UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ÉVALUATION D'UNE APPLICATION DE GESTION DES RESSOURCES HUMAINES ET DE SON PROCESSUS DE DEVELOPPEMENT ET DE MAINTENANCE

RAPPORT DE PROJET TECHNIQUE PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAITRISE EN GÉNIE LOGICIEL

PAR
ABDELLAH MEHIR

JUILLET 2019

REMERCIEMENTS

La réalisation de ce projet n'était pas une tâche facile pour moi, sans l'aide et le bon encadrement de ma directrice de projet Sylvie Trudel. Je n'aurais pas pu mener à terme ce travail tout seul, merci beaucoup beaucoup Sylvie pour ta disponibilité, pour tes conseils précieux, pour ta méthodologie de travail, pour tes compétences.
Un grand merci à Louis Martin, professeur du département d'informatique à l'UQAM pour son précieux outil « rapport-facile » qui m'a beaucoup aidé dans la génération de ce rapport, et par conséquent m'a fait épargner beaucoup de temps.
Je remercie également mon employeur qui m'as donné cette occasion pour réaliser ce travail, et tous les employés d'ACME qui ont accepté de répondre à mes questions.
Un merci spécial à ma femme Siham pour son soutien pendant toutes ces années de ma maitrise, durant lesquelles elle s'est occupé seule de nos trois enfants.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACME Nom générique donné à l'organisation pour fins de confidentialité
AQ ou QA *Quality Assurance*, Assurance qualité
BPMN *Business Process Modeling Notation*, Norme de notation pour la modélisation de processus
CM *Configuration Management*, Gestion de la configuration
CMMI™ *Capability Maturity Model Integration* , Modèle intégré des capacités
CMMI-DEV *CMMI™ for Development*,Développement de produits et services
COSMIC *Common Software Measurement International Consortium*
CFP *COSMIC Function Point*, Point fonction cosmic
DoD *United States Department of Defense*, Département de la Défense des États-Unis
FUR Fonctionnalités Utilisateurs Requises
GA Globalement Acceptable
GDI Gestion Des Individus
GPE Gestion de Projet et Estimation
IEC *International Electro Commission technique*
ISO *International Standards Organisation*, Organisation internationale de normalisation
ITIL *Information Technology Infrastructure Library*, Organisation internationale de normalisation
MA *Measurement and Analysis*, Mesure et analyse
MGL Maitrise en génie logiciel
MVC *Model View Controller*, Modèle Vue Contolleur
ODT Organisation Du Travail
OPD *Organizational Process Development*, Développement du processus organisationnel
OPF *Organizational Process Focus*, Focalisation sur le processus organisationnel
PEM *Process Evaluation Method*, Méthode d'évaluation du processus
PI *Product Integration*, Intégration du produit
PMC *Project Monitoring and Control*, Surveillance et contrôle de projet
PP *Project Planning*, Planification de projet
PPQA *Product and Process Quality Assurance*, Assurance qualité du produit et du processus
PRC PRoCessus
PRL PRoduit Logiciel
RD *Requirements Development*, Développement des exigences
REQM *Requirements Management*, Gestion des exigences

RSKM *Risk Management*, Gestion du risque
S3M *Software Maintenance Maturity Model*
SaaS *Software As A Service*
SEI *Software Engineering Institute*
SMART Spécifique, Mesurable, Atteignable, Réaliste, et limité dans le Temps
SPICE *Software Process Improvement and Capability dEtermination*
SW-CMM *Capability Maturity Model for Software*, Modèle d'évolution des capacités
TFS *Team Fondation System*
TS *Technical Solution*, Solution technique
UQAM Université du Québec à Montréal
Val *Validation*, Validation
Ver *Verification*, Vérification

RÉSUMÉ

Les dernières années chez ACME ont été marquées par un nombre de bogues qui a passablement augmenté, par le nombre important de projets réalisés en parallèle, et en particulier, par un retard considérable sur la plupart des projets. Pour essayer de comprendre la source de ces symptômes, nous avons fait une évaluation du processus logiciel de RH.net, une application de gestion des ressources humaines développée par ACME et utilisée dans le monde pour gérer plus d'un million de dossiers d'employés. Pour y parvenir, nous avons choisi et adapté la méthode PEM puisqu'elle convient bien à notre contexte.

Pour évaluer ce processus logiciel, nous avons fait des entrevues avec sept personnes d'ACME, nous avons aussi révisé et analysé la documentation et les données liées à l'application et aux projets d'évolution. Nous avons noté 29 constats organisés par domaine de processus, parmi les 14 domaines du CMMI pris en considération dans ce projet. À partir de ces constats, nous avons suggéré 16 recommandations regroupées en cinq thèmes, ce qui devrait permettre à ACME d'améliorer son processus logiciel et réduire le nombre de bogues à gérer et à corriger.

Nous avons aussi mesuré objectivement l'efficience du processus logiciel avec les données de sept projets et en appliquant la méthode COSMIC (norme ISO 19761). Cette mesure d'efficience nous a permi d'établir un modèle d'estimation pouvant être une référence pour ACME pour l'estimation des prochains projets. De meilleures estimations devraient alors mener à des budgets et des échéances plus réalistes.

ABSTRACT

The last years at ACME were marked by a number of bugs that has increased significantly, by a large number of projects running in parallel, and by considerable delays on most projects. To try to understand the source of these symptoms, we evaluated the software process of RH.net, a human resources management application used throughout the world to manage over one million employee files. To achieve this, we have chosen and adapted the PEM method, since it was well suited to our context.

To evaluate this software process, we interviewed seven ACME employees, and reviewed and analyzed the documentation and data related to the application and development projects (evolutive maintenance). We noted 29 findings organized by process area, among the 14 CMMI process areas considered in this project. Based on these findings, we made 16 recommendations grouped in five categories. This should allow ACME to improve its software process and reduce the number of bugs to manage and fix.

We also objectively measured the efficiency of the software process with data from seven projects by applying the COSMIC method (ISO Std 19761). This efficiency measure allowed us to propose an estimation model that could become a reference for ACME for estimating future projects. Better estimates should then lead to more realistic budgets and deadlines.

INTRODUCTION

Ce rapport est le résultat d'un projet de fin d'études réalisé dans le cadre de la maitrise en génie logiciel à l'Université du Québec à Montréal (UQAM). Au cours de cette maitrise, on acquière des connaissances et des compétences liées au développement et à la maintenance des logiciels, les aspects techniques autant que les aspects méthodologiques et de gestion ou d'organisation des projets. Ce projet de fin d'études est une occasion de mettre en pratique ces connaissances et compétences dans un cadre pratique, soit celui de l'emploi que j'occupe depuis plus de 10 ans au sein d'une compagnie Montréalaise qui développe une application de gestion des ressources humaines déployée dans plus de 20 pays. L'application de mes connaissances sur des projets de mon quotidien, ainsi que l'occasion d'aider mon employeur à améliorer ses processus de développement, est à la fois un honneur et une grande responsabilité.

Ce projet consiste, dans un premier temps, à évaluer le processus de développement et de maintenance de logiciel et, dans un deuxième temps, à proposer des recommandations pour réduire les symptômes vécus par l'organisation. Compte tenu de la nature confidentielle des informations se trouvant dans ce rapport, nous référons à cette compagnie sous le pseudonyme « ACME » et à son logiciel de gestion des ressources humaines sous le pseudonyme « RH.net ».

Ce rapport est structuré comme suit. Le chapitre 1 décrit la compagnie, son logiciel principal et son processus logiciel ainsi que les différents symptômes des problèmes sous-jacents. Nous enchainons ensuite sur le but de notre projet et les objectifs à atteindre pour réduire ces symptômes. Le chapitre 2 décrit l'état de l'art, soit quelques extraits de la littérature décrivant comment d'autres personnes ou organisations ont pu résoudre des problèmes similaires. En se basant sur les expériences que les autres ont décrites, nous établissons notre méthodologie de travail au chapitre 3 qui consiste en une évaluation sur deux axes principaux : l'axe qualitatif pour le processus logiciel et l'axe quantitatif pour l'efficience de ce processus. Au chapitre 4, nous présentons les constats découlant de notre évaluation qualitative, autant les forces que les points à améliorer, et des conséquences qui y sont liées. Au chapitre 5, nous avons analysons l'efficience du processus quantitativement avec la méthode COSMIC (COSMIC-Team-V4.0, 2015). Au chapitre 6, nous proposons une série de recommandations qui, lorsque mises en œuvre, pourront atténuer les symptômes décrits au chapitre 1. Nous terminons avec une conclusion dans laquelle nous faisons un retour sur le but et les objectifs de notre projet.

CHAPITRE 1

L'ORGANISATION ET SA PROBLÉMATIQUE

1.1 La compagnie ACME

La compagnie ACME développe une solution intégrée de gestion des ressources humaines utilisée dans plus de 20 pays. Pour les besoins de ce rapport et pour respecter la confidentialité de la compagnie, nous appellerons cette application « RH.net ». ACME a deux modèles d'affaires principaux. Le premier est du type *Software As A Service* (SaaS), c'est-à-dire que les clients paient le service d'utilisation de l'application en fonction du nombre de dossiers d'employés gérés dans l'application. Le deuxième modèle d'affaires est un contrat ferme d'achat d'une licence d'utilisation comprenant le service et les mises à jour de l'application. ACME existe depuis 1992. Elle possède des bureaux en France et au Canada. Le développement et l'évolution de RH.net n'ont lieu qu'au Canada, à Montréal. En France, on y fait des ventes et on emploie huit consultants qui font le déploiement et la formation des clients de l'application RH.net.

1.2 L'application RH.net

RH.net est adoptée par des entreprises d'envergure, gérant plus d'un million de dossiers d'employés dans le monde. RH.net est constituée de plusieurs modules : Fiche employé, Santé et sécurité, Gestion du temps et des congés, Planification et saisie des horaires, Formation-carrière et autres modules. Lorsqu'ils achètent RH.net, les clients choisissent des modules spécifiques en fonction de leurs besoins. ACME offre le soutien et la maintenance pour tous ses modules. Les clients demandent souvent à ACME de développer des modules personnalisés à l'intérieur de l'application, ce qui donne lieu à un certain nombre de projets annuellement. De plus, ACME met sur pied ses propres projets d'évolution de RH.net, et ce afin de conquérir de nouveaux marchés, à raison de deux à trois versions par année.

1.3 Équipe de développement et de maintenance

L'équipe de développement et de maintenance d'ACME comporte treize personnes : un gestionnaire de projet, deux analystes, deux personnes d'assurance qualité, six dé-

veloppeurs, un responsable du support technique et un rédacteur technique. Une vue d'ensemble est présentée à la Figure 1.1.

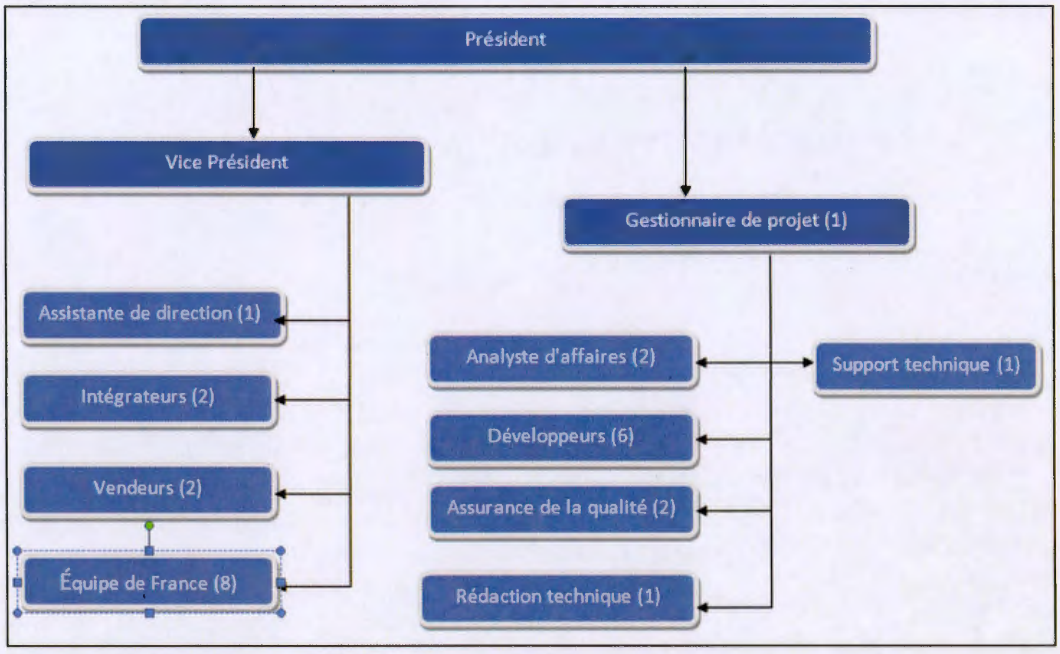

FIGURE 1.1 – Organigramme d'ACME

Nous présentons ici les responsabilités de chacun de ces rôles :

Gestionnaire de projet :

- Suivre les états d'avancement ;
- Assigner les tâches après les avoir priorisées avec le président ;
- Participer de temps en temps dans des dossiers d'analyse ;
- Prendre les décisions sur des points de projet.

Analyste :

- Éliciter, analyser, documenter, et valider les besoins des parties prenantes du projet ;
- Élaborer la solution et la présenter aux développeurs ;
- Faire des tests préliminaires pour être sûr que ce qui est développé reflète bien sa vision pour le produit.

Assurance de la qualité :

- Identifier les types de tests à préparer ou à exécuter selon la nature ou le type de modèle à développer ;
- Élaborer le plan de test ;
- Exécuter l'ensemble des cas de test une fois le codage est terminé ;

— Prendre des notes des problèmes trouvés dans le logiciel des bogues (TFS dans le cas de ACME) ;
— Faire des *Smoké* tests ;
— Dans certains cas, exécuter les cas de tests sur d'autres modules en relation avec les nouvelles fonctionnalités développées.

Développeur :

— Faire une estimation et l'envoyer au chef du projet ;
— Développer les fonctionnalités découlant des besoins spécifiés ;
— Élaborer des tests unitaires (avec *Xunit*) ;
— Exécuter un ensemble d'outils de validation du code comme *style coop* et *Fixoop*.

Support technique :

— Répondre aux appels et les emails des clients ;
— Faire le soutien du premier niveau (problèmes simples de paramétrage, problèmes de navigateur) ;
— Investiguer la nature des problèmes ;
— Enregistrer les incidents dans le logiciel du bogues.

Rédaction technique :

— Faire la traduction anglaise du logiciel ;
— Vérifier les fautes d'orthographe et de syntaxe ;
— Adapter des expressions selon le contexte.

1.4 Environnement de développement

Le développement du logiciel Rh.net se fait avec *Visual Studio*. RH.net est une application Web développée avec le *Framework ASP.NET*, avec une architecture *Web Forms* et le langage C#, sur un système d'exploitation Windows. *SQL server* est utilisé comme serveur de base de données. ACME utilise TFS pour gérer les versions de son logiciel, soit du code source et des scripts de base de données. Elle utilise aussi *Share Point* pour gérer les documents internes. ACME utilise le logiciel TFS pour surveiller les demandes de changement et les incidents de production. TFS permet aussi de suivre toutes les demandes soumises à ACME, de documenter l'échelle de résolution et de gérer les affectations des demandes.

1.5 Processus logiciel de développement et de maintenance de RH.net

En cas de problème avec l'application, les utilisateurs communiquent avec le service à la clientèle d'ACME qui investigue le problème. Quand ce problème s'avère être un bogue, il est communiqué à l'équipe de développement. Depuis trois ans, soit la période

correspondant au départ du directeur informatique qui gérait la planification et le suivi des livraisons, le nombre de bogues et le nombre de demandes de développement sur mesure ont passablement augmenté, ayant pour effet de surchager l'équipe. L'équipe reçoit ses affectations directement du PDG, priorisées en fonction de la pression des clients exercée sur la compagnie. Pour tenter de diminuer son débordement, l'équipe a pu recruter deux développeurs, après plus de six mois de recherche de candidats.

Dans ce contexte, l'équipe a besoin de recadrer son processus de développement afin de devenir à la fois plus efficace – soit de livrer des fonctionnalités, modules ou demandes des clients respectant les besoins et de haute qualité – et plus efficiente – soit d'arriver à livrer plus de fonctionnalités dans un même laps de temps qu'avant.

1.6 Symptômes liés à la qualité et au processus d'évolution de RH.net

1.6.1 Nombre grandissant de bogues et de demandes

Le nombre de bogues et le nombre de demandes de développement sur mesure ont passablement augmenté, ayant pour effet de surcharger l'équipe ces deux dernières années, comme le montre la figure 1.2.

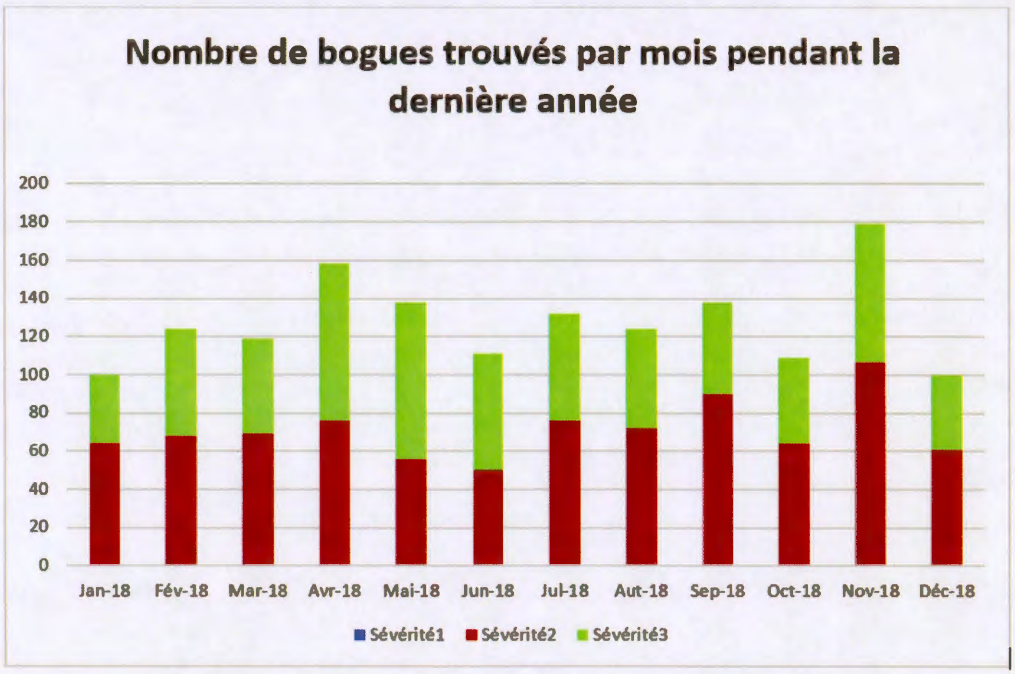

FIGURE 1.2 – Nombre de bogues de la dernière année

1.6.2 Nombre élevé de dossiers ouverts en parallèle

Pour gérer les urgences, puisque le logiciel RH.net est une application en maintenance évolutive, on change souvent les priorités, ce qui oblige les membres d'équipe à changer de contexte de projet souvent, ce qui donne lieu à plusieurs dossiers ouverts en parallèle. Par exemple, un développeur peut travailler sur trois dossiers et plus au cours d'une même journée.

1.6.3 Allongement de la durée d'implémentation d'une nouvelle fonctionnalité

Étant donné qu'il y a plusieurs dossiers ouverts en parallèle, la durée d'implémentation des nouvelles fonctionnalités s'allonge systématiquement, ce qui amène beaucoup de pression sur les développeurs. On demande souvent aux développeurs de faire des heures supplémentaires.

1.6.4 Travaux faits individuellement, pas en équipe

Chacun travaille sur une tâche dans son coin et, à la fin, on livre un produit en rassemblant tous les morceaux. Après, on juge la performance de chaque individu en fonction du temps qu'il a mis pour réaliser sa tâche et le nombre de bogues trouvés dans sa partie. Par conséquent, l'évaluation est basée sur ses données personnelles, ce qui empêche de créer une équipe performante. Il n'y a pas, ou peu, de temps alloué pour construire des relations. Les groupes se concentrent rarement sur la rétroaction entre leurs membres pour améliorer l'efficience du groupe. Après que les nouveaux développeurs se soient adressés aux anciens pour obtenir des informations, le PDG leur a demandé d'aller d'abord le voir afin qu'il détermine lui-même si les autres développeurs peuvent ou pas être *dérangés*.

1.7 Définition de notre projet

1.7.1 But de notre projet

Il nous faut donc découvrir les problèmes à la base de ces symptômes de façon à être en mesure de les régler. Dans le temps alloué à notre projet, nous visons à évaluer la qualité du produit logiciel RH.net et de son processus d'évolution et maintenance afin de déterminer quelles sont les causes de ces symptômes et proposer des recommandations adéquates.

