UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RAPPORT DE PROJET
PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À MONTRÉAL

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE LOGICIEL

PAR
Carl SIMARD

INNOVATION DANS LES PROCESSUS DU DÉPARTEMENT DE MAINTENANCE DES APPLICATIONS JAVA CHEZ CANADIEN NATIONAL

MONTRÉAL, LE 20 DÉCEMBRE 2017

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Madame Sylvie Trudel, directrice de projet
Département d'informatique à l'Université du Québec à Montréal

Monsieur Louis Martin, président du jury
Département d'informatique à l'Université du Québec à Montréal

Monsieur Normand Séguin, membre du jury
Département d'informatique à l'Université du Québec à Montréal

IL A FAIT L'OBJET D'UNE PRÉSENTATION DEVANT JURY ET PUBLIC

LE 19 DÉCEMBRE 2017

À L'UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Remerciements

J'aimerais, avant tout, remercier Madame Sylvie Trudel pour son support, sa motivation et son encouragement. Elle a toujours été optimiste et disponible afin de m'aider dans cette aventure. Elle est extrêmement professionnelle et possède un souci du détail qui fait toute la différence. Je te remercie Sylvie. J'ai beaucoup appris à travailler avec toi.

J'aimerais, aussi, remercier mes confrères chez Canadien National dont Brian Hossack, Ngan Nguyen, Daniel Hauyon, Gustavo De Leon et Karim El-Bachiri. Ils ont eu un esprit ouvert. Leur support et leur participation ont été des facteurs clés à la réussite de ce projet. Leurs recommandations ont été fortement appréciées et ont aidé à innover le monde des technologies de l'information et du génie logiciel.

J'aimerais remercier ma famille et ma belle-famille pour leur support et leur soutien. Ils croient en moi et m'encouragent chaque jour, et cela dans tous les aspects de ma vie. Mes parents et mes beaux-parents sont des exemples à mes yeux et m'aident à grandir. Je suis juste un enfant qui apprend des grands.

Évidemment, j'aimerais remercier ma chère et tendre épouse Mariel Becerril Ruiz qui a toujours su m'encourager. Elle a toujours cru en moi. Elle est mon idole et m'a toujours inspiré à me surpasser afin d'être le meilleur de moi-même. Sa forte énergie et son sourire m'ont toujours aidé à voir la vie du bon côté, même pendant les moments les plus difficiles. Je te le dis plusieurs fois par jour et encore une fois, je t'aime.

En surplus, je remercie Dieu tous les jours. Il nous a donné Matthew Simard Becerril. Notre petit garçon nous donne une joie de vivre qui nous surprend tous les jours. Ma famille est tout ce qui compte pour moi et je me considère privilégié d'être à leur côté et de faire partie de leur vie.

Résumé

Ce présent projet, comme exigence à la Maîtrise en génie logiciel, a été réalisé chez Canadien National dans le but d'innover les processus à l'intérieur du département de la maintenance des applications et systèmes Java du côté des opérations.

La documentation et l'analyse du processus initial m'ont permis d'identifier cinq problèmes majeurs, dont l'absence de vue d'ensemble, la priorisation segmentée, l'état d'avancement inconnu, l'estimation impossible et la non communication des informations de solutions. Pour les résoudre, j'ai fait une étude et une analyse des méthodes et des normes afin de prendre connaissance des meilleures solutions qui ont déjà été implantées dans l'industrie et ainsi s'en inspirer.

Pour assurer le succès de mon projet d'amélioration, j'ai défini une méthodologie et l'ai déployée, incluant une phase de démarrage, de mise en œuvre, d'amélioration et finalement de fermeture. Les objectifs, afin de résoudre les cinq problèmes de départ, ont été atteints grâce à la mise en œuvre du cadre Scrumban à l'intérieur du processus initial. En plus de résoudre la problématique, Scrumban nous aide à être beaucoup plus efficaces et mieux organisés, ce qui nous permet d'ajouter plus de valeur à la compagnie.

Pour soutenir cette mise en œuvre, j'avais sélectionné un premier outil - Kanboard - et l'ai utilisé. Étant donné qu'il n'était pas intégré avec notre outil de gestion des requêtes de changement - HP Service Manager (HPSM) – j'ai dû changer d'outil pour Jira, pleinement intégré avec HPSM, nous évitant ainsi une double saisie des informations. J'ai par ailleurs instauré des rencontres hebdomadaires de synchronisation et de partage des connaissances avec tous les membres de l'équipe de maintenance des applications Java du CN, ce qui permet une plus grande flexibilité d'affectation des membres d'équipe à différentes applications.

Ce projet d'application a été limité à notre département, bien que la méthodologie utilisée soit prévue d'être appliquée aux autres groupes de maintenance logicielle du CN au cours des prochaines années (voire même des prochains mois).

Le cadre Scrumban est plutôt récent dans le domaine du génie logiciel et des technologies de l'information. Le fait de l'avoir mis en pratique nous permet d'être avant-gardistes au sein du CN et va probablement motiver et aider nos collègues à innover comme nous.

Abstract

This project, as a requirement in the Master's degree in software engineering, has been realized at Canadian National Railway with the aim of innovating the processes of the maintenance department of the Java applications and Java systems on the operations unit.

The documentation and the analysis of the initial process have allowed me to identify five major problems, among which the absence of overview, the segmented prioritization, the unknown state of progress, the impossible estimation and the non communication of information of solutions. To solve them, I studied and analyzed methods and standards to acquire knowledge of the best solutions already deployed in the industry and thus be inspired by them.

To ensure the success of my improvement project, I have defined a methodology and have deployed it, including these phases: start-up, implementation, improvement, and closure. The objectives to solve the five initial problems were reached, thanks to the deployment of the Scrumban framework supporting the initial process. Besides solving the problem, Scrumban helps us to be much more effective and better organized, which allows us to add more value to the company.

To support this deployment, I had selected a first tool - Kanboard – and used it. Given that it was not integrated with our change request management tool - HP Service Manager (HPSM) - I had to change the tool for Jira, completely integrated with HPSM, avoiding a double information entry. I have also established weekly meetings with all maintenance department team members as a means for knowledge transfer and to support colleagues when needed. It allows more flexibility in team member's affectation to various applications.

This application project was limited to our department, although the applied methodology is planned to be used by CN's other software maintenance groups during the next years (even next months).

The Scrumban framework is quite recent in the software engineering and information technology domain. Having deployed Scrumban allows us to innovate inside CN and hopefully it will motivate and help our colleagues to innovate too.

Table des matières

Introduction 1

Chapitre 1. Canadien National 3

1.1. Canadien National 3

1.2. Division des technologies de l'information chez Canadien National 5

1.3. L'équipe de maintenance des applications Java – Division des opérations ..7

1.4. Les applications supportées 7

1.4.1. Smartyard 8

1.4.2. EMS 8

1.4.3. WHN 9

1.4.4. Demurrage Preview 9

1.4.5. Oreis 9

1.4.6. Corporate Data WorkQueue 9

1.4.7. FMLM/CarShipment 10

Chapitre 2. Contexte et problématique 11

2.1. Le contexte 11

2.3. Problématique 14

2.3.1. Absence de vue d'ensemble 16

2.3.2. Priorisation segmentée 17

2.3.3. État d'avancement inconnu 17

2.3.4. Estimation impossible 17

2.3.5. Informations des solutions non communiquées 18

2.4. Résumé des conséquences 18

Chapitre 3. État de l'art 19
3.1. La maintenance du logiciel 19
3.2. Les types de maintenance du logiciel 22
3.3. Les normes et modèles de la maintenance du logiciel 23
3.3.1. ISO/IEC 14764, Software Engineering -- Software Life Cycle Processes – Maintenance: 23
3.3.2. Software Maintenance Maturity Model (S3M): 25
3.4. Approche agile émergente de gestion de la maintenance du logiciel 27
3.4.1. eXtreme Programming 29
3.4.2. Scrum 31
3.4.3. Kanban 35
3.4.4. Srumban 37
Chapitre 4. Méthodologie 39
4.1. But et objectifs 39
4.2. Portée 40
4.3. Aperçu de la méthodologie 40
4.4. Démarrage 42
4.4.1. Analyser et sélectionner une méthode agile 42
4.4.2. Analyser et sélectionner un outil Kanban 42
4.5. Mise en œuvre initiale 43
4.5.1. Installer Kanboard 43
4.5.2. Implémenter Scrumban 43
4.5.3. Présenter le processus Scrumban et l'outil Kanban 44
4.6. Améliorations 44
4.6.1. Analyser et améliorer le processus Scrumban 44
4.6.2. Analyser et améliorer l'outil Kanban 45
4.7. Fermeture 45
4.7.1. Documenter le processus Scrumban 45

Chapitre 5. Résultats 47
5.1. Démarrage 47
5.1.1. Processus actuel documenté 47
5.1.2. Méthode Scrumban sélectionnée 47
5.1.3. Outil Kanboard sélectionné 48
5.2. Mise en œuvre initiale 48
5.2.1. Outil Kanban installé 48
5.2.2. Processus Scrumban implémenté avec mesures 50
5.2.3. Équipe formée 54
5.3. Améliorations 54
5.3.1. Processus Scrumban amélioré 54
5.3.2. Outil Jira installé 58
5.4. Fermeture 63
5.4.1. Processus Scrumban amélioré documenté 63
Chapitre 6. Discussion 65
6.1. Meilleure connaissance de la problématique du processus initial 65
6.2. Meilleure connaissance de l'approche et des méthodes agiles 65
6.3. Les défis et obstacles rencontrés 66
6.4. Résistance aux changements 66
6.5. Acceptation de la méthode Scrumban 67
6.6. Meilleure connaissance de la méthode Scrumban au CN 67
Conclusion 69
Bibliographie 73
Annexe 79

Liste des tableaux

Tableau 2.1 - Résumé des problèmes et leurs conséquences.....15
Tableau 4.1 - Liste des problèmes 39
Tableau 4.2 - Méthodologie.....41
Tableau 7.1 - Liste des problèmes 69

Liste des figures

Figure 0.1 - Équipemen du CN xix
Figure 0.2 - Locomotive du CN xx
Figure 0.3 - Train du CN xxi
Figure 0.4 - Exemple d'un Yard xxii
Figure 1.1 - Étendue du réseau ferroviaire du CN 4
Figure 1.2 - Carte du réseau ferroviaire du CN 4
Figure 2.1 - Processus initial de maintenance des applications en Java 12
Figure 3.1 - Scrum et ses pratiques 34
Figure 3.2 - Exemple appliqué de la méthode Kanban 36
Figure 5.1 – Exemple de l'interface de l'outil Kanboard 49
Figure 5.2 – Exemple des étapes du processus Kanban 51
Figure 5.3 - Exemple des étapes du processus Kanban 52
Figure 5.4 - Exemple des étapes du processus Kanban 53
Figure 5.5 - Exemple de notre tableau Kanban avec l'outil Jira 58
Figure 5.6 - Exemple (zoom) de notre tableau Kanban avec l'outil Jira 59
Figure 5.7 - Demande de changement OPSC-224 59
Figure 5.8 - Détail de la demande de changement OPSC-224 1/4 60
Figure 5.9 - Détail de la demande de changement OPSC-224 2/4 61
Figure 5.10 - Détail de la demande de changement OPSC-224 3/4 61
Figure 5.11 - Détail de la demande de changement OPSC-224 4/4 62

Définitions, acronymes et abréviations

CAD	Devise en dollars Canadiens.
CIO	Officier en chef de l'information ou *Chief Information Officer.*
CN	Canadien National.
DB2	IBM DB2 est un système de gestion de base de données.
EMS	Event Management System est une application Java au CN qui permet de faire l'analyse et la gestion des corrections des événements des équipements.
Équipement	Chez CN, un équipement ferroviaire peut transporter du charbon, de l'engrais, des boissons et aliments, des produits forestiers, des produits céréaliers, des métaux et des minéraux, des ventes en gros, du pétrole et des produits chimiques, des cultures spécialisées et finalement des automobiles.

Figure 0.1 - Équipemen du CN[1]

Équipe Java Core Support	Équipe qui fait le support et la maintenance logicielle des applications Java du côté des opérations.
Équipe QA	Équipe Quality Assurance est une équipe qui fait des tests d'acceptation afin de s'assurer de la qualité des applications et des systèmes.
HPSM	HP Service Manager est un outil de gestion qui permet de gérer et suivre les incidents et les changements en production des applications et des systèmes informatiques.

[1] Image tirée de Wikimedia commons. (2014). Tous droits réservés. [Image en ligne] <https://commons.wikimedia.org>

I&T	*Information & Technologies*, division responsable des TI au CN.
Java J2EE	Java Enterprise Edition est une spécification ou un langage de programmation afin d'implémenter des applications d'entreprises.
JUnit	JUnit est une technologie afin de faire des tests unitaires en Java.
Locomotive	Une locomotive est un véhicule ferroviaire qui fait avancer ou reculer un train.

Figure 00.2 - Locomotive du CN[2]

Oracle	Oracle est un système de gestion de base de données.
PHP	Hypertext Preprocessor est un langage de programmation permettant de développer des applications web.
PO	*Product Owner*, ou propriétaire du produit [logiciel].
PTC	Positive train control est un projet au CN qui permet de monitorer et de contrôler le mouvement des trains.
Railinc	Railinc est une compagnie qui donne des services dans l'industrie ferroviaire. Elle est une compagnie innovatrice et elle est une ressource fiable pour les technologies de l'information dans le domaine ferroviaire. <https://www.railinc.com/rportal/web/guest/home>
SM	Scrum Master.
Spring	Spring est un cadre de développement aidant le développement d'application Java J2EE.
SQL	SQL est un langage de requête structurée pour manipuler ou exploiter des bases de données comme Oracle ou DB2.
SQLJ	SQLJ est une technologie afin d'écrire du SQL en Java.

[2] Image tirée de CNR – Canadian National Railways Historical Association. (2014). Tous droits réservés. [Image en ligne] <http://www.cnrha.ca>

SRS	Service reliable system est un système mainframe au CN. En résumé, il a été le premier système au CN et il permet de tout faire afin de livrer la marchandise et donner un excellent service aux clients. Tranquillement, les fonctionnalités sont remplacées par des technologies plus modernes.
Struts	Struts est un framework aidant le développement d'application Java J2EE.
TDD	*Test Driven Development* ou développement piloté par les tests.
TI	Technologies de l'information.
Train	Un train est une composition de une à plusieurs locomotives avec au moins un équipement ferroviaire.

Figure 0.3 - Train du CN[3]

VP	Vice-président.
WHN	Wayside Health Network est un système implémenté en Java qui permet l'analyse des données récoltées sur le terrain afin d'augmenter la sécurité ferroviaire. Dans le cas d'une alerte, un message est envoyé à l'équipe concernée.
WILD	WILD est un acronyme qui signifie Wheel Impact Load Detector. C'est un système qui reçoit l'information des appareils installés dans les yards et sur les voies ferroviaires.

[3] Image tirée de The Sustainable Business Review. (2014). Tous droits réservés. [Image en ligne] <http://www.tsbreview.com>

Yard	Court de triage en français. Emplacement où l'on gère et opère les trains (inspecter, défaire, construire, inspecter de nouveau, faire la maintenance au besoin, etc.) (ex : Taschereau yard à Montréal et Macmil yard à Toronto).

Figure 0.4 - Exemple d'un Yard[4]

[4] Image tirée de Your Railway Pictures.com. (2014). Tous droits réservés. [Image en ligne] <http://yourrailwaypictures.com>

Introduction

Dans le cadre de la Maîtrise en génie logiciel, un projet innovateur doit être réalisé afin que les étudiants puissent bien assimiler les concepts vus en classe. Puisque l'industrie du génie logiciel et des technologies de l'information est de plus en plus en forte croissance, un domaine rempli de belles opportunités s'offre aux étudiants. Que ce soit dans le domaine du Big Data, de l'intelligence artificielle, de la robotique, de l'informatique cognitive, des bases de données, de la sécurité, de l'infonuagique, du data mining, de la gestion de projet ou des processus, les choix de sujets sont nombreux et très intéressants.

