ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION D'UN
D.E.S.S. EN TECHNOLOGIE DE L'INFORMATION
D.E.S.S.

PAR
ABBAS, Abderrahmane

ÉVALUATION DE LA MATURITÉ DU PROCESSUS DES TESTS CHEZ TAKTIKA

MONTRÉAL, LE 19 DÉCEMBRE 2008

CE RAPPORT A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE

Dr. Alain April, Superviseur du projet
Département Génie logiciel & Technologie de l'information
École de technologie supérieure

Dr. Roger Champagne,
Département Génie logiciel & Technologie de l'information
École de technologie supérieure

IL A FAIT L'OBJET D'UN DÉPÔT

LE 19 DÉCEMBRE 2008

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier tout d'abord mon directeur de recherche, **Alain April**, à qui je dois toute ma gratitude de m'avoir accepté en tant qu'étudiant de DESS, et surtout de m'avoir fait confiance à la réalisation de ce rapport. Je me souviens très bien de notre première rencontre, tout de suite, elle a pris un chemin très amical et harmonieux. On y découvre sans aucune explication que ce qui peut être considéré souvent comme une corvée peut se transformer en plaisir avec son lot de satisfaction et de bonne ambiance. Alain April, un homme très accueillant et chaleureux. Je lui dis encore mille fois merci pour ses précieux conseils.

Mohamed Taleb, lui aussi, je le remercie beaucoup de m'avoir aidé, guidé et contribué à la réalisation de mon travail au laboratoire de GELOG. Notre relation a pris rapidement une trajectoire harmonieuse, amicale et solide. Grâce à lui, j'ai pu apprendre et comprendre ce qu'est la recherche. J'avoue que ce n'était pas facile parce qu'il faut vraiment avoir beaucoup de patience. Pour tout ça, je lui dois reconnaissance et sympathie.

Je voudrais également remercier spécialement **Alpha Diallo** qui a contribué à l'élaboration de ce travail en m'ouvrant grandes les portes de l'entreprise TAKTIKA et en me donnant de son précieux temps.

Grand merci, très spécial, à ma mère, à mon père, à mes frères et sœurs et leurs petites familles et tous les autres membres de ma famille de m'avoir enduré et supporté durant toute ses années d'études et surtout de m'avoir encouragé moralement et financièrement. Je leur dois toute ma reconnaissance et merci encore. Je leur dédie ce rapport comme preuve de mon amour.

ÉVALUATION DE LA MATURITÉ DU PROCESSUS DES TESTS CHEZ TAKTIKA

ABBAS, Abderrahmane

RÉSUMÉ

La croissance de l'information dans notre société est l'élément déclencheur de la plupart des changements majeurs dans l'organisation du travail. En effet, l'informatisation des entreprises privées et des organismes publics et parapublics ont conduit à une réingénierie des processus d'affaires au sein de l'entreprise. Les entreprises et les gouvernements embrassent l'ère de l'informatique, car ceci leur permet de réduire considérablement leur budget et mettre plus l'emphase sur leur mission et accroître leur croissance interne et externe sur les moyens dont ils disposent afin d'acquérir une plus grande part de marché.

L'environnement concurrentiel sans précédent dans lequel nous nous situons fait qu'il s'agit désormais d'une question de survie. L'informatique, en particulier le logiciel, a évidemment un rôle à jouer pour y parvenir, en permettant à l'entreprise de devenir plus apprenante et d'avoir une meilleure connaissance de ses clients, de sa compétitivité ou de son environnement. C'est la raison pour laquelle le logiciel doit, en plus du respect des délais et budgets du développement, offrir une haute qualité afin de répondre, entre autres, aux besoins de performance, de fiabilité et de sécurité.

De nombreux travaux ont été élaborés dans le domaine de la qualité logicielle. Divers aspects de la qualité en génie logiciel ont été introduits et portés essentiellement sur : 1) le produit logiciel; et 2) le processus de développement logiciel. À cet effet, plusieurs modèles de maturité ont été développés. Ce travail se focalise principalement sur le processus des tests et le modèle de maturité des tests.

Ce rapport vise à proposer une évaluation de la maturité des tests de l'entreprise TAKTIKA. Cette évaluation est basée sur le modèle de maturité des tests appelé TMM (*Testing Maturity Model*).

Ce document est divisé en trois chapitres. Le premier chapitre présente l'état de l'art sur les tests logiciels en général et sur le modèle de maturité du processus des tests. Le deuxième chapitre va porter sur la conception de l'outil d'investigation fondé sur les travaux de Burnstein et son application dans l'entreprise Taktika. Le troisième et dernier chapitre présente l'interprétation des résultats, l'évaluation des forces et faiblesses de l'approche, les travaux futurs et la conclusion.

Mots-clés: Processus, test, défaillance, qualité du logiciel, cycle de développement de logiciels, TMM, niveau de maturité, évaluation de la maturité.

ÉVALUATION DE LA MATURITÉ DU PROCESSUS DES TESTS CHEZ TAKTIKA

ABBAS, Abderrahmane

ABSTRACT

The growth of information in our society is a trigger for major changes in organizations. Indeed, the computerization of private companies and government agencies has forced reengineering of business processes. Companies and governments embraced the computer age, because it allows them to reduce their costs, concentrate on their mission and increase their productivity allowing greater market share.

The unprecedented competitive environment, in which we are, shows that using technology it is key to survival. Software has a growing role to play in achieving this, allowing the organizations to learn faster and gain a better understanding of its customers, its competitiveness and its environment. Software must be of high quality in order to help the organizations.

Many publications present the challenges of software quality. The two main perspectives are: 1) software product quality and 2) software process quality. Software process quality debates which best practice should be used to produce good software products. Many process models present best practices in the form of a maturity model. This research experiments with the concepts of software testing process and testing maturity model.

More specifically this research aims to provide a software testing process based on an existing maturity model named TMM (*Testing Maturity Model*). We will then use this testing process to assess the maturity of the testing processes used in a company named TAKTIKA.

This research is divided into three chapters. The first chapter presents the state of the art in software testing and the testing maturity model. The second chapter presents the proposed testing maturity assessment process (based on the TMM) followed by a case study applied to TAKTIKA's current testing process. The third chapter presents the results of the testing maturity assessment followed by suggestions for improving the maturity of their testing process. The last chapter presents a conclusion on this research and future work.

Key words: Process, testing, Failure, Software Quality, Software Development Cycle, TMM, Maturity Level, Maturity Assessment.

TABLE DES MATIÉRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET ACRONYMES

ART	Activity, Task and Responsibility
CAF	Capability maturity model Appraisal Framework
CFTL/ISTQB	Comité Français des Tests Logiciels/ International Software Testing Qualifications Board
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
COTS	Commercial Of The Shelf
DoD	Department of Defense
ÉTS	École de Technologie Supérieure
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers, Inc
IIT	Illinois Institute of Technology
ISO	International Organization for Standardization
IT	Information Technology

RP	Rapport de Problèmes
SEI	Software Engineering Institute

SPICE	Software Process Improvement and Capability dEtermination
SWEBOK	Software Engineering Body of Knowledge
S3M	Software Maintenance Maturity Model
TMM	Testing Maturity Model
TMM-AM	Testing Maturity Model-Assessment Model
UQAM	Université du Québec À Montréal

INTRODUCTION

La croissance de l'information dans notre société est l'élément déclencheur de la plupart des changements majeurs dans l'organisation du travail. En effet, l'informatisation des entreprises privées et des organismes publics et parapublics ont conduit à une réingénierie des processus d'affaires au sein de l'entreprise. Les entreprises et les gouvernements embrassent l'ère de l'informatique, car ceci leur permet de réduire considérablement leur budget et mettre plus l'emphase sur leur mission et accroître leur croissance interne et externe sur les moyens dont ils disposent afin d'acquérir une plus grande part de marché.

L'environnement concurrentiel sans précédent dans lequel nous nous situons fait qu'il s'agit désormais d'une question de survie. L'informatique; en particulier le logiciel, a évidemment un rôle à jouer pour y parvenir, en permettant à l'entreprise de devenir plus apprenante et d'avoir une meilleure connaissance de ses clients, de sa compétitivité ou de son environnement. C'est la raison pour laquelle le logiciel doit, en plus du respect des délais et budgets du développement, offrir une haute qualité afin de répondre, entre autres, aux besoins de performance, de fiabilité et de sécurité.

A. Problématique

On entend par logiciel de qualité; un logiciel livré dans les délais, dans les budgets et avec toutes les spécifications du client/utilisateur. Dans le génie logiciel, le processus test représente la dernière ligne de sécurité pour l'atteinte de cette qualité car il représente la dernière phase du cycle du développement, après laquelle le produit sera soumis à l'approbation du client. L'emplacement du processus de tests dans le cycle du développement fait de lui un processus critique qui doit être mature. Pour décider de la maturité des processus en génie logiciel, plusieurs modèles de maturité ont été développés. Leur rôle est d'évaluer ces processus, relever leurs points forts et points faibles pour ensuite guider leur amélioration. Une multitude de modèles a été mise en œuvre pour évaluer le processus des tests de logiciels.

Le Testing Maturity Model (TMM) en un et c'est sur ses outils que nous nous basons pour mener à terme cette étude. Le présent rapport consiste en l'évaluation de la maturité des tests de logiciels de l'entreprise Taktika. Le processus Tests de cette entreprise est analysé sur plusieurs volets tels que la planification, l'organisation, la documentation, la stratégie des tests…etc. cette analyse nous permet de dresser le profile du processus. Sur la base des forces, des faiblesses et du niveau de maturité du processus nous allons émettre des recommandations pour améliorer les activités des tests donc la maturité du processus des tests.

B. À propos du Processus de tests

Le contexte économique de plus en plus concurrentiel et instable a poussé les entreprises à revoir leurs pratiques de travail, et même leurs structures organisationnelles. Pour tenir une place, l'entreprise doit rallier efficacité et souplesse sur tous les niveaux hiérarchiques. Une bonne recette a vu le jour; elle recommande d'adopter le mode de travail par processus et introduire les technologies en particulier les technologies de l'information comme support au processus.

La gestion par processus est venue marquer une nouvelle ère dans la façon de faire des entreprises. Un processus dans sa définition globale consiste en «un ensemble d'activités corrélées ou interactives qui transforment les éléments d'entrée en éléments de sortie» (ISO 9001 :2000). Six paramètres caractérisent un processus; 1) un pilote; qui veille au bon fonctionnement du processus et à son amélioration; 2) des ressources requises; humaines, matérielles et financières; 3) des éléments d'entrée tels les produits et les données; 4) de la valeur ajoutée qui représente la contribution additionnelle du processus dans la réalisation d'un produit ou service; 5) des éléments de sorties tels les produits et les données et 6) un système de mesure et de contrôle.

Dans le domaine du génie logiciel, le processus s'articule sur l'ensemble de méthodes, activités, standards, pratiques, documents et procédures; nécessaire pour le développement

et la maintenance d'un système logiciel (SWEBOK, 2004). Le test est processus du génie logiciel qui regroupe les procédures d'évaluation d'un composant ou système logiciel sur certains aspects tels la performance, la fiabilité, l'utilisabilité, la sécurité…etc.

Autre définition du test : « le test est le processus d'exécution d'un programme ou système dans l'intention de trouver des erreurs» (Myers, 2004). Cette définition restreint l'activité du test à la phase qui suit directement la phase de codage dans le cycle de développement de logiciel or, actuellement tous les travaux dans ce domaine convergent à dire que les tests est une activité qui doit être intégrée à l'ensemble du cycle de vie logiciel. D'ailleurs c'est l'un des quatre objectifs du niveau 3 du TMM.

C. À propos des modèles de maturité

Après avoir introduit les processus à tous les niveaux organisationnels; il a fallu mettre en œuvre des mécanismes pour évaluer leur maturité, les améliorer pour enfin les optimiser et leur procurer une meilleure efficacité. Rappelons par là que la maturité d'une organisation dépond de la maturité des ses processus. Par définition; «la maturité est la capacité d'une organisation par rapport à la rentabilité et l'efficacité de ses processus et pratiques de travail» (Glossaire CFTL/ISTQB, 2005).

Dans le domaine du génie logiciel, de nombreux modèles tels le SPICE, CMM (*Capability Maturity Model*) et CMMI (*Capability Maturity Model Integration*) par la suite, ont été développés pour permettre aux entreprises de mieux évaluer et améliorer leurs processus.

Tous les modèles d'amélioration des processus du génie logiciel qui ont eu une large adoption dans l'industrie sont des modèles de haut niveau, dans le sens ou il traite le processus du génie logiciel en bloc et n'offrent pas un support adéquat pour évaluer des sous-processus du processus de développement logiciel tels la conception et les tests (Burnstein, 2003). Ce manquement fait une des raisons qui ont motivé le développement

d'un modèle complémentaire dit TMM (*Testing Maturity Model*) voué spécialement au processus de tests logiciels.