1.7.2 Approche préconisée

Pour atteindre ce but, nous avons choisi d'évaluer l'application et le processus de développement et de maintenance en cherchant à y identifier les aspects manquants ou inadéquats. Pour évaluer le processus, on compte se baser, entre autres, sur les pratiques recommandées du modèle *Capability Maturity Model Integration for Software* (CMMI). D'autres modèles pourraient aussi être utilisés, tels que le *Software Maintenance Maturity Model* (S3M, ou modèle d'amélioration de la maintenance du logiciel) ou le cadre ITIL (*Information Technology Infrastructure Library*). Pour évaluer l'application RH.net, on compte utiliser la norme ISO/IEC 14598-5 (*Information technology – Software product evaluation – Part 5 : Process for evaluators*). L'évaluation sur ces deux axes, processus et produit logiciel, a été combinée dans une méthode appelée *PEM* (*Process Evaluation Method* (Trudel, Lavoie, Paré, Suryn, 2006)), dans laquelle on détermine la source des modèles d'évaluation au moment de la conception d'une occurence d'un évaluation en fonction des besoins et des buts.

1.7.3 Objectifs du projet

Selon l'équipe de développement et de maintenance, les objectifs d'affaires du projet seraient les suivants :

- Augmenter la qualité du logiciel RH.net en diminuant les taux d'anomalies de 30% au cours des douze prochains mois ;
- Réussir à ce que chaque employé puisse travailler sur un seul projet à la fois dans le but d'augmenter son efficience ;
- Réduire l'écart à plus ou moins de 20% entre la durée estimée et la durée réelle pour réaliser des tâches, dans le but de se donner le temps pour produire un logiciel de qualité du premier coup et d'être ainsi plus efficace.

Pour atteindre ces objectifs d'affaires, nous définissons l'objectif de ce travail à être réalisé avant l'été 2019 :

Comprendre et identifier les forces et faiblesses du processus de développement et de maintenance à l'origine du nombre élevé de bogues :
1. En catégorisant les bogues par phase de développement ;
2. En évaluant le processus logiciel avec les bonnes pratiques de CMMI et S3M ;
3. En élaborant un modèle initial d'estimation du processus d'évolution du logiciel.

1.7.4 Portée

La portée de ce projet sera focalisée sur les éléments suivants :

- Les modules déployés chez les clients existants et qui sont en maintenance évolutive ;
- Les processus mis en place pour développer les nouveaux projets ;
- Les activités de génie logiciel suivantes : gestion de projet, analyse, développement et assurance qualité.

L'étude ciblera seulement la filiale de Montréal et, plus précisément le service ingénierie et développement.

Ce projet va exclure les points suivants :

- Les projets de déploiement d'un module RH.net pour un nouveau client ;
- Le bureau de Paris ;
- Les activités d'administration, de vente et d'intégration.

CHAPITRE 2

ÉTAT DE L'ART SUR LES MEILLEURES PRATIQUES DE DÉVELOPPEMENT ET MAINTENANCE DE LOGICIEL ET LEUR ÉVALUATION

La concurrence féroce dans le marché actuel (logiciel et autres) pousse les organisations constamment à livrer des produits de grande qualité dans les meilleurs délais et avec le coût le moins élevé possible. Ceci passe par l'amélioration continue des processus logiciel. Toutefois, avant de parler d'amélioration d'un processus, encore faut-il qu'il soit déjà défini et qu'il soit utilisé par l'organisation en question. Et le plus important est qu'il soit utilisé par les gens de l'organisation. Le meilleur processus logiciel est celui qui est proche des gens qui vont le mettre en œuvre. Si un modèle de processus logiciel a été développé dans une entreprise au niveau de l'organisation globale, il ne peut être efficace que s'il peut être adapté pour répondre aux besoins de l'équipe de projet qui l'utilise. Dans un cadre idéal, vous créez un processus qui convient le mieux à vos besoins et, en même temps, répond aux besoins plus larges de l'équipe et de l'organisation. Alternativement, l'équipe elle-même peut créer son propre processus et, en même temps, doit répondre aux besoins plus étroits des individus et aux besoins plus larges de l'organisation (Pressman, Maxim, 2015).

2.1 Définition d'un processus logiciel

Un processus est « une séquence d'étapes effectuées dans un but précis » (IEEE-Standards-Board, 1990). Un processus logiciel est « un ensemble d'activités, méthodes, pratiques et transformations exécutées par des individus pour développer et maintenir le logiciel et ses sous-produits » (Paulk, Curtis, Chrissis, Weber, 1993). Dans le contexte de l'ingénierie logicielle, un processus n'est pas une prescription rigide de comment construire un logiciel informatique. Il s'agit plutôt d'une approche adaptable qui permet aux personnes effectuant le travail (l'équipe du logiciel) de choisir l'ensemble d'actions et de tâches appropriées. L'intention est toujours de fournir des logiciels « en temps opportun et avec une qualité suffisante pour satisfaire ceux qui ont parrainé sa création et ceux qui l'utiliseront » (Pressman, Maxim, 2015).

Un processus, logiciel ou pas, peut être modélisé graphiquement. Il existe plusieurs outils pour documenter et modéliser les processus : Visio, Corporate Modeler, ARIS, Mega, Word et PowerPoint, etc. Mais la forme la plus connue est une cartographie de processus

utilisant le *Business Process Modeling Notation* (BPMN) qui est une notation graphique normalisée permettant de modéliser toutes les étapes d'un processus. Elle permet de représenter visuellement une séquence détaillée d'activités nécessaires à la réalisation d'un processus, par rôle impliqué et avec l'illustration d'évènements survenant à des moments précis. Cette notation a plusieurs avantages, grâce à sa représentation visuelle et son détail : elle permet mieux de comprendre le processus et de le mettre en œuvre. De plus, elle offre un langage normalisé à tous les personnels impliqués, ce qui facilite sa modification et son évolution. D'une manière générale un schéma est plus facile à comprendre qu'un texte (Lucidchart, 2018).

Dans la compagnie ACME, l'outil utilisé pour cartographier les processus est Visio.

2.2 Modèles des meilleures pratiques en développement et maintenance de logiciel

Pour aider les organisations à améliorer leur processus, des modèles de meilleures pratiques ont été publiés et sont utilisés à travers le monde par des milliers d'organisations. Parmi ces modèles, nous nous intéressons aux suivants dans notre contexte :

- Le *Capability Maturity Model Integration* (CMMI), et plus particulièrement sa version *CMMI for Dev, v1.3*, adaptée spécifiquement pour le développement de logiciel.
- Le *Software Maintenance Maturity Model* (S3M) ou Modèle de la maturité de la maintenance du logiciel, spécifiquement adapté pour la maintenance de logiciel.

Ces deux modèles parlent de « capacité » et de « maturité » du processus. La capacité d'un processus est définie comme « l'intervalle de résultats espérés en utilisant un processus » (Paulk, Curtis, Chrissis, Weber, 1993). La maturité d'un processus est définie comme « La portée et la granularité de la définition, de la gestion, du contrôle, des mesures et de l'efficience d'un processus » (Paulk, Curtis, Chrissis, Weber, 1993) et définie comme « une représentation d'attributs clés d'une activité organisationnelle spécifiée, ayant un effet direct sur sa progression vers l'accomplissement complet de son potentiel » (Garcia, 1993).

Nous décrivons ci-après ces deux modèles.

2.2.1 Capability Maturity Model Integration (CMMI)

Le modèle CMMI (*Capability Maturity Model Integration*) est un ensemble de bonnes pratiques destinées à aider les organisations à améliorer leurs processus (CMMI-DEV-Team, 2010). Le CMMI a été développé par le *Software Engineering Institute* (SEI) de l'université Carnegie-Mellon, à la demande du département de la Défense des États-Unis (*Department of Defense* ou DoD) et ceci dans le but, d'une part, d'éviter de donner des contrats à des entreprises immatures et, d'autre part, de savoir comment évaluer si une

entreprise est en mesure de livrer un produit de qualité (Wikipedia, 2018a). Le modèle regroupe des pratiques recommandées dans 22 domaines de processus qui peuvent être répartis selon deux représentations :

- La représentation en continu : Dans cette représentation, les 22 domaines de processus sont regroupés en quatre catégories. À chaque domaine de processus est associé un niveau de capacité allant de "0" à "3" (0-Incomplet ; 1-Basique ; 2-Discipliné et 3-Ajusté). Une évaluation du processus selon cette représentation fournit alors un « profil » de capacité du processus logiciel où l'organisation pourrait alors cibler des améliorations sur les domaines de processus dont le niveau de capacité serait le plus faible, au regard de ses besoins spécifiques (CMMI-DEV-Team, 2010 ; Wikipedia, 2018a ; Synechron, 2018).
- La représentation étagée : Dans cette représentation la plus courante, c'est un niveau global de maturité qui va être défini et non un niveau par domaine de processus (Synechron, 2018). À chaque niveau de maturité correspond un sous-ensemble des domaines de processus et des pratiques auxquelles doit répondre l'organisation au niveau visé (Synechron, 2018). Une évaluation du processus selon cette représentation fournit alors un niveau global de la maturité du processus logiciel. Afin d'augmenter le niveau de maturité de son processus logiciel, l'organisation pourrait cibler les domaines de processus du niveau visé dont les objectifs n'ont pas été atteints (CMMI-DEV ; Wikipedia, 2018a).

Dans représentation étagée, les domaines de processus sont regroupés en cinq niveaux de maturité. Les domaines de processus rattachés à un niveau de maturité M ne peuvent être stabilisés et efficaces que si les domaines de processus des niveaux inférieurs ($< M$) sont déjà stabilisés et efficaces. Les cinq niveaux sont (CMMI-DEV-Team, 2010) :

- **Niveau 1 (initial)** : Le niveau où le résultat final est imprévisible (budget, échéancier, portée, qualité). À ce niveau, le succès dépend davantage des individus en place et de leur engagement que de l'application disciplinée de bonnes pratiques. L'environnement de développement est instable. Il n'y pas de gestion de processus - très vulnérable aux crises. L'évaluation de l'efficacité et des performances est absente. La documentation est inexistante ou peu développée (CMMI-DEV-Team, 2010 ; Wikipedia, 2018a).
- **Niveau 2 (discipliné)** : Une gestion de projet élémentaire est définie pour assurer le suivi des coûts, des délais et des fonctionnalités du projet (SQLI-Group, 2019). La discipline nécessaire au processus est en place. Les processus des projets sont disciplinés, ce qui se caractérise par (CMMI-DEV-Team, 2010 ; Wikipedia, 2018a) :
  - Les activités sont planifiées et exécutées conformément à une politique d'organisation.
  - Pas de compromis sur la qualité.
  - Les rôles, responsabilités et parties prenantes sont définis et connus.
  - Les parties prenantes disposent des compétences et des ressources adéquates pour réaliser les produits.

  - La mise en œuvre du processus fait l'objet d'un suivi, de vérifications et d'ajustements si nécessaire.
- **Niveau 3 (défini)** : À ce niveau, l'organisation dispose d'un ensemble de processus normalisés qui sont ajustés par projet, selon le contexte du client, avec des règles fixées par l'organisation. Les processus normalisés sont développés, maintenus, supportés et leur application contrôlée par un groupe rsponsable du processus. Chaque projet capitalise son expérience et permet de bonifier le capital collectif. C'est aussi à ce niveau que les cycles de vie et processus d'ingénierie sont normalisés par typologie de projet. Le niveau 3 met en place des boucles d'amélioration : l'expérience, les forces et difficultés rencontrées lors des développements, sont capitalisées pour améliorer les développements futurs (CMMI-DEV-Team, 2010 ; Wikipedia, 2018a).
- **Niveau 4 (géré quantitativement)** : À ce niveau, les domaines de processus sont sous contrôle statistique (surveillance d'indicateurs quantitatifs et actions correctives s'il y a des dérives). Le cas échéant, on élimine les causes fondamentales de variation. Les méthodes statistiques de contrôle des processus mises en place au niveau 4 permettent de focaliser les actions d'amélioration sur les pratiques pour lesquelles ces actions sont les plus utiles. Au niveau des projets, elles permettent d'identifier des activités n'ayant pas atteint des résultats attendus et ainsi prendre des actions (CMMI-DEV-Team, 2010 ; Wikipedia, 2018a).
- **Niveau 5 (d'optimisation)** : À ce niveau, l'organisation est dans une boucle permanente d'optimisation (réduction des causes communes de variation) des processus et des technologies, optimisation basée sur des analyses coûts/bénéfices. Des analyses causales statistiques menées régulièrement permettent ces améliorations (CMMI-DEV-Team, 2010 ; Wikipedia, 2018a).

« Les boucles permanentes d'optimisation sont mises en place dès le niveau 3. Au niveau 5, celles-ci se basent sur des méthodes statistiques pour focaliser les analyses causales sur les événements dont l'analyse apportera des informations permettant d'optimiser les processus, en évitant de perdre du temps sur l'analyse d'évènements apportant moins d'information. » (Wikipedia, 2018a)

Le CMMI sera le principal modèle sur lequel nous baserons notre projet d'évaluer les processus logiciels d'ACME dans un but d'amélioration. Nous utiliserons la représentation étagée en se limitant au niveau 2 et en incluant quelques domaines de processus pertinents du niveau 3.

FIGURE 2.1 – Les niveaux de maturité de CMMI (tiré de SQLI-Group, 2019)

2.2.2 Software Maintenance Maturity Model (S3M)

Le référentiel S3M (*Modèle de la maturité de la maintenance du logiciel*) est un modèle d'évolution, soit « un moyen permettant de lancer et de guider un programme d'amélioration continue spécialisé pour la maintenance du logiciel » (April, Abran, 2016). Comme tous les modèles, S3M ne vise pas à résoudre tous les problèmes du logiciel, mais vise plutôt à offrir un outil approprié pour améliorer la maintenance des logiciels. « Le modèle S3M a été développé par Alain April et vérifié avec des collaborateurs de recherche du Dr Abran. Il s'inspire du modèle exploratoire mis au point par Mohammed Zitouni sous la direction du Dr Abran » (April, Abran, 2016).

— Architecture du modèle S3M : « Le modèle S3M est fondé sur le concept d'un itinéraire à suivre. Un itinéraire se définit comme un ensemble de pratiques liées qui couvrent plusieurs niveaux sur l'échelle du modèle. Le concept de facette a été introduit afin de créer des regroupemts selon un itinéraire, par exemple : l'ingénierie du changement devra prendre en compte un certain nombre de facettes telles que : a) la documentation technique ; b) la gestion des données » (April, Abran, 2016, p.124).

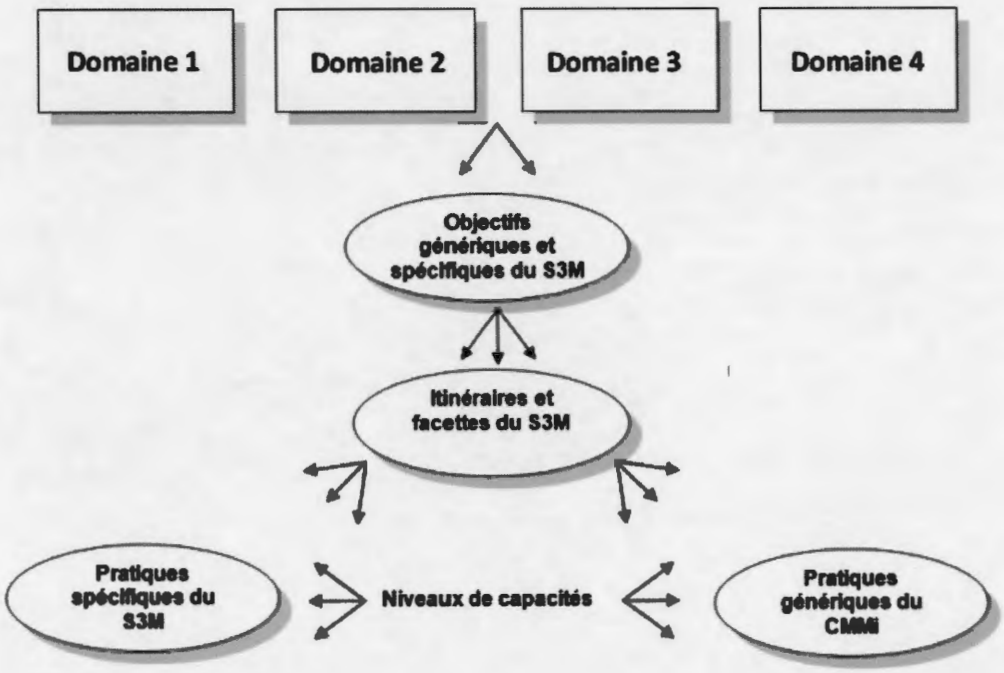

FIGURE 2.2 – Architecture du modéle modèle S3M (tiré et adapté de April, Abran, 2016)

- Les domaines de processus du modèle S3M : Le modèle S3M est subdivisé en quatre domaines de processus (gestion du processus, gestion de projet, ingénierie et support). Ce choix a été inspiré du CMMI, représentation en continu, pour garder une homogénéité entre les deux modèles (voir la Figure 2.3).

Domaines de processus de la maintenances S3M	Secteurs clés de la maintenance
Gestion des processus	1- Focalisation sur le processus maintenance 2- Définition des processus / Services maintenance 3- Formation organisationnelle de maintenance 4- Performance des processus de maintenance 5- Innovation en maintenance et déploiement
Gestion des requêtes	1- Gestion des requêtes de services et événements 2- Planification de la maintenance 3- Suivi et supervision des requêtes de maintenance 4- Gestion d'ententes de service et de sous-traitance
Ingénierie d'évolution	1- Coordination avant livraison et transition 2- Services de support opérationnel 3- Services d'évolution et de correction du logiciel 4- Vérification et validation des changements
Support à l'ingénierie d'évolution	1- Gestion de version et la configuration 2- Assurance qualité (Services, processus et produits) 3- Mesure et analyse de la maintenance 4- Analyse causale et résolution de problème 5- Rajeunissement, retraite et migration

FIGURE 2.3 – Domaines et secteurs du modéle modèle S3M (tiré et adapté de April, Abran, 2016)

— Les niveaux de maturité du S3M : Le modèle de maturité est basé sur une approche pratique et reconnaissable facilement par tous les intervenants dans le domaine de la maintenance du logiciel. April et Abran ont défini les niveaux de maturité du modèle S3M de la manière décrite à la Figure 2.4 (April, Abran, 2016) :

Niveau 0 - Inexistant	Le processus n'est pas effectué par l'organisation, l'organisation n'est pas consciente de l'existence de ce processus ou il n'en existe pas d'évidence.
Niveau 1 - Initial, improvisé	Reconnaissance de la pratique mais elle est faite de façon informelle.
Niveau 2 - Discipliné – Répétable mais intuitif	Conscience de la bonne pratique, mise en place de cette pratique ou exécution d'une pratique similaire.
Niveau 3 - Processus personnalisé	La pratique exemplaire est comprise et effectuée conformément à une procédure organisationnelle documentée.
Niveau 4 - Géré quantitativement et mesurable	La pratique exemplaire est effectuée formellement, est gérée quantitativement, conformément à un objectif et avec des limites établies.
Niveau 5 - En optimisation	La pratique exemplaire est sous contrôle statistique, conformément à un objectif et à l'intérieur des limites établies.

FIGURE 2.4 – Domaines et secteurs du modéle modèle S3M (tiré et adapté de April, Abran, 2016)

Bien que le modèle S3M soit des plus intéressants pour aider ACME à améliorer ses activités de maintenance du logiciel, nous ne l'utiliserons pas dans le cadre de notre projet parce que nous limitons notre portée à des projets de développement. Le modèle S3M demeure néanmoins un modèle à retenir pour l'évolution d'ACME.

2.3 Évaluation des processus logiciels et des logiciels

2.3.1 ISO 15504 pour évaluer le processus logiciel

ISO 15504, norme également connue sous le nom de SPICE (*Software Process Improvement and Capability dEtermination*), est un « cadre pour l'évaluation des processus » développé par le sous-comité technique mixte entre l'ISO (Organisation internationale pour Normalisation) et l'IEC (*International Electro Commission technique*) (Wikipedia, 2018b).

L'ISO 15504 était initialement dérivée de la norme de cycle de vie ISO 12207 et de nombreuses idées de modèles de maturité comme *Bootstrap*, *Trillium* et le *SW-CMM* (*Ca-*

pability Maturity Model for Software, soit le précurseur du CMMI) (Wikipedia, 2018b). La norme ISO 15504 est composée de dix parties. Les parties 1 à 5 sont les parties normatives tandis que les parties 6 à 10 sont des améliorations qui étendent l'utilisabilité et l'adoptabilité des concepts de capacité de processus à de nouveaux domaines de processus et à de nouvelles branches d'activités (HM&S, 2018, traduction libre).