Aujourd'hui, les projets en génie logiciel et des technologies de l'information sont de plus en plus complexes à réaliser et de plus en plus difficiles à livrer à temps. Les équipes multidisciplinaires deviennent de plus en plus grandes et les parties prenantes en veulent toujours plus, et cela avec des budgets limités. Finalement, la pression d'obtenir de bons résultats est souvent énorme pour les gestionnaires de projet et les propriétaires d'applications. Or, pour obtenir ces bons résultats, ils ont besoin d'applications performantes et c'est pourquoi ils demandent des changements aux applications existantes, tout en souhaitant obtenir ces modifications le plus rapidement possible et avec un très haut niveau de qualité.

Le sujet de ce présent projet porte sur l'innovation des processus dans l'équipe de maintenance des applications Java du département des technologies de l'information chez Canadien National, dans un but d'améliorer les performances du processus de maintenance afin de mieux servir les propriétaires d'applications.

Ce présent rapport est composé de six grands chapitres et d'une conclusion. Ce qui suit décrit en quelques mots le contenu de chaque partie. Le premier chapitre décrit sommairement l'entreprise ferroviaire Canadien National et la structure

organisationnelle du département des technologies de l'information, en finissant par le département de maintenance des applications Java. Le deuxième chapitre parle du contexte de la situation et des problématiques rencontrées lors de l'analyse des processus appliqués dans le département de la maintenance des applications Java. Le troisième chapitre contient une revue de l'état de l'art sur les mêmes problématiques rencontrées lors de l'analyse, mais aussi des solutions pouvant y être apportées. Le quatrième chapitre explique la méthodologie utilisée afin de résoudre la problématique expliquée dans le contexte. Il est question aussi des objectifs et de la portée de notre projet. Le cinquième chapitre explique l'expérimentation réalisée et les résultats obtenus. Le dernier chapitre discute des retombées du projet, dont les obstacles et défis rencontrés.

Finalement, la conclusion fait un lien entre les objectifs et les résultats obtenus. Aussi, cette dernière partie donne quelques recommandations afin que les activités réalisées dans ce projet se poursuivent en mode d'amélioration continue.

Chapitre 1. Canadien National

1.1. Canadien National

Canadien National est une compagnie ferroviaire, cotée en bourse, ayant un chiffre d'affaires de plus de 10 milliards de dollars CAD par année et un bénéfice d'exploitation d'environ 4 milliards de dollars CAD par année. Elle gère plus de 23 000 employés, dont environ 900 dans les technologies de l'information. Avec un réseau ferroviaire de plus de 32 000 kilomètres, elle est considérée aujourd'hui comme la plus grande compagnie ferroviaire en Amérique du Nord. Son réseau ferroviaire s'étend de l'est à l'ouest du Canada jusqu'au sud des États-Unis avec plus d'une vingtaine de terminaux intermodaux stratégiques, dont Montréal et Toronto.

Canadien National possède aussi de nombreux partenaires, dont plusieurs installations portuaires par exemple le port de Halifax, Montréal, Prince-Rupert, Vancouver, Nouvelle-Orléans et Québec, afin de livrer la marchandise partout à travers le monde. Avec les 23 terminaux intermodaux, les clients ont accès à 100 % des marchés canadiens et à plus de 75 % des marchés américains. Les deux prochaines figures montrent l'étendue et la carte du réseau ferroviaire du Canadien National.

Figure 1.1 - Étendue du réseau ferroviaire du CN[5]

Figure 1.2 - Carte du réseau ferroviaire du CN[6]

[5] Image tirée de Canadien National. (2014). Tous droits réservés. [Image en ligne] <http://www.cn.ca>
[6] Image tirée de Canadien National. (2014). Tous droits réservés. [Image en ligne] <http://www.cn.ca>

Le Canadien National se spécialise dans le transport de marchandises dont l'industrie de l'automobile, le charbon, l'engrais, les boissons et les aliments, les produits forestiers, les produits céréaliers, les métaux et les minéraux, le pétrole et produits chimiques, les cultures spécialisées, l'intermodal et les chargements surdimensionnés.

L'entreprise ferroviaire met l'accent sur cinq principes dont le service, le contrôle des coûts, l'utilisation des actifs, la sécurité et finalement le personnel. Elle possède un excellent contrôle des coûts et une efficacité hors pair en ce qui concerne la logistique de ses équipements. Aussi, afin de toujours s'améliorer, elle se focalise en ce moment dans l'amélioration de son service à la clientèle.

Finalement, il est possible de connaître l'histoire de l'entreprise sur le site Internet de Wikipédia[7] et de l'Institut de recherche sur l'histoire des chemins de fer du Québec[8].

1.2. Division des technologies de l'information chez Canadien National

La division des technologies de l'information (I&T) est maintenant sous la responsabilité de Monsieur Serge Leduc, vice-président (VP) et officier en chef de l'information (CIO). Depuis son arrivée, le département possède quatre grands piliers dont :

- Engagement auprès de nos partenaires : ce pilier est responsable de l'approche et de la gouvernance en matière d'engagement auprès de leurs partenaires, d'améliorer les processus de planification de projets, de veiller à la demande appropriée avec la liste de projets des technologies de l'information (TI) et finalement d'établir une stratégie de communication pour améliorer les interactions avec leurs partenaires.

[7] Wikipédia. Canadien National. [En ligne] <http://fr.wikipedia.org/wiki/Canadien_National>
[8] Institut de recherche sur l'histoire des chemins de fer du Québec. Inventaire bibliographique de l'IRHCFQ. [En ligne]. <http://www.irhcfq.org/web_fr/bibliographie.html>

- Gestion des applications et livraison : ce pilier est responsable d'établir les disciplines de gestion du portefeuille et la gestion du cycle de vie des produits, de faire passer les services de livraison au niveau supérieur et finalement d'impliquer tout le monde dans l'amélioration des processus.

- Infrastructure et télécommunication : ce pilier est responsable d'aligner la gestion du cycle de vie des produits et les finances du groupe I&T, de mettre en place la stratégie d'approvisionnement de la main-d'œuvre I&T, d'évaluer l'écart fonctionnel du groupe I&T, d'élever le niveau de maturité de tous les centres d'excellence du groupe I&T, de continuer de renforcer les relations de travail avec le groupe Signalisation et Communications et finalement produire le modèle de soutien du projet Positive train control (PTC) pour les composants des TI.

- Planification et stratégie : ce pilier est responsable d'appuyer la transformation des TI, dont obtenir le soutien de la direction, gérer une feuille de route fonctionnelle et un cadre pour les processus et l'information d'entreprise.

Le pilier qui nous intéresse ici est celui de la gestion des applications et livraison. Ce dernier est sous la direction de Monsieur Alain Boucher, directeur sénior des technologies de l'information. L'objectif est d'améliorer la performance et la gouvernance en ce qui concerne la gestion des projets dans les technologies de l'information et le cycle de vie des applications dont la maintenance. Dans ce même pilier, on y trouve la gestion du portfolio des applications existantes en production, la gestion de projets, les solutions d'architecture et finalement l'intelligence d'affaires.

En ce qui concerne la gestion du portfolio des applications existantes en production, elles sont classées selon trois divisions, soit le markéting, les opérations et finalement l'entreprise.

Chez Canadien National, plusieurs technologies sont utilisées et les équipes sont formées en conséquence, et cela dans chacune des divisions.

1.3. L'équipe de maintenance des applications Java – Division des opérations

L'équipe de maintenance des applications Java, division des opérations, est composée de cinq concepteurs/développeurs d'applications et un gestionnaire. Chaque personne possède une expertise assez précise sur une application donnée. Cependant, tout le monde a l'opportunité de travailler sur tout afin d'augmenter leurs connaissances du domaine d'affaires et leurs expertises.

L'équipe de maintenance des applications Java supporte plusieurs applications qui permettent à la compagnie de bien gérer les opérations sur le terrain afin que le personnel de Canadien National soit proactif, sécuritaire et efficace. Puisque les applications et les systèmes ont un impact direct sur les opérations, leur bon fonctionnement est critique. Ainsi, les membres de l'équipe en question ont la responsabilité de s'assurer du bon fonctionnement des systèmes qu'ils supportent et de faire la gestion des changements qu'ils soient adaptatifs ou correctifs.

1.4. Les applications supportées

Chaque application que l'équipe supporte possède ses propres gens d'affaires qui représentent les utilisateurs sur le terrain. Les gens d'affaires ont la responsabilité de faire la demande de nouvelles fonctionnalités et nous faire part des correctifs tout en priorisant les items dans la liste. En d'autres mots, l'équipe collabore directement avec les gens d'affaires et cela quotidiennement, car il n'y a aucun gestionnaire de projets dans le département de maintenance.

Les sous-sections qui suivent donnent un aperçu des applications supportées par l'équipe.

1.4.1. Smartyard

Smartyard est une application utilisée dans les yards afin de gérer les trains et la logistique des équipements. En résumé, les utilisateurs peuvent voir l'estimation de l'heure d'arrivée des trains dans le yard et peuvent commencer à gérer le personnel en conséquence. Cela leur permet d'organiser d'avance l'ouvrage sur le terrain à savoir comment faire les transferts d'équipements ferroviaires d'une voie à une autre afin de reconstruire de nouveaux trains. L'application permet aussi de voir dans quel état sont les équipements et les trains. Prenons par exemple le fait qu'un équipement soit en ce moment en maintenance.

Les technologies utilisées sont le Java J2EE, Swing, Oracle et quelques cadres de développement comme SQLJ, JDBC, JUnit, Struts et Spring.

1.4.2. EMS

EMS est un système en soit avec quelques applications web et back-end qui permet de voir les évènements des équipements et l'information qui a été envoyée à Railinc[9]. Le système permet aussi de voir les exceptions et d'en faire les corrections. Prenons par exemple qu'un départ ne peut se produire avant une arrivée. Les évènements sont utilisés dans le département de finance et comptabilité afin d'obtenir plusieurs statistiques et d'ajuster les prix en conséquence.

Les technologies utilisées sont le Java J2EE, DB2 et quelques cadres de développement comme SQLJ, JDBC, JUnit, Struts et Spring.

---

[9] Railinc. [En ligne] <https://www.railinc.com/rportal/web/guest/home>

1.4.3. WHN

WHN est un système développé en Java où les fonctionnalités principales sont d'analyser l'information reçue du terrain et d'envoyer des alertes au besoin. L'information provient soit de Wayside, Wild ou de Railinc. En d'autres mots, l'objectif principal du système en entier est de vérifier la qualité des équipements afin d'éviter un déraillement sur le réseau ferroviaire pour ainsi augmenter la sécurité.

Les technologies utilisées sont le Java J2EE, DB2 et quelques cadres de développement comme SQLJ, JDBC, JUnit, Struts et Spring.

1.4.4. Demurrage Preview

Demurrage Preview est une application web qui permet aux utilisateurs de gérer les comptes payables des clients.

Les technologies utilisées sont le Java J2EE, GWT, DB2 et quelques cadres de développement JDBC, JUnit, Spring.

1.4.5. Oreis

Oreis est une application web extranet qui permet aux premiers répondants lors d'un déraillement d'obtenir l'information sur la marchandise des équipements.

Les technologies utilisées sont le Java J2EE, DB2 et quelques cadres de développement comme SQLJ, JDBC, Struts, JUnit et Spring.

1.4.6. Corporate Data WorkQueue

Corporate Data Work Queue est une application web intranet qui permet aux utilisateurs de gérer l'information des clients. Les technologies utilisées sont le Java

J2EE, DB2 et quelques cadres de développement comme SQLJ, JDBC, JUnit, Struts et Spring.

1.4.7. FMLM/CarShipment

CarShipment est une application *back-end* qui filtre et analyse l'information reçue des autres systèmes, dont Service Reliable System (SRS), afin d'obtenir plusieurs informations sur les équipements dont l'état dans leur cycle de livraison et plusieurs autres. Les équipements peuvent être dans le yard d'origine, en route, à l'intérieur de 24 heures du yard de destination, dans le yard de destination ou finalement à la destination finale.

FMLM est une application web externe qui utilise CarShipment Service pour obtenir l'information afin de montrer aux clients où sont rendus leurs équipements, dans quelle catégorie ils sont et plusieurs autres dont l'estimation de la date et de l'heure d'arrivée chez le client.

Les technologies utilisées sont le Java, Oracle et quelques cadres de développement comme JDBC, Camel, JUnit, Struts et Spring.

Chapitre 2. Contexte et problématique

Le chapitre 2 discute du contexte et des problématiques rencontrées lors de mon analyse du processus initial du département Java Core Support. Tout d'abord, la première partie explique et discute du contexte et du processus actuels du département en question et la deuxième partie discute davantage des problématiques et leurs conséquences sur la compagnie Canadien National.

2.1. Le contexte

L'équipe de maintenance des applications Java J2EE du côté des opérations chez Canadien National est composée de cinq développeurs et un gestionnaire d'équipe. Chaque membre de l'équipe travaille sur une application donnée pendant quelques semaines et est ensuite réaffecté afin de changer la routine et de rester à jour avec les autres applications et les autres technologies utilisées. Le processus actuel permet de livrer de la valeur en mettant en production les différents changements demandés et implémentés, mais on observe quelques lacunes causant des problèmes majeurs, et cela à court et à long terme. Le diagramme à la page suivante explique le processus initial établi dans le département en question.