D. Organisation du rapport

L'organisation de la suite de ce rapport de projet est comme suit:

Dans le chapitre 1, l'état de l'art du domaine des tests de logiciels est présenté. Le processus de tests typique est défini ainsi que ses activités, un aperçu des modèles de maturités CMM, S3M et SPICE est donnée, enfin une description détaillée du modèle de maturité des tests (TMM) est présentée.

Dans le chapitre 2, l'entreprise Taktika est présentée et l'outil d'investigation des activités de test de celle-ci est conçu. Cet outil est fondé sur les travaux de Burnstein et comprend un questionnaire et une procédure d'évaluation.

Dans le chapitre 3, l'interprétation des résultats de l'évaluation est présentée dans laquelle un niveau de maturité est attribué au processus des tests de Taktika, les points fort et les point faibles de ce dernier sont relevés et des suggestions et recommandations d'amélioration sont émises.

Finalement, une conclusion résume les contributions potentielles de ce travail, évalue les forces et faiblesses de l'approche suivie, et présente les futurs travaux.

CHAPITRE 1

ÉTAT DE L'ART

Ce chapitre présente une revue de littérature concernant deux sujets distincts : 1) le processus des tests dans le domaine du génie logiciel et le modèle de maturité des tests tel que décrit par l'auteure Ilene Burnstein dans son livre «*Practical Software Testing : A Process-Oriented Approach*, 2003»

La première section introduit quelques définitions de base du vocabulaire des tests en génie logiciel. La deuxième section présente le processus de tests typique; ses activités, ses niveaux et sa portée dans le cycle de développement du logiciel. La troisième section explore rapidement quelques modèles de maturité connus en génie logiciel et la quatrième section apporte un aperçu sur le modèle de maturité des tests, à savoir le TMM de Ilene Burnstein.

1.1 Définitions de base

Erreurs (*Errors*)

Une erreur, en génie logiciel, est définie comme une action humaine produisant un résultat incorrect (lEEE Std 610.12-1990). C'est donc l'écart entre une valeur ou condition calculée, observée ou mesurée et la valeur ou condition qui est vraie, spécifiée ou théoriquement correcte (Glossaire CFTL/ISTQB, 2005). Notons que dans le domaine du génie logiciel, la composante humaine inclut, les ingénieurs en génie logiciel, les programmeurs, les analystes et les testeurs.

Défauts (*Defaults*)

Un défaut, en génie logiciel, est défini comme une imperfection dans un composant ou un système qui peut conduire à ce qu'un composant ou un système n'exécute pas les fonctions requises, par exemple une instruction ou une définition de données incorrecte. Un défaut, si rencontré lors de l'exécution, peut causer la défaillance d'un composant ou d'un système.

Défaillances (*Failure*)

Selon l'IEEE Std 610.12-1990; une défaillance, en génie logiciel, est définie comme l'incapacité d'un système ou d'un composant d'exécuter une fonction requise dans les limites spécifiées. Une défaillance peut être produite quand un défaut est rencontré.

Cas de test (*Test Case*)

Un cas de test, en génie logiciel, est défini comme un ensemble de valeurs d'entrée, de pré-conditions d'exécution, de résultats attendus et de post-conditions d'exécution, développées pour un objectif ou une condition de tests particulier, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique (lEEE Std 610.12-1990).

Test (Test)

Un test, en génie logiciel, se défini comme un ensemble de cas de test reliés, ou un ensemble relié de cas de test et de procédures de test (Burnstein, 2003).

Oracle de tests (*Test Oracle*)

Un oracle de test, en génie logiciel, se défini comme un document, ou un composant logiciel qui permet aux testeurs de déterminer si un test a été passé avec succès ou échoué (Burnstein, 2003).

Banc de test (*Test Bed*)

Un banc de test est un environnement qui contient tout le matériel et les outils logiciels nécessaires pour tester un composant ou un système logiciel (Burnstein, 2003).

Qualité logicielle (*Software Quality*)

1. La qualité externe correspond à la mesure dans laquelle un système, un composant système ou un processus répond aux besoins/attentes du client/utilisateur (Burnstein, 2003).
2. La totalité des fonctionnalités et caractéristiques d'un produit logiciel qui influent sur sa capacité à satisfaire des besoins déclarés ou implicites (ISO 9126).

Revues (*Reviews*)

Une revue, en génie logiciel, est une réunion de groupe dont l'objectif est d'évaluer un artefact logiciel ou un ensemble d'artefacts logiciels (Burnstein, 2003).

1.2 Processus de tests type

1.2.1 Description

Un processus, dans le domaine du génie logiciel, est l'ensemble de méthodes, pratiques, standards, documents, activités et procédures que les ingénieurs logiciels utilisent pour développer et maintenir un système logiciel et ses artefacts associés, tels que : la documentation de projet et les plans de test, les documents de conception, le code source et les différents manuels d'utilisateurs et d'opération.

Le processus de test est généralement décrit comme un groupe d'activités effectuées pour évaluer certains aspects d'un composant logiciel. Selon Burnstein, le débogage, ou processus de localisation des erreurs possède trois activités : 1) localiser les erreurs ou les défauts; 2) réparer le code et 3) retester le code (Burnstein, 2003).

1.2.2 Activités du processus de tests

Le 'Software Engineering Body of Knowledge' (SWEBOK) est un document guide initié par l'IEEE-Computer-Society et piloté par les laboratoires de génie logiciel de l'École de Technologie Supérieure (ÉTS) de Montréal et de l'Université du Québec à Montréal

(UQAM). L'objectif de ce guide est de fournir un corpus de connaissances nécessaires à l'ingénieur logiciel. Il présente les domaines de connaissances qui couvrent, entre autres, les processus et activités du cycle de développement de logiciels. Les connaissances nécessaires pour planifier et effectuer des tests de logiciel y sont décrites. Ces activités sont regroupées en une liste de 7 processus:

1. **La planification :** Les activités de tests doivent être planifiées. La planification inclut la coordination du personnel, la gestion des ressources disponibles et la gestion des résultats indésirables. Les considérations majeures de planification sont le temps et l'effort nécessaires pour s'assurer que l'environnement des tests est instauré dans un ensemble de configurations contrôlées.

2. **La génération des cas de test:** La génération des cas de test est dépendante des niveaux de test (unitaires, intégration, systèmes et acceptation) nécessaires et des techniques de test qui seront mises en œuvre. Les cas de test doivent être sous le contrôle de la gestion des configurations du logiciel. Chaque cas de test doit décrire le résultat attendu du test.

3. **Le développement de l'environnement de test :** l'environnement utilisé pour les tests doit être sous contrôle. Il doit faciliter l'exécution des cas de test, et permettre l'enregistrement des résultats de tests, des résultats attendus des scripts et des données de test.

4. **L'exécution des cas de tests :** L'exécution des tests doit suivre les principes de base d'une expérimentation scientifique : Les tests doivent être contrôlés et documentés clairement de sorte à pouvoir reproduire les résultats. Les tests de logiciels doivent être effectués conformément à un plan documenté.

5. **L'évaluation des résultats des tests :** Les résultats d'un test doivent être évalués pour déterminer si le test est réussi ou échoué. Dans la plupart des cas «succès» signifie que le test a été effectué avec succès. Les défauts relevés lors des tests devront être évaluées,

classées par importance et communiquées aux développeurs. Une réunion formelle peut être convoquée pour présenter les résultats d'un cycle de tests et décider de corrections nécessaires avant la reprise des tests.

6. **La documentation des problèmes et le registre de tests :** les tests qui ont échoués et ceux qui ont été réalisés avec succès sont décrits dans un journal (ou registre de tests) afin d'identifier quand le test a été conduit, qui l'a effectué, quelle configuration logicielle et de données a été utilisée et document aussi d'autres informations pertinentes. Les défauts sont souvent enregistrées dans un 'rapport de problème' (RP) pour en faire un suivi détaillé.

7. **Le suivi et correction des défauts :** les défaillances observées durant les tests sont causées essentiellement par des erreurs et défauts dans le logiciel en développement. Les défauts peuvent être analysés pour déterminer quand ils ont étés introduits dans le logiciel et quels types d'erreurs en sont la cause. L'information sur le suivi de défauts est utile pour déterminer quels aspects du processus de l'organisation nécessitent des améliorations.

1.2.3 Niveaux de tests

Il existe plusieurs types de tests logiciels; pour cette raison on les a catégorisés en niveaux de tests. Selon Burnstein; on reconnaît 3 à 4 niveaux (ou phases) de tests : tests unitaires, tests d'intégration, tests systèmes et tests d'acceptation. La **figure 2** présente les niveaux de tests. Chaque niveau peut être constitué de plusieurs sous-niveaux. Chaque niveau possède des objectifs spécifiques, par exemple; au niveau de tests unitaires un seul composant est testé et l'objectif de ces tests consiste à détecter les défauts fonctionnels et structurels de l'unité logicielle. Donnons un autre exemple. Au niveau du test d'intégration, plusieurs composants logiciels sont testés comme un groupe, et le testeur va investiguer les interactions inter-unités. Finalement au niveau du test système tout l'ensemble des composants logiciels, des données et des interfaces est testé, en bloc. Le but principal est

d'évaluer les attributs de fiabilité, de performance et d'utilisabilité, tel qu'il sera perçu par l'utilisateur final. Plusieurs techniques et stratégies sont mises en œuvre pour mener à bien les tests à tous ces niveaux. Selon Burnstein, ces techniques, telles que la technique boite blanche et la technique boite noire, ainsi que les stratégies (aval, amont) sont sélectionnées et varient en fonction de 2 facteurs principaux: 1) le type du système logiciel développé et 2) le paradigme de programmation utilisé par les programmeurs soit; a) le procédural ou b) l'approche orienté-objet (Burnstein, 2003).

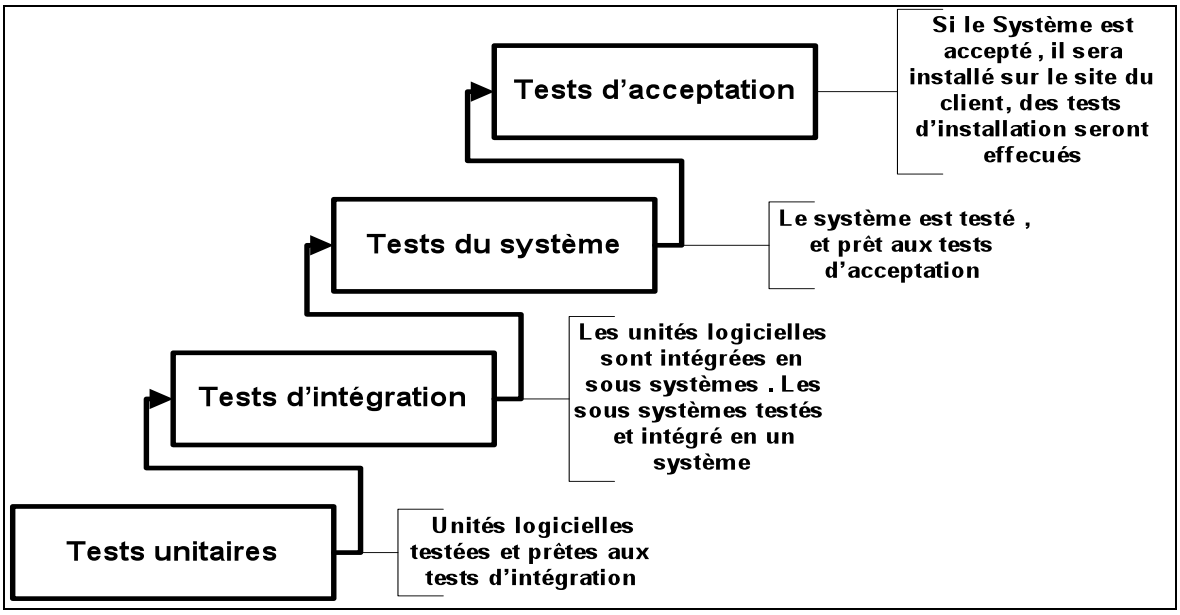

Figure 1 Les niveaux de tests logiciels selon Burnstein.

1.2.3.1 Tests unitaires

La plus petite unité de logiciel, qui peut faire objet d'un test, se nomme l'unité logicielle. Une unité logicielle est traditionnellement perçue comme une fonction ou une procédure dans le paradigme de programmation procédural. Dans le paradigme orienté-objet, les méthodes et classes/objets ont été suggérés, par les chercheurs, comme choix de l'unité

logicielle (Burnstein, 2003). L'unité logicielle peut être compilée et exécutée d'une manière indépendante. On peut donc la tester d'une manière isolée.
Les tests unitaires ont pour objectifs de tester les unités logicielles une à une. Les tests unitaires représentent un premier niveau de tests important. A ce niveau on doit concevoir, exécuter et enregistrer les résultats de test sur des composants de petites tailles. Par exemple on testera la validation d'une donnée ou un calcul spécifique. Ce premier niveau de test permet facilement de localiser les défauts et de les réparer. Une unité logicielle peut aussi être un petit composant acheté d'un fournisseur externe. Ces composants sont souvent appelé composants COTS (*Commercial-Of-The-Shelf*).