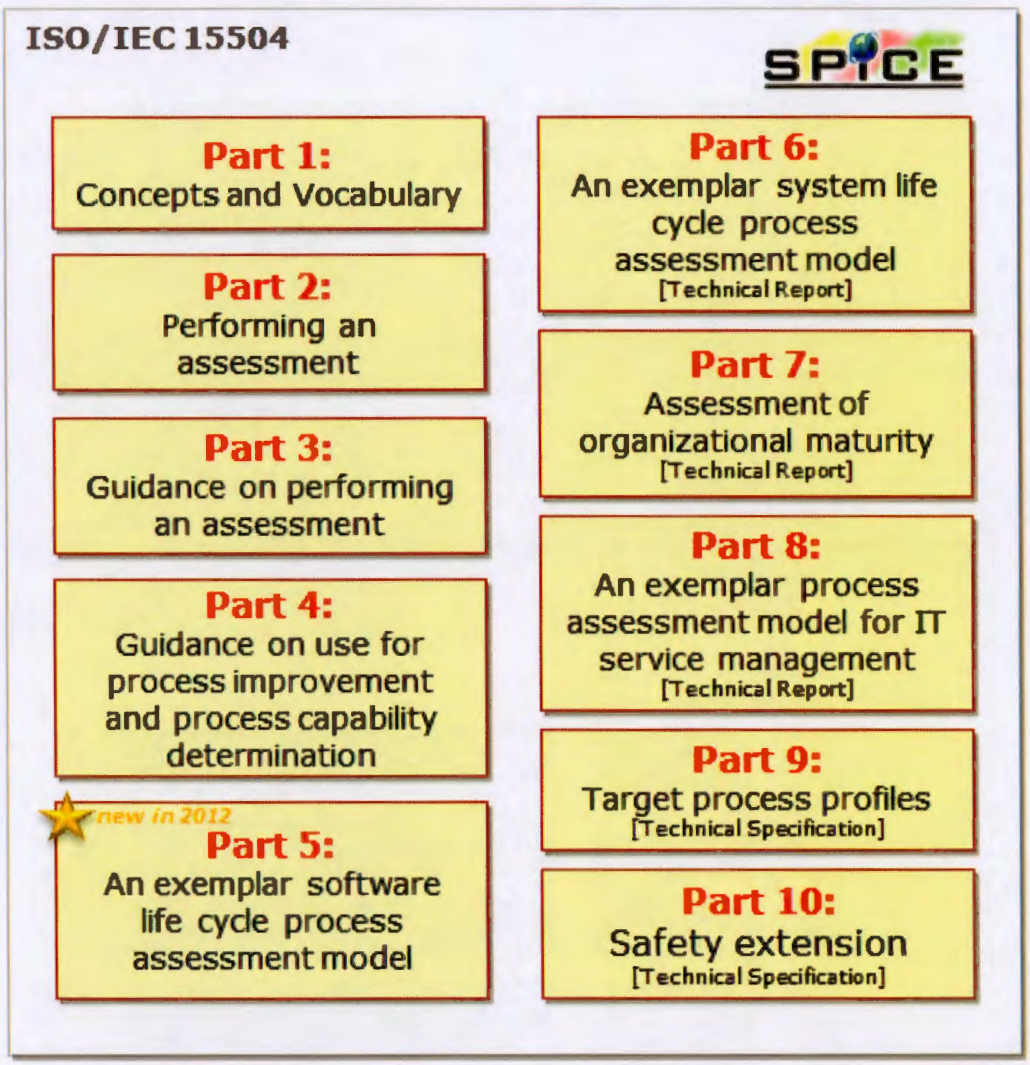

FIGURE 2.5 – La structure de la norme ISO 15504 (tiré de HM&S, 2018)

L'évaluation des processus est basée sur un modèle bidimensionnel contenant une dimension de processus et une dimension de capacité. La dimension de processus est fournie par un modèle de référence de processus externe, qui définit un ensemble de processus caractérisés par des déclarations d'objectifs de processus et des résultats de processus.

La dimension « capacité » comprend un cadre de mesure avec six niveaux de capacité de processus et leurs attributs de processus associés (Wikipedia, 2018b).

Les six niveaux de capacité de processus sont les suivants (Wikipedia, 2018b) :

0) Incomplet ;
1) Exécuté ;
2) Géré ;
3) Établi ;
4) Prévisible ; et
5) D'optimisation.

Les neuf attributs des processus sont les suivants (Wikipedia, 2018b) :

1. : Performance du processus ;
2. : Gestion du rendement ;
3. : Gestion des produits de travail ;
4. : Définition du processus ;
5. : Déploiement de processus ;
6. : Mesure de processus ;
7. : Contrôle de processus ;
8. : Processus d'innovation ;
9. : Optimisation du processus.

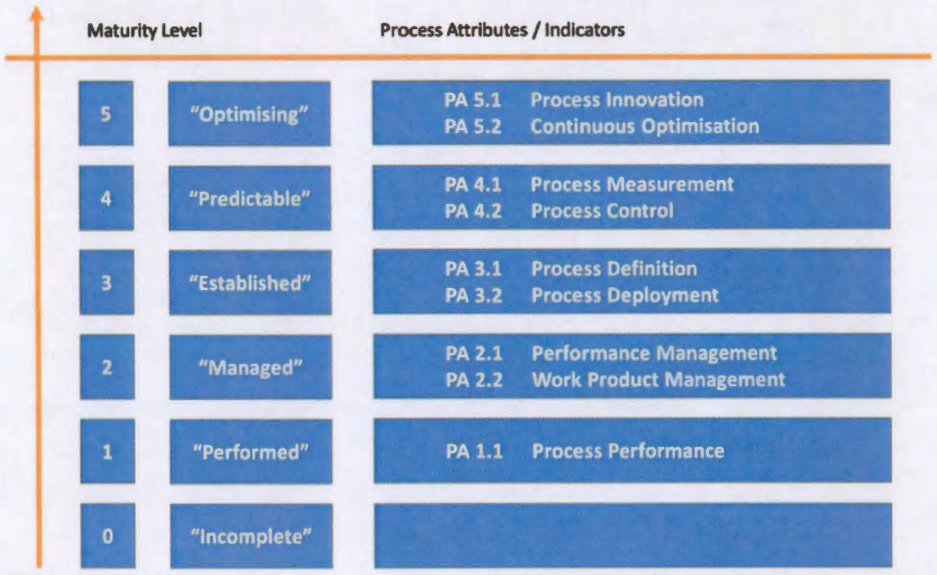

FIGURE 2.6 – Modèle d'évaluation bidimensionnel (tiré de Plays-in-Business, 2019)

Le résultat de l'évaluation consiste en un ensemble d'évaluations d'attributs de processus pour chaque processus évalué, appelé profil de processus, et peut également inclure le niveau de capacité atteint par ce processus.

La réalisation des attributs de processus est mesurée sur une échelle de quatre valeurs :

— N - Non atteint (*Not compliant*) ;
— P - Partiellement atteint (*Partly compliant*) ;
— L - Largement atteint (*Largely compliant*) ;
— F - Complètement atteint (*Fully compliant*).

Le niveau de maturité est déterminé par les mesures d'attributs de processus : pour atteindre un certain niveau de maturité, tous les attributs de processus du niveau en question doivent être réalisés au moins en « L » et tous les attributs de tous les niveaux en dessous doivent être « F » (Plays-in-Business, 2019, traduction libre).

Dans le cadre de notre projet, nous ne cherchions pas à coter les attributs des processus, mais simplement à identifier les forces et les faiblesses des processus d'ACME d'une manière générale. Aussi, étant donné que ce projet et le premier exercice d'évaluation des processus effectué en interne, et par une seule personne, il aurait été difficile d'obtenir un consensus de la cote de réalisation des attributs par 1) une équipe d'évaluation trop limitée et 2) une sélection relativement restreinte du nombre de projets évalués. Par ailleurs, la norme ISO/IEC 15504 étant générique, il nous fallait se rabattre sur une description de mise en œuvre plus pragmatique, et c'est pourquoi nous avons sélectionné la méthode PEM.

2.3.2 Méthode PEM pour évaluer le processus et le logciel

PEM (*Process Evaluation Method*) est une méthode d'évaluation qui combine une méthode d'évaluation qualitative basée sur CMMI pour évaluer et améliorer le processus logiciel d'une organisation, et une approche quantitative. PEM est aussi basée sur ISO/IEC 14598-5 pour permettre, au besoin, de mesurer divers attributs de qualité du logiciel issu du processu évalué.

PEM est destiné aux petites organisations ou aux petits projets, où une évaluation peut être effectuée en moins d'une semaine par un ou deux évaluateurs, fournissant des conclusions sur le processus logiciel et le logiciel.

PEM comporte sept étapes, les six premières sont extraites de l'ISO/IEC 14598-5 et de l'ISO/IEC 15504-5, et la septième a été créée comme une étape à valeur ajoutée pour l'organisation évaluée (voir la figure 2.6). Chaque étape est décrite plus en détail dans la méthode, similaire à une description de cas d'utilisation. Pour chaque étape, il y a une courte description, les objectifs de l'étape, les contributions, les pré-conditions à la réalisation de l'étape, le flux normal d'activités, les flux alternatifs (s'il y en a), la liste des activités de vérification qui sont menées, les sorties, les post-conditions à la réalisation de l'étape et les mesures à prendre (Trudel, Lavoie, Paré, Suryn, 2006).

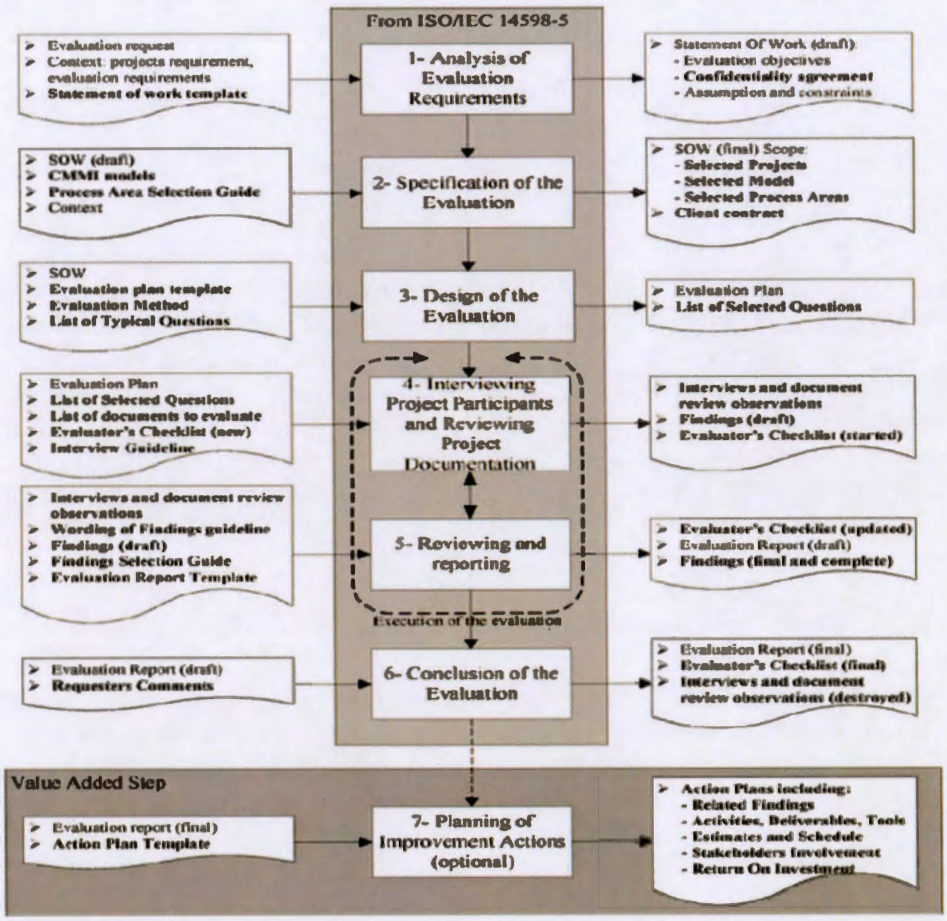

FIGURE 2.7 – Méthode PEM d'évaluation de processus (tiré de Trudel, Lavoie, Paré, Suryn, 2006)

1. **Analyser les exigences d'évaluation** : Décrire les besoins liés à l'évaluation et l'environnement de l'évaluation.
2. **Spécifier l'évaluation** : Définir la portée de l'évaluation à savoir les unités organisationnelles concernées par cette évaluation, ainsi que la sélection des projets et les pratiques logicielles du modèle sélectionné à évaluer.
3. **Concevoir l'évaluation** : Établir un plan d'évaluation détaillé pour décrire la méthode d'évaluation, identifier le personnel requis, planifier leur disponibilité et leur préparation et bâtir la liste des questions pour les entrevues.
4. **Interviewer les participants au projet et examiner le projet** : À cette étape, les évaluateurs doivent examiner la documentation du projet par rapport aux pratiques du modèle de processus sélectionné, ainsi que faire les entrevues avec le personnel identifié dans le plan d'évaluation et examiner les attributs de qualité du logiciel qui auront été mesurés. Cette étape consiste à produire les constats préliminaires tirés des entrevues et des examens.
5. **Documenter et réviser les constats** : Les constats sont rédigés à partir des observations. Chaque constat doit être appuyé par les commentaires des participants dans au moins deux entrevues différentes ou par des commentaires faits par un participant et appuyés par de la documentation examinée. Les constats sont mis en forme dans le rapport préliminaire.
6. **Conclure l'évaluation** : Le rapport préliminaire est présenté aux participants de l'évaluation pour obtenir leur validation des constats, à savoir que la terminologie utilisée est adéquate et comprise de tous et que les constats représentent leur situation. Leurs commentaires sont pris en compte pour rédiger le rapport final. À ce moment, des recommandations sont émises pour chaque groupe de constats afin d'améliorer le processus logiciel.
7. **Planifier des actions d'amélioration** : Bien que ISO/IEC 14598-5 ne mentionne aucune activité après la conclusion de l'évaluation, cette étape de planification d'actions est nécessaire pour fournir de la valeur au demandeur lorsque l'objectif est d'améliorer le processus logiciel. Après une telle évaluation, le demandeur est le plus souvent laissé seul avec une liste de constats importants, sans savoir ce qui pourrait être fait et comment le mettre en œuvre. Un plan d'implantation est utilisé pour définir les actions prioritaires, ainsi que les rôles et les responsabilités des parties prenantes et un ordre de grandeur du retour sur investissement.

Dans le cadre de notre projet, cette méthode PEM d'évaluation documentée était toute indiquée pour en dériver notre méthodologie de travail, décrite au chapitre 3.

2.3.3 Méthode COSMIC

« La méthode COSMIC est une méthode normalisée internationale (ISO 19761) pour mesurer la taille des exigences fonctionnelles de la plupart des domaines logiciels, incluant les logiciels d'application d'affaires (ou « MIS »), les logiciels temps réel, les logiciels d'infrastructures et certains types de logiciels scientifiques et d'ingénierie. » (COSMIC-Team-V4.0, 2015).

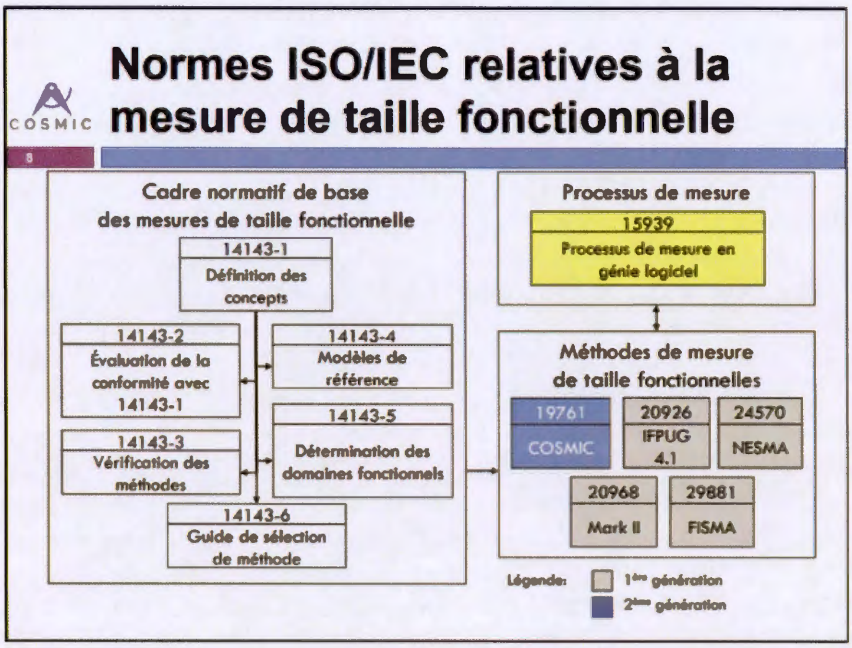

FIGURE 2.8 – Normes ISO relatives à la mesure de taille fonctionnelle (tiré de Trudel, 2018)

La méthode COSMIC (Common Software Measurement International Consortium) fait partie de la deuxième génération de méthodes de mesure de la taille fonctionnelle. Elle a été créée en 1998 par un groupe d'experts de mesurage du logiciel provenant de l'Australie, de l'Europe et de l'Amérique du Nord avec l'objectif de corriger les limitations des méthodes de mesures de la taille fonctionnelle de première génération et de développer une nouvelle méthode de mesure basée sur des principes bien établis du génie logiciel et sur des critères de la métrologie. « Aujourd'hui, la méthode est très largement utilisée partout dans le monde, dans tous les domaines pour lesquels elle a été conçue, tels que la mesure de la taille des logiciels à des fins contractuelles, et elle est appliquée avec succès pour la mesure du rendement, de l'étalonnage et de l'estimation des projets. » (COSMIC-Team-V4.0, 2015).

Cette méthode se base sur les mouvements de données nécessaires à la réalisation des

Fonctionnalités Utilisateurs Requises (FUR). Un mouvement de données est un composant fonctionnel de base qui déplace un seul type de groupe de données. Un groupe de données est un ensemble d'attributs décrivant un objet d'intérêt, par exemple un client ou une transaction. Chaque mouvement de données compte pour un point de fonction COSMIC (PFC, ou CFP en anglais). Il existe quatre types de mouvements de données décrits par COSMIC, soit Entrée (*Entry*), Sortie (*eXit*), éCriture (*Write*) et Lecture (*Read*). L'unité de mesure de la méthode COSMIC est CFP (point de fonction COSMIC ou *COSMIC Function Point*) (Trudel, 2018).

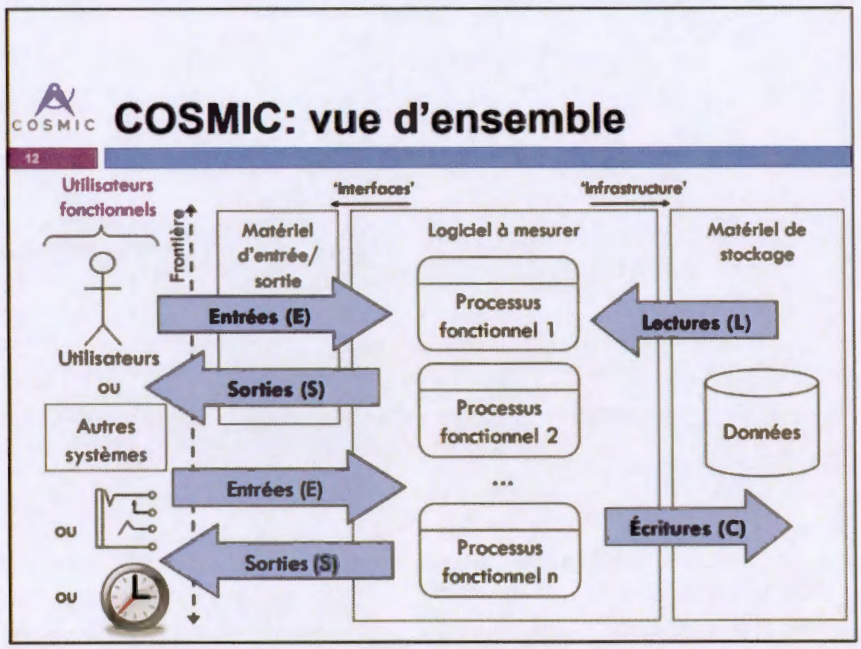

FIGURE 2.9 – COSMIC : vue d'ensmeble (tiré de Trudel, 2018)

Le processus général de mesure COSMIC comprend trois phases (Trudel, 2018) :

1. La phase de la stratégie de mesure : On détermine la raison d'être de la mesure (ex. établir un modèle d'estimation) et la portée de l'exercice de mesure (ex. une liste de projets spécifiques) ;

2. La phase d'arrimage : On applique le modèle générique de logiciel (c.-à-d. FUR, Processus fonctionnels, Groupes de données et Mouvements de données) à chaque logiciel devant être mesuré ;

3. La phase de mesure : On obtient les mesures de taille fonctionnelle en aditionnant les Mouvements de données.

Le résultat obtenu après avoir appliqué le processus de mesure sur un logiciel est une mesure de taille fonctionnelle du logiciel, exprimée en CFP.

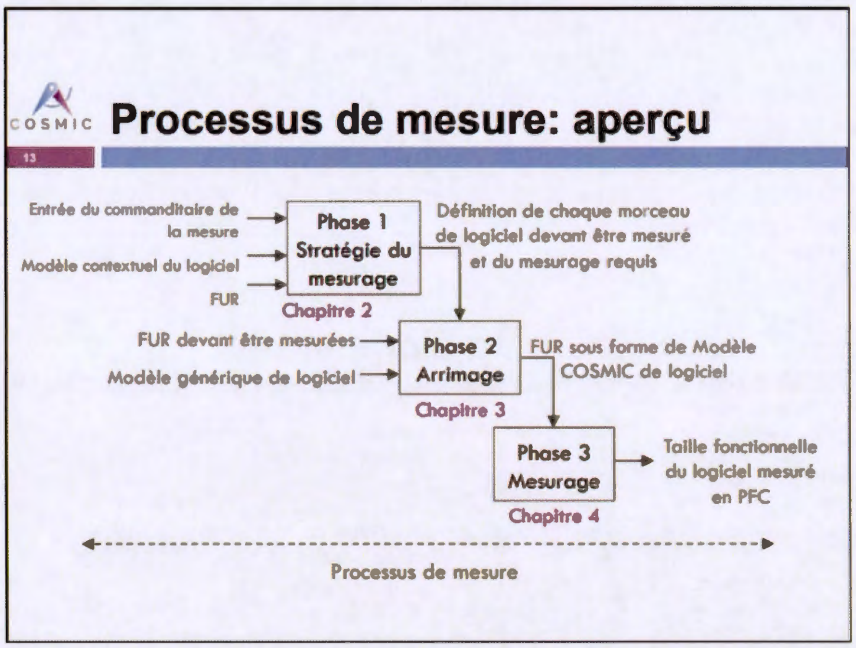

FIGURE 2.10 – Processus de mesure (tiré de Trudel, 2018)

Pour l'aspect quantitatif de l'évaluation des processus, nous avons opté pour une mesure de productivité où on combine l'effort des projets (en heures-personnes) avec la taille fonctionnelle des portions de logiciel de la portée des projets que nous avons évalués.