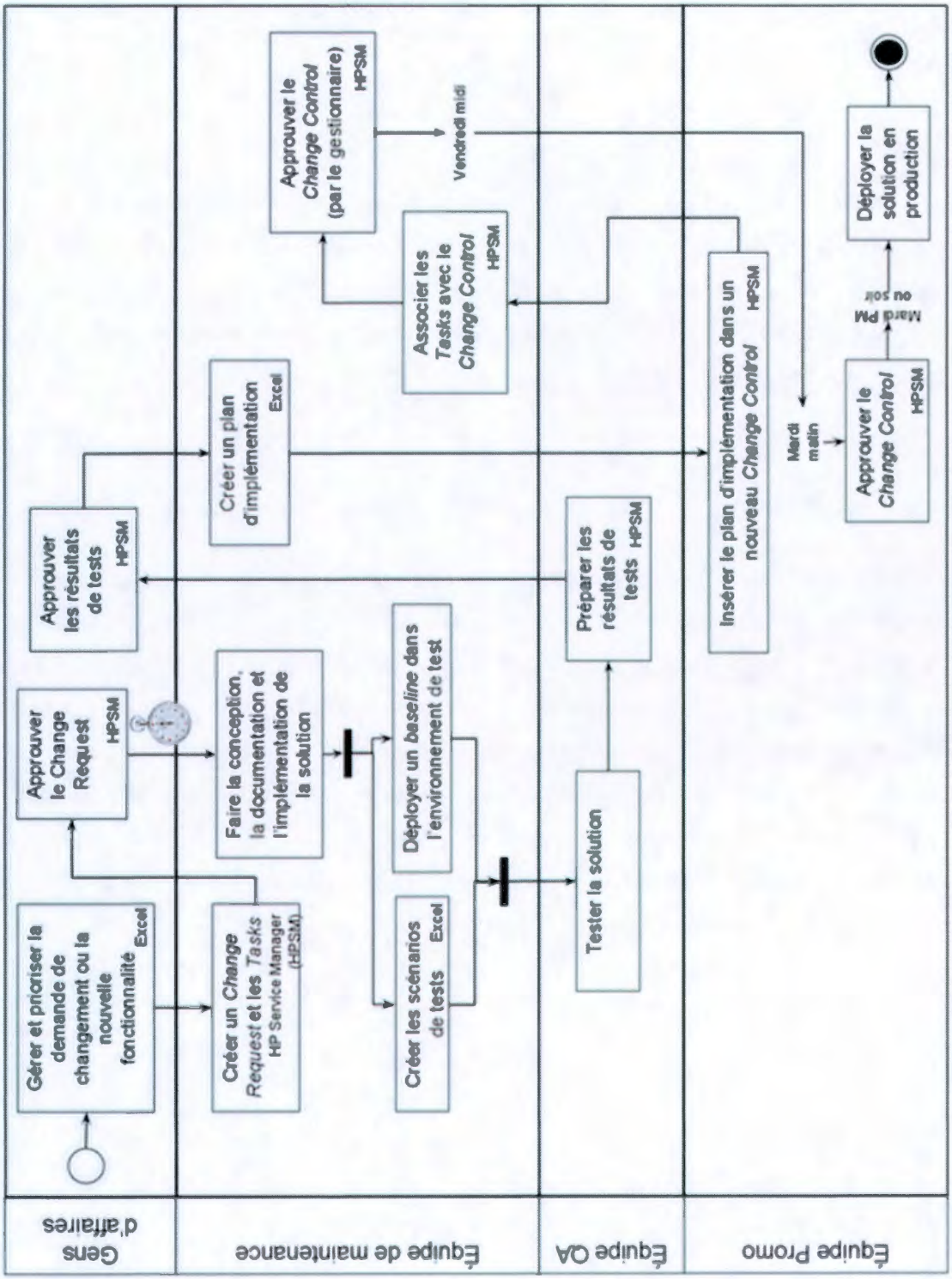


Figure 2.1 - Processus initial de maintenance des applications en Java

2.2. Processus

Tout d'abord, les gens d'affaires vont gérer les demandes de changements et les nouvelles fonctionnalités à l'aide d'un fichier Excel et/ou l'outil HP Service Manager. En d'autres mots, chaque application a son propre fichier Excel. Quelques fois, la demande de changement ou la nouvelle fonctionnalité va être créée à l'aide d'un *Change Request* et quelques *Tasks* dans HP Service Manager (HPSM). Les *Change Requests* et les *Tasks* sont obligatoires pour déployer le nouveau référentiel en production. Ils sont créés par un développeur et approuvés par les gens d'affaires.

En parallèle, les développeurs peuvent décider de travailler sur une application donnée sans connaître la liste des demandes de changements et des nouvelles fonctionnalités. Ils doivent s'assoir avec le développeur qui possède le fichier Excel et lui demander sur quoi il peut travailler, car il n'a pas l'information. Dans le contexte initial, il y a une priorisation des demandes de changements et des nouvelles fonctionnalités, mais seulement par application et non dans l'ensemble.

Une fois que le développeur décide de travailler sur une demande de changement ou sur une nouvelle fonctionnalité et qu'il a bien compris les besoins d'affaires, il peut commencer à travailler sur la conception et la documentation. Ensuite, une fois que la conception semble la plus adéquate, il peut travailler sur le développement de la solution. Une fois le travail terminé et testé sur son ordinateur, il pourra créer un nouveau référentiel à l'aide d'une compilation et le déployer dans les environnements de tests.

Dès que le nouveau référentiel est déployé dans l'environnement de tests, un nouveau plan de tests est élaboré afin de faire des tests de non régression et des tests de la nouvelle fonctionnalité et/ou des demandes de changements. Il est à noter qu'un développeur peut travailler sur plusieurs changements avant de faire un nouveau référentiel.

Les tests peuvent être faits par l'équipe QA ou par un autre développeur qui pourra mettre le chapeau de testeur pendant quelques heures. Cela dépend de l'application

et des changements. Prenons par exemple l'application SmartYard. Celle-ci est habituellement testée par l'équipe QA, car les impacts sont directement liés avec la performance des opérations sur le terrain. Une fois que les tests sont terminés, et cela avec succès, ils sont associés avec les *Tasks* concernées dans HPSM.

La dernière étape avant la mise en production est de créer un *Change Control*, soit un plan d'implémentation, et approuver les résultats des tests et le Change Control. Une fois que tout est approuvé, il ne reste plus qu'à déployer le nouveau référentiel en production.

En général, les processus du département des technologies de l'information chez Canadien National possèdent une bonne pratique et une maturité jugées adéquates par le personnel et la direction. Cependant, le processus possède quelques problèmes expliqués à la section suivante.

2.3. Problématique

Après avoir analysé le contexte et le processus initial se trouvant dans le département de la maintenance des applications Java, il est facile de constater qu'il y a, à ce moment, quelques problèmes. Comme mentionné plus haut, le processus initial permet de livrer de la valeur à la compagnie, mais quelques faiblesses et goulots causent indirectement des impacts négatifs.

Le tableau suivant résume les problèmes et les conséquences observées lors de l'analyse du contexte de la situation et du processus initial. Les paragraphes suivants en discutent davantage.

Tableau 2.1 - Résumé des problèmes et leurs conséquences

Problèmes	Conséquences
Absence de vue d'ensemble : Chaque développeur qui termine une solution doit choisir parmi les prochaines demandes de changements, donc fouiller dans de multiples listes Excel. Pour chaque élément examiné, le développeur doit essayer de comprendre le besoin avant de décider s'il est apte à sélectionner cet élément.	— Ralentissement du démarrage du cycle de développement. — Perte d'efficacité, donc moins de nouvelles fonctionnalités en production au courant de l'année.
Priorisation segmentée : Il est impossible de connaître les priorités des demandes de changement parmi les différentes applications. La priorité par application est connue, mais ne l'est pas inter-applications.	— Les nouvelles fonctionnalités déployées ne sont pas nécessairement celles qui ajoutent le plus de valeur pour la compagnie.
État d'avancement inconnu : Il est difficile de connaître l'état d'avancement des demandes de changements et des nouvelles fonctionnalités.	— Une nouvelle fonctionnalité reste parfois bloquée dans une phase du cycle de développement. — Pas de retour possible aux gens d'affaires sur l'état d'avancement.
Estimation impossible : Il est difficile et parfois impossible de faire des estimations réalistes.	— Impossible de gérer les attentes des clients, ce qui les irrite. – Diminution de la qualité du service et la satisfaction du client.

Problèmes	Conséquences
Informations des solutions non communiquées : Les personnes autres que celles ayant développé une solution n'ont aucune idée des composantes ajoutées, modifiées ou supprimées ni des technologies changées. Chaque développeur documente les solutions à sa manière et parfois ces informations sont placées sous SharePoint. Toutefois, les fichiers étant nombreux, il est difficile pour une autre personne de fouiller et trouver rapidement l'information dont elle aurait besoin.	– Rend la maintenance corrective et adaptative beaucoup moins efficace, car il y a une courbe d'apprentissage lorsqu'on doit faire une correction rapide en production.

2.3.1. Absence de vue d'ensemble

Puisque chaque membre de l'équipe travaille sur une application donnée et que chaque application a sa propre liste de changements et correctifs à faire, il est difficile d'avoir une vue globale pour l'ensemble des applications. Cette vision en lorgnette par application rend difficile de connaitre les demandes de changements des autres applications, ce qui a comme conséquence de ralentir le démarrage du cycle de développement d'une nouvelle fonctionnalité. La personne qui désire travailler sur une autre application doit vérifier avec les personnes concernées et analyser de nouveau la liste afin de faire un choix. Le choix est basé sur les applications et non basé sur la valeur d'affaires des demandes de changement. L'efficacité du processus est donc impactée de façon négative, ce qui ralentit la vélocité. Ainsi, le nombre de nouvelles fonctionnalités déployées en production n'est pas à son maximum et le coût par demande de changement est fortement plus haut qu'il ne le devrait.

2.3.2. Priorisation segmentée

Les priorités sont établies pour les demandes de changement par application dans un chiffrier Excel. Ces priorités sont « High », « Medium » ou « Low », et les gens d'affaires trient la liste dans l'ordre souhaité. Bien qu'aucune valeur d'affaires monétaire ne soit inscrite à chaque demande de changement, l'impact sur les affaires y est inscrit dans l'analyse des besoins : ex. empêcher un train de dérailler, augmenter l'efficience d'un processus utilisateur, ajouter de l'information manquante dans le système, etc. Cependant, il n'y a aucune priorisation entre les demandes de changements de l'ensemble de toutes les applications. Ainsi, il est difficile de choisir en fonction des valeurs ajoutées aux clients internes. Ce qui donne comme problématique que les nouvelles fonctionnalités déployées en production ne sont pas nécessairement celles qui ajoutent le plus de valeur pour la compagnie.

2.3.3. État d'avancement inconnu

Dans le contexte actuel, il est difficile de savoir qui travaille sur quoi exactement et comment les choses avancent. Cela cause problématique, car il arrive souvent qu'une application ne change pas pendant plusieurs semaines et même pendant plusieurs mois, parce que personne ne travaille dessus. Aussi, il arrive parfois qu'il y ait des changements qui ont été commencés, mais qui se perdent en cours de route parce que la personne a décidé de travailler, un matin, sur une autre application. Finalement, l'état de ce changement ne change jamais, car personne n'est au courant.

2.3.4. Estimation impossible

Il arrive souvent que les gens d'affaires veuillent obtenir une estimation de l'échéancier afin de connaître la date de la mise en production de leurs nouvelles fonctionnalités. Cependant, il est souvent difficile de donner une bonne estimation et parfois c'est même impossible, car le processus initial n'a aucune mesure. Ceci cause plus que

souvent de la frustration et a comme conséquence de donner un mauvais service à la clientèle d'affaires à l'interne de la compagnie.

2.3.5. Informations des solutions non communiquées

Finalement, les personnes n'ont aucune idée des nouvelles fonctionnalités implémentées et des technologies utilisées. Ceci rend très difficile la maintenance corrective, car les personnes ne sont pas au courant des nouvelles fonctionnalités et des technologies utilisées. Ceci rend aussi la maintenance beaucoup moins efficace, car il doit y a voir une courbe d'apprentissage lorsqu'il est temps de faire une correction rapide en production.

2.4. Résumé des conséquences

En résumé, les conséquences des problèmes expliqués dans cette section ne sont pas à négliger, car elles ont des impacts négatifs envers la compagnie. Il y a une perte d'efficacité dans le processus donc moins de valeur ajoutée à la fin de l'année. Le personnel coûte cher, alors il faut optimiser le rendement et diminuer la perte de temps et les goulots. Les coûts par demande de changement sont plus élevés qu'il ne le faudrait et la satisfaction de la clientèle interne est laissée à risque. Les gens d'affaires ne savent jamais ou presque quand leurs changements demandés seront déployés en production.

Chapitre 3. État de l'art

Le présent chapitre met en revue les stratégies, théories et approches qui ont été expliquées à travers toutes les lectures qui ont été faites afin de bien comprendre les problématiques et les solutions potentielles qui auraient été appliquées par d'autres dans des contextes similaires. Cette revue de l'état de l'art nous permettra aussi d'analyser un peu plus en profondeur les méthodologies qui sont expliquées et proposées afin de résoudre les problématiques de ce présent projet.

Les présentes problématiques sont un sujet très à la mode dans le domaine du génie logiciel et surtout dans celui de la gestion de projet en technologie de l'information. Il y a en effet plusieurs ouvrages qui discutent et donnent des modèles sur l'amélioration des processus en maintenance du logiciel. Pour les lecteurs qui veulent en savoir un peu plus sur le sujet, plusieurs autres articles, mémoires et livres sont mentionnés à la fin de ce rapport à la section des bibliographies.

3.1. La maintenance du logiciel

Lors d'un projet d'un nouveau logiciel, plusieurs phases sont accomplies afin de livrer un produit de haute qualité aux clients. Prenons par exemple un modèle en V ou en cascade très simple. Il y a l'analyse des besoins, la conception du nouveau système, le développement, les tests, la livraison, l'enseignement aux utilisateurs et finalement la transition des connaissances et du produit à l'équipe de maintenance. Le cycle de développement d'un logiciel se termine normalement quand ce dernier a été déployé avec succès en production, incluant la garantie, et après la fermeture du projet[10].

[10] Cycle de développement logiciel dans Wikipédia. (2015). [En ligne] Récupéré le 16 novembre 2017 de https://fr.wikipedia.org/wiki/Cycle_de_d%C3%A9veloppement_(logiciel)

Heureusement pour les utilisateurs, le cycle de vie du logiciel ne s'arrête pas là. En théorie, lors du projet, une transition des connaissances du produit est faite à l'équipe de maintenance qui s'assurera, par la suite et en priorité, de la continuité de son intégrité et de sa disponibilité tout en ajoutant de la valeur afin d'évoluer le produit et cela jusqu'à son retrait qui mettra fin à son cycle de vie.

En d'autres mots, la maintenance du logiciel consiste en une série d'activités permettant d'assurer un minimum d'efficacité dans les processus du département afin de maintenir et d'évoluer le logiciel de façon continue, tout dépendamment des besoins et du budget.

La norme ISO/IEC 14764:2006 distingue en détail six différents processus de maintenance logicielle (ISO 2006) dont :

1. L'implémentation;
2. L'analyse et la résolution de problème;
3. La modification du logiciel;
4. L'acceptation de la modification;
5. La migration;
6. La mise à la retraite.

L'implémentation consiste à décrire le plan et les processus du département de la maintenance. Ce processus se fait de façon continue et en parallèle avec les autres processus mentionnés plus haut. Il est unique et s'adapte à chaque département de maintenance tout dépendamment des valeurs et des besoins de la compagnie. Prenons par exemple la figure 2.1 de ce présent rapport qui décrit les processus du département de la maintenance de notre présent projet.

L'analyse et la résolution de problème consistent à reproduire le problème et le besoin du client, et d'analyser la meilleure solution à l'intérieur d'un budget donné. Afin d'y arriver, une analyse des impacts sera faite, laquelle inclura le nouveau changement.

La modification du logiciel consiste en toutes les étapes, dépendamment de la méthodologie du département de la maintenance, afin de modifier et de développer les nouveaux changements du logiciel. Cela comprend la conception, le plan de tests, la programmation, les tests unitaires et la mise à jour de toute documentation.

L'acceptation de la modification consiste à faire les tests d'acceptation décrits dans les plans de tests et de vérifier si les résultats obtenus correspondent à ceux désirés. Obligatoirement, les personnes qui font les tests sont différentes de ceux qui ont travaillé sur la modification. Parfois, c'est un autre département, appelé QA, qui est spécialisé dans les tests et parfois ce sont les clients eux-mêmes ou les utilisateurs. Après avoir terminé, la modification est acceptée ou rejetée par les testeurs.

La migration consiste à migrer le logiciel ou le système dans un nouvel environnement. Une des raisons principales est d'augmenter l'efficacité, la maintenance et la performance du système. Aussi, il arrive souvent que les outils utilisés deviennent obsolètes et ne soient plus supportés par les fournisseurs. Prenons par exemple le système EMS qui devra être migré de Websphere 7 à Websphere 9, car IBM ne supportera plus la version 7 à la fin de l'année 2017.

La mise à la retraite consiste à retirer le logiciel de l'environnement de production. Normalement, il est remplacé par un nouveau système utilisant une technologie différente. Afin d'y arriver, une profonde analyse doit être faite afin d'évaluer les besoins, les coûts et les impacts. Habituellement, le nouveau système est déployé plusieurs semaines en production avant la mise à la retraite du logiciel à retirer. Cela permet aux utilisateurs d'avoir une phase d'adaptation et de tranquillement migrer leurs opérations vers le nouveau système.