1.2.3.2 Tests d'intégration

Les tests d'intégration ont deux objectifs majeurs. Ils consistent à : 1) détecter les défauts dans les interfaces des unités logicielles préalablement testées et 2) assembler les unités en un sous-système opérationnel (en un sous-système complet prêt pour les tests système) (Burnstein, 2003). Burstein propose que dans le cas du paradigme procédural deux stratégies d'intégration sont typiquement mise en œuvre dans l'industrie: 1) le test descendant, (*Top-down Testing*) qui est une approche incrémentale des tests d'intégration où les composants en haut de la hiérarchie sont testés d'abord. Les composants testés sont ensuite utilisés pour tester des composants de niveaux inférieurs. Le processus est répété jusqu'à ce que les composants de plus bas niveau aient été testés; et 2) le test ascendant (*Bottom-up Testing*) où le niveau le plus bas des composants sont testés d'abord, et ensuite utilisés pour faciliter les tests des composants de plus haut niveau. Ce processus est répété jusqu'au test du composant le plus haut de la hiérarchie du logiciel.

Burnstein propose aussi que dans le cas du paradigme orienté objet, dont la structure du programme et les types de collaboration entre unités ne sont pas identiques à celles du langage procédural, les 2 stratégies présentées ci-haut (descendante et ascendante) ne sont pas applicables. Une autre approche est utilisée en industrie et se nomme: test de regroupement d'objets (*Object Cluster Testing*). Burnstein décrit cette technique qui

consiste à regrouper les classes reliées. Les classes reliées sont celles qui collaborent entre elles pour assurer une fonctionnalité du système. (Burnstein, 2003). Ces ‘clusters’ seront par la suite testés comme des sous systèmes du paradigme procédural.

1.2.3.3 Tests systèmes

Les tests systèmes sont typiquement effectués à la suite des tests d’intégration. Le système logiciel est alors entièrement assemblé et tous ses composants, données et interfaces sont testés. À cette phase les développeurs/testeurs entament leurs tests de l’ensemble du système. Les tests du niveau système sont souvent complexes et peuvent nécessiter beaucoup d’effort et de ressources spécialisées, c’est pour cette raison qu’il est préférable qu’ils soient élaborés par une équipe compétente de testeurs spécialisés. Ce niveau de test est critique car c’est à cette étape qu’il faut évaluer les critères qualité du système avant de le soumettre aux tests d’acceptation ou tests Alpha/Beta qui impliquent les utilisateurs finaux (Burnstein, 2003).

Il existe plusieurs types de tests système, ces types sont :

- Tests fonctionnels
- Tests de performance
- Tests de stress
- Tests de configuration
- Tests de sécurité
- Tests de récupération
- Tests de fiabilité
- Tests d’utilisabilité

Un système peut ne pas subir tous ces types de tests. La décision de choix des types à exécuter dépend des exigences qualité, domaine d’utilisation, industrie, caractéristiques du système logiciel et de la disponibilité des ressources.

1.2.3.4 Test d'acceptation

Suite au succès des tests du niveau système et après avoir corrigé tous les défauts découverts, le système va typiquement passer l'ultime étape de validation qui correspond aux tests d'acceptation par les utilisateurs. C'est à ce niveau que les utilisateurs vont jouer un rôle plus important dans le processus de tests. À ce niveau, le système sera testé dans l'environnement spécifié par l'utilisateur et qui simulera la réalité. Les tests d'acceptation cherchent à valider si le système développé rencontre tous les besoins spécifiés par le client ou utilisateur lors de la phase de spécifications des exigences fonctionnelles et non fonctionnelles. Si le client est satisfait et donne son approbation suite aux tests d'acceptation le système subira typiquement un seul autre test : le test d'installation. Le test d'installation vise à s'assurer que la mise en production finale a bien fonctionné.

Dans le cas de logiciel développé destiné à un grand public, les tests d'acceptation ne sont pas possible et seront donc simulés par l'éditeur de logiciel. Ces tests seront suivis de tests de lancement au grand public. Deux types de tests sont appliqués dans ce cas; les tests Alpha et les tests Beta. Les tests alpha sont effectués sur le site de développement, des membres et utilisateurs potentiels de l'organisation sont invités à utiliser les logiciels dans leur travail quotidien. Les problèmes encourus doivent être rapportés et seront évalués. Les tests Beta consistent à envoyer le logiciel à un ensemble d'utilisateurs choisis. Ces derniers installent le logiciel et l'utilisent dans des conditions de travail réelles, enregistrent les problèmes rencontrés et les envoient. Ces problèmes seront évalués selon leur importance et réparés rapidement ou dans une prochaine version.

Notons qu'il existe un autre type de tests dits tests de régression. Les tests de régression ne sont pas un niveau de tests mais c'est une activité qui consiste à retester le logiciel sur lequel, des changements ont été apportés. L'objectif des tests de régression est de s'assurer que la nouvelle version du logiciel garde les attributs de l'ancienne version et que les changements n'ont pas introduits de nouveaux défauts. Les tests de régression peuvent être effectués sur tous les niveaux précédemment décrits.

1.2.4 Processus tests et cycle de vie logiciel

Le domaine des tests logiciels a évolué et est maintenant devenu une profession avec ses propres certifications et programmes d'enseignements. Les tests ne sont plus considérés comme une activité qui commence seulement après la complétude de la phase de codage, avec le but limité de détection des dysfonctionnements. Le test de logiciel est maintenant considéré comme une activité qui devrait englober l'ensemble du processus de développement et de maintenance et est même devenu une partie importante de la construction du produit final. En effet, la planification des tests devrait commencer avec la première étape des exigences du nouveau logiciel. Les plans de tests et les procédures doivent être systématiquement et continuellement développés, et peuvent-être raffinés tout au long du développement du logiciel. La planification et les activités de conception des tests constituent, elles-mêmes, une contribution utile aux concepteurs pour mettre en évidence les faiblesses potentielles telles que les contradictions, les omissions ou les ambiguïtés dans la documentation. (SWEBOK, 2004).

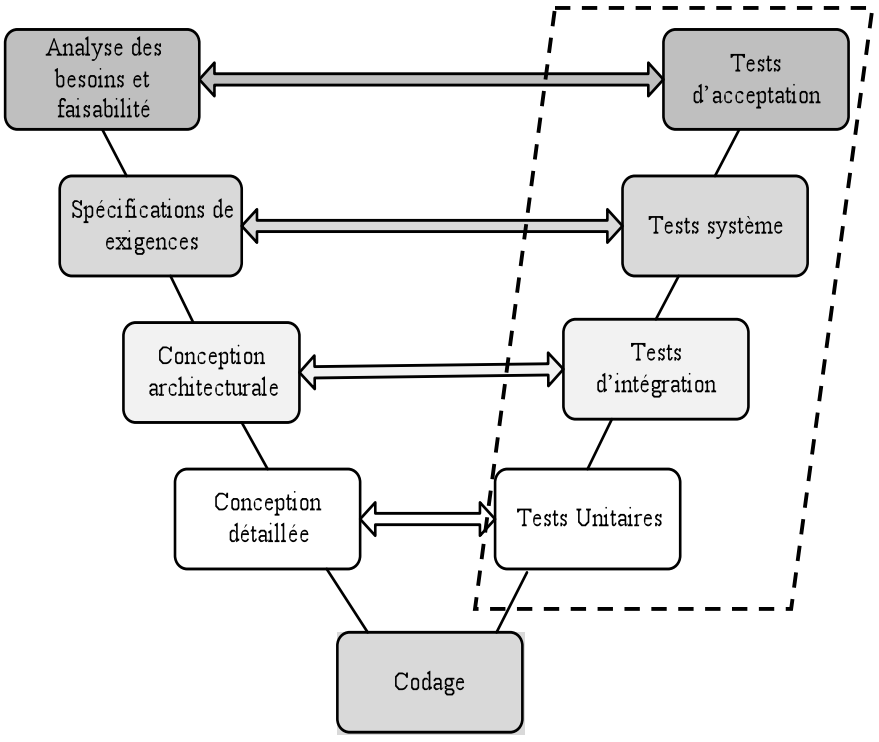

Figure 2 Cycle de vie en V.

1.3 Aperçu sur les modèles de maturité en génie logiciel

Après l'introduction massive de l'approche par processus dans les organisations, le besoin d'évaluer les processus et de les amener à des niveaux d'efficacité supérieure est apparu. À cet effet; plusieurs modèle dit de maturité ont été développés et utilisés. Le domaine du génie logiciel ne fait pas l'exception, il a vu naitre une multitude de modèle de maturité.

Une étude commandée par le département de la défense américaine (DoD) était à l'origine de l'apparition du premier modèle de maturité consacré aux organisations du développement du logiciel. Cette étude révélait qu'une infime proportion (moins de 5%) des projets confiés à des sous-traitants informatiques se terminait à temps, dans les budgets et avec un fonctionnement adéquat (Richard Basque, 2004). Qu'en est-il des 95% qui restent? Une horrible perte certainement. Ceci conduit le DoD à la création du SEI (*Software Engineering Institue*), hébergé à l'université Carnegie Mellon, avec le mandat de régler la crise du logiciel. Watts Humphrey et son équipe ont publié en 1991 la première version du modèle en question (*Capability Maturity Model for Software*) désigné habituellement par SW-CMM ou tout simplement CMM. Selon Humphrey et al le CMM définit essentiellement les attributs d'un processus mature et discipliné de chaque domaine du génie logiciel et spécifie les pratiques à suivre pour atteindre la maturité. Il décrit aussi le chemin de l'évolution pour progresser aux niveaux supérieurs d'efficacité, c.-à-d, allant des processus ad hoc et immatures à des processus disciplinés et matures avec amélioration de la qualité (Humphrey et al.). CMM définit cinq niveaux de maturité au processus génie logiciel (1-Initial, 2-Reproductible, 3-Défini, 4-Géré Contrôlé et 5-Optimisé). Après le CMM, d'autre modèle ont été mis en œuvre soit pour adresser les manquements soit pour se spécialiser dans des processus plus spécifiques. Parmi ceux-ci on peut citer : L'ISO/IEC 15504, le S3M et le TMM.

Le modèle ISO/IEC 15504, est une norme internationale pour l'évaluation des processus du génie logiciel. En fait tous les modèles de maturité (tel que le CMMI, le S3M et le TMM) doivent se conformer à cette norme internationale. Développé en 2003 sous les

auspices d'ISO et d'IEC. Cinq guides composent la version complète de la norme ISO/IEC 15504. Le premier guide, le ISO/IEC 15504-1 présente les concepts et le vocabulaire. Le second guide, le ISO/IEC 15504-2 présente l'élaboration d'une évaluation, le 15504-3 fournit les directives de réalisation de l'évaluation, le 15504-4 est un guide d'utilisation pour l'amélioration et la détermination de la capacité du processus et le 15504-5 présente un exemple de modèle d'évaluation des processus. Ces guides ont été publiés entre 2003 et 2006 (http://www.sei.cmu.edu/cmmi/faq/15504-faq.html#Q241). Un sixième guide le 15504-6 vient compléter le standard, il a été publié en 2008 et présente un exemple de modèle d'évaluation des procédés du cycle de vie d'un système.

Le modèle S3M (Software Maintenance Maturity Model) est conçu spécialement pour l'amélioration continue du processus de maintenance du logiciel. Il a été développé par les professeurs Alain April et Alain Abran de L'École de Technologie Supérieure en 2006 Il est fondé sur la version continue du CMMI et l'approche exploratoire de Abran & Zitouni (A. April et A. Abran, 2006). Il définit une structure étagée de six niveaux de maturité : 0) incomplet,1) défini, 2) géré, 3) établi, 4) maitrisé et 5) optimisé (A. April et al, Software Maintenance Maturity Model (SMmm): The software maintenance process model).

Le TMM quant à lui, définit le guide de bonnes pratiques pour l'évaluation et l'amélioration du processus des tests logiciels. La section suivante présente sa structure et ses composants car nous allons l'utiliser en détail dans notre recherche.

1.4 Le TMM (*Testing Maturity Model*)

Le modèle de maturité des tests à été développé en 1996 par Ilène Burnstein à l'Institut de technologie de l'Illinois (IIT). Plusieurs raisons ont motivé le développement d'un modèle de maturité voué spécialement au processus test, parmi celles-ci citons :

- Une demande croissante pour des logiciels de haute qualité avec un minimum de défauts;

- Le test est un processus important dans la discipline du génie logiciel;
- Les modèles d'évaluation et d'amélioration existants n'ont pas suffisamment pris en compte le processus test.

À partit du dernier facteur, nous comprendrons que le modèle de maturité des tests est apparu pour compléter les autres modèles, particulièrement le CMM, duquel le TMM s'est largement inspiré. Il lui empreinte l'architecture étagée de la démarche d'amélioration de processus.