CHAPITRE 3

MÉTHODOLOGIE D'ÉVALUATION DU PROCESSUS LOGICIEL

Notre approche retenue pour l'amélioration des processus de la compagnie ACME est basée en grande partie sur la méthode PEM adaptée à notre contexte (voir la section 2.3.2 pour la description de la méthode PEM). Elle se compose de trois phases :

1. Démarrage, composée des activités suivantes :
   - Expliquer et présenter l'objectif du travail à la direction d'ACME ;
   - Spécifier les paramètres de l'évaluation ;
   - Préparer l'évaluation.
2. Évaluation, composée des activités suivantes :
   - Faire les entrevues avec le personnel sélectionné et les gestionnaires ;
   - Réviser et analyser la documentation et les données liées à l'application, aux projets d'évolution et aux opérations de maintenance ;
   - Rédiger les constats ;
   - Réviser les constats et rédiger le rapport brouillon ;
   - Envoyer le rapport brouillon aux participants pour validation.
3. Conclusion, composée des activités suivantes :
   - Finaliser le rapport d'évaluation ;
   - Discuter des résultats avec le président.

3.1 Aperçu de la méthodologie

Méthodologie			
Phases	**Intrants**	**Activités**	**Extrants**
Démarrage	• Opportunité d'amélioration identifiée	• Expliquer et présenter l'objectif du travail (3.2.1)	• Objectifs de l'évaluation • Accord de la direction pour faire l'évaluation
	• Normes et modèles de l'industrie • Liste des projets de RH.net	• Spécifier les paramètres de l'évaluation (3.2.2)	• Liste des projets sélectionnés • Modèles sélectionnés : CMMI et S3M • Liste des pratiques et domaines de processus sélectionnés parmi les modèles
	• Méthode PEM	• Préparer l'évaluation (3.2.3)	• Méthode sélectionnée et adaptée : PEM • Liste du personnel et des gestionnaires sélectionnés pour les entrevues • Questionnaire d'entrevue • Mesures de la qualité de RH.net • Plan de l'évaluation (horaire des activités avec estimation d'effort et durée)
Évaluation	• Plan de l'évaluation • Questionnaire d'entrevue • Liste des documents à évaluer • Mesures de la qualité de RH.net • Liste de vérification des activités à faire (nouvelle)	• Faire les entrevues avec le personnel sélectionné et les gestionnaires (3.3.1) • Réviser et analyser la documentation et des données liées à l'application, aux projets d'évolution et aux opérations de maintenance (3.3.2) • Faire les mesures (3.3.3)	• Observations (notes manuscrites et confidentielles de l'évaluateur) • Résultats de mesure (taille, effort, durée) • Constats (brouillon) • Liste de vérification des activités à faire (débutée)
	• Observations • Guide de sélection et rédaction des constats • Constats (brouillon)	• Rédiger les constats (3.3.4) • Envoyer le rapport brouillon aux participants pour validation (3.3.5)	• Rapport d'évaluation (brouillon) : o Constats (non validés) • Liste de vérification des activités à faire (mise à jour)
Conclusion	• Rapport d'évaluation (brouillon) • Commentaires des participants	• Finaliser le rapport d'évaluation (3.4.1)	• Rapport d'évaluation (final) : o Constats (validés) o Recommandations
	• Rapport d'évaluation (final)	• Discuter des résultats avec le président (3.4.2)	• Liste des améliorations considérées

FIGURE 3.1 – Aperçu de la méthodologie

3.2 Phase 1 : Démarrage

La phase de démarrage vise à confirmer le soutien de la direction pour cette évaluation et à se préparer adéquatement à la réaliser.

3.2.1 Expliquer et présenter l'objectif du travail

Le but de cette activité est d'expliquer et présenter l'objectif de l'évaluation [1] au président et au gestionnaire. Compte tenu des défis auxquels je faisais face de plus en plus dans mon travail, j'ai constaté l'opportunité d'amélioration, de laquelle a germé l'idée de faire une évaluation des processus et du produit logiciel pour améliorer la situation. Cette idée a été présentée la première fois dans une rencontre d'évaluation annuelle de

1. Voir le chapitre 1 pour les énoncés d'objectifs de cette évaluation.

l'organisation. Depuis ce temps, la rencontre avec le président a été repoussée plusieurs fois jusqu'à la fin du mois de mars 2018 où une rencontre rapide avec le président m'a permis d'avoir un accord verbal pour aller de l'avant avec cette évaluation.

3.2.2 Spécifier les paramètres de l'évaluation

Le but de cette activité est de définir la portée de l'évaluation selon les dimensions suivantes : les projets, les modèles de meilleures pratiques et les pratiques retenues de ces modèles.

— **Dimension « projets »** : Les projets sélectionnés (voir le tableau 3.1) devaient être des projets en cours ou des projets mis en production au cours de la dernière année, et ceci dans le but d'avoir des résultats qui reflètent les pratiques actuelles de la compagnie ACME. Le périmètre de l'évaluation inclut les activités de gestion de projet, d'analyse, de développement logiciel et d'assurance qualité [2].

Tableau 3.1 – Liste des projets sélectionnés

No. Projet	Nature du projet	Date de déploiement	Durée (jours)	Effort (heures)
A	Ajout	2017-08-15	183	265,00
B	Ajout	2017-09-22	142	181,50
C	Ajout	2017-08-29	161	526,00
D	Ajout	2016-06-04	54	164,25
E	Ajout	2018-05-22	85	363,25
F	Ajout	2018-05-24	109	312,75
G	Ajout	2018-06-20	95	137,75
H	Modification	2018-04-03	522	1347,25

— **Dimension « modèles de meilleures pratiques »** : Compte tenu que la nature des activités de développement logiciels est surtout constituée de projets d'évolution et de maintenance, nous avons sélectionné les modèles CMMI et S3M.

— **Dimension « pratiques sélectionnées »** : Presque tous les domaines de processus des niveaux de maturité 2 et 3 du CMMI ont été sélectionnés pour cette évaluation, sauf *Gestion des accords avec les fournisseurs* (SAM) (ACME n'utilise pas de sous-traitance), *Formation organisationnelle* (OT), *Analyse et prise de décision* (DAR) et *Gestion de projet intégré* (IPM) (ACME n'applique pas les pratiques de ces domaines) qui n'ont pas été retenus puisqu'ils ne sont pas pertinents dans notre contexte actuel. Du modèle S3M, nous avons sélectionné

2. L'assurance qualité est le nom donné chez ACME aux activités de test.

les secteurs clés de la maintenance équivalents aux domaines de processus sélectionnés du CMMI (voir le tableau 3.2).

Tableau 3.2 – Domaines de processus du CMMI sélectionnés pour notre évaluation

Sigle	Domaines de processus sélectionnés du CMMI	Catégorie	Niveau de maturité
CM	Gestion de la configuration	Support	2
PPQA	Assurance qualité processus et produit	Support	2
MA	Mesure et analyse	Support	2
PP	Planification de projet	Gestion de projet	2
PMC	Surveillance et contrôle de projet	Gestion de projet	2
RSKM	Gestion des risques	Gestion de projet	3
REQM	Gestion des exigences	Ingénierie	2
PI	Intégration de produit	Ingénierie	3
RD	Développement des exigences	Ingénierie	3
TS	Solution technique	Ingénierie	3
VAL	Validation	Ingénierie	3
VER	Vérification	Ingénierie	3
OPD	Définition du processus organisationnel	Gestion de processus	3
OPF	Focalisation sur le processus	Gestion de processus	3

L'architecture du modèle S3M est similaire à celle du modèle CMMi pour le logiciel. Cet arrimage au modèle CMMi permet ainsi de s'y référer plus facilement et d'utiliser certains documents existants du SEI, lorsqu'ils sont pertinents, pour mieux comprendre certaines activités de la maintenance du logiciel. Nous ne retenons du modèle S3M que les domaines processus correspondants à ceux sélectionnés du CMMi (voir le tableau 3.3).

Tableau 3.3 – Domaines de processus et secteurs clés du S3M sélectionnés pour notre évaluation

Domaines de processus S3M	Secteurs-clés de la maintenance	Pratiques S3M
Management des processus	Focalisation sur les processus	1.0.1
Management des processus	Définition des processus/services	2.0.1
Management des requêtes	Management des requêtes de services et événements	1.0.1

Domaines de processus S3M	Secteurs-clés de la maintenance	Pratiques S3M
Management des requêtes	Planification de la maintenance	2.0.1
Management des requêtes	Suivi et supervision des requêtes	3.0.1
Evolution du logiciel	Coordination avant livraison et transition	1.0.1
Evolution du logiciel	Services de support opérationnel	2.0.1
Evolution du logiciel	Services d'évolution et de correction	3.0.1
Evolution du logiciel	Vérification et validation	4.0.1
Support à l'évolution du logiciel	Management de versions et de configuration	1.0.1
Support à l'évolution du logiciel	Assurance qualité services, processus et produits	2.0.1
Support à l'évolution du logiciel	Analyse et mesure des services/logiciels	3.0.1

3.2.3 Préparer l'évaluation

Le but de cette activité est d'établir un plan d'évaluation détaillé pour décrire la méthode de l'évaluation, identifier le personnel requis et planifier leur disponibilité. Le tableau 3.4 résume cette planification. La durée indique le temps pendant lequel nous aurons besoin des différents participants mentionnés. L'effort prévu inclut l'effort pour préparer et faire l'activité et celui de rédiger les livrables issus de l'activité.

Tableau 3.4 – Plan d'évaluation détaillé

Activité	Durée	Livrable	Effort prévu(h)
Présentation de démarrage	60 min	Présentation de démarrage (PPT)	6h
Entrevue avec le gestionnaire	30 min	Observations	4h
Entrevue avec l'analyste d'affaires 1	30 min	Observations	4h
Entrevue avec l'analyste d'affaires 2	30 min	Observations	4h
Entrevue avec le résponsable de l'assuance qualité 1	30 min	Observations	4h
Entrevue avec le résponsable de l'assuance qualité 2	30 min	Observations	4h
Entrevue avec développeur 1	30 min	Observations	4h
Entrevue avec développeur 2	30 min	Observations	4h

Activité	Durée	Livrable	Effort prévu(h)
Revue des documents du processus de développement	1 sem.	Observations	20h
Rédiger le rapport d'évaluation (brouillon)	4 sem.	Constats bruts	60h
Vérifier et valider le rapport avec la directrice	4 sem.	Rapport brouillon vérifié	40h
Valider le rapport brouillon	1 sem.	Commentaires des participants	8h
Rapport d'évaluation (final)	8 sem.	Constats validés et recommandations	30h

3.3 Phase 2 : Évaluation

Cette section décrit les activités qui ont été réalisées lors de la phase de l'évaluation. Cette phase a permis de rédiger le rapport d'évaluation (brouillon) et les constats (non validés).

3.3.1 Faire les entrevues

Cette activité consiste à mener des entrevues avec les personnes sélectionnées suivant le plan d'évaluation. Elle permet entre autres de collecter les informations concernant les phases de gestion de projet, d'analyse, d'assurance qualité et de développement adoptées par ACME, et avoir l'opinion des différents participants sur le processus actuel et leur satisfaction. À cet effet, un ensemble de questions sont préparées pour aider à réaliser les évaluations dans de bonnes conditions (voir Annexe 01).

Cette étape permet d'obtenir les observations et les résultats de l'évaluation qui seront par la suite formulés et validés par les participants.

3.3.2 Réviser et analyser la documentation

Cette activité est une autre tâche importante du plan d'évaluation. Elle permet de réviser et d'analyser plusieurs produits de travail pertinents, les processus documentés pour les différentes activités d'ACME, les documents de spécifications, les plans de test, les activités de gestion de projet, et tout autre artéfact produit ou utilisé par les équipes des projets.

Au cours de notre projet, nous avons consulté un ensemble d'artéfacts qui sont produits ou utilisés par les projets. La plupart de ces artéfacts sont stockés sous *Share Point*. Le tableau 3.5 donne la liste et le nombre des artéfacts évalués.

Tableau 3.5 – Liste des documents révisés et analysés

Artéfact	**Type**	**Nombre**
Spécifications détaillées	Document Word	10
Document de gestion de projet	Chiffrier Excel	2
Processus de demande de changement	Document Word, fichier Visio	2
Processus de développement	Document Word, fichier Visio	2
Listes de vérifications pour les développeurs	Document Word	1
Plan de test	Document Word	3
Application web interne	Wiki	1 (30 pages)

Comme pour les entrevues, cette étape aussi permet d'obtenir des notes et des observations qui vont aider dans la rédaction du rapport d'évaluation (brouillon).

3.3.3 Mesurer la taille fonctionnelle

Pour mesurer la taille fonctionnelle nous avons utilisé la méthode COSMIC (voir le chapitre 2). Nous avons utilisé un chiffrier Excel dont le gabarit nous a été fourni lors du cours MGL7240 (Mesures et développement logiciel). Ce gabarit est composé de quatre onglets :

- **Stratégie** : fournit les informations suivantes : le nom de la personne qui a fait les mesures, la raison d'être de la mesure, la portée de la mesure, les couches de l'application, le niveau de décomposition, les utilisateurs fonctionnels, le niveau de granularité et la liste des documents utilisés.
- **Mesures** : fournit les informations suivantes : le type de changement (ajout, modification, retrait ou autre), le système mesuré, les références (no de cas d'utilisation, images, etc.), le processus fonctionnel, le groupe de données, les mouvements de données, le nombre d'entrées, le nombre de sorties, le nombre de lectures, le nombre d'écritures, le total en CFP et un commentaire.
- **Sommaire** : fournit le nombre d'entrées, le nombre de sorties, le nombre de lectures, le nombre d'écritures, le total en CFP, groupés par système et processus fonctionnel.
- **Paramètres** : fournit des paramètres avec leur description pour faciliter les mesures.

Les projets sélectionnés pour les mesures avaient les caractéristiques suivantes :

- Projets récents ;
- Projets représentatifs des projets habituels d'ACME ;

— Projets faits par des développeurs différents ;
— Projets ayant appliqué le processus logiciel tel qu'on le connait.

Nous nous sommes limités à sept projets compte tenu du temps que nous avions pour faire cette étude. Aussi, nous croyons que ce nombre est suffisant pour élaborer nos premiers modèles de productivité de l'effort (le nombre d'heures d'effort réel par CFP) et de la durée (le nombre de CFP livrés par semaine) et, par extension, nos premiers modèles d'estimation de l'effort (le nombre estimé d'heures d'effort pour un projet de X CFP) et de la durée (le nombre estimé de semaines pour un projet de X CFP).

Par ailleurs, nos mesures de taille fonctionnelle seront vérifiées par un autre mesureur certifié sur la méthode COSMIC.

3.3.4 Rédiger les constats

Cette activité permet de formuler des constats à partir des notes et des observations collectées lors des activités d'évaluation et d'analyse des documents de la campagne ACME. L'aspect important de cette activité est de s'assurer que la façon de phraser chacun des constats ne permettra pas d'identifier l'individu qui en est à l'origine, afin d'éviter qu'il puisse être blâmé par la suite. Nous voulons aussi nous assurer de la confidentialité des observations et des notes. Pour cette raison, les observations et les notes ne deviendront un constat que si les mêmes observations ont été faites dans deux entrevues distinctes (ou plus) ou ressorties dans une entrevue et l'analyse d'un artéfact ou décument.

Toute observation ou note non corroborée ne sera pas prise en compte dans la rédaction des constats, puisque qu'elles seront davantage annecdotiques que factuelles.

Le livrable principal de cette activité est le rapport brouillon, dans lequel chacun des constats est identifié de façon unique. Les constats sont aussi vérifiés par la directice de ce projet technique afin d'en assurer le caractère anonyme et l'adéquation de leur rédaction.

3.3.5 Envoyer le rapport brouillon aux participants pour validation

Durant cette activité, le rapport brouillon issu de l'activité précédente est présenté aux participants pour validation. Cette démarche vise à s'assurer que l'interprétation de leurs affirmations et que la confidentialité des entretiens soit préservée. Nous demandons alors aux participants de nous envoyer leurs commentaires par courriel, s'ils en ont, à savoir : les constats à modifier, les constats à corriger, les constats qui seraient manquants selon eux, et les constats superflus. Pour chaque commentaire, nous leur demandons d'identifier le constat pour lequel une modification du rapport serait requise, sauf pour les constats manquants, bien sûr.

Nous accordons un délai d'une semaine aux participants pour nous envoyer leurs commentaires.

3.4 Phase 3 : Conclusion

Cette section décrit les activités qui ont été réalisées lors de la phase de la conclusion. Cette phase permet de finaliser le rapport d'évaluation et de discuter les résultats avec le président.

3.4.1 Finaliser le rapport d'évaluation

Après le délai accordé pour la validation des constats, nous allons intégrer leurs commentaires pertinents au rapport. Le chapitre 4 va contenir les constats sur le processus mis en place par ACME ainsi qu'une liste des facteurs pouvant expliquer le nombre élevé de bogues d'ACME sur la plupart de projets sélectionnés.

À ce moment seulement, nous pouvons rédiger les recommandations qui seront fondées sur les constats validés. Pour chaque recommandation, nous allons estimer grossièrement :

- leur priorité de mise en oeuvre (basse, moyenne, élevée) [3] ;
- leur horizon de mise en œuvre (court, moyen ou long terme) [4] ;
- le niveau d'effort requis (faible, moyen, élevé) [5] ; et
- le degré d'autonomie de mise en œuvre par les employés actuels d'ACME (faible, moyen, élevé) [6].

Le chapitre 5 de ce projet va contenir ces recommandations qui seront présentées au président à la fin de ce projet. Le présent rapport constitue le rapport final de cette évaluation.

---

3. Selon notre imterprétation des retombées potentielles et positives sur le processus de développement ou la qualité.

4. Court terme : moins de 3 mois ; Moyen terme : 3 à 12 mois ; Long terme : plus de 12 mois.

5. Faible effort : facilement intégrable dans les activités quotidienne sans dérangement significatif pour les projets en cours ; Effort moyen : moins de 80 heures, soit l'équivalent de deux semaines d'effort pour une personne ou moins ; Effort élevé : plus de 80 heures.

6. Le degré d'autonomie indiquera les besoins en formation, le coaching requis ou le besoin d'avoir recours à des experts externes. Élevé : nous avons les compétences requises parmi les membres d'équipe ; Moyen : une formation d'appoint doit être suivie par un ou plusieurs membres d'équipe (lectures dirigées, formation en classe ou en ligne, coaching à temps partiel, etc.) ; Faible : ACME n'a pas les compétences requises et doit faire appel à des experts externes.

3.4.2 Discuter des résultats avec le président

À la fin de ce projet, une rencontre est prévue avec le président pour, d'une part, présenter les recommandations et, d'autre part, proposer de l'aide à ACME pour élaborer un plan d'action s'il souhaite aller de l'avant avec les recommandations.

CHAPITRE 4

CONSTATS DÉCOULANT DE L'ÉVALUATION

Le présent chapitre comporte les résultats obtenus dans le cadre de l'évaluation, soit les constats qui y sont liés pour chaque domaine de processus parmi les domaines du CMMI pris en considération dans ce projet. Ces constats sont regroupés par catégorie des domaines de processus :

- Gestion de projet
- Gestion de processus
- Processus de support
- Ingénierie

4.1 Gestion de projet

Selon le Guide du corpus des connaissances en gestion de projet (PMBoK), la gestion de projet consiste à :
« appliquer des connaissances, des compétences, des outils et des techniques aux activités d'un projet pour répondre à ses exigences. La gestion de projet s'effectue à l'aide de l'application appropriée et de l'intégration des processus de gestion de projet : démarrage, planification, exécution, surveillance et contrôle, et clôture. La gestion d'un projet comprend généralement, mais sans s'y limiter : identifier les besoins, répondre aux divers besoins, préoccupations et attentes des parties prenantes lors de la planification et de l'exécution du projet, maintenir et mettre en place une communication active et efficace entre les parties prenantes, gérer les parties prenantes pour répondre aux exigences du projet et créer des produits livrables, équilibrer les contraintes parfois orthogonales du projet, qui incluent la portée, la qualité, l'échéancier, le budget, les ressources et les risques.
Les caractéristiques et les circonstances spécifiques du projet peuvent influer sur les contraintes sur lesquelles l'équipe de gestion de projet doit se concentrer. La relation entre ces facteurs est telle que si l'un de ces facteurs change, au moins un autre facteur risque d'être affecté. Par exemple, si le calendrier est raccourci, le budget doit souvent être augmenté pour ajouter des ressources supplémentaires » (traduction libre de Project-Management-Institute, 2013).