La norme ISO/IEC 14764:2006 décrit en détail et beaucoup plus en profondeur le mécanisme des différents processus. La norme discute des intrants, des tâches et des sortants pour chacun eux. Il est possible de l'utiliser afin d'établir et d'améliorer les processus des départements de la maintenance. Il est aussi à noter que la documentation est primordiale dans la norme et que la plupart des sortants incluent plusieurs mises à jour de la documentation.

3.2. Les types de maintenance du logiciel

On retrouve généralement quatre types de maintenance du logiciel dont la corrective, l'adaptative, la perfective et l'évolutive. E.B. Swanson a identifié les trois premiers tandis que la norme ISO/IEC 14764:2006 a ajouté la dernière[11].

1. Maintenance corrective : modification du logiciel ou du système pour régler un défaut afin d'éviter de reproduire une erreur donnée. Prenons par exemple un système qui montre un équipement ferroviaire avec un certain statut tandis que la réalité est autre. Il est possible que le système ne gère pas bien certains scénarios en production qui n'ont pas été pensés lors de la conception du système.

2. Maintenance adaptative : modification du logiciel ou du système afin qu'il demeure exact et utilisable lors d'un changement dans son environnement externe. Prenons par exemple un logiciel de paiement qui doit être adapté lors d'un changement de la TVQ ou de la TPS. Prenons aussi par exemple le changement d'image d'une compagnie. Il est fort probable que les couleurs et les logos des logiciels de la compagnie en question devront être modifiés. En général, les logiciels sont toujours en évolution afin d'augmenter les fonctionnalités ou d'augmenter l'efficacité des processus et des opérations des clients et des utilisateurs.

3. Maintenance perfective : modification du logiciel afin d'augmenter son efficacité et sa maintenabilité. Prenons par exemple un logiciel qui a un problème de performance après un certain ajout de nouvelles fonctionnalités. Il est fort possible que le client demande une maintenance perfective afin d'éliminer les lacunes de performance.

---

[11] Maintenance du logiciel dans Wikipédia. (2015). [En ligne]. [En ligne] Récupéré le 16 novembre 2017 de <https://fr.wikipedia.org/wiki/Maintenance_du_logiciel>

4. Maintenance préventive : modification du logiciel afin de détecter en amont les défauts avant qu'ils se manifestent et qu'ils créent des erreurs. Prenons par exemple un logiciel qui envoie un message à l'équipe de maintenance l'informant qu'il n'a pas reçu d'information d'un autre système quelconque depuis un certain temps donné. Le fait que l'équipe de maintenance puisse vérifier la situation avant qu'une erreur se produise ajoute beaucoup valeur au système.

Dans l'équipe de Maintenance Java du CN, les listes de maintenance des applications contiennent principalement les éléments de maintenance adaptative sans toute fois négliger la maintenance préventive et perfective. Évidemment, la maintenance corrective se fait aussi, mais elle est plus rare.

3.3. Les normes et modèles de la maintenance du logiciel

Dans la littérature, on trouve principalement deux publications majeures relatives aux normes et modèles de bonnes pratiques de maintenance du logiciel : la norme ISO/IEC 14764 sur le cycle de vie de la maintenance du logiciel et le modèle de maturité spécifique à la maintenance du logiciel (April et Abran 2016).

3.3.1. ISO/IEC 14764, Software Engineering -- Software Life Cycle Processes – Maintenance:

ISO/IEC 14764 est une norme internationale décrivant les bonnes pratiques de la maintenance dans le domaine de l'ingénierie logicielle. La norme commence à expliquer différents termes et définitions comme les types de maintenances qui sont mentionnés plus haut. Ces mêmes types de maintenance y sont aussi très bien expliqués un peu plus loin dans la norme. Ensuite, comme mentionnée précédemment, la norme explique plus en profondeur les six différents processus qui doivent se retrouver, idéalement, dans un département de maintenance du logiciel.

Chacun des processus possède des entrants, des tâches, des contrôles, du support et finalement des sortants. Il est possible d'avoir plus de détails sur ce sujet au début de ce présent chapitre.

La norme mentionne aussi les différentes possibilités de fonctionnement entre le client et l'équipe de maintenance. Est-ce une équipe à l'interne, une firme externe ou l'entreprise ayant conçu le produit qui va s'occuper de la maintenance du produit ? Doit-il y avoir du transfert de connaissance chez le client afin de maintenir le produit ? À quelle fréquence et à quelle période peut-on livrer une nouvelle version en production ? Dans le cas d'un sinistre, après combien de temps le produit doit être opérationnel? Il est facile de constater, ici, qu'il y a plusieurs choses à penser quand le temps est venu de donner le produit à une équipe de maintenance du logiciel.

La norme discute aussi des outils nécessaires afin de livrer un produit de qualité comme les environnements de tests et d'autres guides de développement et de maintenance comme la norme ISO/IEC TR 14471 (ISO 2017). La documentation doit être primordiale afin que tous les membres de l'équipe aient l'accessibilité à l'information et plusieurs mesures quantitatives, comme la vélocité, doivent être prises afin de faire de l'amélioration continue et de mieux estimer les demandes de changements. La norme ISO/IEC 15939 (ISO 2006) discute plus en détail des processus de prise de mesures. La norme ISO/IEC 14764 suggère et explique la possibilité d'introduire un membre de l'équipe de maintenance du logiciel dès le début du projet afin d'avoir une meilleure connaissance du domaine d'affaires et du produit, que ça soit au niveau de la conception ou au niveau plus technique. La norme ISO/IEC 12207 (ISO 2008) discute de ce sujet plus en profondeur. Une longue section est dédiée à la maintenabilité du produit, car il y a une forte corrélation entre la maintenabilité et la maintenance logicielle. La norme résume à haut niveau les différents éléments à considérer afin d'augmenter la maintenabilité du produit. Prenons comme exemple la portabilité du produit, l'infrastructure choisie, la structure des données, le code et les choix de nomenclature, la qualité des tests unitaires, etc. La norme discute aussi de la transition qui passe de l'équipe de projet à l'équipe de maintenance du logiciel. En théorie, un plan de transition doit être documenté afin de

faire que le transfert de connaissance soit un succès. Si la documentation n'est guère suffisante, la norme suggère à l'équipe de maintenance du logiciel de la compléter pendant la phase de transition.

La dernière partie de la norme ISO/IEC 14764 discute des stratégies de maintenance du logiciel. Cette section discute à haut niveau les différentes stratégies à planifier dont le concept de maintenance, le plan de maintenance et l'analyse des ressources. Le concept de maintenance est documenté dans le plan de maintenance et doit être travaillé dès le début du projet. Cette section traite des sujets à discuter et non des solutions proposées, car elles dépendent des besoins de l'entreprise et du client. Même si cette section est très intéressante, les détails ne seront pas discutés dans ce présent rapport, car ils s'appliquent un peu moins à notre projet, mais un peu plus à une entreprise qui doit créer une nouvelle équipe de maintenance du logiciel.

3.3.2. Software Maintenance Maturity Model (S3M):

Le modèle de maturité de la maintenance du logiciel S3M est un compendium des meilleures pratiques liées à la maintenance du logiciel (April et Abran 2005) et principalement utilisé comme guide d'amélioration des processus de maintenance logiciels des organisations (Counet 2006) (April et Abran 2005). Le modèle S3M se base sur le CMMI (Software Engineering Institute) bien que ce dernier ne répondait pas aux besoins de la petite maintenance.

Le modèle S3M n'est pas un processus ou une méthode en soi, mais plutôt un modèle de processus qui doit s'adapter en fonction des besoins d'affaires de l'organisation.

Le modèle S3M comprend six niveaux dont :

- Niveau 0 – *Unstructured* : Comme le décrit le niveau 0, il s'applique aux organisations qui appliquent peu ou pas de notions du modèle et qui, fort probablement, n'en obtiennent pas les avantages.

- Niveau 1 – *Executed* : Le niveau 1 s'applique aux organisations qui sont conscientes des activités de maintenance préconisées par le modèle, mais qui non aucun processus d'établi et ni documenté. L'organisation dépend de l'expertise d'une ou plusieurs personnes qui font les activités d'une façon aléatoire et non systématique.

- Niveau 2 – *Managed* : Le niveau 2 s'applique aux organisations qui mettent en œuvre les activités de maintenance préconisées par le modèle, notamment de décrire un processus pour un département en particulier et non pour l'ensemble de l'organisation. En bref, il n'est pas uniformisé à travers toute la compagnie.

- Niveau 3 – *Established* : Le niveau 3 s'applique aux organisations qui répondent au niveau 2, plus les caractéristiques suivantes (Arnaud Counet, 2007) :

  - *La pratique exemplaire suggérée par le modèle est exécutée ;*
  - *Les procédures utilisées sont les mêmes dans toutes les unités de l'entreprise ayant une même fonction ;*
  - *Un ensemble limité de mesures est développé, collecté, validé et utilisé ;*
  - *Les employés savent comment exécuter un processus ;*
  - *Le temps et les ressources nécessaires sont alloués afin d'atteindre les buts identifiés ;*
  - *Des collections de techniques, de modèles et d'informations sont mises en place ;*
  - *Le processus est constamment utilisé par le personnel ;*
  - *Les caractéristiques du processus et les activités clés sont mesurées.*

- Niveau 4 – *Predictable* : Le niveau 4 s'applique aux organisations qui mettent en évidence des pratiques exemplaires de façon formelle, ayant un but défini et leurs propres limites.

- Niveau 5 – Dynamique : Le niveau 5 s'applique aux organisations qui suivent le niveau 4 et qui mettent en place un contrôle statistique. La caractéristique la plus

importante de ce niveau est l'accent sur l'amélioration continue des processus et cela de façon constante.

3.4. Approche agile émergente de gestion de la maintenance du logiciel

Depuis 2001, une nouvelle approche de développement logiciel a émergé : l'approche agile, définie par quatre valeurs – dont les individus et leurs interactions, du logiciel fonctionnel, la collaboration avec le client et l'acceptation du changement – et douze principes qui sont décrits plus bas (Beck et al. 2001). Cette approche agile est mise en œuvre par un certain nombre de méthodes, dont les plus connues sont *eXtreme Programming* (XP) (Beck 2000) et *Scrum* (Sutherland, Schwaber et Beedle 2001).

En résumé, l'approche agile est caractérisée par des cycles de développement itératifs, incrémentaux et adaptatifs jusqu'à la fin du projet. Il n'y a pas de règle générale sur la durée d'un cycle, mais il est suggéré qu'elle soit constante. Voici les douze principes en question de l'approche agile (Manifeste Agile 2001) :

- *Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.*
- *Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.*
- *Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.*
- *Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.*
- *Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.*

- *La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.*

- *Un logiciel opérationnel est la principale mesure d'avancement.*

- *Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.*

- *Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.*

- *La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.*

- *Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.*

- *À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.*

Le livre *Choisir l'agilité* (Boisvert et Trudel 2011) explique beaucoup plus en détail l'approche agile. Il cible surtout les personnes travaillant dans des projets en technologie de l'information et génie logiciel et qui ont, en général, des problèmes de délais de livraison, de suivi des échéanciers et des tâches, de fonctionnalités inadéquates, et ayant aussi un budget serré.

Les expériences démontrent que la majorité des projets utilisant l'approche agile résolvent les différentes problématiques mentionnées dans le chapitre précédent. Aussi, les projets utilisant l'approche agile ont démontré qu'il était plus facile de prendre des mesures, dont la vélocité, et d'être constant entre les différents cycles (Boisvert et Trudel 2011).

3.4.1. eXtreme Programming

En résumé, la méthode *eXtreme Programming* (XP) implémente les valeurs et principes de l'approche agile à travers treize différentes pratiques.

La première est qu'il est primordial que le client passe beaucoup de temps avec l'équipe de développement afin de confirmer plusieurs détails et de s'assurer d'être toujours sur le droit chemin.

La deuxième est que le client crée les différents scénarios désirés afin que l'équipe de développement évalue le temps nécessaire pour ainsi permettre de les prioriser.

La troisième pratique est l'intégration continue. Cette pratique permet d'intégrer la nouvelle fonctionnalité au produit aussi tôt celle-ci terminée et d'en faire les tests d'acceptation. Une fois tous les tests d'acceptation approuvés, la nouvelle fonctionnalité est considérée comme intégrée au produit.

La quatrième pratique est la livraison avec un minimum de fonctionnalités, mais livrer souvent, afin d'éviter les problèmes d'intégration. Cela a pour effet de réduire grandement les coûts de livraison, d'une part parce qu'ils sont étalés, mais surtout parce que les problèmes sont traités à mesure.

La cinquième pratique est le fait d'éviter les heures supplémentaires. Le concept étant qu'un employé fatigué est un employé non productif et pouvant facilement insérer des défauts dans le produit.

La sixième pratique est l'implémentation des tests fonctionnels. Avant chaque itération, l'équipe de développement crée les tests fonctionnels avant même de commencer à développer. Quand tous les tests fonctionnent et sont approuvés, la phase de développement de l'itération courante est considérée terminée.

La septième pratique est l'implémentation des tests unitaires la pratique du *Test Driven Development* (TDD), ou développement piloté par les tests. L'idée est de créer le code des tests unitaires avant même de développer le code de la nouvelle

fonctionnalité. Ainsi, il est facile de constater dans le futur si l'intégration d'une nouvelle fonctionnalité causera des défauts ailleurs.

Afin d'éviter une intégration difficile, la huitième pratique met l'accent sur les choix de conception les plus simples. Si la conception devient trop complexe, c'est probablement parce que l'équipe est dans l'erreur.

La neuvième pratique est la convention des métaphores. Il est essentiel que toutes les parties prenantes du projet utilisent le même vocabulaire afin d'éviter des ambiguïtés et la perte de productivité.

La dixième pratique est le remaniement du code afin de toujours l'améliorer, notamment en terme de maintenabilité. Cette pratique nous mène à la onzième pratique qui est la propriété collective du code. Le code n'appartient à aucun membre de l'équipe en particulier[12] et tous les développeurs sont encouragés à faire de l'amélioration au besoin sans toutefois changer les fonctionnalités.

La douzième pratique est la convention de nommage afin d'avoir une norme sur la façon d'écrire le code et de nommer les variables, méthodes, objets, fichiers, etc.

Finalement et non le moindre, la treizième pratique est la programmation en binôme. Le fait d'être deux développeurs devant l'écran diminue le risque d'insertion de défauts, favorise les discussions quant aux choix des algorithmes et donne une forte motivation afin d'être productif et présent tout en assurant une revue en continu par les pairs.

Le livre *Extreme Programming Explained* (Beck et Andres 2004) explique des pratiques d'ingénierie utilisant quelques notions de l'approche agile et de la méthode Scrum afin de livrer un produit de qualité en production, par incréments successifs. La

[12] En réalité, le code appartient au client, soit celui qui paie pour le développement. Cette pratique a été incluse dans la méthode XP pour éviter les cas où un développeur aurait un trop fort sentiment d'appartenance de son code ... et les comportements indésirés qui en découlent.

méthodologie met aussi beaucoup l'accent sur les pratiques du TDD et de la programmation en binôme (*pair programming*).

3.4.2. Scrum

En résumé, Scrum est un cadre méthodologique se basant sur l'approche agile qui a comme objectif de livrer un produit final découpé en des intervalles de temps appelés sprints. En général, les sprints n'ont pas de durée prédéfinie d'un projet à l'autre. La durée des sprints est définie au début de chaque projet et doit être constante jusqu'à la fin du projet, à moins de nécessiter un ajustement sans équivoque. Il est important de mentionner que les activités du cycle de développement du logiciel ne sont pas dans la portée de Scrum et plusieurs autres approches ou méthodes peuvent être couplées avec elle, comme XP. En cohérence avec l'approche agile, la méthode Scrum se base sur trois piliers, dont la transparence, l'inspection et l'adaptation. La transparence permet d'avoir un langage commun pour toutes les parties prenantes afin de faciliter la communication. De façon régulière, l'inspection permet à l'équipe de s'assurer d'être sur le droit chemin. Cela permet d'éviter de dériver rapidement ce qui causerait du retard et des coûts au projet de façon inutile. Finalement, ce qui nous amène à l'adaptation. Scrum se permet d'être adaptatif afin de s'ajuster tout dépendamment de l'expérience vécue sur le terrain. En d'autres mots, on dit qu'il est empirique.