1.4.1 Composition du TMM

Le TMM se compose de deux éléments majeurs :

- **Un ensemble de niveaux**
  Chaque niveau, à l'exception du premier, possède ses propres caractéristiques. Celles -ci sont décrites en termes d'objectifs organisationnels et d'efficacité du processus test à ce niveau là. La structure du niveau consiste en :

  - **Objectifs de maturité.** Les objectifs de maturité identifient les objectifs d'amélioration des tests qui doivent être atteints en vue de parvenir à la maturité de tel niveau. Pour être placé à un niveau donné, l'organisation doit atteindre tous les objectifs de maturité de ce niveau. Les niveaux de maturité du TMM sont montrés à la figure 3.

  - **Sous-objectifs de maturité support.** Ils définissent la portée, les limites et les accomplissements nécessaires pour un niveau donné.

  - **Activités, Tâches et Responsabilités (ATR).** Les ARTs abordent les questions d'implémentation et d'adaptation organisationnelle à chaque niveau du TMM. les tâches et activités de support sont identifiées et les responsabilités assignées aux groupes appropriés.

La figure suivante montre la structure du modèle TMM et les objectifs de chaque niveau de maturité.

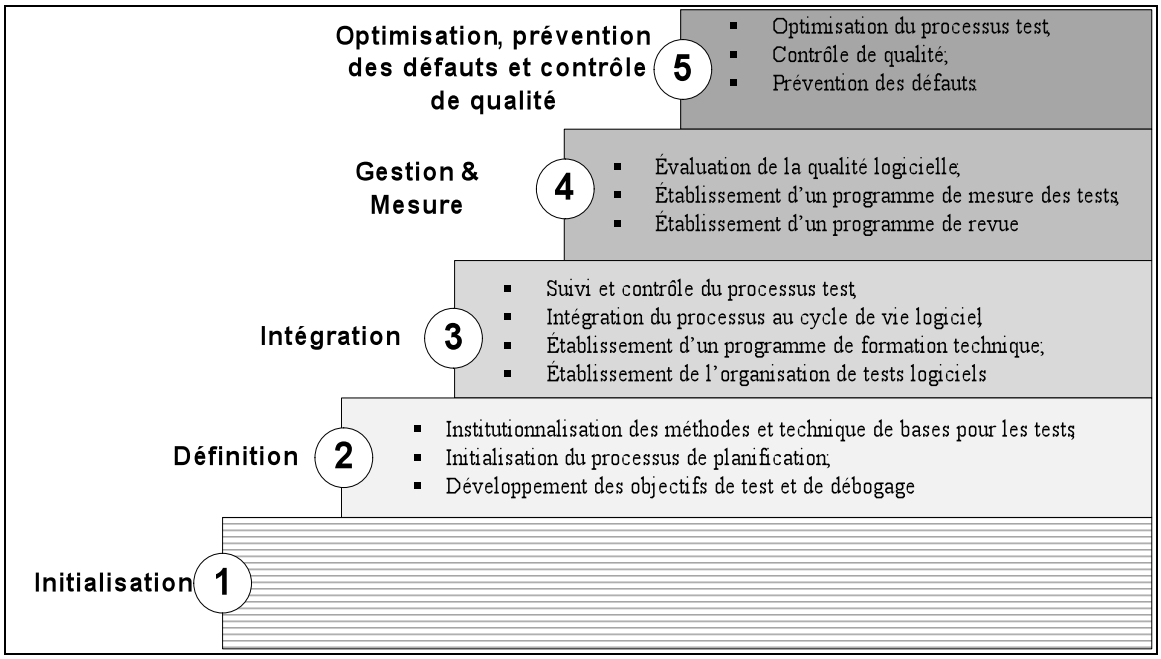

Figure 3 Les cinq niveaux du TMM.

- **Le modèle d'évaluation (*TMM-Assessment Model*)**

Le modèle d'évaluation de la maturité des tests permet de :

- Déterminer le niveau de maturité des tests logiciels;
- Identifier les points forts et points faibles du processus des tests logiciels;
- Développer un plan d'action pour l'amélioration du processus des tests logiciels;
- Identifier les sous-processus candidats à la réutilisation.

Le modèle d'évaluation de la maturité des tests (TMM-AM) est fait de 3 composants inspirés du CMM, SPICE et CAF. Ce sont :

1) Un questionnaire d'évaluation;
2) La procédure d'évaluation;

3) La formation de l'équipe et les critères de sélection.

1.4.2 Structure du TMM

Dans le TMM, les processus des tests de logiciel sont catégorisés selon leurs maturités. Cinq niveaux de maturité composent le TMM, chaque niveau définit un ensemble d'activités (ou objectifs de maturité) à mettre en œuvre pour y accéder. Pour atteindre un niveau supérieur de maturité, le processus de tests doit satisfaire les objectifs de maturité à ce niveau et maintenir les objectifs acquis des niveaux inferieurs.

1.4.2.1 Niveau 1 : Initialisation

À ce niveau de maturité, le processus test est chaotique, il est mal définit et souvent confondu avec le débogage c.-à-d. un processus qui consiste à trouver, analyser et éliminer les causes de défaillance dans les logiciels (Glossaire CFTL/ISTQB, 2005). Aussi, la documentation est minimale ou absente et les logiciels sont fournis sans assurance qualité. L'activité de test est considérée marginale; elle manque toujours de ressources et de compétence en la matière.

1.4.2.2 Niveau 2 : Définition de phase

Dans ce niveau, le processus test est séparé du débogage. Il est défini comme la phase qui suit le codage et ses activités sont planifiées. Même à ce niveau, le processus est considéré immature vu que la planification des tests est dépendante de la phase codage et aucun test ne peut être exécuté sans la disponibilité du code. Quoique le processus soit défini et que des techniques et méthodes soient utilisées; le logiciel produit reste vulnérable aux problèmes de qualité du fait que la planification de tests se fait tard dans le cycle du développement logiciel, et les erreurs se propagent à partir des phases de spécification et conception jusqu'à la phase de codage.

Rappelons que les objectifs de ce niveau de maturité sont : 1) Institutionnalisation des techniques et méthodes de base pour les tests 2) Initialisation du processus de planification et 3) Développement des objectifs de test et de débogage.

1.4.2.3 Niveau 3 : Intégration

Contrairement au processus de niveau de maturité 1 ou 2, celui du niveau 3 n'est plus considéré comme la phase qui suit le codage mais il est intégré à tout le cycle de vie logiciel. Pour la planification; elle commence à la phase de spécification et continue tout au long du cycle de développement. À ce niveau, les objectifs des tests se focalisent sur les exigences basées sur les besoins du client/utilisateur, et sont utilisé pour la conception des cas de test. L'activité de test est reconnue professionnelle et le processus est apparent dans l'organigramme de l'entreprise; on y trouve une équipe de test, et une organisation de formation technique dans les tests. Les organisations à ce niveau réalisent l'importance des revues dans le contrôle de qualité, elles ne disposent pas d'un programme de revue formel et les revues n'ont pas encore pris place dans l'ensemble du cycle de développement. Un programme formel de mesure de tests n'est pas encore établi pour quantifier bon nombre d'attributs du processus et du produit.

Pour résumer; rappelons les objectifs du niveau 3 de maturité : 1) Suivi et contrôle du processus test, 2) Intégration du processus test au cycle de vie logiciel, 3) Établissement d'un programme de formation technique et 4) Établissement de l'organisation des tests.

1.4.2.4 Niveau 4 : Gestion et mesure

Les objectifs à ce niveau sont 1) Évaluation de la qualité du logiciel, 2) Établissement d'un programme de mesure et 3) Établissement d'un programme de revue.
Ceci dit qu'au niveau 4 de maturité; le processus test est mesuré et quantifié. Les revues à chaque phase du processus de développement sont utilisées comme activité de contrôle de test et de qualité. Ce sont un compliment des tests basés sur l'exécution pour la détection

de défauts et pour l'évaluation et l'amélioration de la qualité logicielle. Le modèle en V étendu, tel que montré par la figure 2 peut faire un bon support pour l'implémentation de cet objectif. Les attributs de qualité sont entre autres; la fiabilité, l'utilisabilité et la maintenabilité. Notons au passage qu'à ce niveau de maturité et pour les besoins de réutilisation, éventuellement dans les tests de régression, une base de données est utilisée pour la collecte et l'enregistrement des cas de test de tous les projets. Chaque défaut est enregistré et un niveau de sévérité lui est attribué.

1.4.2.5 Niveau 5 : Optimisation, prévention des défauts et contrôle de qualité

À ce stade de maturité, le processus test a traversé 4 niveaux, il est à présent géré et clairement défini. Ses couts et son efficacité sont contrôlés. Au niveau 5 de maturité la vision est orientée vers l'amélioration continue des mécanismes en place. Les objectifs ce niveau de maturité sont : 1) Optimisation du processus des tests, 2) Contrôle de la qualité et 3) Prévention des défauts.

À ce niveau; des outils totalement automatisés font le support de l'exécution des cas de test. Des outils fournissent aussi le support pour la conception des cas de test, la maintenance des items reliés aux tests, la collection et l'analyse des défauts.

1.5 Sommaire

Ce chapitre a présenté une vue globale sur le domaine des tests logiciels. Il a mis en évidence l'évolution de l'art et mis l'accent sur les pratiques qui le régissent actuellement. Nous avons introduit quelques définitions du vocabulaire des tests de logiciels, défini le processus de tests typique, ses niveaux et ses activités. Nous avons donné un court aperçu de quelques modèles de maturité et finalement nous avons porté une attention particulière au modèle de maturité des tests (TMM). La structure de celui-ci a été définie, ses niveaux de maturité ont été décrits et les activités requises dans chaque niveau ont été énumérées.

Le chapitre suivant présente l'investigation du processus de tests logiciels chez l'entreprise Taktika. Cette investigation est fondée sur les outils proposés par Burnstein.

CHAPITRE 2

CONCEPTION DE L'OUTIL D'INVESTIGATION FONDÉ SUR LES TRAVEAUX DE BURNSTEIN ET APPLICATION DANS L'ENTREPRISE

Ce chapitre présente l'outil d'évaluation du processus des tests de l'entreprise TAKTIKA. Cette évaluation va être élaborée à l'aide du modèle d'évaluation (AM-Assessment Model) fourni par le TMM.

La première section est une description de l'organisation TAKTIKA. La deuxième section introduit le TMM-AM et met en évidence les étapes de la procédure d'évaluation. La troisième section présente le questionnaire d'évaluation tel que décrit par Burnstein.

2.1 Description de l'entreprise TAKTIKA

Taktika Management, de son nom complet, est une entreprise de consultation en matière de gestion. Elle excelle particulièrement dans l'implantation des systèmes de gestion par l'approche processus. Taktika est composée d'un ensemble multidisciplinaire d'experts et consultants exerçant dans le diagnostic et l'accompagnement des entreprises clientes dans leurs structurations et l'optimisation de leurs projets. L'équipe Taktika est composée d'experts normatifs, d'auditeurs, d'analystes d'affaires, de gestionnaires de projets, de conseillers en ressources humaines, de formateurs et de spécialistes en gestion du changement.

Aussi, pour mieux compléter les besoins spécifiques de ses clients; l'entreprise travaille en partenariat avec un ensemble d'entreprise spécialisées. Les partenaires potentiels de Taktika sont : l'école de technologie supérieure, Lixar, Créatech et Microsoft.

La figure suivante illustre la structure organisationnelle de l'entreprise Taktika.