Les constats liés à la Gestion de projet sont tirés des domaines de processus REQM (Gestion des exigences), PP (Planification de projet), PMC (Suivi et supervision de projet) et RSKM (Gestion du risque) :

- « Le but de REQM est de gérer les exigences des produits et composants de produit du projet, et d'assurer l'alignement de ces exigences sur les plans et les produits d'activité du projet » (CMMI-DEV-Team, 2010, p.419).
- « Le but de PP est d'établir et de maintenir les plans qui définissent les activités de projet » (CMMI-DEV-Team, 2010, p.349).
- « Le but de PMC est de fournir une appréciation de l'avancement du projet, de telle sorte que des actions correctives puissent être prises quand la performance du projet s'écarte de façon significative du plan » (CMMI-DEV-Team, 2010, p.339).
- « Le but de RSKM est d'identifier des problèmes potentiels avant qu'ils ne surviennent, de telle sorte que les activités pour traiter les risques puissent être planifiées et déclenchées au besoin tout au long de la vie du produit ou du projet afin que les impacts nuisibles à l'atteinte des objectifs soient atténués » (CMMI-DEV-Team, 2010, p.417).

4.1.1 La gestion de projet chez ACME

Chez ACME, on distingue deux types de projets :

— Mise en production d'un nouveau client lors de la signature d'un contrat : c'est un type de projet bien structuré avec un plan de projet élaboré avec un budget et une date de livraison ;
— Développement de nouvelles fonctionnalités : notre étude concerne ce dernier type de projet pour lequel nous donnons plus de détails ci-après.

Chez ACME, la gestion des projets de développement comprend les activités suivantes :

— Définition des tâches et responsabilités par le gestionnaire
— Affectation des tâches de développement aux membres de l'équipe selon leur expertise
— Présentation des spécifications par l'analyste aux développeurs et personnel de l'AQ
— Estimation de l'effort par les développeurs et le personnel d'AQ
— Validation des estimations par le gestionnaire
— Priorisation des tâches par le gestionnaire
— Suivi hebdomadaire de l'état d'avancement lors d'une rencontre appelée *kick-off* le lundi matin, avec ou sans publication d'un compte-rendu
— Utilisation de *Excel* et *Free Mind* pour faire le suivi et le découpage du projet.

4.1.2 Constat GPJ-1 : Sous-estimation systématique de la plupart des projets

Chez ACME, au début de chaque nouveau développement, une estimation est faite par le gestionnaire, l'analyste ou le développeur ou tous. Sauf que ces estimations ne sont pas fondées sur des arguments solides, soit par une sous-estimation de la quantité de travail du côte du gestionnaire et de l'analyste, soit par une lecture superficielle des spécifications de la part du développeur. Les estimations des développeurs sont régulièrement diminuées par le gestionnaire au moment de leur validation. Aussi, les estimations validées ne tiennent pas compte des tâches en parallèle comme la correction de bogues urgents, l'aide qu'un membre expérimenté peut fournir à un nouveau développeur, les demandes de compilation d'une version de l'application, les demandes expresses de tâches sur des projets clients et l'investigation sur d'éventuels problèmes de l'application.

Par conséquent, la plupart des projets ont connu beaucoup de retards et de démotivation des développeurs (un employé non motivé ne peut pas fournir un bon rendement ou tirer du plaisir dans son travail). On peut même aller des fois jusqu'à l'escamotage du processus pour livrer plus vite, sachant le travail incomplet, et qui sera retourné par l'équipe QA de façon à donner davantage de temps au développeur pour compléter. Une conséquence directe de cette sous-estimation est une augmentation du nombre de bogues.

4.1.3 Constat GPJ-2 : Nombreux travaux en parallèle pour chaque développeur

Comme le produit RH.net est déjà développé et commercialisé, ACME a adopté une stratégie pour maintenir ce produit pour qu'il soit à la hauteur des attentes des clients et au niveau des produits concurrents. Pour cette raison, la date de livraison d'un nouveau projet de développement peut être repoussée à une date indéterminée dans le cas où le président décide que certains ou tous les développeurs doivent travailler sur un projet client, une demande de changement client, ou bien la correction de bogues urgents. Tout cela fait qu'un développeur peut travailler sur deux ou plusieurs projets en parallèle. Par conséquent, le développeur perd sa concentration et sa motivation et une conséquence directe est une augmentation du nombre de bogues et le non respect de l'échéancier des projets de développement.

4.1.4 Constat GPJ-3 : Personnel insuffisant pour effectuer la gestion de projet

Chez ACME, il y a une seule personne affectée pour la gestion des deux types de projet, soit le gestionnaire. Pour assurer la satisfaction des clients, cette personne passe plus du temps à gérer les projets clients que de gérer les projets de développement. Plusieurs intervenants m'ont confirmé que la plupart du temps, ils doivent attendre plusieurs heures pour avoir des réponses sur des points en suspens ou des questions. En conséquence, cette perte de productivité étire les échéanciers.

4.1.5 Constat GPJ-4 : Blâmes publics envers les membres d'équipe lors du *kick-off* du lundi matin

Chez ACME, on fait une rencontre chaque lundi matin (*kick-off*). Cette rencontre est une occasion pour le gestionnaire, les analystes, les testeurs et les développeurs de donner un bref résumé sur les tâches achevées dans la semaine passée et donner le plan de match pour la semaine en cours, ainsi que pour discuter avec les autres membres de l'équipe des problèmes rencontrés. Cependant, selon plusieurs intervenants, le président n'hésite pas à blâmer certains membres d'équipe en public, principalement pour le non-respect de l'échéance, ce qui rend cette rencontre désagréable. Par ailleurs, le gestionnaire n'intervient pas lors de ces blâmes. Certains intervenants m'ont confié qu'ils cherchent des excuses pour ne pas assister à la rencontre ou même pour s'absenter le jour de cette recontre. Les conséquences de ce comportement sont :

— Démotivation du personnel
— Données incomplètes pour le suivi des projets
— Non amélioration du processus de développement
— Obstacle à la collaboration et l'entraide entre les membres de l'équipe.

4.1.6 Constat GPJ-5 : Processus normalisé

Chez ACME, un processus normalisé de toutes les activités permet d'alléger énormément la planification du projet. Dans tous les projets de développement, un certain nombre d'activités sont menées par le gestionnaire, ce qui lui permet d'avoir un certain contrôle sur son projet :

— Au début de chaque nouveau de projet, le gestionnaire définit les tâches et responsabilités et affecte des tâches de développement aux membres de l'équipe selon leur expertise.
— Avant le début de chaque tâche, les développeurs et le personnel d'AQ estiment l'effort, ce qui donne au gestionnaire une idée de la quantité de travail nécessaire.
— Le gestionnaire priorise régulièrement les tâches, pour être conforme aux tendances du marché et aux attentes des clients.
— Une rencontre hebdomadaire chaque lundi matin (*kick-off*) permet au gestionnaire de faire un suivi de projet en se basant sur les informations des tâches achevées dans la semaine précédante, en comparant l'effort réel de la réalisation avec l'estimation initiale et d'ajuster le calendrier en conséquence et prendre des mesures correctives lorsque des écarts ont été identifiés.
— Le gestionnaire planifie les dates de compilation à des jalons précis (que l'équipe appelle *milestones*), où une version intermédire de l'application exécutable pour fins de test est produite par un développeur senior.

4.1.7 Constat GPJ-6 : Gestion de risque informelle et réactive

Malgré l'absence d'un plan formel de la gestion des risques, ACME réagit souvent au bon moment et de la bonne manière quand un risque se manifeste et ceci grâce aux facteurs suivants :

1. En se basant sur l'expérience accumulée durant plusieurs années pour résoudre des risques similaires.
2. Les dirigeants connaissent bien la capacité et les limites du logiciel RH.net. Par conséquent, ils ne s'engagent pas dans des demandes qui peuvent provoquer d'énormes surprises ou qui sont trop risquées (évitement de certains types de risques).
3. Une flexibilité dans l'affectation des tâches permet de corriger les problèmes.

Par conséquent, les risques qui se manifestent sont traités rapidement bien que ce soit de manière réactive.

4.1.8 Constat GPJ-7 : Peu ou pas de travail d'équipe

Selon plusieurs intervenants, chaque développeur travaille sur une tâche dans son coin et, à la fin, il livre un produit en rassemblant tous les morceaux. À part quelques revues par les pairs pour les nouveaux développeurs juste avant l'intégration, le reste du travail se fait de façon individuelle.

Aussi, l'évaluation de performance des individus est fonction du temps qu'il met pour réaliser sa tâche et le nombre de bogues trouvés dans sa partie. Cette mesure, à elle seule, favorise les comportements individualistes. Il n'y a pas de mesures de performance d'équipe, donc on ne favorise pas le travail d'équipe. Or, des études ont démontré que les travaux faits en équipe permettaient aux membres d'équipe d'être plus performants, de développer un meilleur sentiment d'accomplissement, d'être davantage fiers de leur travail, de générer des produits de plus grande qualité, d'améliorer leurs compétences plus rapidement grâce au *feedback* des autres et d'être plus motivés et heureux au travail.

4.2 Gestion de processus

Selon le CMMI pour le développement, la gestion de processus aborde « les activités transversales aux projets relatives à la définition, à la planification, au déploiement, à la mise en œuvre, à la surveillance, au contrôle, à l'évaluation, à la mesure et enfin à l'amélioration des processus. Les domaines de processus de la gestion de processus basique permettent à l'organisation de documenter et de partager les meilleures pratiques, les actifs de processus organisationnels et les retours d'expérience » (CMMI-DEV-Team, 2010).

Les constats liés à la Gestion de processus sont tirés des domaines de processus OPD (Développement du processus de l'organisation) et OPF (Focalisation sur le processus de l'organisation) :

- « Le but de OPD est d'établir et de gérer un ensemble utilisable d'actifs de processus organisationnels, de normes d'environnement de travail et de règles et directives pour les équipes, pour améliorer en permanence les processus et autres actifs de processus » (CMMI-DEV-Team, 2010, p.247).
- « Le but de OPF est d'aider à la planification, l'amélioration et au déploiement des processus organisationnels en s'appuyant sur l'expériences de l'équipe de projet et sur sa compréhension approfondie des forces et des faiblesses des processus actuels » (CMMI-DEV-Team, 2010, p.261).

Pour qu'un processus organisationnel demeure efficace et efficient, il est important qu'une organisation puisse l'améliorer de manière continue. Le CMMI recommande d'identifier les occasions d'amélioration, de prendre action pour améliorer, déployer et communiquer à tous les nouvelles versions du processus.

4.2.1 Constat PRS-1 : Processus organisationnel défini

ACME a défini clairement son processus organisationnel en incluant les cycles de vie du logiciel et les différentes activités à mener durant toutes les phases de développement de logiciel. Ces informations sont stockées sur *SharePoint* et sur le wiki de l'organisation. Des lignes directrices sont données pour les activités suivantes : Développement par prototypage, Convention de codage et de nomenclature (*coding standard*), Tests unitaires, Compilation, Révision du code source et Implémentation d'une modification. Ce processus défini est bien connu des membres de l'équipe qui travaillent dans l'organisation depuis plusieurs années, mais il est peu connu des nouveaux employés.

4.2.2 Constat PRS-2 : Processus non communiqué et mise en œuvre non surveillée

Malgré tout l'effort qu'ACME a mis pour mettre en place son processus organisationnel, il n'est pas connu par les nouveaux arrivants. Aussi, sa mise on œuvre n'est pas surveillée lorsqu'appliquée par les membres expérimentés de ACME. En conséquence, le processus n'est pas appliqué uniformément et les mesures s'y rapportant sont difficiles à interpréter. Par ailleurs, on constate une augmentation du nombre de bogues rapportés dans le logiciel bien que ce processus vise à minimiser le nombre de bogues.

4.2.3 Constat PRS-3 : Pas de mécanisme d'amélioration continue

L'équipe d'ACME ne prend pas conscience des problèmes de son processus afin de les éliminer et ainsi l'améliorer. Par exemple, il n'y a pas de rétrospective à la fin d'un

cycle ou d'un projet, malgré le fait que plusieurs membres de l'équipe connaissent cette technique. En conséquence, les mêmes problèmes et erreurs sont répétées dans plusieurs projets. Cela amène un surplus de travail à refaire qui vient directement impacter l'efficience de l'équipe.

Par ailleurs, selon les intervenants, le processus d'ACME n'a jamais été révisé depuis son déploiement initial et plusieurs de ses éléments ne sont plus d'actualité ou plus utilisés. Par conséquent ce processus est devenu moins efficace.

4.3 Processus de support (SUP)

Selon le CMMI pour le développement (CMMI-DEV-Team, 2010), le domaines de processus de la catégorie Support couvrent « les activités qui soutiennent le développement de produit et sa maintenance. Ils concernent les processus employés dans le contexte de la réalisation d'autres processus. Généralement, ces processus portent sur les domaines qui sont ciblés par le projet. Ils peuvent intéresser des processus qui s'appliquent plus généralement à l'organisation. Par exemple, 'Assurance qualité processus et produit' peut être associé à tous les domaines de processus pour permettre d'obtenir une évaluation objective des processus et des produits d'activités décrits dans tous les domaines de processus » (CMMI-DEV-Team, 2010).

Les constats liés aux processus de support sont tirés des domaines de processus CM (Gestion de configuration), MA (Mesure et analyse) et PPQA (Assurance qualité processus et produit) :

- « Le but de CM est d'établir et de maintenir l'intégrité des produits de travail d'une façon adéquate, complète, précise et de les protéger contre toute modification non autorisée et ceci par l'identification, contrôle et audits des éléments de configuration » (CMMI-DEV-Team, 2010, p.185).
- « Le but de MA est de développer et maintenir une capacité à mesurer utilisée pour soutenir les besoins d'information de gestion. Autrement dit, le but de la mesure est d'analyser et de s'assurer que chaque partie appliquée dans les projet obtienne des données de qualité qui vont l'aider à prendre ses décisions » (CMMI-DEV-Team, 2010, p.185). Les objectifs de la mesure peuvent couvrir plusieurs éléments tels que le budget, les délais, la qualité et les performances du produit et du projet. Plus précisément, les éléments jugés importants pour l'entreprise doivent être définis comme des objectifs.
- « Le but de PPQA est de fournir aux intervenants une vision objective sur l'assurance qualité des processus et produits de travail associés. Le PPQA est mené pour identifier les problèmes de non-conformité liés à ces processus et produits de travail, puis pour suivre ces problèmes de non-conformité jusqu'à leur clôture » (CMMI-DEV-Team, 2010, p.185).

4.3.1 Constat SUP-1 : Normes de programmation définies et appliquées

Afin que tout développeur puisse lire le code de quelqu'un d'autre, ACME exige que le codage soit uniforme. C'est pourquoi des documents décrivant les normes internes sont disponibles pour référence. Ces document fournissent une description complète, précise et détaillée de la syntaxe et de la sémantique. Ils donnent des détails sur l'organisation des fichiers et leurs nomenclatures, les *name spaces* et leur utilisation, les déclarations de classes et d'interfaces, l'utilisation des commentaires, l'initialisation et la nomenclature des variables, l'utilisation des méthodes anonymes et les expressions Lambda.

Pour assurer le respect de les conventions de nommage, ACME a mis en place plusieurs outils et méthodes :

— Utilisation de gabarits : ACME a développé des modèles, dès que le développeur les utilise, une grande partie des règles des normes internes sont automatiquement implémentées et respectées.
— Utilisation des outils de vérification de style comme *Style Coop* et *fixCoop* où chaque utilisateur peut configurer l'ensembles des règles qui répondent à son besoin.

Et finalement, le code des employés récents (moins d'un an à l'emploi d'ACME) est révisé notamment pour assurer l'application des normes internes (voir le domaine VER dans la catégorie *Ingénierie*).

4.3.2 Constat SUP-2 : Outils mis en place pour la gestion des modifications

Pour gérer l'ensemble des modifications apportées au cours de l'évolution du système, ACME a mis en place un ensemble d'outils. En d'autres termes, il s'agit de l'ensemble des processus permettant d'assurer la conformité d'un produit aux exigences, tout au long de son cycle de vie. De plus, pour des raisons de sécurité, ACME a mis en place des contrôles : des procédures sont définies et implémentées pour stocker et archiver les données critiques (toutes données ayant un impact sur les données et systèmes des clients) de façon à ce que ces données demeurent accessibles et utilisables, et pour gérer les sauvegardes et les récupérations. Pour réaliser ceci, ACME utilise *Team Foundation System* (TFS) pour gérer les versions de son logiciel, soit du code ou des scripts de base de données, et pour surveiller les demandes de changement ainsi que les incidents de production. TFS permet de suivre toutes les demandes soumises à ACME et de documenter l'échelle de résolution ainsi que le processus d'affectation des demandes. *Sharepoint* est utilisé pour la gestion des versions de toute la documentation.

4.3.3 Constat SUP-3 : Gestion de l'intégrité des versions de RH.net

Chez ACME, les environnements de test, de développement et de production sont séparés. Il y a aussi une procédure (manuelle) pour compiler une version.

Bien qu'il n'y ait pas d'audit de configuration, TFS a permis d'automatiser le processus pour produire une version intègre d'un référentiel de RH.net. Chaque version du logiciel RH.net est sur une branche séparée de TFS. Il y aussi une politique pour numéroter les versions et faire une distinction entre les versions en production et celles en développement. Un logiciel de contrôle de version permet de conserver une piste de vérification sur toutes les versions déployées en production.

4.3.4 Constat SUP-4 : Documents relatifs à la gestion de configuration peu fréquentés

ACME a créé un certain nombre de procédures et *checklists* contenant l'information relative à la gestion de configuration : compilation, déploiement, *check-in*, scripts de base de données, configuration d'un poste de travail, installation des outils, comment utiliser TFS (ex. pour créer une nouvelle branche de code), et utilisation de scripts de personnalisation. Ces documents sont soit dans le *wiki* de l'équipe, soit sous SharePoint.

L'utilisation de ces documents relatifs à la gestion de configuration est peu fréquente car le nombre de visualisations de ces documents montre que personne ne les a consulté (pour la plupart des documents consultés, il y a seulement une seule visualisation). Donc on peut se demander si les pratiques qui y sont définies sont réellement appliquées. Cependant, leur contenu est transmis verbalement quand, par exemple, un nouveau collègue demande à un ancien comment faire telle ou telle procédure.

4.3.5 Constat SUP-5 : Planification informelle des mesures et analyse

Chez ACME, quelques mesures sont utilisées pour la gestion des projets, des activités de développement et de la qualité du produit :

1. Qualité du produit : Avec TFS, on collecte sur demande le nombre de bogues par statut (statuts : en investigation, ouvert-pas commencé, ouvert-en correction, ouvert-corrigé, fermé), par module, par client, par sévérité (1=*Hot fix*, 2=*Urgent* à livrer en moins de 7 jours, 3=Bogue à prioriser normalement, 4=Demande de changement ou amélioration). Les bogues de sévérité 2 et 3 sont extraits chaque lundi matin en tant qu'intrant au *kick-off*.
2. Effort réel : ACME utilise RH.net pour la gestion des feuilles de temps de son personnel. Toutes les feuilles de temps doivent être envoyées au gestionnaire pour approbation chaque vendredi soir avant de quitter. Une fois approuvées, les feuilles de temps servent notamment à calculer la paie et cumuler l'effort par projet.
3. Estimations d'effort et de durée : Les estimations faites au début de chaque projet, soit par l'analyste, soit par les développeurs, sont détaillées dans un chiffrier Excel à l'aide d'un gabarit défini puis envoyé au gestionnaire. Ces estimations lui servent pour mesurer l'écart entre le nombre d'heures réelles et celles estimées.

Malgré qu'il y ait plusieurs mesures collectées et parfois analysées, il n'y a pas de plan de mesure formel où chaque mesure serait liée à un objectif d'affaires. Cependant, les estimations d'effort et de durée deviennent des objectifs d'affaires à atteindre.

4.3.6 Constat SUP-6 : Pas de contrôle de qualité sur les mesures

Il n'y a aucune spécification explicite sur les méthodes de collecte de données. Par exemple, un testeur peut attribuer la sévérité 2 à un simple bogue de terminologie et, au contraire, il peut attribuer la sévérité 3 ou 4 à un bogue bloquant. L'absence de contrôle de qualité ne permet pas de garantir que les données sont recueillies correctement. En conséquence, les analyses faites sur ces données parfois erronées peuvent être tout autant erronées, menant à des décisions discutables.

4.3.7 Constat SUP-7 : Résultats de mesures non communiqués

Les résultats du processus de mesure et d'analyse ne sont pas communiqués ou, dans les meilleurs cas, sont présentés d'une manière ambigüe et incompréhensible aux personnes intéressées. Par exemple, dans le portail de TFS, il y a un diagramme qui présente le nombre de bogues corrigés par sévérité et par personne. Les développeurs ont exprimé un malaise à l'effet qu'un bogue corrigé par eux, mais généré par quelqu'un d'autre, leur était systématiquement attribué puisqu'ils l'ont corrigé. Donc, la pertinence d'avoir ces données par personne demeure douteuse.