Les sprints sont incrémentaux et immuables. En d'autres mots, chaque sprint possède son propre carnet qui contiendra un certain nombre d'éléments à développer, tout dépendamment de la vélocité de l'équipe, et sur lesquels l'équipe s'est engagée en début d'itération pendant la réunion de planification. Une fois fait, le carnet de sprint est verrouillé et aucun ajout n'est fait à moins d'un cas extrême. N'oublions pas encore une fois que Scrum est adaptatif. Le fait de faire des insertions dans le carnet de sprint pourrait perturber l'itération en cours et rendre l'engagement caduc, car la durée du cycle est déjà prédéfinie d'avance. Probablement, il y aura un manque de temps au bout du compte afin de compléter tous les éléments en cours d'itération. Afin d'arriver

au bout du sprint avec tous les éléments complétés, une mêlée quotidienne de moins de 15 minutes est faite et cela à chaque jour. L'objectif est simplement de se synchroniser en faisant un tour de table pour discuter de l'état d'avancement et des problèmes en cours. L'objectif n'est pas de résoudre les problèmes en cours, mais uniquement de les rendre visibles pour qu'ils puissent être traités par la suite. La discussion pourra continuer après la mêlée avec les parties concernées. Comme mentionné, chaque sprint contient un carnet, mais le projet aussi en possède un : le carnet de produit. Le carnet de produit contient tous les éléments à développer du client. Ces éléments doivent aussi être priorisés par le client afin de faciliter la prise de décision pour les prochains sprints. Il est à noter, aussi, que le carnet de produit peut contenir les défauts à corriger et les améliorations techniques à implémenter, comme la performance.

Au début du projet, il y a un sprint 0 (zéro) représentant la phase de démarrage et qui permet à l'équipe et au client de définir la vision et les objectifs du projet, ainsi que le carnet de produit initial. Par la suite, chaque cycle est incrémenté et consiste en une réunion de planification, des activités de développement entrecoupées de mêlées quotidiennes, une revue de sprint et finalement une rétrospective. À chaque sprint, l'équipe assure la transparence de l'avancement des travaux en rendant disponibles les données de progression et en calculant sa vélocité.

Comme mentionné, la réunion de planification permet de définir les tâches associées aux éléments à développer pendant le sprint courant. La revue du sprint permet de valider que les éléments développés sont complétés. L'ensemble de tous les éléments complétés depuis le début du projet constitue l'incrément du produit. Le calcul de vélocité est simplement le nombre point d'efforts (*user story points*) complétés dans le sprint par l'équipe. Ainsi, le projet possède un calcul de vélocité moyenne dans son ensemble, ce qui permet de mieux estimer les prochaines itérations. La transparence sur l'avancement des travaux est souvent illustrée avec le diagramme de burndown montrant la quantité de travail restant sur une période de temps donnée. Cela permet de mieux visualiser l'état d'avancement du sprint et de s'ajuster en conséquence. Finalement, la rétrospective consiste à examiner les façons de faire de l'équipe dans

un but de s'améliorer. Typiquement, l'équipe fera un résumé des points forts et des points à améliorer du dernier sprint afin de s'ajuster en conséquence pour le sprint suivant.

Différents rôles s'appliquent avec Scrum, dont le propriétaire du produit (appelé *Product Owner* ou PO), le Scrum Master (SM) et les membres d'équipe. Le PO, souvent le client lui-même ou son représentant, est responsable de décrire et d'analyser les besoins et les demandes. Il est responsable du carnet de produit et de la priorisation par valeur d'affaires des éléments qu'il contient. Il travaillera de façon proactive et rétroactive avec l'équipe de développement afin de s'assurer d'être toujours sur le droit chemin. L'équipe de développement, constituée de 3 à 9 personnes, est évidemment une équipe multidisciplinaire ayant toutes les compétences pour réussir le projet, dont probablement des analystes fonctionnels, des développeurs logiciels, des administrateurs de bases de données, etc. Finalement, la méthode Scrum n'inclut pas un gestionnaire de projet, mais bien un Scrum Master ou « maître de mêlée ». Le SM s'assure du bon déroulement des pratiques Scrum en tant que facilitateur d'une équipe auto-organisée. Cela permet à l'équipe de s'assurer d'un bon déroulement et des bonnes pratiques afin de profiter à 100 % des avantages de cette méthode. L'image qui suit montre la méthode Scrum et ses pratiques.

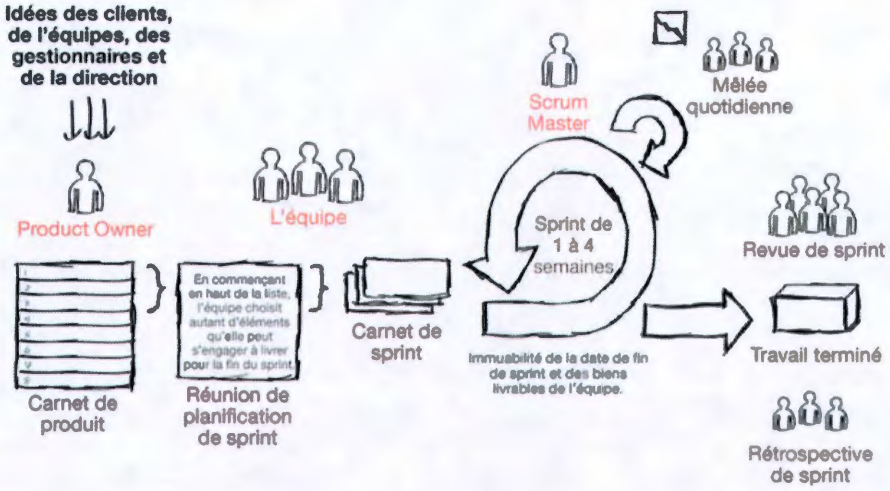


Figure 3.1 - Scrum et ses pratiques[13]

Bien que ces méthodes (XP et Scrum) soient surtout appliquées en développement logiciel, elles peuvent aussi être appliquées à des projets de maintenance sous certaines conditions. D'abord, il doit y avoir du travail d'équipe dont la taille serait de 3 à 9 personnes (Sutherland et Schwaber 2013), le travail doit pouvoir être découpé et réalisé en itérations de durée fixe et l'équipe doit être en mesure de prendre un engagement en début d'itération pour livrer un sous-ensemble d'éléments de son carnet de produit pour la fin de l'itération. Or, en maintenance du logiciel, ces conditions sont souvent absentes. Bien que les valeurs et les principes agiles puissent s'appliquer en maintenance, ces méthodes populaires le sont moins.

Dans le cas où une organisation a besoin d'une méthode dont le cycle incrémental est de durée variable, où l'engament à livrer un contenu déterminé à une date déterminée ne peut être pris parce qu'un nombre d'incidents de la production vient perturber les travaux en cours et où les travaux sont davantage faits individuellement plutôt qu'en

[13] Image tirée de AIM Services. (2017). Tous droits réservés. [Image en ligne] <http://www.aim-services.ch/blog/>

équipe, alors elle se tournera vers une autre méthode : Kanban (Kniberg et Skarin 2009).

Pour les lecteurs qui désirent en savoir davantage sur la méthode Scrum, le livre SCRUM (Aubry 2013) cible les personnes travaillant dans des projets en technologie de l'information et génie logiciel qui utilisent l'approche agile et qui aimeraient augmenter leur efficacité en utilisant la méthode Scrum.

3.4.3. Kanban

En résumé, Kanban signifie « panneau » ou « tableau » en japonais et a permis, dans les années 1950, à la compagnie Toyota d'avoir une meilleure méthode de gestion de la production. Cette méthode permettait de limiter les stocks d'une ligne de production en amont pour mieux répondre aux besoins en aval. L'objectif général de cette méthode est d'avoir une amélioration continue des processus en cours afin d'augmenter la fluidité tout en élimant les goulots et en limitant les stocks à une certaine limite donnée. Cette méthode s'est avérée particulièrement utile pour la maintenance du logiciel.

Le tableau Kanban pour IT inclut des colonnes représentant chacune des étapes du processus du département de la maintenance afin de livrer une demande de changement en production (c'est-à-dire tout élément de maintenance requis pour un système donné). Il inclut aussi des cartons représentant les demandes de changements et les urgences. À chaque fois qu'une demande de changement ou une urgence passe à la prochaine étape dans le processus en question, le carton se déplace aussi en conséquence. Comme mentionné plus haut, l'objectif est d'adopter une approche agile tout en limitant les travaux en cours par étape du processus et non par itération. L'avantage de Kanban par rapport à Scrum est qu'il est beaucoup plus flexible aux urgences. Il permet aussi de mieux visualiser les états d'avancement des demandes de changements et de voir les goulots. Une règle dit, en maintenance logicielle, que chaque colonne devrait se limiter au nombre de cartons au maximum à

1,5 fois le nombre de personnes aptes à faire cette étape (Morisseau 2014). Par exemple, pour une équipe de six développeurs, la colonne de développement pourrait avoir une limite maximale de neuf cartons. Il n'y a pas de règle pour la limite minimale. C'est au département à analyser et ajuster ses limites aux besoins. L'image qui suit montre la méthode Kanban et ses pratiques.

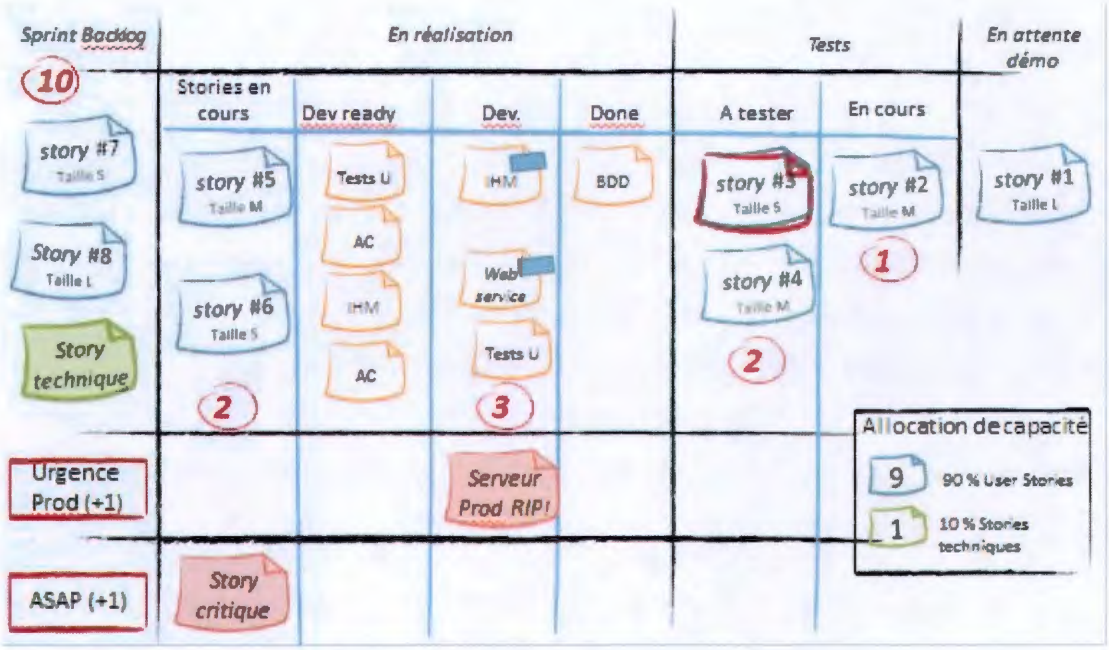


Figure 03.2 - Exemple appliqué de la méthode Kanban[14]

Le livre Kanban pour IT (Morisseau 2014) cible les personnes travaillant dans les technologies de l'information et génie logiciel, mais surtout en mode maintenance plutôt qu'en mode projet et qui ont, en général, des problèmes de délai de livraison, de suivi des échéanciers et des tâches, de fonctionnalités, et ayant, aussi, un budget serré.

[14] Image tirée de Morisseau Consulting. (2017). Tous droits réservés. [Image en ligne] <http://www.morisseauconsulting.com>

3.4.4. Srumban

La méthode Kanban peut aussi être couplée avec celle de Scrum afin de tirer au maximum les avantages des deux. Cette méthode qui devient de plus en plus populaire s'appelle Scrumban. Prenons par exemple la vélocité, ce qui permet de mieux estimer les nouvelles demandes de changements, les rencontres quotidiennes ou hebdomadaires afin de mieux comprendre les états d'avancement et les problèmes en cours et finalement, la planification afin de bien comprendre les demandes de changements et les revues du produit afin de mieux comprendre l'implémentation au niveau technique. Chacune des deux méthodes possède des forces et des limites. Le fait de les unifier permet d'augmenter les chances de succès dans le monde de la maintenance logicielle.

Ainsi, il est facile des constater que l'état de l'art sur la maintenance du logiciel nous converge vers l'approche agile et Scrumban. La raison principale est le fait que cette solution couvre à 100 % les problématiques discutées dans le chapitre précédent.

Le chapitre suivant discutera de la méthodologie utilisée afin d'analyser et d'implanter l'approche agile avec le couplage des méthodes Scrum et Kanban afin de résoudre les problématiques en cours.

Chapitre 4. Méthodologie

Après avoir fait plusieurs lectures sur les solutions abordées dans le précédent chapitre, j'ai développé une méthodologie afin d'implémenter la meilleure solution à notre problématique. Ce présent chapitre explique cette méthodologie utilisée pour permettre la réussite de notre projet.

4.1. But et objectifs

Le but de ce projet est bien sûr d'analyser et de résoudre la problématique discutée au chapitre 2. En ce qui concerne les objectifs, le tableau suivant reprend les problèmes et donne un objectif spécifique à chacun d'eux.

Tableau 4.1 - Liste des problèmes

Problèmes	Objectifs
Absence de vue d'ensemble	L'équipe doit être en mesure d'obtenir l'information des demandes de changement dans un seul endroit commun.
Priorisation segmentée	Toutes les demandes de changement de toutes les applications doivent être priorisées en un seul endroit commun.
État d'avancement inconnu	L'équipe doit être en mesure d'obtenir l'état d'avancement à jour de chacune des demandes de changement de toutes les applications.
Estimation impossible	L'équipe doit être en mesure de donner des estimations pour chacune des demandes de changement.
Informations des solutions non communiquées	L'équipe doit être en mesure de faire du support sur tous les changements et nouveautés technologiques déployés en production.