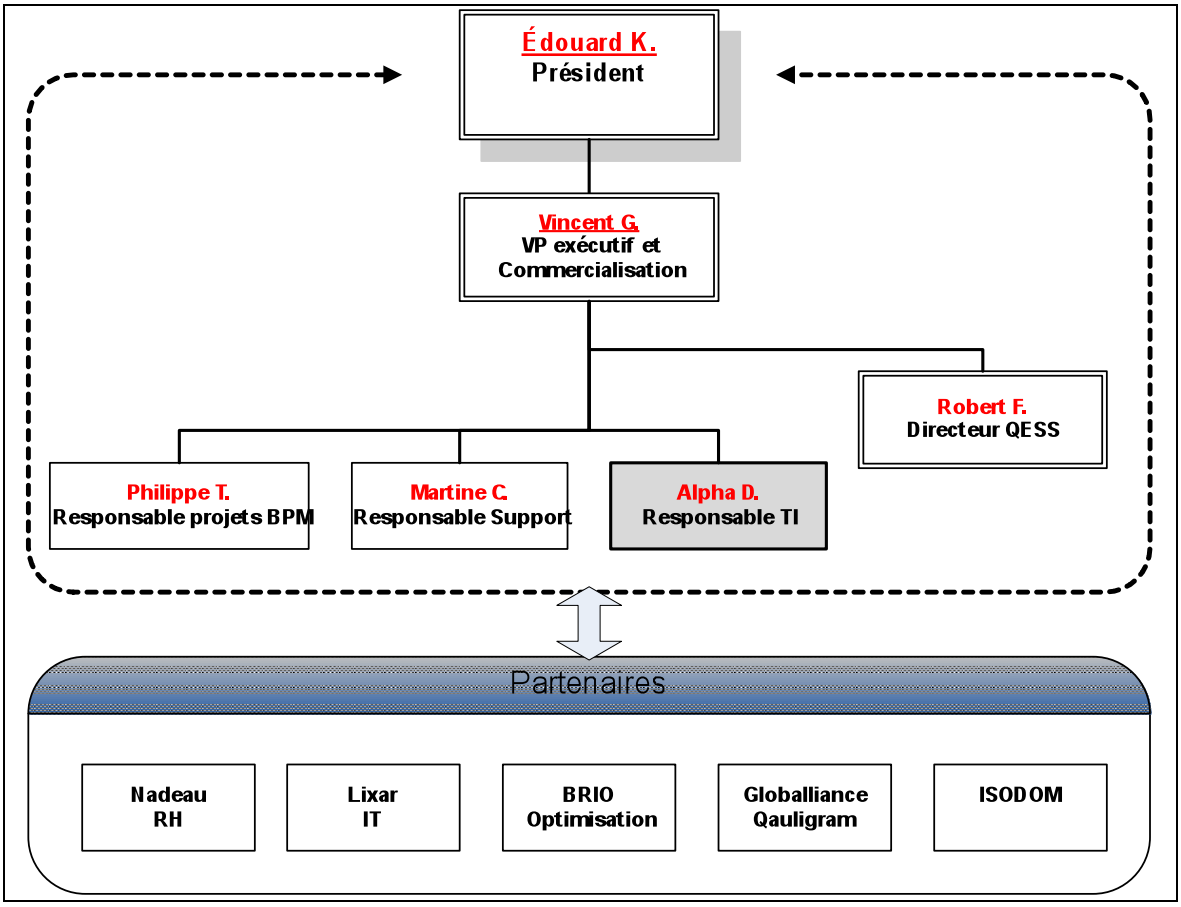

Figure 4 Organigramme de Taktika.

Comme nous pouvons le constater dans la description; Taktika ne développe pas de solutions logicielles. Elle compte néanmoins s'y investir dans le futur proche. Sa première expérience dans le domaine du logiciel consiste en l'élaboration des tests de Qualigram 2.7 développé par son partenaire Lixar IT.

2.2 Introduction au modèle d'évaluation TMM-AM

Comme nous l'avons mentionné dans la section 1.4.1 du chapitre précédent; le TMM fournit aussi, en plus de l'ensemble des niveaux, un modèle d'évaluation. Ce dernier définit

trois composants nécessaires à l'élaboration de l'évaluation des processus de tests logiciels. Pour rappel, ces composants sont : 1) la sélection et la formation de l'équipe de l'évaluation; 2) la procédure d'évaluation et 3) le questionnaire d'évaluation.

Tout au long de cette étude du processus des tests logiciels de Taktika, nous allons nous baser sur la procédure et le questionnaire d'évaluation décrits dans le TMM-AM.

2.2.1 Procédure d'évaluation des processus tests logiciel

La procédure d'évaluation du TMM-AM est constituée d'un ensemble d'étapes qui guident l'équipe d'évaluation. Les principaux objectifs de la procédure d'évaluation de la maturité des tests logiciels sont :

1. développer le profile du processus des tests et déterminer le niveau de maturité;
2. guider l'organisation dans le développement du plan d'action pour l'amélioration du processus des tests logiciels;
3. s'assurer que l'évaluation est effectuée avec une utilisation efficace des ressource de l'organisation;
4. guider l'équipe d'évaluation dans la collecte, l'organisation et l'analyse des données d'évaluation.

La procédure d'évaluation est exécuté sur six étapes, tel que montré par la figure suivante. Ces étapes seront expliquées au fur et à mesure que l'évaluation du processus des tests de Taktika.

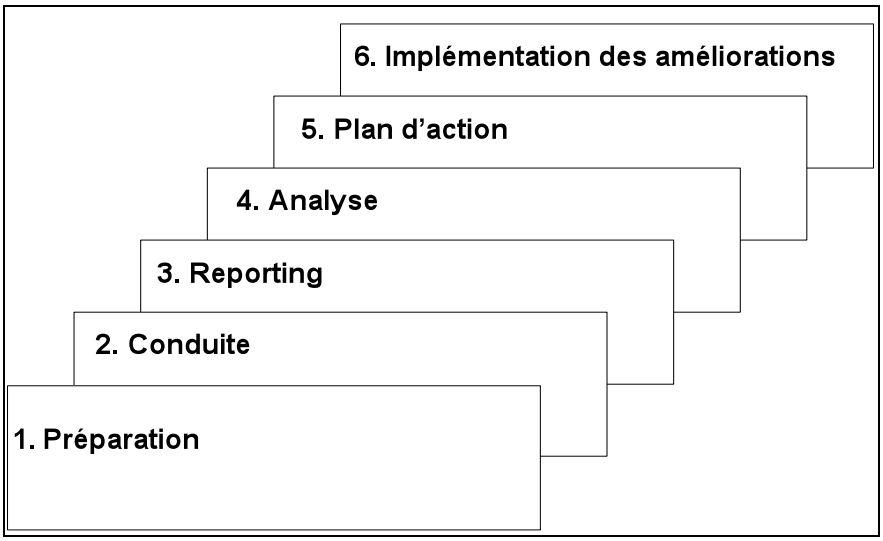

Figure 5 Étapes de la procédure d'évaluation TMM.

Cette étude s'inspire essentiellement des activités pratiquées durant les tests du logiciel Qualigram 2.7 auxquels nous avons participé ainsi que des entrevues organisées avec le responsable TI de l'entreprise Taktika.

2.2.1.1 Préparation de l'évaluation

Dans l'étape de la préparation, il est supposé sélectionner et former l'équipe qui va faire l'évaluation. Et comme ce travail fait l'objet du projet technique pour l'obtention d'un DESS en technologie de l'information; il n y aura pas de formation d'équipe d'évaluation au sein de Taktika.

2.2.1.2 Conduite de l'évaluation

Cette étape consiste à la collecte de l'information utile à l'élaboration de l'évaluation du processus des tests logiciels. Cette information est obtenue à l'aide du questionnaire d'évaluation, des interviews avec les personnes pouvant fournir des éléments pertinents

pour l'activité de l'évaluation ainsi que dans des documents relatifs aux tests dans l'organisation.

2.2.1.3 Documentation de l'évaluation

Dans cette étape nous allons dresser le profile du processus des tests logiciels. Ce profile fournit un sommaire global sur l'état actuel du processus des tests. Il doit contenir le niveau de maturité attribué, les objectifs et sous objectifs de maturité satisfaits, non satisfaits, non applicables aux cas en étude ainsi que ceux qui n'ont pas été évalués.

Le profile en question sera construit à partir du questionnaire rempli par les répondants, des données recueillies lors des entrevues avec le responsable des TI et lors des activités des tests de Qualigram 2.7 ainsi que dans les documents élaborés pour les tests tels que les check-lists de tests et le plan des tests.

2.2.1.4 Analyse des résultats de l'évaluation

Dans cette étape, l'équipe de l'évaluation en association avec les gestionnaires et les ingénieurs de la qualité analyse les résultats de l'évaluation pour identifier les objectifs et les priorités de l'amélioration. Cette approche de priorisation est appelée procédure de classement (ou Ranking procedure). La priorité d'amélioration est attribuée aux activités les plus importantes du processus des tests. L'activité possédant la haute priorité est classée en premier et ainsi de suite. Un plan d'action peut être développé en premier pour les activités les plus prioritaires, comme décrit dans la prochaine étape. Les activités moins prioritaires peuvent être adressées dans les prochains plans d'action.

2.2.1.5 Plan d'action

Le plan d'action peut être implémenté sous forme d'ateliers, il décrit les activités, les ressources et le calendrier pour améliorer les pratiques courantes et implanter les pratiques absentes, de façon à ce que l'organisation puisse accéder au niveau de maturité supérieur.

Ce plan d'action est semblable à un plan de projet. Il doit inclure les objectifs des améliorations, les tâches, les activités, responsabilités, les couts estimés et les bénéfices. Les risques associés aux actions doivent être analysés et un système de suivi d'état doit être décrit.

2.2.1.6 Implémentation des améliorations

Après le développement et l'approbation du plan d'action, la mise en œuvre des améliorations peut être lancée. Cette mise en ouvre peut s'effectuer par un ou plusieurs projets qui seront gérés et suivis pour assurer la progression des actions et atteindre les objectifs cibles.

Dans cette étude, nous nous limiterons à l'interprétation des résultats donc à l'étape analyse des résultats de l'évaluation. Le plan d'action ne sera pas développé vu qu'il nécessite plus d'effort et de ressources. Le plan et l'implémentation des améliorations peuvent être l'objet de travaux futurs. Néanmoins, nous allons émettre des recommandations pour l'amélioration des activités des tests de logiciels.

2.3 Le questionnaire d'évaluation TMM

Il existe plusieurs choix d'instruments d'évaluation incluant les check-lists et les questionnaires. Le questionnaire a été adopté pour l'évaluation de la maturité des tests pour les raisons suivantes :

- il facilite l'intégration avec les autres instruments d'évaluation des processus;
- il assure la couverture de toutes les activités, tâches, et responsabilités associées à chaque niveau de maturité;
- il fournit un Framework de collection et stockage des informations,
- il offre aux évaluateurs des lignes directrices en ce qui concerne les aspects nécessitants des interviews.

Le questionnaire ne suffit pas à lui seul comme source de données pour l'évaluation. Les informations fournies par le questionnaire doivent être confirmées avec celle collectées lors des interviews et l'examen des documents pertinents.

Le questionnaire TMM contient 8 sections qui sont :

I. Instructions d'utilisation;
II. Identification et background du répondant;
III. Description du background de l'organisation;
IV. Questions sur les objectifs et sous-objectifs de maturité;
V. Questions sur l'utilisation des outils de test;
VI. Contexte et tendances des tests;
VII. Recommandations pour l'amélioration du questionnaire;
VIII. Glossaire des termes utilisés dans les tests.

Notons que la section V qui comporte des questions sur l'utilisation des outils de tests n'est pas déterminante pour l'évaluation de la maturité d'un processus de tests de logiciel (Burnstein, 2003). À cet effet nous ne l'avons pas incluse dans le questionnaire remis aux répondants de Taktika, et ce au même titre que le glossaire des termes. Ceci rend le questionnaire moins long pour la lecture et plus souple pour le remplissage. Le questionnaire est mis en annexe dans ce rapport.

2.4 Sommaire du chapitre

Dans chapitre nous avons mis l'accent sur l'outil d'investigation sur lequel nous nous sommes basés pour élaborer l'évaluation du processus des tests logiciels de Taktika. L'organisation et la mission de l'entreprise ont été décrites pour cerner le contexte du processus en question. Dans l'outil d'investigation qui est basé sur le TMM-AM, nous avons présenté en profondeur la procédure d'évaluation ainsi que le questionnaire qui nous a servi d'instrument pour l'évaluation.

Dans le chapitre qui suit, nous allons interpréter les résultats de l'évaluation en analysant les informations collectées lors des entrevues avec le responsable de TI et les données mises dans les réponses au questionnaire. Cette interprétation se soldera par l'identification des forces et faiblesses du processus des tests de Taktika ainsi que son niveau de maturité, à partir de là nous allons aussi adresser une liste de recommandation d'amélioration des pratiques actuelles. L'approche sera aussi sujette à une analyse; ses forces et faiblesse seront énumérées.

CHAPITRE 3

INTERPRÉTATION DES RÉSULTATS, ÉVALUATION DES FORCES ET FAIBLESSES DE L'APPROCHE, TRAVAUX FUTURS ET CONCLUSION

Ce chapitre présente l'interprétation des résultats de l'évaluation du processus des tests de logiciels de Taktika, les forces et faiblesses de l'approche utilisée par l'entreprise dans ses activités de tests de logiciels ainsi que nos suggestions et recommandations d'améliorations du processus actuel.

La première section décrit la procédure de classement telle que définie par Burnstein. La deuxième section fournit une interprétation des résultats obtenus par l'application de la procédure de classement aux données fournies par le questionnaire et autres. La troisième section présente les forces et faiblesses du processus à l'étude et ce, suivant les résultats de l'évaluation et les recommandations du TMM. La quatrième section fournit un ensemble de recommandations d'amélioration des pratiques établies afin de réduire les points faibles du processus.

3.1 La procédure de classement (*Ranking Procedure*)

La procédure de classement pour le TMM-AM est un mécanisme qui se base sur les données recueillies par les différents instruments d'évaluation (questionnaire, entrevues...etc.) pour situer le processus des tests logiciels sur l'échelle de maturité définie par TMM. Les résultats de la procédure de classement produits en sortie sont 1) le numéro de maturité du processus des tests et 2) un rapport sommaire des forces et faiblesses du processus évalué. Ces informations représentent une partie du profile du processus à l'étude. L'algorithme de classification consiste à évaluer les sous-objectifs de maturité, les objectifs de maturité et enfin le niveau de maturité. Cet algorithme définit quatre catégories de classification pour les objectifs et sous-objectifs de maturité (Burnstein, 2003). Un objectif ou sous-objectif de maturité peut être :

1. Satisfait : s'il est implémenté et institué par l'organisation tel que défini par le TMM ou une alternative satisfaisante est mise en place.
2. Non satisfait : s'il est faiblement implémenté et faiblement institué par rapport sa définition dans le TMM et qu'aucune alternative satisfaisante n'est mise en place.
3. Non applicable : s'il n'est pas pertinent du point du vue de l'environnement de l'organisation.
4. Non évalué : si les résultats de l'évaluation ne le couvrent pas adéquatement ou il n'est pas dans la portée de l'évaluation TMM courante.