4.3.8 Constat SUP-8 : AQ des produits logiciels adéquate

Les contrôles mis en place donnent une assurance qualité adéquate, du fait que les nouvelles fonctionnalités se conforment aux spécifications, aux normes de développement et aux exigences de qualité et qu'un plan d'assurance qualité contribue à obtenir la qualité exhortée dans la définition des exigences et la politique de l'assurance qualité d'ACME.

ACME alloue des personnes et des ressources pour l'assurance qualité du produit. Après chaque nouveau développement ou changement, l'équipe de l'assurance qualité élabore et exécute un plan de test, en adéquation avec la définition des besoins du nouveau projet ou de la demande du changement et avec la politique et les normes de qualité d'ACME. dans le plan du test on indique l'objectif, la nature et l'étendue des test à faire.

Le responsable de l'assurance qualité utilise TFS pour documenter et communiquer les écarts détectés durant différentes étapes de test ainsi que pour faire le suivi des problèmes de non-conformité jusqu'à leur clôture.

4.3.9 Constat SUP-9 : Absence de toute activité d'assurance qualité du processus

Selon plusieurs intervenants, ACME ne mène presque aucune activités pour l'assurance qualité de son processus, à l'exception d'un audit fait par un auditeur externe en 2011 sur les activités de contrôle reliées au développement, au soutien et à l'hébergement de RH.net. Depuis ce temps, aucune activité d'AQ des processus n'a été observée. Ce rapport d'évaluation est la seule activité formelle depuis 2011 qui cherche à améliorer le processus.

4.4 Constat liés à l'ingénierie (ING)

Selon le CMMI pour le développement, les domaines de processus de la catégorie Ingénierie couvrent « les activités de développement et de maintenance des diverses disciplines de l'ingénierie. Les domaines de processus de cette catégorie intègrent également les processus liés aux différentes disciplines de l'ingénierie au sein d'un seul processus de développement de produit, autorisant ainsi une stratégie d'amélioration de processus orientée produit. Une telle stratégie cible les objectifs d'entreprise essentiels plutôt que les disciplines techniques spécifiques » (CMMI-DEV-Team, 2010, p.68).

Les observations liées à l'ingénierie sont tirées des domaines de processus RD (Développement des exigences), TS (Solution technique), PI (Intégration de produit), Ver (Vérification) et Val (Validation) :

- « Le but de RD est d'obtenir et expliciter, d'analyser et d'établir les exigences client, produit et composants de produit » (CMMI-DEV-Team, 2010, p.401).
- « Le but de TS est de sélectionner, de concevoir et d'implémenter des solutions aux exigences. Les solutions, conceptions et implémentations couvrent les produits, les composants de produits et les processus du cycle de vie liés aux produits en question, en tout ou en partie, selon ce qui convient » (CMMI-DEV-Team, 2010, p.455).
- « Le but de PI est d'assembler le produit à partir des composants de produit, de s'assurer que le produit assemblé se comporte correctement (autrement dit possède les fonctionnalités et les attributs de qualité nécessaires) et de le livrer » (CMMI-DEV-Team, 2010, p.323).
- « Le but de VAL est de démontrer qu'un produit ou un composant de produit satisfait à l'utilisation prévue lorsqu'il est placé dans l'environnement cible » (CMMI-DEV-Team, 2010, p.477).
- « Le but de VER est de s'assurer que les produits d'activité sélectionnés respectent les exigences spécifiées qui les concernent » (CMMI-DEV-Team, 2010, p.487).

4.4.1 Constat ING-1 : Environnements de test définis et utilisés

ACME a mis en place les environnements de test suivants :

- **Robot** : Ensemble de tests lancés dans les versions de production ou à la fin de la version de développement. L'application ne sera pas considérée comme GA (globalement acceptable), que si tous les tests passent avec succès.
- **Tests unitaires** : Une fois le codage terminé, les développeurs élaborent des tests unitaires (avec Xunit) pour s'assurer que les fonctionalités dévelopées répondent bien aux demandes initiales et qu'aucune régression n'a été introduite. Ces tests permettent très rapidement d'évaluer la qualité d'une version et de garantir qu'aucune fonctionnalité antérieure n'a été affectée par des modifications.
- **AQ** : Exécution de l'ensemble des cas de tests fonctionnels une fois le codage terminé.
- ***Build* de nuit** : Afin de s'assurer que toutes les modifications au code source sont compatibles entre elles, l'application est compilée automatiquement toutes les nuits. Si des bris sont découverts, ceux-ci sont corrigés le lendemain matin.

Ces environnements permettent aux membres d'équipe d'effectuer des tests exhaustifs afin de déceler et corriger le plus de bogues possible avant la livraison d'une version. Les membres d'équipe expérimentés connaissent tous ces environnements et s'en servent quotidiennement.

Cependant, les nouveaux employés n'ont pas une connaissance systématique de ces environnements, notamment lorsqu'il s'agit de reproduire l'environnement d'un client afin de répliquer un bogue. Dans ce cas, un membre d'équipe expérimenté doit passer du temps avec eux pour expliquer et montrer le fonctionnement d'une réplication d'environnement client.

4.4.2 Constat ING-2 : Revues par les pairs du code des employés récents

La vérification du code par les pairs : Lors de ces inspections/révisions de code, le réviseur procède généralement par analyse statique du code. Cette révision est faite sous la forme d'un atelier avec le développeur et un réviseur qui commente les erreurs potentielles, les améliorations à apporter pour rendre le code plus résilient ou améliorer les performances ainsi que la maintenabilité. Cette révision permet aussi un transfert de connaissances du code et des meilleures pratiques utilisées par chacun.

Cependant, seul le code des employés récents est révisé par un développeur expérimenté, plus tout ce qui touche le module de sécurité et pour lequel auncun bogue n'a été rapporté au cours des deux dernières années. Donc les bogues sont trouvés dans les autres modules pour lesquels ces bogues sont rarement critiques. Au cours de l'année 2018, 1533 bogues ont été identifiés dans 14 modules, dont environ 63% auraient été générés par des développeurs expérimentés.

4.4.3 Constat ING-3 : Architecture du logiciel définie et stable

Chez ACME, on touche rarement à l'architecture du logiciel. Toute évolution est conçue pour réutiliser le cadre de développement de l'organisation où l'architecture est prédéfinie. Chaque ajout ou modification du produit doit suivre les directives de ce cadre de développement. Aussi, quand les développeurs reçoivent la version finale du document des spécifications détaillées, ils commencent à implémenter le code source de la solution sans avoir à effectuer le développement des documents de conception en premier puisque la conception « générique » existe. Et à cause de l'utilisation de gabarits, chaque développeur est obligé d'implémenter toutes les méthodes définies dans le cadre de développement, et dans le bon ordre d'exécution.

Cependant, cette architecture (basée sur des *web forms*) est considée comme étant obsolète par plusieurs développeurs qui préfèreraient une achitecture de type MVC (*Model - View - Controller*) plus récente. Les deux types d'architecture s'utilisent avec ASP.net. Par ailleurs, certains développeurs ont quitté ACME parce qu'ils souhaitaient travailler avec des architectures plus modernes. Cette architecture est donc un frein à l'embauche de nouveaux développeurs.

4.4.4 Constat ING-4 : Exigences validéés et vérifiées par plusieurs personnes

ACME se préoccupe de ce que les exigences et les attentes documentées soient comprises et révisées par tous les membres des projets, chacun ayant son champ d'intervention.

À chaque fois que l'analyste d'affaires rédige un document d'exigences d'un nouveau module, ou suite à un changement dans la législation, plusieurs rencontres sont organisées avec le président pour s'assurer que le document reflète bien ses idées et ses orientations, avec les personnels en France pour s'assurer de la conformité avec la législation française, avec les développeurs expérimentés afin de déterminer la portée et la faisabilité des solutions proposées, avec les responsable de l'assurance qualité pour vérifier l'exactitude et la testabilité des spécifications d'exigences et sa cohérence avec la description du produit. Aussi, une rencontre a lieu avec un autre analyste pour la validation par un pair.

De plus, selon les intégrateurs, les clients sont très satisfaits des modules de RH.net et la majorité des clients renouvellent leur contrat SaaS année après année.

4.4.5 Constat ING-5 : Processus de développement itératif et incrémental

Chez ACME, le processus de développement est itératif et incrémental. Les fonctionnalités sont implémentées et déployées au fur et à mesure sous forme de *builds* successifs. Cela permet aux analystes de valider très tôt l'exactitude des fonctionnalités et de les démontrer au président qui agit en tant que client à l'interne. Cela permet aussi aux testeurs de se familiariser avec la nouvelle application et de préparer les cas de tests. La

livraison en production chez un client externe n'est cependant effectuée que losque le module complet est développé.

4.4.6 Constat ING-6 : Processus de *build* manuel et trop lent

Étant donné que le processus de *build* est manuel, ça prend plus de six heures pour compiler une version. Le processus de *build* est considéré très lent par les développeurs, puisque le *build* ne se fait qu'une fois par semaine. Donc, un bogue introduit dans la semaine 1 est détecté au cours de la semaine 2 et ne peut pas être confirmé comme étant corrigé par l'équipe QA avant la troisième semaine. Ces délais sont induits par la lenteur du processus de *build* car les boucles de *feedback* sont d'au moins une semaine.

4.5 Sommaire des constats

Suite à notre évaluation, on a constaté qu'ACME mène plusieurs activités recommandées par le CMMI pour les quatre catégories de domaines de processus, Gestion de projet, Gestion de processus, Support et Ingénierie. Parmi les forces, on cite : processus organisationnel défini et normalisé, gestion de risque adéquate bien que parfois réactive, normes de programmation définies et appliquées, outils mis en place pour la gestion des modifications, gestion de l'intégrité des versions de RH.net, AQ adéquate des produits logiciels, environnements de test définis et utilisés, architecture du logiciel définie et stable, exigences validées et vérifiées par plusieurs personnes, processus de développement itératif et incrémental et développement par prototypage.

Cependant, ACME devrait porter une attention particulière aux faiblesses suivantes :

Gestion de projet :

- Sous-estimation systématique de tous les projets : selon la plupart des intervenants, aucun projet n'a respecté l'échéancier initial, notamment parce que l'estimation de cet échéancier ne tient pas compte des corrections prioritaires de bogues pour certains clients qui émergent en cours de réalisation.
- Nombreux travaux en parallèle pour chaque développeur : à cause de la priorité changeante des projets due aux corrections de bogues prioritaires des clients.
- Personnel insuffisant pour effectuer la gestion de projet : une seule personne pour gérer deux types de projet.
- Culture de blâme contre-productive lors des rencontres de suivi de projet : plusieurs employés évitent cette rencontre importante devant l'injustice perçue de recevoir des blâmes à cause des délais non respectés alors qu'ils ne contrôlent absolument pas les paramètres d'une estimation souvent imposée, leur charge de travail et les priorités changeantes.

Gestion de processus :

— Processus non communiqué et mise en œuvre non surveillée : le processus n'est pas connu par les nouveaux arrivants.
— Pas de mécanisme d'amélioration continue : absence de rétrospective à la fin d'un cycle ou d'un projet.

Support :

— Documents relatifs à la gestion de configuration peu consultés : le nombre de visualisations de ces documents montre que personne ne les a consultés.
— Planification informelle des mesures et leur analyse, pas de contrôle de qualité sur les mesures et résultats de mesures non communiqués : absence du plan de mesure formel où chaque mesure serait liée à un objectif d'affaires.
— Absence de toute activité d'assurance qualité du processus : aucune activité pour l'assurance qualité de son processus depuis 2011.

Ingénierie :

— Revues par les pairs du code des employés récents : une restriction de la revue par les pairs du code sur les employés récents seulement, bien que des bogues identifiés soient attribuables à des employés expérimentés.
— Architecture du logiciel définie et stable mais montrant des signes d'obsolescence : une architecture du logiciel basée sur les *web forms* de *ASP.net*, cependant les développeurs préfèreraient travailler avec MVC, ce qui empêche d'attirer des nouveaux talents ou de les retenir.

CHAPITRE 5

ÉLABORATION D'UN MODÈLE INITIAL D'ESTIMATION DE PROJET

Dans ce chapitre, nous décrivons l'étude quantitative de la productivité du processus de développement d'ACME. Nous avons mesuré la productivité des sept premiers projets en comparant leur taille fonctionnelle mesurée avec la méthode COSMIC, soit la norme ISO 19761, avec l'effort de réalisation [1].

5.1 Application de la méthode COSMIC

La taille fonctionnelle a été mesurée en CFP avec la méthode COSMIC.

5.1.1 Phase 1 : Stratégie de mesure

La raison principale pour appliquer la mesure de la taille fonctionnelle est d'élaborer un premier modèle de productivité et d'estimation pour ACME.

Après avoir appliqué nos critères de sélection, nous avons retenu sept projets de développement réalisés par quatre développeurs différents.

5.1.2 Phases 2 et 3 : Arrimage et mesure

Après avoir appliqué la méthode COSMIC, nous obtenons les données de mesure dans le tableau 5.1.

Les mesures détaillées ont été enregistrées dans un chiffrier Excel dont le gabarit nous a été fourni lors du cours MGL7240 (Mesures et développement logiciel).

1. le 8ième projet n'a pas pu être mesuré à l'intérieur des délais de ce travail.

Tableau 5.1 – Mesures de taille fonctionnelle pour les projets sélectionnés

Projet	Nbre PF	Entrées	Sorties	Lectures	éCritures	Taille (CFP)
A	10	6	25	25	25	81
B	7	6	19	18	21	64
C	18	43	70	67	40	220
D	8	13	22	22	21	78
E	15	15	52	46	17	130
F	12	20	34	31	9	94
G	8	2	10	9	8	29
H	n/d	n/d	n/d	n/d	n/d	n/d
Total	78	105	232	218	141	696
Moyenne	11,1	15,0	33,1	31,1	20,1	99,4
Écart Type	4,1	13,8	20,1	19,5	10,8	61,3

5.1.3 Vérificaion des données de la taille fonctionnelle

Le mesureur a préalablement suivi une formation sur la méthode COSMIC. De plus, les mesures de la taille fonctionnelle ont été vérifiéés par la professeur Sylvie Trudel, mesureure cerifiée sur la méthode COSMIC, pour en assurer la qualité avant que nous les utlisions dans nos analyses.

5.2 Mesures de l'effort et de la durée

L'effort de chacun des projets en heures-personnes a été extrait du système de feuilles de temps. Nous croyons que cet effort est juste car le temps supplémentaire y est inscrit puisqu'il peut être repris par les employés. Cet effort comprend toutes les activités de développement (analyse, conception, programmation, test). Cet effort comprend la contingence des risques appliquée. L'effort n'inclut pas le déploiement ni la correction des bogues post-production.

La durée d'un projet a été définie comme étant la différence en mois entre la date de la dernière tâche et la date de la première tâche saisies dans les feuilles de temps. La plupart du temps, la phase de développement ne commence pas immédiatement après la phase d'analyse, donc on peut avoir une période où le projet est ouvert même si aucune tâche n'est en cours, soit une période d'inactivité. Nous avons mesuré également cette inactivité, comme le font les bases de données internationales de productivité telles que ISBSG (www.isbsg.org), et l'avons soustrait de la durée brute pour obtenir une durée nette.

Le tableau 5.2 donne les résultats de la taille, de l'effort et de la durée (en semaines) pour chacun des sept projets, avec les totaux et les moyennes.

Tableau 5.2 – Mesures de base pour les projets sélectionnés

Projet	**Taille (CFP)**	**Effort (heures)**	**Durée (semaines)**
A	81	265,00	8
B	64	181,50	5
C	220	526,25	14
D	78	164,25	6
E	130	363,25	9
F	94	312,75	8
G	29	137,75	3
H	n/d	n/d	n/d
Total	696	1950,75	53
Moyenne	99,4	278,68	7,6
Écart Type	61,3	136,74	3,5

5.3 Relation entre la taille fonctionnelle et l'effort

Pour établir notre modèle de productivité, nous devons mettre en relation la taille et l'effort. D'une manière simpliste, nous pouvons faire la simple division mathématique des totaux de l'effort par la taille, auquel cas nous obtenons 2,80 heures/CFP. Cependant, cette façon simpliste ne nous permet pas de calculer une marge d'erreur. Pour calculer cette marge d'erreur, nous analysons les données sous forme d'une droite de régression linéaire simple, telle que présentée à la figure 5.1.

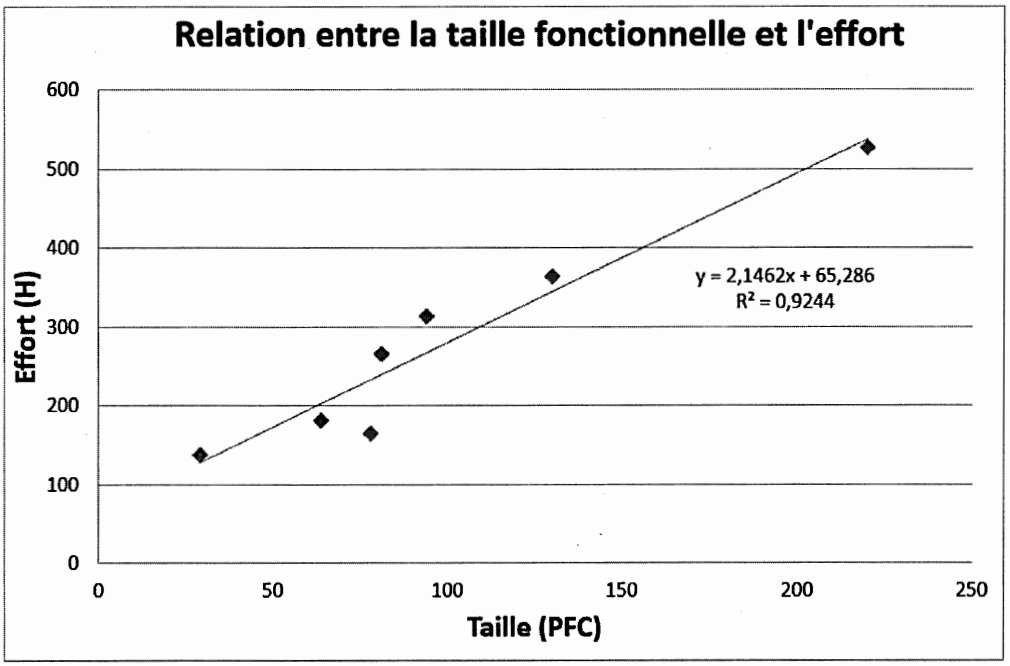

FIGURE 5.1 – Relation entre taille fonctionnelle et effort

La corrélation entre la taille fonctionnelle et l'effort est très forte, comme l'indique le coefficient de corrélation R2 de 0,924. Ce graphique donne également la fonction de régression linaire simple, y = 2,146x + 65,29, soit un effort variable de 2,146 heures/CFP et un effort fixe de 65,29 heures par projet.

Typiquement, la marge d'erreur sur l'effort se calcule comme étant plus ou moins l'erreur-type. Selon nos calculs dans Excel, l'erreur-type est de 41,18 heures. Dans ce cas, l'équation d'estimation de l'effort (c.-à-d. notre modèle d'estimation initial) devient :

Modèle initial d'estimation de l'effort
Effort estimé (en CFP) = 2,146x + 65,29 +/- 41,18

Donc, pour un nouveau projet de 100 CFP, l'effort estimé devrait être entre 239 et 321 heures, selon le niveau de risque perçu, à savoir si ces risques sont plus ou moins importants en lien avec les risques des projets ayant servi à construire notre modèle.

5.4 Relation entre la taille fonctionnelle et la durée

Pour établir notre modèle du taux de livraison, en CFP/semaine, nous avons analysé la relation entre la durée nette (semaine) et la taille (CFP). D'une manière simpliste, ACME réalise en moyenne 13,13 CFP/semaine. Pour mieux comprendre la marge d'erreur, nous avons également produit une analyse de régression linéaire simple, illustrée à la figure 5.2.

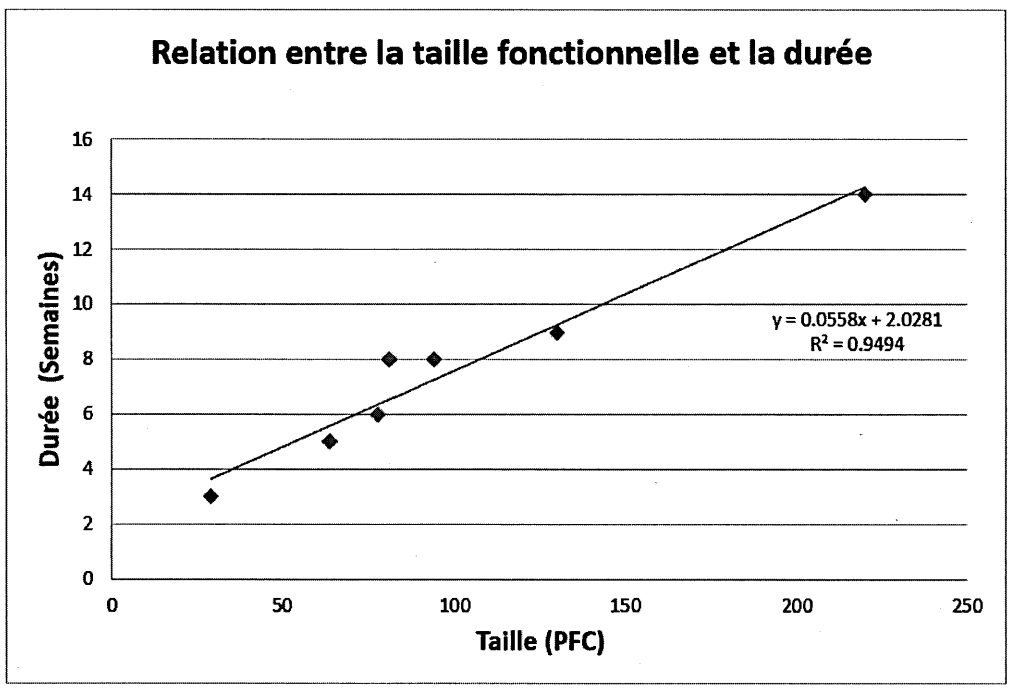

FIGURE 5.2 – Relation entre taille fonctionnelle et la durée

Tout comme la relation entre la taille fonctionnelle et l'effort, la corrélation entre la taille fonctionnelle et la durée est très forte, comme l'indique le coefficient de corrélation R2 de 0,949. Ce graphique donne également la fonction de régression linaire simple, y = 0,055x + 2,03, soit une durée variable de 0,055 semaines/CFP et une durée fixe de 2,03 semaines par projet.