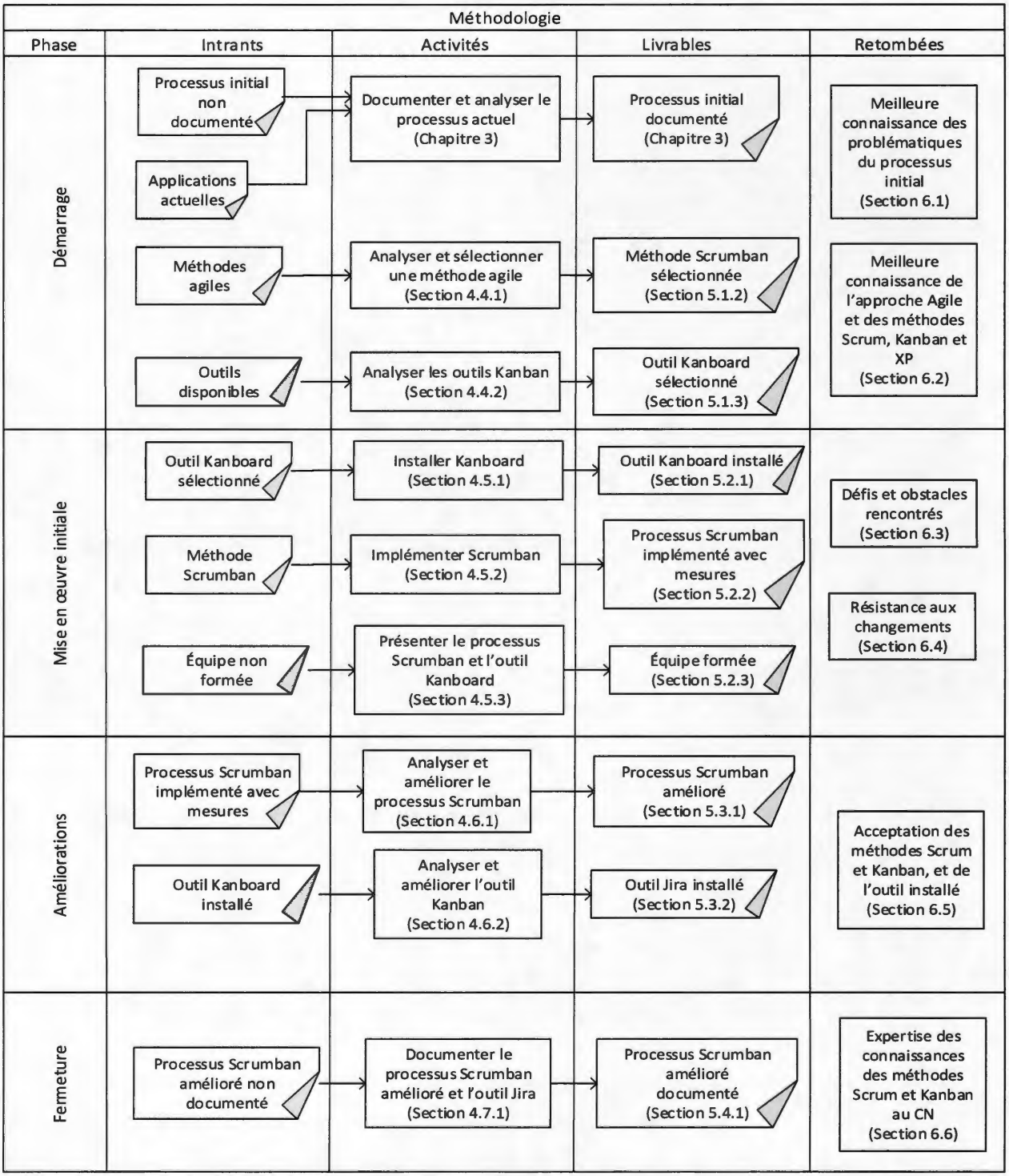
4.2. Portée

La portée du projet touche seulement l'équipe de la maintenance des applications Java sous la division des opérations et quelques gens d'affaires qui y sont reliés. Dans le cas où l'innovation prendrait de l'ampleur dans la compagnie, un transfert de connaissance pourra se faire aux différents départements ayant les mêmes besoins. En résumé, ça pourrait toucher, à plus long terme, toute la division de la maintenance des systèmes informatiques chez Canadien National.

4.3. Aperçu de la méthodologie

Le prochain diagramme montre la méthodologie utilisée afin de résoudre les problématiques mentionnées plus haut. Le texte qui suit explique plus en détail chaque activité avec ses intrants et ses livrables.

Tableau 4.2 - Méthodologie



4.4. Démarrage

4.4.1. Analyser et sélectionner une méthode agile

L'équipe de maintenance des applications Java du côté des opérations chez Canadien National ont entendu parler de Scrum et de ses bienfaits lors des projets. Ainsi, le gestionnaire de l'équipe se demandait s'il était possible d'implémenter Scrum avec nos processus.

Pour faire d'une pierre deux coups, j'ai eu la chance de travailler pendant plus d'un an avec le projet FMLM chez Canadien National afin de bien comprendre le domaine d'affaires, la conception du nouveau système et ses technologies, et surtout la méthodologie utilisée dont l'agilité avec Scrum comme cadre méthodologique.

L'objectif principal était d'obtenir l'expertise du nouveau système FMLM. Spécialement les applications CarShipment et CarShipment Service, car elles sont maintenant sous la responsabilité de l'équipe de maintenance. Aussi, il était pertinent de se familiariser avec l'approche agile et le cadre Scrum afin d'évaluer sa pertinence dans le monde de la maintenance.

Évidemment, l'évaluation de l'approche et du cadre ne s'est pas arrêtée là. J'ai fait aussi plusieurs lectures sur le domaine en question afin de comparer les avantages et les inconvénients entre les cadres Scrum, Kanban et Scrumban. En accord avec l'équipe, j'ai sélectionné la méthode Scrumban.

4.4.2. Analyser et sélectionner un outil Kanban

Puisque la solution sélectionnée afin d'éliminer les problématiques mentionnées plus haut est Scrumban, l'analyse et la sélection d'un outil Kanban est nécessaire. Les premiers critères de sélection sont simplement le fait qu'il soit gratuit ou libre, qu'il soit facile à utiliser, qu'il soit sécuritaire et qu'il fonctionne avec des privilèges, qu'il puisse

être installé de façon simple sur une machine à l'interne et surtout qu'il soit convivial pour les usagers. J'ai analysé plusieurs outils et Kanboard a été sélectionné, car il répondait à tous mes critères de sélection.

4.5. Mise en œuvre initiale

L'objectif de la mise en œuvre initiale est d'implémenter le nouveau processus avec le nouvel outil Kanban afin d'en faire l'essai. En d'autres mots, c'est un peu comme un prototype à l'essai où l'équipe se mouille le pied avant de sauter dans la piscine.

4.5.1. Installer Kanboard

Cette présente activité comprend l'installation temporaire de l'outil Kanboard sur mon ordinateur avec un serveur PHP qui a déjà été installé dans le passé. Le serveur PHP permet de faire fonctionner des applications web qui ont été développées avec la technologie PHP.

Le prochain chapitre discutera, aussi, un peu plus des résultats avec quelques images de l'outil installé.

4.5.2. Implémenter Scrumban

Après avoir lu et analysé Scrumban, j'ai fait une analyse afin de bien créer le tableau Kanban avec des mesures pour chacune des colonnes qui correspondent à chacune des étapes du processus du département de la maintenance.

Par la suite, j'ai fait une analyse afin d'intégrer dans le processus courant des réunions de mêlée et de transfert de connaissances.

Le prochain chapitre discutera un peu plus des résultats du processus Scrumban implémenté avec mesures.

4.5.3. Présenter le processus Scrumban et l'outil Kanban

Cette activité m'a permis de présenter aux membres de l'équipe de la maintenance la méthodologie Scrumban et de ses bienfaits dans le domaine de la maintenance logicielle. Pour ce faire, deux réunions ont été nécessaires. La première consistait à la méthodologie Scrumban en question et la deuxième consistait au nouveau processus implémenté avec ses mesures et l'outil Kanboard. Ayant maintenant une équipe formée sur le sujet, il a été facile d'obtenir du feedback et cela très rapidement.

Le prochain chapitre discutera, aussi, des feedbacks des membres de l'équipe de la maintenance.

4.6. Améliorations

La phase d'amélioration permet d'analyser et d'améliorer le nouveau processus Scrumban implémenté dans la phase précédente. Cette phase a été pratiquée par tous les membres de l'équipe.

4.6.1. Analyser et améliorer le processus Scrumban

Le seul moyen d'analyser et d'améliorer le processus Srumban est de l'utiliser au quotidien. Avec le temps, quelques ajustements ont été nécessaires afin d'augmenter la fluidité et la vélocité pour ainsi donner comme extrants un processus amélioré. Il est à noter que cette activité s'est faite de façon continue afin de toujours bien s'ajuster et que tous les membres de l'équipe de maintenance y ont participé afin de maximiser les valeurs ajoutées.

Tout d'abord, il y a eu une petite courbe d'ajustement et un peu de discipline pour mettre à jour l'information dans l'outil Kanban. Cependant, les membres le l'équipe ont vraiment adoré le nouveau processus, surtout les réunions hebdomadaires. Ils se sont rendu compte assez vite des avantages d'être au courant de ce qui se passe dans le département.

4.6.2. Analyser et améliorer l'outil Kanban

Comme le processus Scrumban, l'outil Kanban doit être utilisé et analysé afin d'être amélioré. Avec le temps, de nouveaux critères ont été sélectionnés. La recherche et l'analyse du nouvel outil ont été transférées dans un autre département responsable de la normalisation des outils chez Canadien National. Après plusieurs semaines, ils nous sont revenus avec comme extrant l'outil Jira.

La section des résultats donne plus de détails sur les raisons des changements et des améliorations apportées.

4.7. Fermeture

La phase de fermeture est simplement celle où les dernières activités sont faites afin de fermer mon projet. Dans ce cas-ci, il s'agissait de seulement compléter la documentation sur le processus amélioré.

4.7.1. Documenter le processus Scrumban

Cette activité m'a permis de documenter le processus Scrumban dans le département de la maintenance, tout en incluant l'outil Jira. Cela me permettra de faire du transfert de connaissances aux nouveaux employés et aux autres départements de maintenance chez Canadien National.

Chapitre 5. Résultats

Cette section explique en détail les résultats et les livrables issus des activités expliquées dans le chapitre précédent.

5.1. Démarrage

5.1.1. Processus actuel documenté

Voir le chapitre 3 Contexte et problématique.

5.1.2. Méthode Scrumban sélectionnée

L'analyse des différentes méthodologies m'a fait converger vers la méthode Scrumban pour plusieurs raisons. Tout d'abord, même si la méthodologie Scrum apportait beaucoup d'avantages lors de son utilisation, le projet FMLM a démontré que la méthodologie Scrum avait ses limites concernant les urgences. Puisque les carnets de sprint étaient immuables lors des différentes itérations, les usagers devaient attendre jusqu'à la livraison de la prochaine itération avant de voir les erreurs et les défauts se régler en production. Dans le domaine de la maintenance, ça ne serait aucunement acceptable.

Après mon analyse de la méthodologie Kanban et quelques études de cas, il est facile de constater que son utilisation dans le domaine de la maintenance semble très intéressante. En résumé, la méthodologie Kanban permet de régler la majorité des problématiques mentionnées dans ce rapport. Cependant, elle n'est pas complète.

Prenons par exemple le transfert de connaissances et les réunions hebdomadaires. Afin d'y arriver, il fallait y intégrer quelques notions de la méthodologie Scrum. Heureusement, comme mentionné dans les chapitres précédents, cette intégration existe depuis quelques années seulement et s'appelle Scrumban.

En bref, la méthode Scrumban résout toutes les problématiques discutées dans ce présent rapport. C'est pour cette raison qu'elle a été sélectionnée.

5.1.3. Outil Kanboard sélectionné

L'outil Kanboard[15] pour Kanban a d'abord été sélectionné, car il est simple et gratuit. Il est facile à utiliser et les fonctionnalités sont complètes. En d'autres mots, c'est un outil parfait pour un projet pilote. Il est à noter qu'il y a plusieurs excellents outils qui simulent Kanban, dont par exemple Trello[16]. Trello aurait été aussi un excellent choix. La grande différence entre les deux c'est qu'il faut installer Kanboard sur un serveur PHP tandis que Trello est utilisé en ligne. Chez Canadien National, il est fortement recommandé d'avoir nos outils installés à l'interne afin d'éviter des problèmes de sécurité, dont la confidentialité, la disponibilité et l'intégrité.

5.2. Mise en œuvre initiale

5.2.1. Outil Kanban installé

L'installation de l'outil Kanboard a été quand même facile à faire. Les étapes sont bien expliquées sur le site web de Kanboard. Cependant, mon choix des colonnes pour représenter les étapes du processus a été un peu plus difficile. À première vue, ça semble facile, mais ce ne l'est pas. Plusieurs ajustements ont été faits tout au long du

[15] KB Kanboard. [En ligne] <http://kanboard.net>
[16] Trello. [En ligne] <https://trello.com>

projet afin de détailler ou simplifier le processus en question. L'outil Kanboard est aussi très facile à comprendre et en même temps facile d'utilisation. La figure suivante montre un exemple de l'outil Kanboard avec les différentes étapes du processus pour les requêtes de changements adaptatifs et correctifs.

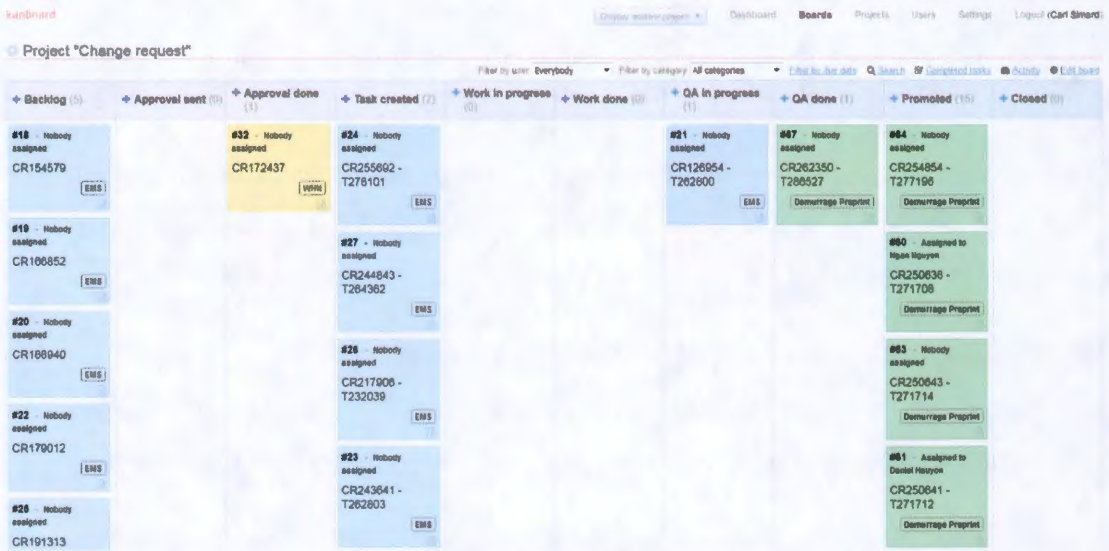


Figure 5.1 – Exemple de l'interface de l'outil Kanboard

Voici les différentes colonnes établies dans le tableau Kanban :

1. Backlog
2. Approval sent
3. Approval done
4. Task created
5. Work in progress
6. Work done
7. QA in progress
8. QA done
9. Promoted
10. Closed

Une fois les colonnes créées, il suffit simplement de créer des cartes et de les gérer tout le long du cycle de développement du nouveau changement. Il est possible de

lire la documentation sur le site web de Kanboard afin de bien comprendre l'utilisation de l'outil.

La section suivante explique chaque étape mentionnée ci-dessus, les limites et les données prises.

5.2.2. Processus Scrumban implémenté avec mesures

Cette section discute du nouveau processus avec l'implantation de Scrumban. Ça ressemble beaucoup à celui mentionné à la section 3.4.4, mais avec quelques particularités essentielles.

Tout d'abord, il y a une notion de limites inférieures et supérieures avec Scrumban. Elles seront expliquées pour chacune des étapes. Aussi, il y a une notion de vélocité et de temps. Encore une fois, ces notions seront expliquées à chacune des étapes.