En plus des ces quatre catégories de classification, Burnstein introduit le concept 'degré de satisfaction' pour déterminer si la satisfaction d'un sous objectif de maturité est très haute, haute, moyenne, faible ou très faible. Le degré de satisfaction est décrit dans le tableau suivant.

Tableau 1

Calcul du degré de satisfaction des sous objectifs de maturité

Degré de satisfaction	% des réponses 'Oui'	Évaluation du sous objectif de maturité
Très haut	> 90 %	Satisfait
Haut	70 – 90 %	Satisfait
Moyen	50 – 69 %	Satisfait
Faible	30 – 49 %	Non Satisfait
Très faible	< 30 %	Non Satisfait

Une fois les sous objectifs de maturité classifiés; les objectifs de maturité peuvent être évalués comme le montre le tableau ci-après.

Tableau 2

Classement des objectifs de maturité

Catégorie des sous objectifs de maturité (en %)	Objectif de maturité
% des sous-objectifs satisfaits >= 50%	Satisfait
% des sous-objectifs satisfaits < 50%	Non satisfait
% des sous-objectifs non applicable >= 50%	Non applicable
% des sous-objectifs non évalué >= 50%	Non évalué

3.2 Interprétation des résultats de l'évaluation

3.2.1 Classement des sous objectifs du niveau 2 du TMM

Les informations contenues dans le questionnaire et portant sur les niveaux de maturité ont été sondées et à l'aide de la procédure de classement; les sous-objectifs de maturité sont catégorisé tels que montré par le tableau suivant.

Tableau 3

Classement des sous-objectifs de maturité du processus des tests de Taktika pour le niveau 2 du TMM

Objectif de maturité du niveau 2 TMM	Sous objectif de maturité	Oui	Non
1. Développement des objectifs et politique de tests et de débogage	**1.1** Un comité de tests et débogage est formé et doté de financement et d'appui. Le comité développe, documente, distribue et supporte des procédures, des objectifs et des politiques pour le test et le débogage. Les objectifs, politiques et procédures, une fois approuvés, sont mis en place et périodiquement revues.	✓	
	1.2 Des politiques/objectifs de test sont pris en compte dans des plans des projets/ tests.	✓	
	1.3 Un système de classification basique de défauts est établi et un référentiel de défauts est mis en place.	✓	
	1.4 De simples mesures de tests sont identifiées et recueillies.	✓	
2. Initialisation du processus de planification des tests	**2.1** Un comité à l'échelle de l'organisation, pour la planification des tests est formé et dotée de support et de financement. Il élabore, documente, distribue et supporte des procédures, des objectifs, des politiques pour la planification des tests. Les objectifs, politiques et procédures, une fois approuvés sont mis en place, et révisés périodiquement.		✓

Tableau 3

Classement des sous-objectifs de maturité du processus des tests de Taktika pour le niveau 2 du TMM (Suite)

Objectif de maturité du niveau 2 TMM	Sous objectif de maturité	Oui	Non
	2.2 Des modèles de plans (Template) pour tous les niveaux de tests sont développés, enregistrés et distribués aux gestionnaires de projet et les développeurs/testeurs. D'autres documents connexes sont identifiés, et prescrits en fonction de la politique.	✓	
	2.3 La formation technique est disponible pour couvrir l'utilisation des modèles de plans de tests et le développement des plans de tests.		✓
	2.4 Une procédure est mise en place pour inclure les exigences générées par les utilisateurs comme intrants nécessaires pour le plan de tests.		✓
	2.5 Des outils basiques de planification et des mesures de testes sont évalués, et appliqués.	✓	
3. Institutionnalisation des techniques et méthodes de base pour les tests	**3.1** Un comité sur la technologie des tests est formé et doté de financement et de soutien. Il étudie, évalue et recommande des techniques, des méthodes de tests, une série de formes simples et des outils pour support. Il élabore les politiques, les procédures et les documents, et ceux-ci sont distribués. Une fois approuvés, ils sont mis en place et périodiquement revus.		✓
	3.2 Une formation technique et des outils de base sont disponibles pour appuyer l'utilisation des techniques et méthodes de tests.		✓
	3.3 Les test du logiciel sont planifiés et mis en œuvre aux niveaux de tests unitaires, d'intégration, du système, et d'acceptation conformément à la politique.	✓	
	3.4 Des Stratégies de base pour les tests (boites blanches/noires), les techniques et les méthodes sont utilisées dans l'organisation pour concevoir les cas de tests. L'interaction entre les développeurs (testeurs) et d'autres staffs techniques (par exemple, des concepteurs, des analystes des exigences) est promue pour identifier les questions de la testabilité, encourager le développement de		

	meilleures représentations logicielles pour les méthodes de tests boîtes blanches/ noires, et pour prendre en charge plusieurs niveaux de tests.	✓	

3.2.2 Classement des objectifs du niveau 2 du TMM

En appliquant la procédure de classement des objectifs de maturité, définie par Burnstein et illustré dans le Tableau 2, nous pouvons aisément conclure que d'après le Tableau 3 que les objectifs de maturité 1 et 3 sont satisfaits par le processus des tests tel qu'établi par Taktika, tandis que l'objectif 2 reste non satisfait.

Tableau 4

Classement des objectifs de maturité du processus des tests de Taktika pour le niveau 2 du TMM

Objectifs de maturité	**Satisfait**	
	Oui	**Non**
Développement des objectifs et politique de tests et débogage	Oui	
Initialisation du processus de planification des tests		Non
Institutionnalisation des techniques et méthodes de base pour les tests	Oui	

3.2.3 Détermination du niveau de maturité du processus Tests de Taktika

D'après Burnstein, pour atteindre un niveau de maturité du modèle TMM, le processus des tests doit satisfaire les critères définis à ce niveau ainsi que ceux des niveaux bas, autrement dit; le processus doit atteindre tous les objectifs de ce niveau de maturité et maintenir satisfaits ceux des niveaux plus bas.

Le processus des tests de Taktika ne satisfait que deux objectifs de maturité sur l'ensemble des trois exigibles au niveau 2 du TMM. Ceci fait que le processus des tests de logiciels de

Taktika est de niveau 1. La classification des objectifs de maturité (niveau 2 du TMM) du processus des tests de Taktika est illustrée par le tableau 4 en haut.

3.3 Identification des forces et faiblesses du processus Tests de Taktika

Le processus des tests de logiciels de l'entreprise Taktika présente quelques points forts et un certain nombre de faiblesses. Suivant le classement des sous-objectifs de maturité élaboré dans l'évaluation de ce processus; nous pouvons relever ses forces et ses faiblesses. Le tableau suivant fournit un résumé des points forts et points faibles du processus des tests évalué sur le projet Qualigram 2.7.

Tableau 5

Sommaire des forces et faiblesses du processus de tests de logiciel de Taktika

Forces	Faiblesses
▪ Classification des bugs selon leurs degrés de sévérité. ▪ Utilisation de mesures pour les activités de tests. ▪ Communication entre le groupe des tests et les développeurs.	▪ Le processus de tests est faiblement défini. ▪ Absence de stratégies et politique de test ▪ Planification inadéquate et absence de formation pour initier à l'utilisation des modèles et outils de planification des tests. ▪ Documentation incomplète. ▪ Absence d'expérience et d'expertise dans les tests de logiciels, et aucune formation technique n'est disponible pour appuyer l'utilisation de méthodes et techniques de tests. ▪ Pas d'affectation à temps plein pour les activités des tests ▪ Les exigences des clients ne sont pas prises en compte pendant la planification des tests.

3.4 Recommandations d'amélioration selon les directives de Burnstein

Burnstein décrit l'ensemble d'activités, tâches et responsabilités (ARTs) qui doit être élaboré dans chaque niveau de maturité. Ces ARTs sont définis pour chaque objectif de maturité et présentés sous trois perspectives ou points de vue critiques qui sont : les gestionnaires, les développeurs/testeurs et les clients/utilisateurs.

En ce qui concerne notre cas, nous allons reprendre les ARTs de chaque objectif de maturité du niveau 2 (définition de phase) du TMM et qui implémentent les pratiques absentes ou mal établies chez Taktika.

Objectif de maturité 2.1 Développement des objectifs et politique de tests et de débogage

Bien que cet objectif soit satisfait dans l'organisation Taktika, certain sous objectifs ne le sont pas. Une série d'activités, tâches et responsabilités sont exigées.

- **Point de vue des gestionnaires**

  - Mettre à la disposition des participants aux tests et débogage la documentation relative aux objectifs et politiques des tests et débogage.
  - Assurer une formation adéquate pour mener à bien les objectifs et politiques de tests et de débogage.
  - Réviser périodiquement la politique et les objectifs de tests et de débogage.

- **Point de vue des développeurs/testeurs**

  - Se familiariser avec les objectifs et politique de tests/débogage approuvés, émettre de suggestions quand nécessaire et se tenir à jour des révisions effectuées sur les objectifs et la politique.

- Participer activement dans les revues périodiques de la politique et des objectifs de tests/débogage.

- **Point de vue des clients/utilisateurs**

  - Les clients/utilisateurs sollicités par les gestionnaires émettent des feedback sur les objectifs et politique de tests/débogage. Les points de vue de ces groupes doivent être pris en compte par les gestionnaires et le groupe d'assurance de la qualité du logiciel.

Objectif de maturité 2.2 Initialisation du processus de planification des tests

L'objectif 'Initialisation du processus de planification des tests' n'est pas satisfait par le processus Tests de Taktika, suivant la démarche de Burnstein nous recommandons les ARTs suivantes :

- **Point de vue des gestionnaires**

  - Former un comité pour la planification des tests et lui attribuer des pouvoirs et des ressources.
  - Développer une politique de planification qu'il faut documenter et distribuer à tous les participants aux tests. Cette politique doit être révisée périodiquement avec le staff technique.
  - S'assurer que tous les projets sont en accord avec la politique de planification.
  - Développer des modèles de plans de tests et s'assurer qu'ils sont utilisés conformément dans tous les projets.
  - S'assurer que les développeurs/testeurs complètent tous les documents post-tests tels que le rapport d'incidents et les registres de tests.

- Participer à des cours de formation pour la planification des tests, l'emploi des modèles de plan de tests et outils de planification, l'identification et l'estimation des risques liés aux tests.

- **Point de vue des développeurs/testeurs**

  - Assister à des cours de formation dans la planification des tests, l'utilisation des outils de planification et des modèles de plans et l'identification des risques de tests.
  - Assister le gestionnaire de tests pour déterminer les objectifs de tests, les coûts et les risques à tous les niveaux de tests.
  - Assister le gestionnaire de tests dans le choix des meilleures méthodes, procédure et outils.
  - Développer les spécifications de cas de tests, les spécifications de la procédure des tests et d'autres documents.
  - Compléter les documents post-test tels que le rapport d'incidents et les registres de tests.

- **Point de vue des clients/utilisateurs**

  - Formuler clairement les exigences
  - Fournir des intrants au plan de tests d'acceptation, les exigences fonctionnelles et les attributs liés à la performance doivent être spécifiés clairement et quantitativement si possible.

Objectif de maturité 2.3 Institutionnalisation des techniques et méthodes de base pour les tests

Cet objectif de maturité est moyennement satisfait dans l'organisation Taktika, bon nombre d'activités reste à performer.

- **Point de vue des gestionnaires**

  - Former un comité qui se charge d'identifier, évaluer et recommander des techniques, des méthodes et des outils de tests.
  - S'assurer que les recommandation du comité sont documentées, distribuées et approuvées.
  - S'assurer que des politiques et standards sont désignés pour la promotion de l'institutionnalisation des méthodes de conception de tests basées sur les boites blanches et boites noires.
  - S'assurer que les politiques et standards couvrent tous les niveaux de tests.
  - S'assurer que les développeurs/testeurs ont acquis la formation nécessaire pour comprendre et appliquer les méthodes de tests boites blanches et boites noires et pour développer les cas de test, les procédures de tests, les registres de test et les rapports d'incidents.
  - S'assurer que les développeurs/testeurs ont acquis la formation nécessaire pour l'élaboration des tests sur tous les niveaux.
  - Allouer des ressources et délais adéquats pour la conception et l'exécution des tests boites blanches/noires pour tous les niveaux ainsi que pour analyser les résultats des tests.
  - Réviser périodiquement les outils, méthodes et techniques de tests.

