, le processus d'ingénierie de qualité intégré dans la phase test QETP est représenté et décrit, ainsi que sa documentation, ses caractéristiques et son apport souhaité en matière de qualité.

CONCLUSION

À terme de ce travail, l'objectif initial était de créer un processus d'ingénierie de qualité QETP qui peut être intégré au sein de la phase test du cycle de vie de développement logiciel.

Ce but a été bel et bien atteint. Et les principales contributions de cette recherche sont, premièrement, la réalisation d'un modèle d'ingénierie de qualité à cinq étapes qui côtoient le déroulement des activités de test logiciel, tout en assurant une haute qualité du processus de test et du produit testé. Et cette qualité est garantie essentiellement grâce à un ensemble des normes connues et validées qui ont servi à la construction du QETP et demeurent nécessaires pour son application.

Deuxièmement, le QETP est intégré dans le processus de test logiciel (qui a été adapté en ayant recours aux normes et standards internationaux). Cette intégration a permis d'avoir un seul processus pour la réalisation des tests, et le suivi de sa qualité. Cela permet de rassembler les efforts, regrouper les responsables de la qualité et les équipes de développement, dans le but d'éviter les procédés parallèles et la perte du temps et des ressources.

La troisième contribution de ce travail de recherche se résume au fait que les problématiques du test logiciel qui peuvent affecter la qualité du produit ou la diminuer ont été identifiées. L'identification de ces lacunes permet aux praticiens, et aux organisations de localiser les défauts dans leurs politiques et méthodes de travail, et par la suite les contourner.

Finalement, et après une mise en pratique prévue, il reste à en déduire que l'adoption de QETP maximise l'apport issu des efforts et acquis de l'ingénieur qualité et lui facilite l'intégration dans le groupe et la coopération avec l'équipe de développement. Ce qui favorise, nettement, la bonne gestion des ressources et la bonne qualité du produit.

Bien que ce travail a permis de concrétiser l'objectif de la recherche (l'intégration de l'ingénierie de la qualité dans la phase test), une application pratique du QETP reste fortement recommandée afin de pouvoir quantifier ce bénéfice et évaluer le processus.

Et parce que toutes les phases de cycle de vie logiciel participent à l'amélioration de la qualité du produit et non seulement le test, même s'il reste d'une importance majeure, des pratiques de l'ingénierie de qualité dans tout le processus de développement est recommandée. L'objectif sera donc d'obtenir un processus d'ingénierie de qualité intégré dans le SDLC en entier, et qui couvre ses différentes phases.

BIBLIOGRAPHIE

Firesmith, Donald G. 2013. Common System and Software Testing Pitfalls: How to Prevent and Mitigate Them: Descriptions, Symptoms, Consequences, Causes, and Recommendations. Addison Wesley, 320 p.

Glenford, J. Myers, Corey Sandler et Tom Badgett. 2012. The Art of Software Testing, 3e éd. Hoboken, N.J : John Wiley & Sons, 240 p.

Hetzel, Bill. 1998. The Complete Guide to Software Testing, 2e éd. New York : John Wiley & Sons, 284 p.

International Organization for Standardization. 2005. Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. ISO/IEC FDIS 25000. International Organization for Standardization, 43 p.

International Organization for Standardization. 2007. Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements. International standard, ISO/IEC 25030:2007. International Organization for Standardization, 34 p.

International Organization for Standardization. 2007. Systems and software engineering — Software life cycle processes. International standard, ISO/IEC FDIS 12207:2007 IEEE Std 12207-2007. International Organization for Standardization, 122 p.

International Organization for Standardization. 2010. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. International standard, ISO/IEC FDIS 25010:2010. International Organization for Standardization, 34 p.

International Organization for Standardization. 2011. Ingénierie du logiciel — Profils de cycle de vie pour très petits organismes. Rapport technique, ISO/CEI TR 29 110-5-1-2:2011. International Organization for Standardization, 44 p.

International Organization for Standardization. 2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process. ISO/IEC FDIS 25040. International Organization for Standardization, 48 p.

International Organization for Standardization. 2012. Software and Systems Engineering — Software Testing — Part 1 : Concepts and Definitions. ISO/IEC DIS 29119-1. International Organization for Standardization, 57 p.

International Organization for Standardization. 2012. Software and systems engineering -- Software testing -- Part 3 : Test documentation. ISO/IEC/IEEE 29119-3:2013. International Organization for Standardization, 127 p.

International Organization for Standardization. 2012. Systems and Software Engineering — Software Testing — Part 4 : Test Techniques. ISO/IEC CD-3 29119-4. International Organization for Standardization, 137 p.

International Organization for Standardization. 2012. Software and Systems Engineering — Software Testing — Part 2 : Test Processes. ISO/IEC/IEEE DIS 29119-2.2 : 2012. International Organization for Standardization, 64 p.

International Organization for Standardization. 2012. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation guide for developers, acquirers and independent evaluators. International standard, ISO/IEC 25041:2012. International Organization for Standardization, 52 p.

International Organization for Standardization. 2014. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Planning and management. International standard, ISO/IEC 25001:2014. International Organization for Standardization, 13 p.

International Organization for Standardization. 2014. Systems and software engineering — System life cycle processes. ISO/IEC/IEEE FDIS 15288:201x. International Organization for Standardization, 108 p.

Laporte, Y. Claude, Alain April. 2011. L'assurance qualité logicielle, 2e éd. Paris : Hermès, 373 p.

Lewis, William E. 2009. Software Testing and Continuous Quality Improvement, 3e éd. Boca Raton : Auerbach Publications, 704 p.

O'Regan, Gerard. 2014. Introduction to Software Quality. Cham : Springer International Publishing, 369 p.

P. Bourque et R.E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014.

Perry, E. William. 2000. Effective Methods for Software Testing, 2e éd. New York : John Wiley & Sons, 812 p.

Schulmeyer, G. Gordon. 2008. Handbook of Software Quality Assurance, 4e éd. Norwood, Mass. : Artech House, 485 p.

Software testing and ISTQB. 2015. « 7 Fundamentals Principles of Software Testing as per ISTQB ». In Software testing and ISTQB. En ligne. < http://www.softwaretestingandistqb.com/7-fundamental-principles-of-software-testing-as-per-istqb/ >. Consulté le 31 Mars 2016.

Suryn, Witold et Girard, Daniel. 2005. Suryn-Abran consolidated quality lifecycle (CQL) model - The applicative evolution. In Business Information System 2005 - 8th International Conference on Business Information Systems (Poznan, Poland, Apr. 20-22, 2005), p. 126-143. Poznan, Poland : Uniwersytet Ekonomiczny w Poznaniu.

Suryn, Witold, Kahlaoui, Abdelileh et Georgiadou, Elli. 2005. Quality engineering process for the Program Design Phase of a generic software life cycle. In 13th International Conference on Software Quality Management (Cheltenham, UK, Mar. 21-23, 2005), p. 253-266. British Computer Society, Swindon.

Suryn, Witold. 2013. Software Quality Engineering : A Practitioner's Approach. Hoboken: Wiley, 191 p.

Tian, Jeff. 2005. Software Quality Engineering:Testing, Quality Assurance, and Quantifiable Improvement. Hoboken, N.J. : Wiley, 403 p.

Tilley, Scott et Brianna Floss. 2014. Hard Problems in Software Testing : Solutions Using Testing as a Service (TaaS). Morgan & Claypool Publishers, 126 p.

Wagner, Stefan. 2013. Software product quality control. Heidelberg, New York : Springer, 219 p.