Comme mentionné dans le processus initial, les requêtes de changements seront toujours crées sous l'application HP Service Manager. Une fois faite, une carte sera créée avec l'outil Kanboard et sera insérée dans la colonne *backlog*. Chacune des applications ou systèmes est représenté par une couleur différente. Prenons par exemple EMS qui est représenté par des cartes bleues. En théorie, il pourrait y avoir autant de cartes que l'on veut. Il n'y pas de limite. Il est ensuite possible d'envoyer la demande au gestionnaire afin d'approuver la nouvelle requête de changement. Cette demande est faite à partir de HP Service Manager. Une fois faite, la carte sera déplacée dans la colonne *Approval sent*. En fait, l'élément est en mode attente. Une fois que l'élément est approuvé, la carte devra être déplacée dans la colonne *Approval done*. Finalement, à partir de HP Service Manager, il sera possible de créer les tâches. Encore une fois, il n'y pas de limite. En théorie, il pourrait y avoir autant de cartes dans les colonnes que l'équipe désire avoir. À part le nombre de cartes dans les colonnes, les données ne seront pas, ici, intéressantes, car les requêtes de changement pourraient rester dans ces colonnes pendant très longtemps. C'est simplement une

question de priorité. La figure suivante montre une partie du tableau Kanban avec les 4 premières étapes du processus.

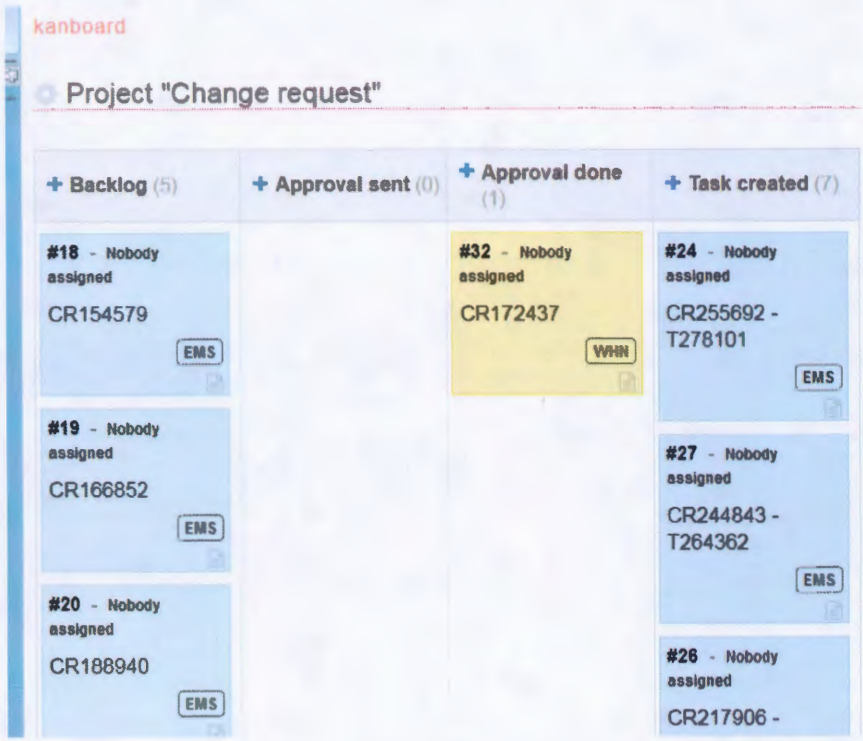


Figure 5.2 – Exemple des étapes du processus Kanban

Une fois qu'une requête de changement a été assignée à quelqu'un et que la personne commence à travailler dessus, la carte qui représente cette requête est déplacée dans la colonne *Work in progress* et elle est assignée, en même temps, à cette même personne. La limite inférieure devrait être en théorie de 5 éléments et la limite supérieure devrait être au maximum de 10 éléments. Chaque personne de l'équipe, sauf le chef, devrait en théorie travailler au moins sur une requête et au maximum sur deux requêtes en même temps. Il n'est guère recommandé de travailler sur trois requêtes ou plus en même temps à moins d'une urgence. Le fait d'avoir des

limites assure l'obtention d'un minimum de vélocité, sans toutefois engorger le processus.

Une fois que le travail est terminé et que le tout est fonctionnel, la carte de la requête est déplacée dans la colonne *Work done* jusqu'au commencement des tests en UAT. Dans le cas où il y aurait des erreurs lors des tests, la carte retournerait dans la colonne Work in progress et ainsi de suite jusqu'à l'acceptation des tests. Une fois terminée, la carte est déplacée dans la colonne *QA done*. La figure suivante montre une partie du tableau Kanban avec les 4 prochaines étapes du processus.

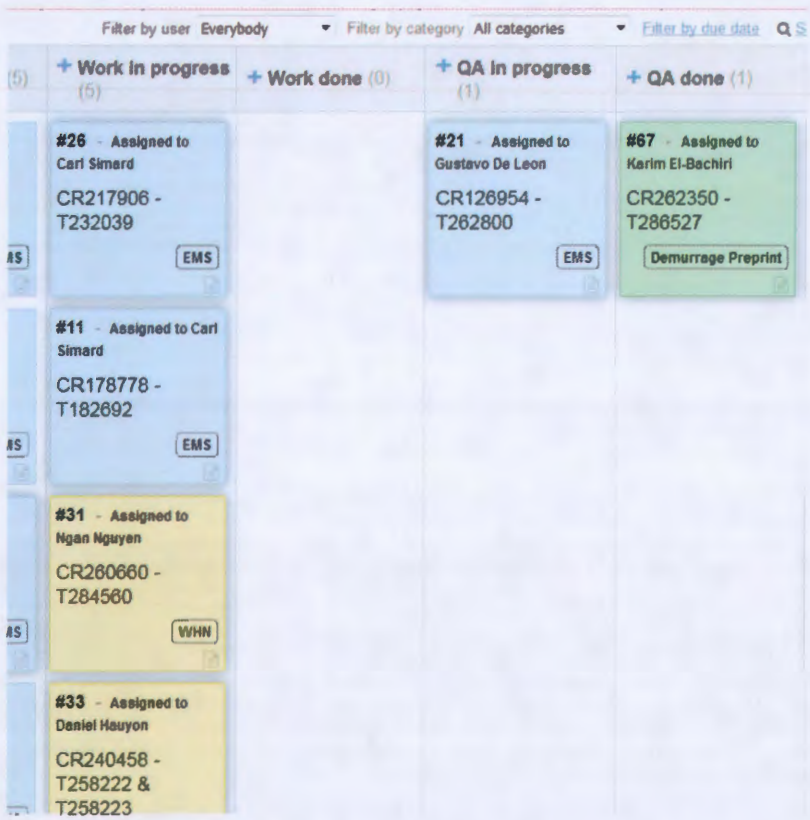


Figure 5.3 - Exemple des étapes du processus Kanban

Finalement, il ne reste plus qu'à préparer et faire le déploiement du nouveau changement en production. Pour ce faire, un plan d'implémentation doit être créé et ajouté dans une des tâches qui ont été créées avec la requête de changement. Une fois le déploiement terminé et le tout validé, la carte est déplacée dans la colonne *Promoted*. La figure suivante montre une partie du tableau Kanban avec les 5 dernières étapes du processus.

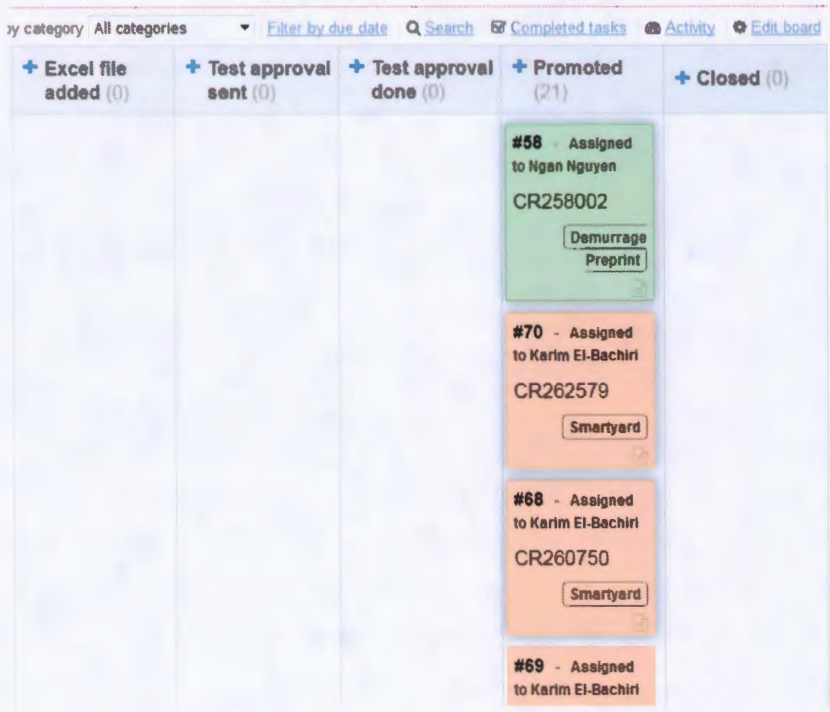


Figure 5.4 - Exemple des étapes du processus Kanban

Finalement, j'ai instauré une rencontre d'une heure par semaine avec tous les membres de l'équipe afin de discuter des états d'avancement, des problématiques rencontrées et des solutions possibles afin d'aider les personnes bloquées. En plus, cette séance permet de faire du transfert de connaissances et de discuter des points forts et des points faibles de la méthode Scrumban pour finalement faire quelques recommandations et ajustements.

5.2.3. Équipe formée

Une fois que les étapes et les limites ont bien été établies, j'ai fait un transfert de connaissance à l'équipe. Une séance de deux heures a été nécessaire afin d'expliquer notre projet, Scrumban et son utilisation. La partie la moins facile a été d'expliquer les objectifs de l'utilisation du cadre Scrumban avec nos processus en cours. Le fait d'obtenir des données, d'avoir des limites et de trouver les goulots afin d'augmenter la fluidité et la vélocité du processus courant et cela de façon continue n'était pas nécessairement le bienvenu pour tout le monde. Quelques personnes ne voyaient aucunement les valeurs ajoutées. La section 6.4 sur la résistance des changements en discute plus en profondeur.

Finalement, tous les membres de l'équipe ont été très motivés et impliqués dans notre projet.

5.3. Améliorations

Cette section discute de l'évaluation du nouveau processus et de l'outil Kanban, et explique les recommandations qui ont été suggérées par les membres de l'équipe. Puisque tous les membres de l'équipe se sont impliqués, l'efficacité du nouveau processus a fortement été améliorée tout au long du projet.

5.3.1. Processus Scrumban amélioré

Pendant quelques semaines, j'ai essayé d'utiliser le nouveau processus avec le cadre Scrumban afin de l'évaluer et d'ajuster les différentes étapes et leurs limites. EMS a été l'application Java qui a été utilisée, car je travaillais dessus presque qu'à 100 % de mon temps. Plusieurs discussions et ajustements ont été apportés au tableau Kanban. Prenons par exemple le fait qu'il y avait un tableau pour les requêtes de changements et un autre pour les différentes tâches associées aux requêtes de changement. Avant même la fin du projet pilote, le tableau pour les tâches a été retiré,

car le comportement des cartes, l'information et les données étaient pratiquement identiques à ceux du premier tableau. Il n'y avait vraiment pas de valeur ajoutée à avoir les deux en parallèle.

5.3.1.1. Les points qui ont été appréciés

Le point fort le plus important qui est ressorti est le fait de voir toute l'information sur un même tableau. Avant, il était pratiquement impossible d'avoir un état global de toutes les requêtes de changement. Ensuite, le fait d'obtenir quelques données et d'en discuter lors de nos réunions permettait de trouver des goulots et des points faibles. L'outil est facile à utiliser et il y a beaucoup de valeurs ajoutées. Prenons par exemple le fait de voir dans une colonne qu'une carte reste trop longtemps sans bouger, il est maintenant plus facile d'aborder une discussion afin de trouver une solution à son blocage. Finalement, le fait de suivre les étapes dans le tableau Kanban assure un ordre adéquat et non aléatoire comme avant.

Les réunions permettent de discuter des différentes requêtes de changement, de leur état d'avancement et des défis rencontrés. Cependant, ce que l'équipe adore le plus est le fait d'en discuter. Cela permet à tout le monde de comprendre ce qui se passe en général. Cela permet de mieux comprendre les nouvelles fonctionnalités des applications que l'équipe supporte. À long terme, il est facile de perdre beaucoup de temps si une personne doit toujours faire du rattrapage de connaissances.

5.3.1.2. Les points qui ont été critiqués

Le point le plus important parmi les critiques a été celui de la duplication. En d'autres mots, l'outil Kanban utilisé ne permet pas de synchroniser l'information avec celle de HP Service Manager. Aussi, le monitoring afin d'obtenir des données comme le temps passé d'une carte dans une colonne, le nombre de cartes dans le tableau, la vélocité et autres est manuel et pénible. D'ailleurs, il arrive souvent que les membres de

l'équipe négligent ces informations, ce qui rend les données erronées. Les membres de l'équipe ne sont probablement pas à l'aise avec le monitoring du processus.

5.3.1.3. Les points qui ont été recommandés

La première recommandation est l'utilisation d'un outil Kanban qui pourrait se synchroniser avec HP Service Manager. L'aspect le plus difficile, ici, est de trouver un outil professionnel avec un prix raisonnable.

La deuxième recommandation est d'avoir un outil où les données peuvent être obtenues de façon automatique. Ceci dit, il faut absolument que quelqu'un dans l'équipe ait la responsabilité de suivre le processus afin d'obtenir des données plus adéquates. Après un certain temps, il sera intéressant de constater l'évolution de la fluidité et de la vélocité du processus. En plus, ces mêmes données pourront être utilisées afin de mieux estimer les échéanciers futurs.

Après une longue recherche et analyse par un groupe du CN externe à l'équipe de maintenance, l'outil Jira[17] a été sélectionné, installé et testé.

5.3.1.4. Les points qui ont été ajustés

Après avoir installé et analysé l'outil Jira, le gestionnaire de l'équipe désirait réduire le nombre de colonnes à 3 dont :

1. *To do* : Cette section contient les demandes de changement à faire.
2. *In progress* : Cette section contient les demandes de changement en développement et en test.

[17] Atlassian - Jira. [En ligne] <https://fr.atlassian.com/software/jira>

3. *Done* : Cette section contient les demandes de changement déployées en production.

La raison principale de diminuer le nombre de colonnes à 3 est qu'il n'était aucunement nécessaire d'avoir plusieurs colonnes dans le tableau. La valeur ajoutée n'était pas vraiment là. Les autres colonnes étaient négligeables par rapport à la colonne *In progress*. Il est à noter qu'il est toujours possible de remodifier le tableau si le besoin se fait sentir. Une colonne UAT ou *In testing* aurait aussi été pertinente, mais le gestionnaire préfère s'en passer.

L'équipe a aussi décidé de ne pas mesurer la vélocité, car il y a trop de facteurs qui entrent en ligne de compte. Prenons par exemple les urgences et les réunions générales. Aussi, l'estimation de nos demandes de changements est très volatile. Plusieurs demandes de changement sont souvent complexes et prennent beaucoup de temps à concevoir et à développer tandis que d'autres peuvent prendre seulement une à deux semaines. Cependant, le temps nécessaire au cycle de développement de la demande de changement est automatiquement enregistré dans l'outil Jira. Dans le besoin d'une estimation un peu plus précise, il sera facile d'aller voir dans la banque et de trouver une demande de changement terminée avec une complexité semblable afin de s'en servir comme modèle.

Les limites des colonnes ont été rapidement mises à l'épreuve et ont été finalement modifiées à sans limite. La raison est que, généralement, un membre de l'équipe va travailler sur une ou deux demandes de changements, mais certaines personnes vont travailler jusqu'à dix demandes de changement en même temps. Prenons par exemple l'application Smartyard. Le gestionnaire va donner à un membre de l'équipe plusieurs demandes de changement à développer et va les promettre aux clients pour la prochaine version à déployer en production. Ansi, même si le développeur travaille sur une demande de changement à la fois, le gestionnaire va toutes les mettre dans la colonne *In progress*.