Typiquement, la marge d'erreur sur la durée se calcule comme étant plus ou moins l'erreur-type. Selon nos calculs dans Excel, l'erreur-type est de 0,863 semaines. Dans ce cas, l'équation d'estimation de la durée (c.-à-d. notre modèle d'estimation initial) devient :

Modèle initial d'estimation de la durée
Durée estimée (en semaines) = 0,055x + 2,03 +/- 0,86

Donc, pour un nouveau projet de 100 CFP, la durée estimé devrait être entre 6,7 et 8,4 semaines.

5.5 Limitation sur la validité des données

Les modèles initiaux d'estimation d'effort et de durée sont sujet à être réévalués lorsque des données supplémentaires seront disponibles. Pour le moment, ces modèles d'estimation sont limités par le volume relativement faible des données nous ayant permis de les élaborer. Par ailleurs, il est toujours possible que les erreurs ou omissions se soient glissées dans les feuilles de temps et que l'effort ait quelques imprécisions. Cependant, il faut savoir que les mêmes données de feuilles de temps sont utilisées par ACME pour faire la facturation. En ce sens, elles sont réputées plutôt fiables. Il en est de même pour la durée.

Quant aux données de la taille fonctionnelle, des variations pourraient exister si les exigences ayant servi à la mesure s'avéraient ambiguës ou incomplètes. Cependant, étant donné que cette mesure a été faite par l'un des développeurs, c'est la taille réelle de ce qui a été développé qui a été mesurée, réduisant ainsi le risque d'erreurs.

CHAPITRE 6

RECOMMANDATIONS D'AMÉLIORATION DES PROCESSUS

Dans ce chapitre, nous proposons des recommandations fondées sur les constats validés dans le chapitre 4 et sur les bonnes pratiques recommandées dans le domaine de gestion du logiciel, tel qu'on retrouve dans le CMMI. Chaque recommandation est décrite par les caractérstiques suivantes :

1. Le **lien** avec un ou plusieurs constats ;
2. La **priorité de mise en œuvre** selon notre interprétation des retombées potentielles et positives sur le processus de développement ou sur la qualité ;
3. Un **horizon de mise en œuvre**, c'est-à-dire le moment dans le temps où la recommandation pourrait être déployée entièrement, avec la convention suivante :
   — Court terme : moins de trois mois ;
   — Moyen terme : trois à douze mois ;
   — Long terme : plus de douze mois.
4. Le **niveau d'effort** requis avec la convention suivante :
   — Faible effort : facilement intégrable dans les activités quotidiennes, sans dérangement significatif pour les projets en cours ;
   — Effort moyen : moins de 80 heures, soit l'équivalent de deux semaines d'effort ou moins pour une personne ;
   — Effort élevé : plus de 80 heures.
5. Le **degré d'autonomie de mise en œuvre** par les employés actuels de ACME avec la convention suivante :
   — Élevé : ACME a les compétences requises parmi les membres d'équipe ;
   — Moyen : une formation d'appoint doit être suivie par un ou plusieurs membres d'équipe (lectures dirigées, formation en classe ou en ligne, coaching à temps partiel, etc.) ;
   — Faible : ACME n'a pas les compétences requises et doit faire appel à des experts externes.
6. La **priorité suggérée**, soit l'urgence perçue de mettre en œuvre cette recommandation, avec la convention suivante :
   — Élevée : ACME devrait entreprendre cette recommandation rapidement (moins de trois mois) car elle apportera beaucoup de valeur avec peu d'effort ;
   — Moyenne : Recommandation ayant beaucoup de valeur mais nécessitant un peu plus d'effort (entre trois et douze mois) ;

— Faible : Recommandation qui apporte de la valeur mais dont les risques ou les efforts de mise en œuvre sont plus élevés (dans plus de douze mois).

Ces recommandations sont regroupées selon les thèmes suivants et décrites de façon à faciliter l'élaboration d'un plan d'action :

— Gestion de projet et estimation (GPE) ;
— Gestion des individus (GDI) ;
— Organisation du travail (ODT) ;
— Processus (PRC) ;
— Produit logiciel (PRL).

6.1 Recommandations liées à la gestion de projet et estimation (R-GPE)

Nous proposons les recommandations suivantes en lien avec la gestion de projet et l'estimation :

— R-GPE-01 : Affecter un coordonnateur de projets et d'activités ;
— R-GPE-02 : Développer et maintenir des modèles d'estimation réalistes ;
— R-GPE-03 : Mettre en place un plan de mesure basé sur des objectifs clairs ;
— R-GPE-04 : Évaluer les risques de manière plus rigoureuse.

R-GPE-01 : Affecter un coordonnateur de projets et d'activités				
Constats (2)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
GPJ-3 ; PRS-2 ;	Moyen	Faible	Élevée	Court terme

— But : Donner plus de temps pour la gestion de projet de développement pour minimiser les pertes de temps entre les questions des membres de l'équipe et les réponses de gestionnaire.
— Description :
La personne recrutée devra assumer la résponsabilté à plein temps de coordination des activités et des projets. Elle sera responsable avec l'équipe de développement de livrer un produit de qualité tout en respectant les délais. Elle devra jouer un rôle de facilitateur. Le coordinateur des activités et des projets devra posséder les compétences pour accomplir les tâches suivantes :
— Réévaluer au besoin les modèles d'estimation de durée et d'effort des projets (voir R-GPE-02) ;
— Aider les membres d'équipe à planifier et mener toutes les activités nécessaires pour atteindre les objectifs du projet ;

— Mener avec *leadership* des rencontres de coordination avec l'équipe, en portant une attention particulière pour qu'il n'y ait jamais de blâme mais plutôt des solutions lorsque les performances s'éloignent trop des estimations ;
— Être disponible pour répondre aux questions des membres d'équipe ;
— Réorganiser les affectations des membres d'équipe pour réduire leur nombre de tâches parallèles ;
— Établir et maintenir des communications efficaces et continues avec l'équipe de développement.

R-GPE-02 : Développer et maintenir des modèles d'estimation réalistes				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
GPJ-1 ;	Faible	Élevée	Élevée	Court terme

— But : Établir des estimations réalistes afin de mieux gérer les attentes des parties prenantes.
— Description :
Pour éviter d'établir des estimations toujours trop petites, il est recommandé d'établir des modèles d'estimation simples mais fiables qui tiennent compte des données historiques de productivité du processus, du fait que chacun travaille en ce moment sur un trop grand nombre de tâches en parallèle, que plusieurs font du surtemps (bien que repris par la suite) afin de livrer plus rapidement et que de nouveaux bogues urgents surgissent régulièrement. Ces modèles d'estimation pourraient être basés sur la taille fonctionnelle des tâches comparée avec l'effort (Trudel, 2018). Cette taille fonctionnelle ayant déjà été mesurée sur sept projets avec la méthode COSMIC (COSMIC-Team-V4.0, 2015), il sera plus facile d'établir un premier modèle de départ. Par la suite ce modèle devra évoluer avec les améliorations régulières qui seront apportées au processus (lien avec la recommandation R-PRC-04). Pour que cette pratique soit bien comprise et utilisée, il est recommandé que les analystes et les développeurs intéressés suivent une formation sur la méthode COSMIC, dont l'un des avantage de son application est la rédaction plus efficiente (plus rapidement) et efficace (moins de défauts et d'ambiguïtés) des descriptions des fonctionnalités.

R-GPE-03 : Mettre en place un plan de mesure basé sur des objectifs clairs				
Constats (3)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
SUP-5 ;SUP-6 ; SUP-7 ;	Faible	Élevée	Élevée	Court terme

— But : Se doter d'indicateurs objectifs facilitant la prise de meilleures décisions.
— Description :
Un plan de mesure efficace pour une PME devrait être simple, tel que tenir sur trois petits onglets dans un chiffrier Excel :

1. **Objectifs visés** : Pour le moment, nous avons trois grands objectifs généraux. Afin de prendre un peu d'avance, nous proposons leur rédaction SMART (Spécifique, Mesurable, Atteignable, Réaliste, et limité dans le Temps) comme suit :
   a. Réduire le nombre de bogues trouvés d'au moins 30% au cours des douze prochains mois, dans le but d'augmenter la qualité et la productivité.
   b. Réduire l'écart à plus ou moins 20% entre la durée estimée et la durée réelle pour réaliser des tâches, dans le but de se donner le temps pour produire un logiciel de qualité du premier coup et être ainsi plus efficace.
   c. Limiter à deux le nombre de tâches en parallèle de chaque développeur, et ce dès le mois prochain, dans le but d'augmenter leur efficience.
2. **Mesure de base** (ex. Taille fonctionnelle, Effort, Durée, Nombre de bogues). Pour chaque mesure de base, on doit savoir :
   — Qui mesure ;
   — Quand mesure-t-on ;
   — Comment mesure-t-on ;
   — Avec quel outil ;
   — Sur quoi porte la mesure (ex. Projet, Tâche ou autre) ;
   — Où stocke-t-on les données ;
   — Comment assure-t-on la qualité de la donnée.
3. **Indicateurs et mesures dérivées** (ex. Modèle d'estimation, Nombre de bogues trouvés par semaine et cumulatif). Pour chaque mesure dérivée ou indicateur, on doit connaitre :
   — À quel(s) objectif(s) cet indicateur est-il lié ;
   — Qui produit cet indicateur ;
   — Quand le produit-on (fréquence et/ou moment dans le temps : ex. chaque semaine, à chaque livraison en production, à chaque fin de tâche, etc.) ;
   — Comment calcule-t-on cet indicateur (formule et/ou graphique) ;
   — Comment doit-on interpréter ou analyser les valeurs au regard des objectifs à atteindre ;
   — Quel type de décision devrait être basée sur ces résultats ;

— À qui est-il destiné ;
— Quel est le moyen de communication de cet indicateur ;
— Où stocke-t-on les résultats d'analyse.

R-GPE-04 : Évaluer les risques de manière plus rigoureuse				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
GPJ-6 ;	Faible	Élevée	Faible	Court terme

— But : Éviter les *mauvaises* surprises affectant négativement les durées des tâches.
— Description :
Avec l'aide de TFS, créer et maintenir un registre des risques, soit les problèmes potentiels qui ne sont pas encore survenus et qui peuvent affecter les délais, la productivité ou la qualité. Pour chaque risque, on décrit brièvement le problème, le niveau d'impact, la ou les actions concrètes pour éviter le risque et la ou les actions concrètes à enclencher si le risque se manifeste.
Aussi, avec l'aide de TFS, créer et maintenir un registre des obstacles pour chaque problème survenu, avec la ou les actions qui ont été appliquées pour le résoudre. Cette liste d'obstacles devrait être utilisée comme source d'information pour les rétrospectives (lien avec la recommandation R-PRC-02 - Faire des rétrospectives à chaque mois).

6.2 Recommandations liées à la gestion des individus (R-GDI)

Nous proposons les recommandations suivantes en lien avec la gestion des individus :

— R-GDI-01 : Valoriser le travail d'équipe ;
— R-GDI-02 : Cesser de blâmer les employés ;
— R-GDI-03 : Former les développeurs sur les nouvelles technologies.

R-GDI-01 : Valoriser le travail d'équipe				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
GPJ-7 ;	Faible	Élevée	Élevée	Court terme

— But : Avoir une équipe fonctionnelle avec objectifs communs afin d'en augmenter la productivité.

— Préambule :
Pour atteindre ce but, il faut éviter les cinq dysfonctions d'une équipe à savoir :
— Absence de confiance ;
— Peur du conflit ;
— Absence de l'engagement ;
— Evitement de la responsabilité ;
— Inattention portée aux résultats.
Au contraire de ces cinq dysfonctions, chacun des membres d'une équipe doit acquérir et appliquer les cinq comportements souhaités (Confiance, Courage, Engagement, Responsabilité, Attention aux résultats) pour que l'équipe fonctionne bien. Ces cinq comportements que l'on peut assimiler à des niveaux de maturité d'équipe se construisent les uns sur les autres. Le niveau supérieur ne pouvant être élaboré que si le niveau inférieur est là et consolidé (XXl-Strategie, 2019).

1. **Confiance** : Une chose est sûre, on ne peut pas construire une équipe sans confiance. Il faut que les membres de l'équipe se fassent confiance entre eux, qu'ils admettent leurs erreurs et leurs lacunes et qu'ils demandent de l'aide au besoin (XXl-Strategie, 2019) .

2. **Courage** : Ce niveau est réussi si les membres de l'équipe sont capables d'exprimer et défendre leurs points de vue mais aussi d'écouter ceux des autres pour pouvoir argumenter. Il faut laisser place aux désaccords, à la critique constructive (XXl-Strategie, 2019).

3. **Engagment** : Ce troisième niveau, construit sur le niveau du courage (acceptation des conflits productifs), est celui de l'engagement. Si les objectifs ne sont pas clairs ou difficilement réalistes dans les circonstances, il est difficile de se rallier à un objectif commun (XXl-Strategie, 2019) .

4. **Responsabiltié** : Le quatrième niveau accessible uniquement, tout comme les autres niveaux, en cas de réussite du niveau précédent, est l'acceptation d'être redevable des actions acceptées. Un échec à ce niveau se voit par la fuite de certains membres de l'équipe face à leurs responsabilités ou bien la complaisance dans des résultats médiocres par rapport à l'objectif fixé (XXl-Strategie, 2019).

5. **Attention aux résultats** : On se soucie des résultats collectifs, sans se concentrer sur les comportements individuels. Tous ensemble, on célèbre les succès et on souffre des échecs, tout en apprenant (XXl-Strategie, 2019).

— Description : Pour valoriser le travail d'équipe on doit :
— Commencer à travailler avec des groupes de deux personnes et plus suivant l'envergure de chaque projet ;
— Étendre la pratique des revues par les pairs (voir R-PRL-01). La revue par les pairs encourage la communication et brise les silos, ce qui augmente le

transfert de connaissances ;
— Mettre en place des outils de partage, de cette façon tous peuvent partager les problèmes et des documents intéressants pour que les autres membres de l'équipe soient au courant et augmentent leur autonomie pour agir ;
— Évaluer les membres de l'équipe en se basant sur les résultats collectifs, et non pas sur la performance individuelle, pour favoriser l'entraide et le focus sur des objectifs communs.

R-GDI-02 : Cesser de blâmer les employés

Constats (1)	Effort	Autonomie	Priorité	Horizon
GPJ-4 ;	Faible	Élevée	Élevée	Court terme

— But : Motiver les employés davantage pour augmenter la productivité, l'engagement, la responsabilité et la fierté du travail bien fait.
— Description :
La rénumération n'est pas une source de motivation pour les employés ; au pire, c'est une source de démotivation si des employés se sentent sous-payés, ce qui n'est pas le cas chez ACME où les salaires sont perçus comme compétitifs. Au lieu de blâmer les individus en les mettant dans une situation de déni, d'accusation, de justification, de culpabilité ou d'obligation, il faut les responsabiliser avec les trois clés de responsabilisation d'un individu : «
— l'intention d'atteindre le conflit constructif basé sur les faits ;
— la prise de conscience de soi ;
— Se confronter » (Laramee, 2016).
Au-delà de ça, il faut prendre le temps de remercier sincèrement les membres de l'équipe, des félicitations publiques leur redonneront confiance. Les employés apprécient énormément la valorisation de leur implication.

R-GDI-03 - Former les développeurs sur les nouvelles technologies				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
ING-3 ;	Faible	Moyenne	Élevée	moyen terme

— But : Motiver les employés davantage pour augmenter la productivité et la qualité d'un produit davantage pérenne.
— Description :
La formation est un levier de motivation très efficace car cela démontre qu'ACME est prête à investir dans le développement des ses employés. Ainsi, il vont se sentir valorisés, surtout dans une période où il y a une pénurie de développeurs.

6.3 Recommandations liées à l'organisation du travail (ODT)

Nous proposons les recommandations suivantes en lien avec l'organisation du travail :

— R-ODT-01 : Limiter à deux tâches en parallèle par développeur, en tout temps ;
— R-ODT-02 : Réorganiser le travail pour améliorer la correction des bogues.

R-ODT-01 : Limiter à deux tâches en parallèle par développeur, en tout temps				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
GPJ-2 ;	Faible	Élevée	Élevée	Court terme

— But : Augmenter la productivité et la qualité.
— Description :
Les développeurs affectés à des projets pourront se concentrer sur une seule tâche. S'ils sont bloqués avec un problème sur cette tâche qu'il n'est pas possible de résoudre dans l'immédiat, alors ils peuvent travailler sur la deuxième tâche. Mais pas davantage si on veut optimiser la productivité et la qualité.

R-ODT-02 : Réorganiser le travail pour améliorer la correction des bogues				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
GPJ-2 ;	Faible	Élevée	Élevée	Court terme

— But : Rendre le processus plus efficace en augmentant la capacité des développeurs à se concentrer sur un type de tâche pendant une période donnée.
— Description :
Il est recommandé ici d'explorer différentes façons d'affecter les tâches de telle sorte que les développeurs affectés à des projets puissent s'y concentrer tandis que d'autres se consacrent à la correction de bogues. Le personnel doit absolument être en rotation périodique entre les tâches de développement et celles de correction des bogues (ex. 2 pers x 3 jrs/sem ; 1 pers. x 1 sem. ; 6 pers. x 2 jrs/-sem ; etc.) afin d'éviter la démotivation si plusieurs d'entre eux détestent passer trop de temps à corriger des bogues. Par ailleurs, quand un développeur corrige des bogues et qu'il est en mesure de comprendre l'origine de l'erreur, il est alors mieux équipé pour éviter ce type d'erreur dans le futur et pour contribuer à améliorer le processus, d'où l'importance que chacun puisse être en rotation sur la correction des bogues.

6.4 Recommandations liées aux processus (R-PRC)

Dans le cas des processus, leur mise en œuvre et leur amélioration, nous croyons qu'il est important de s'assurer que ces processus soient connus et appliqués par le personnel. Nous devons donc voir à consolider et à améliorer les acquis des processus (AQ des processus) en mettant en place les trois recommandations suivantes :

— R-PRC-01 : Communiquer davantage les processus et les environnements de test ;
— R-PRC-02 : Faire des rétrospectives à chaque mois ;
— R-PRC-03 : Faire un audit régulier des processus.

R-PRC-01 : Communiquer davantage les processus et les environnements de test				
Constats (4)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
PRS-1 ; PRS-2 ; ING-1 ; SUP-4 ;	Faible	Élevée	Élevée	Court terme

— But : S'assurer que tous appliquent adéquatement les processus et les environnements de test afin d'optimiser la productivité et la qualité.
— Description :
Ici, il est nécesssaire de commencer par une mise à niveau des connaissances de tous sur les processus et les environnements de test. Pour ce faire, une séance d'une heure devrait être suffisante, pendant laquelle un développeur senior passe en revue les processus et des descriptions des environnements de test tels que décrits dans notre *wiki interne* et dans *SharePoint*. Ensuite, ces informations devront systématiquement être communiquées à chaque nouveau membre d'équipe. Par ailleurs, lorsque nous aurons tous une compréhension commune des processus et des environnements de test, il sera plus facile d'en proposer des améliorations lors de séances de rétrospective (lien avec la recommandation R-PRC-02 - Faire des rétrospectives à chaque mois).

R-PRC-02 : Faire des rétrospectives à chaque mois				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
PRS-3 ;	Faible	Élevée	Élevée	Court terme

— But : Améliorer le processus régulièrement afin d'en augmenter la productivité ou d'optimiser la qualité du logiciel.
— Description :
La question principale à se poser est « Compte tenu des problèmes qui ont émergé ce mois-ci, qu'aurait-on pu faire autrement afin que ces problèmes n'aient pas existé ? ». Au-delà de cette question, l'équipe pourrait déterminer :
  — Ce qu'on veut faire davantage ;
  — Ce qu'on veut cesser de faire ;
  — Ce qu'on veut commencer à faire ;
  — Ce qu'on doit faire différemment.