- **Point de vue des développeurs/testeurs**

  - participer dans le comité désigné pour identifier, évaluer et recommander des techniques, des méthodes et des outils de tests.
  - assister à des sessions de formation sur les outils, techniques de tests et niveaux de tests, travailler avec des collègues expérimentés pour acquérir de l'expérience dans l'application des méthodes basées sur les boites blanches/noires et l'élaboration des tests unitaires, tests d'intégration, tests système et tests d'acceptation.

- Concevoir les cas de tests, les spécifications de tests et les procédures de tests conformément aux connaissances sur les méthodes, techniques et stratégies de tests.
- Installation de l'environnement matériel et logiciel nécessaire à l'exécution des tests.
- Exécuter les cas de tests à tous les niveaux.
- Enregistrer les données relatives aux tests.
- Enregistrer les données relatives aux défauts.
- Travailler avec les spécificateurs et les concepteurs pour voir leurs représentations du logiciel.
- Se référer au référentiel de défauts pour se renseigner sur les défauts identifiés dans le passé pour des projets similaires.
- Servir de mentor pour les collègues qui veulent acquérir de l'expérience dans la mise en œuvre des tests à tous les niveaux.
- Travailler avec les clients/utilisateurs pour le développement des cas d'utilisation ou autre description d'interactions utilisateur-machine et les critères d'acceptation nécessaires aux tests multi-niveaux.
- Travailler avec les clients/utilisateurs lors des tests d'acceptation pour s'assurer de leur satisfaction.
- Participer dans la révision périodique des méthodes et techniques de tests.

- **Point de vue des clients/utilisateurs**

  - Fournir un staff de liaison pour interagir avec le staff de tests.
  - Travailler avec les analystes pout s'assurer que les exigences sont complètes, claires et testables.
  - Participer dans les tests d'acceptation.
  - Émettre des feedback et le rapport de problèmes de façon à ce que les problèmes peuvent être traités lors des tests d'acceptation.
  - Participer dans le développement des cas d'utilisation.

3.5 Autres recommandations d'amélioration

Nous suggérons aussi quelques standards pour adresser certaines activités du processus de tests.

1. Documentation : L'IEEE Standard for Software Test Documentation (IEEE Std 829-1983) décrit tous les documents associés aux tests. Ces documents sont : 1) le plan de tests, 2) spécification de conception des tests, 3) spécification des cas de tests, 4) spécification de la procédure de tests, 5) rapport de transmission des items de tests, 6) registre des tests, 7) rapport d'incidents de tests et 8) rapport sommaire des tests. Nous suggérons ce standard à l'entreprise Taktika pour compléter la documentation des activités de tests.

Nous avons constaté un manque sur cet aspect, et même les documents élaboré ne répondent à aucune norme ou standard. Nous nous contentons de fournir par ici les composants d'un plan de tests. Ces composants sont :

1. Identifiant du plan de test
2. Introduction
3. Items à tester
4. Fonctionnalités à tester
5. Approche de tests
6. Critères d'échec/réussite
7. Critères de suspension et de reprise
8. Livrables
9. Tâches de test
10. Environnement de test
11. Responsabilités
12. Effectif et besoins en formation
13. Calendrier des tests

14. Risques et contingences
15. Coûts des tests
16. Approbations

2. Classification des erreurs : adoption du standard IEEE Guide to Classification for Software Anomalies pour la classification des anomalies rencontrées lors des activités des tests de logiciel.

3. Définition du processus de tests : pour la définition du processus des tests de logiciel; Taktika peut se fier soit à SWEBOK soit au TMM pour la prise de connaissance de toutes les activités qui doivent être mise en œuvre par le processus de tests de logiciels.

3.6 Sommaire du chapitre

Dans ce chapitre nous avons présenté l'analyse et l'interprétation des résultats de l'évaluation conformément aux travaux de Burnstein. Nous avons présenté la procédure de classement fournie avec le TMM et qui décrit le mécanisme d'analyse des informations du questionnaire et des entrevues. Nous avons par la suite appliqué cette procédure au processus de tests de l'entreprise Taktika, les résultats on été interprétés. Sur la base de cette interprétation les forces et faiblesses des activités de tests de Taktika on été relevées et en leur fonction un ensemble de recommandation à été émis et quelques standards on été suggérés à l'entreprise pour mieux définir certaines activités de son processus de tests de logiciel.

Dans ce qui suit une conclusion générale sera présentée et fournira un sommaire sur l'étude élaborée, un sommaire des contributions et les travaux futurs.

CONCLUSION

A. Sommaire de l'étude

Dans le présent rapport, une évaluation des processus des tests de logiciels de l'entreprise Taktika Management a été proposée. Cette évaluation a été effectuée à l'aide du modèle TMM et met en évidence les forces et faiblesses de l'approche tests mise en œuvre par Taktika.

Pour mener à bien cette évaluation, une revue de littérature sur l'état de l'art des tests de logiciels a été élaborée. La revue de littérature donnée au premier chapitre, avait traité deux composants majeurs du domaine des tests ce sont : 1) le processus typique des tests et 2) les modèle de maturité et particulièrement le TMM. Le processus des tests a été décrit sous tous les angles, on a énuméré et défini ses activités, ses niveaux ainsi que sa portée dans le cycle de développement de logiciel. Puisqu'il fournit exclusivement un Framework d'évaluation et d'amélioration des tests de logiciels;, le modèle TMM a été choisi et mis au premier plan des autres modèles de maturité du génie logiciel, ce modèle se démarque aussi par sa puissance et sa parfaite intégration aux autres modèles tels que CMM et CMMI.

L'outil d'investigation a été conçu conformément aux travaux de Burnstein, et nous l'avons utilisé pour structurer la démarche d'évaluation et définir les mécanismes de collecte et d'analyse d'information. Les informations collectées à l'aide du questionnaire TMM et les entrevues avec les testeurs de Taktika ont été traitées à l'aide de la procédure de classification. Les résultats produits en sortie nous ont fournit une base de connaissances qui nous a permis de déterminer le niveau de maturité TMM du processus des tests et d'identifier les forces et faiblesses de celui-ci. À partir de ces faiblesse et en guise de recommandations d'amélioration des pratiques actuelles; un ensemble de suggestions a été émis pour rendre le processus des tests plus efficace et plus mature.

B. Sommaire des contributions

Cette étude est une évaluation réelle, première en son genre du processus des tests de logiciels de l'entreprise Taktika. De ce fait ce travail peut être perçu comme :

- Un référentiel des forces et faiblesses du processus tel qu'il est implanté actuellement;
- Un élément sur lequel on peut s'appuyer pour développer le plan d'action et la mise en œuvre des améliorations prioritaires;
- Une démarche pratique d'évaluations qui pourra être adoptée pour la réalisation des prochaines évaluations des pratiques des tests;
- Un exemple concret pour l'évaluation des processus des tests des petites entreprises de développement de logiciels.

C. Forces et limites de l'approche TMM

Plusieurs avantages sont reconnus au modèle de maturité des tests, parmi ses avantages on site :

- Le TMM est un modèle consacré au seul processus des tests de logiciels, ceci fait de lui un modèle qui donne aux organisations la possibilité de réaliser des évaluations pour un domaine spécifique qui est le domaine des tests;
- Le TMM s'associe parfaitement avec les autres modèles ayant la même architecture dite étagée tels que le CMM et le SE/SW CMMI (version étagée), il peut aussi être associé à d'autres standards à architectures continues tels ISO/IEC 12207 et ISO/IEC 15504 pour adresser le processus des tests;
- Les organisations, particulièrement les petites, n'ont pas besoin (sauf peut être les débutantes) de recourir aux services de consultants externes ou organismes de standardisation pour l'évaluation de leurs processus de tests. Cette dernière peut être effectuée par une équipe interne;

- Le TMM représente une source riche en connaissances du domaine des tests, il peut donc servir de moyen de formation en la matière;
- La démarche TMM est incrémentale; les objectifs et sous objectifs de maturité ainsi que les pratiques décrites dans les ART peuvent être introduites progressivement dans l'organisation.

Le modèle TMM quoiqu'il définisse une démarche puissante qui prend en charge l'évaluation et l'amélioration du processus de tests de logiciels, il présente quelques faiblesses au niveau de la mise en œuvre de l'évaluation. Ces faiblesses se résument comme suit :

- La démarche s'appuie beaucoup plus sur ce qui s'écrit dans l'organisation que sur ce qui se fait réellement;
- Les tests sont un sous processus du processus de développement de logiciels, il reste difficile de l'isoler des autres processus. D'autres pratiques sont nécessaires pour le bon fonctionnement des tests. ces pratiques telles que la gestion des configurations, la gestion des exigences et la planification des projets si elles ne sont pas bien élaborées fragiliseront le processus des tests qui sera évalué en conséquence.
- La structure étagée du TMM fait que l'évaluation est faite pour un seul niveau de maturité, cette évaluation ne fournit pas des éléments sur les capacités de l'organisation dans les niveaux supérieurs.

D. Travaux futurs

Ce rapport est élaboré dans le cadre du projet technique du DESS en technologie de l'information, il fournit une évaluation de l'organisation Taktika dans le domaine des tests de logiciels. Il peut être un élément déclencheur pour une multitude de travaux dans le futur proche. Et comme Taktika compte s'investir dans le développement de logiciels ces travaux peuvent être :

- Implantation du niveau 2 TMM dans l'organisation Taktika;
- Évaluation du processus de développement de logiciel suivant CMM ou CMMI;

BIBLIOGRAPHIE

April Alain. et Abran Alain. 2007. Améliorer la maintenance du logiciel. Longueuil, Québec: Loze-Dion, 337 p.

April, Alain et al. 2004. «Software Maintenance Maturity Model (SMmm): The software maintenance process model». En ligne. 30 p. < http://www.compaid.com/caiinternet/ezine/alainapril-maintenancemodel.pdf>. Consulté le 23 novembre 2008.

Basque, Richard. 2004. *CMMI : un itinéraire fléché vers le Capability Maturity Model Integration*. «InfoPro». Paris: Dunod, 196 p.

Myers, G. J. 2004. *The Art of Software Testing*. 2e éd. Hoboken, N.J.: John Wiley & Sons, 234 p.

IEEE Computer Society. 2004. *Guide to the Software Engineering Body of Knowledge.* Los Alamitos, California, 204 p.

IEEE Computer Society. 1990. *lEEE Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology.*

IEEE Computer Society. 1983. *IEEE Std 829-1983: IEEE Standard for Software Test Documentation*

Burnstein Ilene, 2003. *Practical Software Testing: A Process-Oriented Approach*. «Springer professional computing». New York: Springer, 709 p.

Burnstein Ilene, Suwanassart Taratip, Carlson Robert. 1996. «Developing a Testing Maturity Model; Part I ». In *Le site du* Software Technology Support Center. En ligne <http://www.stsc.hill.af.mil/crosstalk/1996/08/developi.asp>. Consulté le 15 octobre 2008.

Burnstein Ilene, Suwanassart Taratip, Carlson Robert. 1996. «Developing a Testing Maturity Model; Part II ». En ligne. 9 p. < http://www.improveqs.nl/pdf/Crosstalk%20TMM%20part%202.pdf >. Consulté le 20 octobre 2008.

Burnstein Ilene, Suwanassart Taratip, Carlson Robert. 1996. « Developing a Testing Maturity Model for Software Test Process Evaluation and Improvement ». *Test Conference, 1996. Proceedings, International*, (20-25 Oct. 1996), p. 581 – 589. Washington, DC, USA: IEEE Computer Society.

International Software Testing Qualification Board. 2005. *Glossaire CFTL/ISTQB des termes utilisés en tests de logiciels* Version 1.0FR.

Software Engineering Institute, Carnegie Mellon. 2008. «ISO/IEC 15504». In *Le site du* Software Engineering Institute|Carnegie Mellon. En ligne <http://www.sei.cmu.edu/cmmi/faq/15504-faq.html#q241>. Consulté le 23 novembre 2008.

ANNEXE

QUESTIONNAIRE TMM

Ce qui suit est le questionnaire d'évaluation que nous avons soumis aux répondants de Taktika. Celui-ci est le questionnaire que le responsable TI nous a retourné. Notons que ses réponses sont en couleur bleue.

Important

This questionnaire is being developed as part of the assessment software testing process of Taktika. This assessment is contained in the technical project DESS IT (École de Technologie Supérieure).

After filling this questionnaire please return it to Abderrahmane Abbas at: **abderrahmane.abbas.1@ens.etsmtl.ca** or **abderrahmane.abbas@ gmail.com**

Section 1: Instructions for the Respondent

Please read and answer the following questions carefully using the knowledge and experience gained from working on current projects in your organization. Many of the technical terms are defined in the glossary section of this document. If you wish to comment on any of the questions or qualify your answers, please use the comment space provided. Your answers will be held in confidence.

For the maturity goal questions found in Section 4, four possible choices are offered as follows:

1. **YES.** Check when a practice is well established and consistently performed.
2. **NO.** Check when the practice is not well established or is inconsistently performed

3. **DOES NOT APPLY.** Check when you have the required knowledge of the project or organization, but you believe that the question does not apply to the project or organization.
4. **NOT KNOWN.** Check when you are uncertain as to how to answer this question, or you do not have the proper information or experience.

Only one of the choices for each question should be selected, and all of the questions should be considered. Please continue on to the remainder of the questionnaire, and thank you for your help.

Section 2: Respondent Identification and Background

In evaluating the TMM data it will help assessors to have information about the software engineering background, technical skills, and the current duties of each respondent.

1. **Respondent Identification**
   - Name : Alpha Diallo
   - Position : Responsible TI
   - Project Name : Qualigram
   - Telephone :
   - Email : alpha.diallo@taktikamanagement.com
   - Date :3-12-2008

2. **Respondent Background**

   Which best describes your current position?
   - **Manager**
   - Senior or upper management Project manager
   - Test manager
   - Software quality assurance group leader

- Software engineering process group leader
- Test-related subgroup leader (*please specify the name of the subgroup*)
- Other (*please specify*)

- **Technical Staff**
- **Software engineer**
- Test engineer
- Programmer (developer)
- Analyst
- Software quality assurance group member
- Software engineering process group member
- Test-related subgroup member (*please specify*)
- Other (*please specify*)

3. Current Duties and Responsibilities

Which test-related activities are you actively engaged in (you can check more than one)?

- Test policy and goal development
- Test planning
- Test case and test procedure design
- Test execution
- Collection and analysis of test-related measurements
- Defect data collection
- Maintenance of defect database
- Standards development
- Reviews and audits
- Status tracking
- Training
- Metrics definition

- Hiring and recruiting
- User/client communications
- Process control
- Defect prevention
- Technology transfer
- Process assessment
- Process improvement
- Reliability modeling and engineering
- Usability testing
- Tool evaluation
- Process reuse

4. **Have you received any TMM training?**
   - Yes (*please describe*)
   - No

5. **What is the extent of your experience in the software industry?**
   - Your present organization Number of years 5 months
   - Overall industry experience Number of years 1 year
   - Overall testing experience Number of years

6. **Have you participated in other types of software testing appraisals?**
   - Yes (*please describe*)
   - No

Section 3: Organizational Background

1. **Describe as best you can the type of your organization.**
   - Develops military software
   - Develops application software

- Develops telecommunication software
- Software quality assurance/software testing/certification
- Other (*please describe*)

2. **Is the majority (greater than 50%) of the software developed for internal or external use?**
   - Internal
   - External
   - Doesn't apply

3. **How many people are employed in the organization being assessed?**
   - Total number of employees: 12
   - Number engaged in software development and/or maintenance : 2
   - Number engaged in software testing : 4

4. **Please describe the percent of staff engaged in testing as follows:**
   - Full time
   - Part time
   - Consultants

5. **Are the people involved in process improvement in your organization well respected with regard to their technical and management skills? (Please circle one: 1 means no respect and 5 means highly respected.)**

   1 2 3 4 5

6. **Are the responsibilities for test process improvement clearly defined and supported? (Please circle one: 1 means not defined nor supported, and 5 means well defined and supported.)**

1 2 3 4 5

7. **Does the organization under assessment have a software engineering process group or a similar unit?**

Yes No

8. **How is the testing group organized? (please select one)**
   - Developers do the testing
   - Test group within development, report to project manager
   - Separate test group, report to test manager
   - Part of Software Quality Assurance group

9. **How would you generally characterize the nature of your testing process?**
   - Ad Hoc
   - Informal
   - Somewhat structured
   - Highly structured

10. **How frequently do project managers have to meet the challenges of changing customer requirements?**
    - Never
    - Rarely
    - Frequently
    - Very frequently

Section 4: Maturity Goal Questions

Please answer each of the following questions with either a YES, NO, DOES NOT APPLY, or NOT KNOWN response. Comments may be entered after each question.

TMM Level 2: Phase Definition

Maturity Goal 2.1: Develop Testing and Debugging Goals and Policies

The purpose of this goal is to differentiate clearly the processes of testing and debugging. The goals, tasks, activities, and tools for each must be identified. Responsibilities for each must be assigned. Policies must be established by management to accommodate and institutionalize both processes.

Maturity Sub-goals that support this goal are:

2.1.1. An organization-wide committee(s) or group on testing and debugging is formed and provided with funding and support. The committee(s); develops, documents, distributes, and supports procedures, goals, and policies for testing and debugging. The goals, policies, and procedures, once approved, are put in place and periodically reviewed.

2.1.2. Testing and debugging policies/goals are reflected in project/test plans.

2.1.3. A basic defect classification scheme is established, and a basic defect repository is put into place.

2.1.4. Simple testing and debugging measurements are identified and collected.

QUESTIONS	YES	NO	DOES NOT APPLY	NOT KNOWN
Has a committee(s) on testing and debugging been established?	✓			
Have policies, goals, activities, and tools for the testing process been identified, documented, and approved?	✓			
Have policies, goals, activities and tools for the debugging process been identified, documented, and approved?				✓
Is the process of testing defined?	✓			
Is the process of debugging defined?				✓
Have the policy documents on debugging been distributed to project managers and developers (testers)?		✓		
Do the software developers (testers) follow a written organizational policy for testing when test planning?			✓	
Do the developers follow a written organizational policy for debugging?				✓
Are basic measurements used to determine achievement of testing goals?	✓			
Are basic measurements used to determine achievement of debugging goals?				✓

Have the testing policies and goals been developed with inputs from user/client groups with respect to their needs?	✓			
Have the debugging policies and goals been developed with input and feedback from user/client groups with respect to their needs?				✓
Has a basic defect classification scheme been developed?	✓			
Has a defect repository been established?	✓			
Do developers/testers log defects into the repository on a consistent basis?	✓			
Are testing/debugging policies and goals periodically reviewed?	✓			

Maturity Goal 2.2: Initiate a Test Planning Process

The purpose of this goal is to establish a test planning process on an organization-wide basis. Test planning involves stating test objectives, analyzing risks, outlining strategies, and developing test design specifications, and test cases. A test plan must also address the allocation of resources, the costs, and the responsibilities for testing on the unit, integration, system, and acceptance levels.

Maturity Sub-goals that support this goal are:

2.2.1. An organization-wide committee, or group on test planning is formed and provided with funding and support. The committee; develops, documents, distributes, and supports

procedures, goals, and policies for test planning. The goals, policies and procedures, once approved are put in place, and periodically reviewed.

2.2.2. Test plan templates for all levels of testing are developed, recorded and distributed to project managers and developers/testers for use in organizational projects. Other required tested-related documents are identified, and prescribed according to organizational policy.

2.2.3. Technical training is available to cover use of test plan templates and development of test plans.

2.2.4. A procedure is put in place to include user-generated requirements as inputs to the test plan.

2.2.5. Basic planning tools and test measurements are evaluated, and applied.

QUESTIONS	YES	NO	DOES NOT APPLY	NOT KNOMN
Has an organization-wide test planning committee or group been established?		✓		
Is there an organizational policy, and are there procedures for test planning?		✓		
Have the policy and procedures been distributed and approved?		✓		
Is there adequate support and funding for test planning for all projects?		✓		
Are test goals/objectives used as a basis for test planning?	✓			

Have test plan templates been developed and distributed to project managers?	✓			
Are there appropriate planning tools available for test planning?	✓			
Have project managers been trained in the use of templates and planning tools?	✓			
Have developers (testers) been trained in the use of templates and planning tools?	✓			
Are developers (testers) trained properly to develop test specifications, test designs, and test cases for the test plan?	✓			
Are test-related risks considered when developing test plans?	✓			
Are estimates (time, budget, tools) available from past projects for use in test planning?		✓		
Is test planning done at the unit level?	✓			
Is test planning done at the integration level?	✓			
Is test planning done at the system level?	✓			
Is test planning done at the acceptance level?	✓			
Is there a procedure in place for soliciting user/client input for test planning where appropriate (e.g., in acceptance test planning)?	✓			

Do developers (testers) have the opportunity to give inputs to the test plan at all levels?	✓			
Is the test planning process reviewed on a periodic and/or event-driven basis?		✓		
Are other test-related items such as test transmittal reports, test logs, test incident reports, and test summary reports defined in organizational documents?		✓		
Are other test-related items such as test transmittal reports, test logs, test incident reports, and test summary reports completed for each project?		✓		
Does management support interactions between project (test) managers, developers (testers), designers, and analysts to support test planning?	✓			
Are basic test measurements specified in the test plan at all levels?	✓			
Do developers (testers) collect and store basic test-related measurements?	✓			
Are the basic test measurements used to ensure that basic testing goals have been met?	✓			

Maturity Goal 2.3: Institutionalize Basic Testing Techniques and Methods

The purpose of this maturity goal is to improve test process capability by applying basic testing techniques and methods. How and when these techniques and methods are to be applied and any basic tool support for them should be specified clearly in testing policies

and plans. Various basic techniques and methods that are often used in the testing process are black box and white box test design strategies, use of a requirements validation matrix, and the division of execution-based testing into sub-phases such as unit, integration, system and acceptance testing. Some testing tools that support use of these techniques and methods are static and dynamic analyzers, coverage analyzers, test data generators, and error checking tools.

Maturity Sub-goals that support this goal are:

2.3.1. An organization-wide committee or group on test technology is formed and provided with funding and support. The committee studies, evaluates and recommends a set of basic testing techniques and methods, and a set of simple forms, and tools to support them. It develops relevant polices, procedures, and documents, and these are distributed. When approved, they are put in place and periodically reviewed.

2.3.2. Technical training and basic tools are available to support use of testing techniques and methods.

2.3.3. Software testing is planned and implemented at the unit, integration, system, and acceptance levels according to policy.

2.3.4. Basic testing strategies (white/black box), techniques and methods are used organization-wide to design test cases. Interaction between developers (testers) and other technical staff (e.g., designers, and requirements analysts) is promoted to identify testability issues, encourage development of software representations useful for white/black box testing methods, and to support multiple levels of testing.

QUESTIONS	YES	NO	DOES NOT APPLY	NOT KNOWN
Has a committee or group been formed to evaluate and recommend a set of basic testing techniques, methods, and tools?		✓		
Have the recommendations of the group been documented, distributed, and approved?		✓		
Have basic tools and techniques been included in test policies?		✓		
Have appropriate forms and templates been designed to support basic testing techniques?	✓			
Are adequate resources provided by upper management to support the use of basic testing techniques and methods, as well as basic testing tools?		✓		
Have developers (testers) been trained to apply the basic tools, forms, and methods?		✓		
Are basic testing techniques and methods applied on an organization-wide basis?		✓		
Are basic testing tools applied on an organization-wide basis?		✓		
Are basic testing techniques and methods reviewed periodically?		✓		

Do testing policy statements include the requirement for multilevel testing?		✓		
Is testing planned and implemented at multiple levels (unit, integration, system, etc.)?	✓			
Are the basic testing techniques and tools described in policy statements applied at all levels of testing?		✓		
Are the testing techniques and tools to be applied described in multilevel test plans?	✓			
Does upper management support interaction between analysts, designers, and developers (testers) to ensure testing issues are addressed by these groups?	✓			

Section 5: Testing Trends Questions

As in the case of the testing tool use questions, the testing trend questions are designed to provide a broader view of the testing process to assessors. They play no formal role in a TMM ranking. Assessors can use the responses as they see fit to assist in the assessment process.

1. From your perspective, what are the major strengths of the testing process in your organization today?

- Our bug tracking systems and we trying to automatise the tests

2. From your perspective, what are the major weaknesses of the testing process in your organization today?

- No test automatisation

3. What changes has the organization made to improve the testing process over the last 2–5 years?
- System for bug tracking, we move from the excel sheet to bug tracking system (sharepoint)

4. How would you rate the overall effectiveness of the testing process today compared to 2 years ago? Please select one of the following choices: same, improved, greatly improved, little improved, worse, don't know.
- Improve

5. Compare the fraction of time, and resources allocated for testing now and 2 years ago. Please select one of the following choices: same, largely increased, largely decreased, about the same, don't know.
- Increased from 1 to 3

6. Compare the current number of full-time testers in the organization to the number available 2 years ago. Please select one of the following choices: same, increased, decreased, don't know.
- Increased

7. Compare the current number of part-time testers in the organization to the number available 2 years ago. Please select one of the following choices: same, increased, decreased, don't know.
- Increased

8. From your perspective, over the last 2 years has communication between testers, developers, and managers: improved, stayed the same, gotten worse, don't know?
- Improved

9. From your perspective has management addressed the needs of testers/developers? Yes, no, to some extent, don't know.

- Yes

Section 6: Comments from Respondents

This section is reserved for a respondent to comment on the nature of the TMM Questionnaire itself. Respondents may comment on any aspect of the questionnaire, for example, on the clarity of the questions, the organization of the questions, the completeness, and usability of the document. Comments provide useful feedback for questionnaire maintainers who can then consider appropriate changes. Respondents may use the blank section provided below and/or add supplemental pages with their comments.

- The questions are clear and useful