  En réponse à ces quatre points, il devient beaucoup plus facile de faire des actions concrètes d'amélioration des processus pour lesquelles les membres d'équipe et

les gestionnaires seraient d'accord. Les rétrospectives bien menées permettent un appropriation des processus par les membres d'équipe qui sont alors plus enclins à les respecter. Une bonne rétrospective dure deux heures ou moins, et il ne faut pas chercher à résoudre d'un coup tous les problèmes. C'est plus efficace de ne résoudre que les deux ou trois problèmes les plus importants puis de reporter les autres problèmes à la prochaine rétrospective, d'où l'importance d'en faire régulièrement.

R-PRC-03 : Faire un audit régulier des processus				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
SUP-9 ;	Moyen	Moyenne	Moyenne	moyen terme

— But : S'assurer que les processus définis sont effectivement suivis et les améliorer régulièrement.
— Préambule : Si les acquis des processus sont ce qu'ils sont aujourd'hui, c'est qu'ils sont le reflet des divers apprentissages et expériences permettant de minimiser certains risques, problèmes ou bogues. Donc, ACME a un intérêt à ce que chaque membre d'équipe comprenne et applique les processus en place, en plus de les améliorer avec les nouvelles expériences vécues. Des processus non suivis n'ont aucune valeur pour ACME, même que cela nuit à ACME en augmentant les risques.
— Description : Passer en revue les produits et activités du logiciel, donc faire un audit interne, pour vérifier qu'ils sont conformes aux procédures et normes applicables. Quand un élément des processus n'a pas été suivi, l'audit devrait faire ressortir essentiellement deux choses :

1. Si les raisons pour ce manquement sont valables, une flexibilibité devrait être apportée au processus.
2. Si les raisons pour ce manquement ne sont pas valables, une formation d'appoint (d'une durée de moins d'une heure, typiquement) devrait être dispensée aux responsables de ce manquement.

Puis, il faut fournir les résultats de cet audit aux responsables d'ACME et aux membres d'équipe. Cet audit interne pourrait être fait en profondeur une fois par an ou plus souvent. Pour minimiser l'effort de ces audits, ACME devrait aussi améliorer ses processus d'une manière plus continue avec ces mécanismes :

— À la fin de chaque revue par les pairs (voir R-PRL-01 et R-PRL-02) : on peut vérifier l'origine d'un problème identifié et proposer une amélioration à l'un ou l'autre des éléments des processus (procédures, normes, gabarits, nomenclature, etc.).
— Pendant les rétrospectives à chaque mois (voir R-PRC-02) : c'est une bonne occasion pour se questionner sur les processus et leur utilisation.

— Ajouter un projet « Amélioration des processus » dans TFS et l'assigner à un responsable de l'équipe pour effectuer cette tâche, en plus de budgéter un peu de temps chaque mois pour que ces améliorations puissent se concrétiser. Les éléments de ce projet (tâches) peuvent alors être priorisés et suivis comme avec n'importe quel projet.

6.5 Recommandations liées au produit logiciel (PRL)

RH.NET est une application appréciée par ses utilisateurs et les clients d'ACME. L'application fonctionne bien, mais le nombre de bogues identifiés est en croissance depuis plusieurs années. Pourtant, la qualité du produit est une chose avec laquelle on ne veut pas faire de compromis. Il n'est donc pas viable pour ACME de poursuivre dans la même veine quand l'objectif visé est de réduire le nombre de bogues. C'est pour cette raison qu'il est nécessaire de consolider les acquis actuels du produit et de continuer à satisfaire les clients, notamment en ayant de meilleures activités menant à la qualité du logiciel. Pour y parvenir, nous faisons les recommandations suivantes en lien avec le produit Logiciel :

— R-PRL-01 : Étendre la pratique de revues par les pairs ;
— R-PRL-02 : Mieux comprendre pourquoi certains bogues ont émergé ;
— R-PRL-03 : Migrer l'application vers une architecture plus contemporaine (MVC) ;
— R-PRL-04 : Automatiser le *build* et les tests.

R-PRL-01 : Étendre la pratique de revues par les pairs

Constats (1)	Effort	Autonomie	Priorité	Horizon
ING-2 ;	Faible	Élevée	Élevée	Court terme

— But : Identifier et corriger davantage de bogues à la source, ce qui réduit grandement le coût de correction et augmente la productivité et la qualité.
— Préambule : La revue de code (de l'anglais « code review ») est un examen systématique du code source d'un logiciel. (Wikipedia, 2019). L'activité de révision de code par les pairs a montré son efficacité pour améliorer la qualité du code, la maintenabilité et la sécurité du logiciel, pour trouver des bogues ou des vulnérabilités potentielles ou pour corriger des erreurs de conception. Une étude a démontré que la revue de code peut réduire les coûts de projet jusqu'a 44% (Sprunk, 2019). Les objectifs escomptés de la revue de code sont :
  — Améliorer la qualité du code
  — Favoriser la collaboration, le travail en équipe

  - S'assurer que les normes de programmation sont appliquées
  - Détecter de façon précoce des défauts
  - Former les développeurs par transfert des connaissances
- Description : Actuellement, la revue par les pairs est restreinte au code des nouveaux développeurs alors que nos données récentes révèlent que 63% des bogues ont été générés par des employés expérimentés, travaillant dans un contexte sous pression et sans aide de leurs coéquipiers (travail individuel). Donc, il suffit d'étendre la revue par les pairs au code de tous, avant que le code de la tâche ne soit envoyé en test (au groupe QA).

R-PRL-02 : Mieux comprendre pourquoi certains bogues ont émergé				
Constats (2)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
SUP-8 ; ING-4 ;	Faible	Élevée	Élevée	Court terme

- But : Éviter d'introduire les mêmes type de bogues plusieurs fois.
- Description :
  La première chose à faire est de distinguer entre les bogues trouvés en production et les bogues trouvés en développement. Après ça, il faut comprendre dans quelle phase du processus de développement les bogues de production ont été introduits : analyse, conception ou codage. Un simple champ dans TFS serait suffisant et permettrait de mettre l'accent sur cette phase pour éviter d'autres bogues de même nature. Il faut penser à vérifier si des améliorations peuvent être apportées aux plans de test afin que ces types de bogues puissent être détectés avant la mise en production.
  Même pour les autres bogues (ceux trouvés avant la mise en production), nous devrions nous demander ce qu'on doit faire pour que ces bogues ne puissent pas se reproduire. Donc, les bogues corrigés au cours d'une semaine, avec l'information sur la raison pourquoi un tel bogue a existé, pourraient être brièvement présentés au *kick-off* du lundi de façon à ce que tous soient au courant et puissent activement participer à éviter ces types de bogues dans le futur. C'est une autre manière d'améliorer le processus en continu. L'identification de l'origine de chaque bogue ne prend que quelques secondes au développeur qui est responsable de sa correction et un champ dans TFS peut être rendu disponible pour consigner cette information. Par la suite, des améliorations découlant de ces informations devraient être apportées aux acquis du processus pour que cela serve à tous mais surtout aux nouveaux développeurs.

R-PRL-03 : Migrer l'application vers une architecture plus contemporaine (MVC)				
Constats (2)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
SUP-8 ; ING-4 ;	Élevé	faible	Moyenne	moyen terme

— But : Motiver les employées davatange et augmenter la robustesse et la maintenabilité du logiciel.
— Préambule :
Les développeurs favorisent ASP.NET et MVC car ils simplifient les parties complexes de *ASP.net Web Forms* sans compromettre la puissance et la flexibilité de la plate-forme ASP.NET. Le modèle MVC sépare l'application en trois parties : Modèle, Vue et Contrôleur. « Les principaux avantages des modèles MVC sont une séparation claire des problèmes, une installation de tests unitaires et un contrôle accru des URL et du HTML. Le modèle MVC n'utilise pas *Viewstate*, *Postbacks*, les contrôles serveur et les formulaires basés sur serveur qui permettent un contrôle total sur l'application et le code HTML rendu par les vues. Le modèle MVC utilise des URL basées sur *Representational State Transfer (REST)*, au lieu des extensions de noms de fichiers utilisées par le modèle *Web Form* afin que nous puissions créer des URL d'optimisation de moteur de recherche (SEO) publiées par l'application. » (Traduction libre Weblogs.asp.net, 2019).
— Description :
Étant donné que le logiciel RH.NET contient plusieurs modules, il est préférable de choisir un module de taille moyenne et d'une complexité moyenne pour avoir une idée sur le type de problème, quand on doit le résoudre, la quantité de travail et l'effort requis pour faire la migration de toute l'application comme projet pilote en parallèle avec la version de développement. Pour ce faire, il y a une *API Web ASP.NET* qui permet de créer ou de remodeler des applications *ASP.NET Web Forms* vers le modèle MVC, en déplaçant le code du fichier *codebehind* vers un contrôleur d'API Web. L'article dans ce lien (https ://msdn.microsoft.com/en-us/magazine/jj991978.aspx) donne plus de détails sur cet API et le processus de migration d'une manière générale.

R-PRL-04 : Automatiser le build et les tests				
Constats (1)	**Effort**	**Autonomie**	**Priorité**	**Horizon**
ING6	Faible	Élevée	Élevée	Court terme

— But : Diminuer le temps de *build* de 6 heures à 10 minutes ou moins, tout en exécutant des tests automatisés plus fréquemment.

— Préambule :
L'automatisation des *builds* est devenue une nécessité, pas un luxe, surtout pour assurer l'intégration continue et l'efficience du processus. En plus d'augmenter l'efficience du processus, cette automatisation devrait permettre d'identifier des bogues pendant le développement, soit avant même de se rendre à l'étape de QA, ce qui réduit significativement l'effort requis pour les gérer et les corriger. Elle contribue à l'amélioration de la qualité du produit en automatisant les tâches qui permettent de packager le logiciel et en supprimant les tâches manuelles.

— Description :
En plus d'automatiser le *build*, il faut aussi :
  — Faire davantage de tests automatisés (unitaires, d'intégration et fonctionnels) par les développeurs (installer *SonarQube*, par exemple pour avoir une idée sur le pourcentage de couverture de test).
  — Exécuter les cas de test préparés dans le plan de test par les développeurs avant les *checkins* du code.

  Actuellement, il y a des outils sur le marché ayant montré leur efficacité pour l'automatisation des *builds*. On peut citer *Jenkins* et *Hudson* qui ont plusieurs déclencheurs permettant de lancer le *build* et exécuter les tests automatisés. Exemple de déclencheur pour Jenkins : 1) démarrer une tâche de *build* une fois qu'une autre s'est terminée ; 2) lancer des *builds* à des intervalles périodiques (ex. à chaque heure). Pour plus de détails sur l'installation et la configuration nous recommandons de consulter les références suivantes :
  — https ://www.c-sharpcorner.com/article/continuous-deployment-with-jenkins-and-net/
  — https ://www.codeproject.com/Tips/899766/How-To-Automate-Build-Process-of-ASP-NET-Websites

6.6 Sommaire des recommandations

Tableau 6.1 – Sommaire des recommandations

Recommandation	Priorité	Horizon de mise en oeuvre	Niveau d'effort
R-GPE-01 : Affecter un coordonnateur de projets et d'activités	Élevée	Court terme	Moyen
R-GPE-02 : Développer et maintenir des modèles d'estimation des délais afin qu'ils soient réalistes	Élevée	Court terme	Moyen
R-GPE-03 : Mettre en place un plan de mesure basé sur des objectifs clairs	Moyenne	Moyen terme	Moyen
R-GPE-04 : Évaluer les risques de manière plus rigoureuse	Moyenne	Moyen terme	Faible
R-GDI-01 : Valoriser le travail de l'équipe	Élevée	Court terme	Faible
R-GDI-02 : Motiver les employés davantage	Élevée	Court terme	Faible
R-GDI-03 : Former les développeurs sur les nouvelles technologies	Élevée	Court terme	Faible
R-ODT-01 : Limiter à 2 tâches en parallèle, par développeur, en tout temps	Élevée	Court terme	Moyen
R-ODT-02 : Réorganiser le travail pour améliorer la correction des bogues	Élevée	Court terme	Faible
R-PRC-01 : Communiquer davantage les processus et les environnements de test	Élevée	Court terme	Moyen
R-PRC-02 : Faire des rétrospectives à chaque mois	Élevée	Court terme	Faible
R-PRC-03 : Faire un audit régulier des processus	Moyenne	Moyen terme	Moyen
R-PRL-01 : Étendre la pratique de revues par les pairs	Élevée	Court terme	Faible
R-PRL-02 : Mieux comprendre pourquoi certains bogues ont émergé	Élevée	Court terme	Faible
R-PRL-03 : Migrer l'application vers une architecture plus contemporaine (MVC)	Élevée	Court terme	Faible
R-PRL-04 : Automatiser le build	Élevée	Court terme	Moyen

CONCLUSION

Notre projet a consisté à évaluer le processus de la compagnie ACME et son logiciel RH.net. Pour y parvenir, nous avons fait des entrevues avec le personnel sélectionné (deux analystes, trois développeurs et deux testeurs) et le gestionnaire. Nous avons aussi révisé et analysé la documentation et les données liées à l'application et aux projets d'évolution. Nous avons fait ressortir 29 constats organisés par domaine de processus parmi les domaines du CMMI pris en considération dans ce projet. Ces constats sont regroupés par catégorie des domaines de processus :

— Gestion de projet (7 constats) ;
— Gestion de processus (7 constats) ;
— Processus de support (9 constats) ;
— Ingénierie (6 constats).

En tenant compte de ces constats, dont la plupart ont été révisés par les participants, nous avons élaboré 16 recommandations organisées par thème :

— Gestion de projet et estimation (4 recommandations) ;
— Gestion des individus (3 recommandations) ;
— Organisation du travail (2 recommandations) ;
— Processus (3 recommandations) ;
— Produit logiciel (4 recommandations).

Chaque recommandation a un lien avec un ou plusieurs constats, une priorité, un horizon de mise en œuvre, un niveau d'effort requis et un degré d'autonomie du personnel d'ACME pour la réaliser.

Nos objectifs étaient :

— Objectif 1 - Catégoriser les bogues par phase de développement : cet objectif est en cours de réalisation.

— Objectif 2 - Évaluer le processus logiciel avec les bonnes pratiques de CMMI et S3M : cet objectif est atteint par l'évaluation décrite dans le chapitre 3, les constats au chapitre 4 et les recommandations proposées au chapitre 6.

— Objectif 3 - Élaborer un modèle initial d'estimation : cet objectif est atteint par les résultats présentés dans le chapitre 5.

Ce projet était pour moi une bonne occasion d'approfondir mes connaissances dans les modèles des meilleures pratiques en développement et maintenance de logiciel, en particulier le CMMI et le S3M, d'apprendre comment mener des évaluations, d'organiser

mes idées et mes observations pour les présenter et, pour la première fois, de prendre connaissance de la plupart de processus mis en place par ACME.

Depuis 2001, ACME n'as pas fait d'audit de ses processus. Donc ce travail est une bonne occasion pour revoir ces processus. Dans ce sens, je serais prêt à aider ACME, si la direction prend la décision de mettre en place un plan de mise en œuvre pour nos recommandations. Je suis convaincu que ce travail va aider beaucoup ACME à résoudre plusieurs de ses problèmes, en particulier les problèmes d'estimation et le nombre élevé de bogues.

ANNEXE

Questionnaire

Rôle	Objectif de l'activité	Questions
Gestionnaire de projet	- Collecter les informations concernant la phase de la gestion de projet adoptée par ACME	1. Comment vos projets sont-ils gérés (processus et outils) ?
	- Comprendre les activités menées par le gestionnaire de projet pour assurer le bon fonctionnement de l'équipe de développement	2. Comment estime-t-on le coût ?
		3. Comment planifie-t-on les opérations ?
		4. Comment affecte-t-on les personnes ?
		5. Comment mesure-t-on l'avancement d'un projet ?
		6. Comment gère-t-on les changements ?
		7. Comment gère-t-on les risques dans vos projets ?
Analyste	- Collecter les informations concernant La phase d'analyse de projet adoptée par ACME	1. Décrivez-moi les activités menées pour élaborer les documents d'analyse ?
	- Comprendre le processus mis en place et les différentes activités menées par l'analyste pour élaborer les spécifications	2. Est-ce que vous utilisez un gabarit ou un modèle pour les documents d'analyse ?
		3. Comment assure-t-on la qualité de vos livrables ?
		4. Comment communiquez-vous avec les autres membres de l'équipe ?

Rôle	Objectif de l'activité	Questions
Responsable assurance qualité	- Collecter les informations concernant la phase de test et de validation de projet adoptée par ACME	1. Quels types activités fait-on pour assurer la qualité ?
	- Comprendre les activités menées par l'équipe de l'assurance qualité pour livrer un produit de haute qualité.	2. Comment communiquez-vous avec les autres membres de l'équipe ?
		3. Quels sont les outils utilisés pour l'assurance qualité ?
Développeurs	- Collecter les informations concernant la phase de développement de projet adopté par ACME	1. Décrivez-moi les activités menées pendant la phase de développement ?
	- Comprendre les activités menées par les développeurs pour livrer un produit répondant aux exigences	2. Quelles meilleures pratiques ont été adoptées durant vos développements ?
		3. Comment vous assurez-vous de respectez ces pratiques ?
		4. Comment vous assurez-vous de livrer de la qualité ?

BIBLIOGRAPHIE

APRIL, A. et ABRAN, A., 2016. *Améliorer la maintenance du logiciel.* 2e éd. Québec : Loze-Dion.

CMMI-DEV-TEAM, 2010. *CMMI pour le développement, Version 1.3.* Pittsburgh, PA, États-Unis : Software Engineering Institute.

COSMIC-TEAM-V4.0, 2015. *La méthode COSMIC pour mesurer la taille fonctionnelle des logiciels version 4.0.* 2015e éd. Montréal, Canada.

GARCIA, S., 1993. *Principles of Maturity Modeling, Proceedings of SEI Symposium.* Pittsburgh, PA, USA : Software Engineering Institute, Carnegie Mellon University.

HM&S, 2018. *What is SPICE 1-2-3 ?* Consulté le 21 avril 2018. Disponible à l'adresse : http://www.spice12drive.com/cms/en/about-spice-1-2-1.html.

IEEE-STANDARDS-BOARD, 1990. *IEEE standard glossary of software engineering terminology.* New York : The Institute of Electrical; Electronics Engineering. ISBN 155937067x.

LARAMEE, E., 2016. *L'agilité à 3 niveaux.*

LUCIDCHART, 2018. *Qu'est-ce que la norme de modélisation des processus métier.* Consulté le 22 avril 2018. Disponible à l'adresse : https://www.lucidchart.com/pages/fr/quest-ce-que-le-BPMN.

PAULK, M.C., CURTIS, B., CHRISSIS, M.B. et WEBER, C.V., 1993. *Capability Maturity Model for Software, Version 1.1.* Software Engineering Institute.

PLAYS-IN-BUSINESS, 2019. *ISO/IEC 15504 — SPiCE.* Consulté le 18 mai 2019. Disponible à l'adresse : http://www.plays-in-business.com/isoiec-15504-spice/.

PRESSMAN, R.S. et MAXIM, B.R., 2015. *Software Engineering : A Practitioner's Approach.* 8e éd. New York (NY) : McGraw-Hill.

PROJECT-MANAGEMENT-INSTITUTE, 2013. *A Guide to the Project Management Body of Knowledge, Fifth edition.* 5e éd. Pennsylvania, USA : Project Management Institute; Project Management Institute. ISBN 9781935589679.

SPRUNK, M., 2019. *Selected rules of thumb in software engineering.* Consulté le 30 avril 2019. Disponible à l'adresse : https://www.sw-engineering-candies.com/blog-1/rules-of-thumb-in-software-engineering.

SQLI-GROUP, 2019. *Capability Maturity Model Integration (MS) expliqué simplement.* Consulté le 19 mars 2019. Disponible à l'adresse : http://www.labri.fr/perso/xblanc/data/CP/CMMI_pr%C3%A9sentation_V7.pdf.

SYNECHRON, 2018. *Le CMMI.* Consulté le 5 avril 2018. Disponible à l'adresse : https://www.fimarkets.com/pages/cmmi.php.

TRUDEL, S., 2018. *Cours MGL7240 : Mesures et développement logiciel, été 2018.* Montréal, Canada.

TRUDEL, S., LAVOIE, J.M., PARÉ, M.C. et SURYN, W., 2006. *PEM : The small company-dedicated software process quality evaluation method combining CMMISM and ISO/IEC 14598.* 11.

WEBLOGS.ASP.NET, 2019. *ASP.net MVC Vs ASP.net Web Form.* Consulté le 17 février 2019. Disponible à l'adresse : https://weblogs.asp.net/shijuvarghese/asp-net-mvc-vs-asp-net-web-form.

WIKIPEDIA, 2018a. *Capability Maturity Model Integration.* Consulté le 3 avril 2018. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Capability_Maturity_Model_Integration.

WIKIPEDIA, 2018b. *ISO/IEC 15504.* Consulté le 21 mars 2018. Disponible à l'adresse : https://en.wikipedia.org/wiki/ISO/IEC_15504.

WIKIPEDIA, 2019. *Revue de code.* Consulté le 28 février 2019. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Revue_de_code.

XXL-STRATEGIE, 2019. *Les 5 dysfonctionnements d'une équipe / La pyramide de Lencioni.* Consulté le 11 février 2019. Disponible à l'adresse : https://xxl-strategie.fr/index.php/2017/11/08/les-5-dysfonctionnements-dune-equipe-la-pyramide-de-lencioni/.