En résume, l'utilisation du nouveau processus et du cadre Scrumban donne plus d'avantages que d'inconvénients. En plus, le fait de faire des recommandations va

permettre de résoudre quelques faiblesses. Cependant, l'évaluation ne devrait jamais être considérée comme terminée. L'évaluation et les recommandations doivent être un processus en mode continu afin de toujours apporter des améliorations dans le département de la maintenance.

5.3.2. Outil Jira installé

L'outil Jira version v7 est un outil de gestion de demandes de changement et de billets d'urgence qui a été développé par la compagnie Atlassian Corporation Pty Ltd. Il a été installé par un autre département du CN qui l'avait auparavant analysé afin de s'assurer qu'il correspond aux standards et normes du département des technologies de l'information. Les figures suivantes montrent un exemple de notre tableau Kanban.

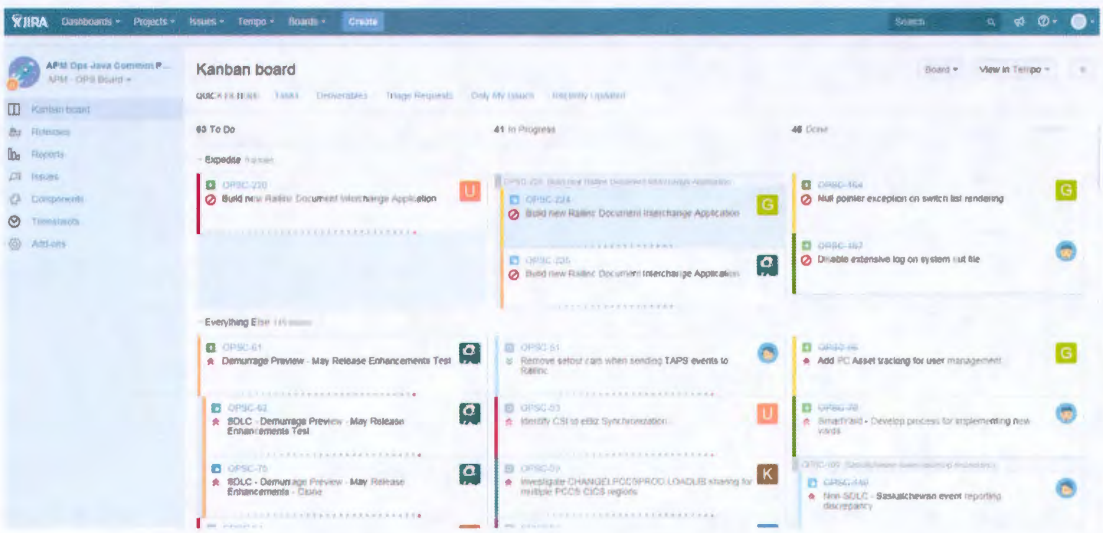


Figure 5.5 - Exemple de notre tableau Kanban avec l'outil Jira

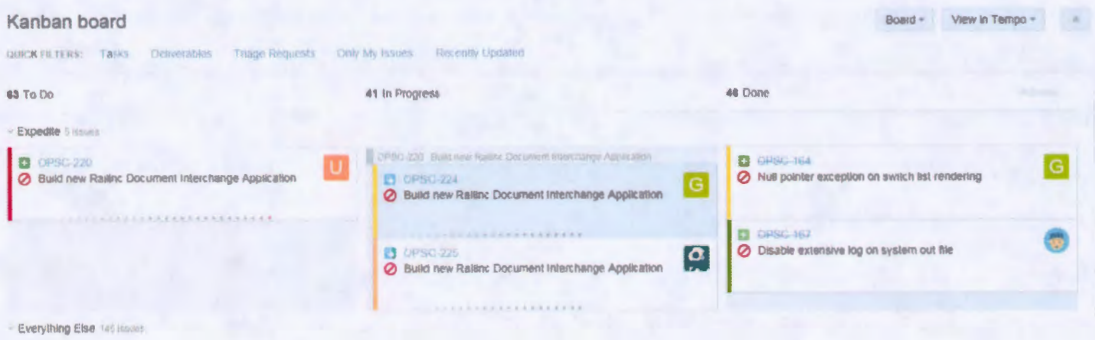


Figure 5.6 - Exemple (zoom) de notre tableau Kanban avec l'outil Jira

Comme mentionné à la section 5.3, il est possible de voir trois colonnes dans tableau Kanban ayant chacun 63, 41 et 46 demandes de changement. Les cartes sont priorisées en ordre d'importance de haut en bas. Il est même possible de constater une section Expedite dans le haut du tableau afin de se concentrer sur eux. Prenons la demande de changement OPSC-224 qui se trouve en haut de la colonne *In progress*. La figure suivante montre la demande de changement.

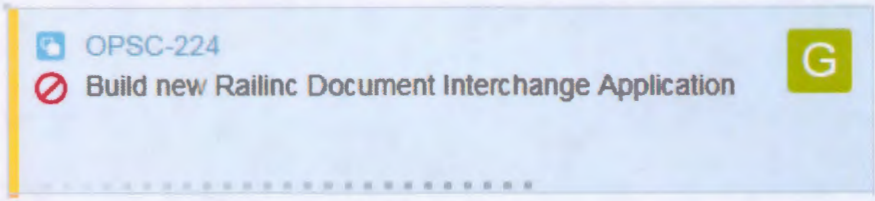


Figure 5.7 - Demande de changement OPSC-224

Le logo G et la couleur jaune à gauche de la carte signifient que la demande de changement est assignée à Gustavo. Il est évidemment possible d'ouvrir la carte afin de voir ses paramètres.

Les figures suivantes montrent plusieurs paramètres de la demande de changement OPSC-224 dont la description, les gens, les détails et le temps. La première figure n'est pas très claire, mais elle donne un bon aperçu de la page. La prochaine figure montre l'information concernant le type de travail, l'estimation, le code du budget, le numéro de la demande de requête et celui de la tâche qui se trouvent dans HP Service Manager. Lors de la création d'une nouvelle demande de changement, une demande de requête est automatiquement crée dans l'outil HP Service Manager. Cette fonctionnalité était nécessaire pour la survie du projet en question. La figure 5.10 montre une brève description de la nouvelle fonctionnalité et finalement la figure 5.11 montre les gens assignés, les dates de création et de mise à jour, et non le moindre le suivi du temps de travail qui reste à faire.

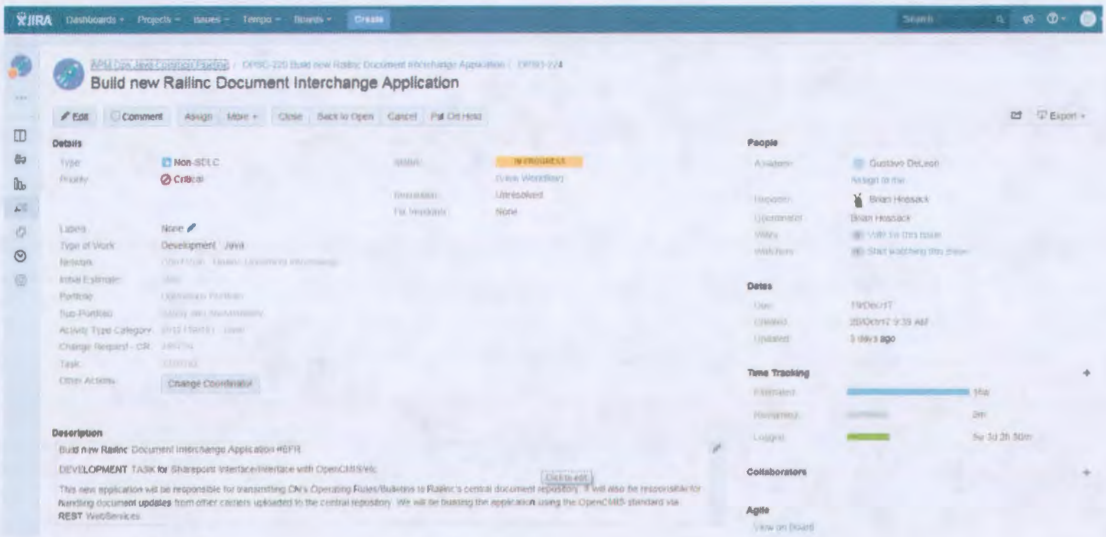


Figure 5.8 - Détail de la demande de changement OPSC-224 1/4

Figure 5.9 - Détail de la demande de changement OPSC-224 2/4

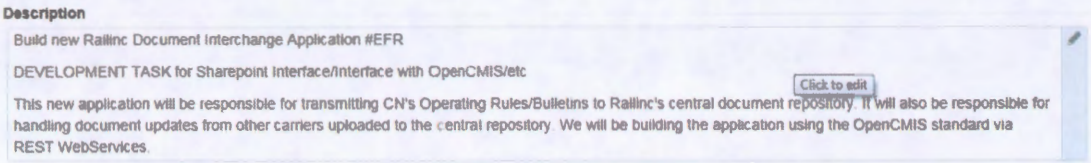


Figure 5.10 - Détail de la demande de changement OPSC-224 3/4

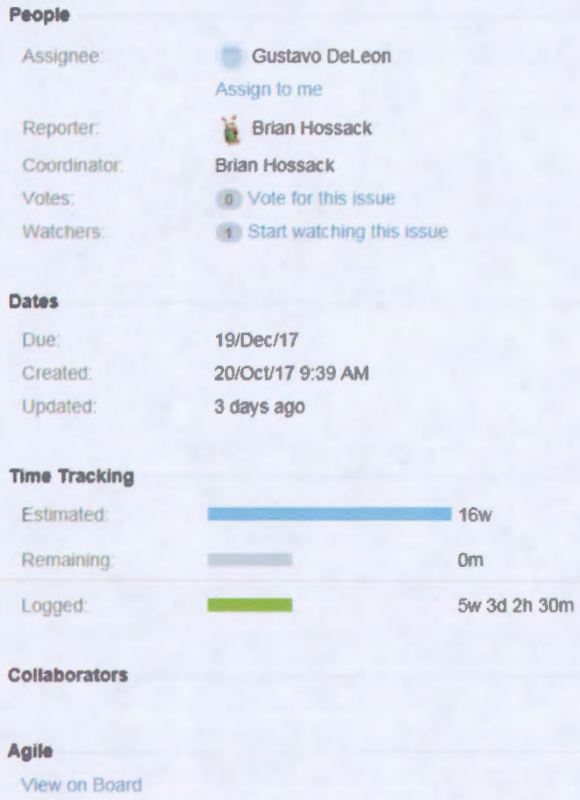


Figure 5.11 - Détail de la demande de changement OPSC-224 4/4

L'outil Jira est complet et professionnel et l'équipe est très satisfaite de ses fonctionnalités. Il y a encore beaucoup de possibilités et de choses à apprendre avec l'outil, dont faire des rapports de vélocité, des graphiques de *burndown* et des flux de travaux (*workflows*).

5.4. Fermeture

5.4.1. Processus Scrumban amélioré documenté

J'ai créé un document sur le processus amélioré afin de faire du transfert de connaissance aux nouveaux employés et aux autres départements de maintenance. La première section fait un résumé du processus initial sans pour autant parler de la problématique mentionnée à la section 2.3. La section suivante fait aussi un résumé de Scrumban pour finalement terminer avec l'utilisation de l'outil Jira.

Chapitre 6. Discussion

Ce présent chapitre discute des retombées des activités qui ont été faites tout au long du projet, ainsi que des défis et obstacles rencontrés. Les paragraphes suivants expliquent plus en détail chacun d'eux.

6.1. Meilleure connaissance de la problématique du processus initial

Puisqu'il n'y avait aucune documentation sur le processus initial et que chaque membre de l'équipe travaillait de façon individuelle, il était difficile de bien voir la problématique. Le fait que j'ai documenté le processus initial et le contexte du projet a permis de mieux saisir les faiblesses du département et de surtout de mieux les comprendre et les communiquer.

6.2. Meilleure connaissance de l'approche et des méthodes agiles

Mon activité d'analyse des méthodes agiles et des outils a été beaucoup plus difficile et a pris beaucoup plus de temps que la première activité de démarrage. Cependant, les connaissances acquises ont une valeur inestimée. Le fait de connaître en profondeur les différentes méthodes agiles et pratiques dans le domaine du développement et de la maintenance logicielle permet d'être un meilleur conseiller et de mieux comprendre les conversations entre les experts du domaine.

6.3. Les défis et obstacles rencontrés

Même si mon projet s'est bien déroulé dans son ensemble, quelques défis et obstacles ont fait surface.

Le premier défi que j'ai rencontré a été de convaincre le gestionnaire du département chez Canadien National de pouvoir faire un projet innovateur au sein de l'organisation afin d'améliorer les processus dans le département de la maintenance des applications Java et de résoudre une problématique jugée importante.

Tout d'abord, il a fallu que je lui explique et démontre la problématique en question. En plus, le gestionnaire avait une grosse crainte et une forte rétention face aux changements des processus déjà établis. Il a aussi fallu que je lui assure que les changements apportés seront évalués et une décision finale sera prise par lui-même afin de savoir s'il est avantageux de continuer dans la même direction ou non. Le fait qu'il avait l'option de revenir en arrière le rassurait énormément. Ainsi, l'approbation a été donnée et l'ouverture de mon projet a vu le jour dès le lendemain.

Le plus grand défi a été de trouver un outil Kanban qui correspondait davantage aux besoins chez Canadien National afin de continuer à utiliser le cadre Scrumban. Le fait qu'il fallait gérer l'information à deux endroits différents causait un gros irritant.

6.4. Résistance aux changements

Le plus grand obstacle que j'ai rencontré a été une forte résistance de quelques employés envers notre projet. La majorité des collègues ont adoré l'idée. Cependant, certaines personnes n'aimaient vraiment pas le fait qu'il y avait maintenant un suivi des états d'avancement des changements adaptatifs et correctifs des applications en production. La solution a été que je m'assois avec les personnes concernées et que j'écoute leurs points de vue en ce qui concerne le projet innovateur. Après une écoute attentive, un point important est ressorti davantage par rapport aux autres. Certaines personnes ne voulaient pas être suivies afin d'évaluer leur performance. En plus, elles

n'aimaient vraiment pas le fait d'être obligées de gérer leurs cartes à travers l'outil Kanban. Elles trouvaient que nous ajoutions des responsabilités pour rien. La solution a été de leur expliquer que l'objectif est de suivre la vélocité du processus et aucunement celle des personnes. Les données sont des données d'équipe et non individuelles. En plus, après avoir fait un transfert de connaissances et expliqué davantage les bienfaits de Scrumban, elles se sentaient plus motivées et, par ce fait, ont décidé de l'essayer et de s'impliquer.

Il est possible de trouver dans la section bibliographie deux excellents livres qui parlent du coaching et mentorat (Kinlaw 2005), et de la gestion des conflits (Pastor et Bréard 2011). Ils m'ont énormément aidé à relever ce défi et à résoudre la problématique de la résistance au changement causé par notre projet.

6.5. Acceptation de la méthode Scrumban

Après quelques améliorations et adaptations au processus Scrumban dans le département, l'équipe a accepté l'implantation de la méthode. L'ensemble des activités précédentes ont permis de livrer chez Canadien National un processus utilisant la méthode Scrumban. L'équipe est passée d'un processus initial mal compris à un processus solide et bien établi.

6.6. Meilleure connaissance de la méthode Scrumban au CN

La création de mon document afin de présenter la méthode Scrumban à d'autres équipes chez Canadien National permettra de donner une meilleure connaissance à quelques employés dans les différents départements de la maintenance du logiciel chez Canadien National. De ce fait, le transfert de connaissance se fera à son rythme à travers les autres employés intéressés.

Conclusion

Mon projet a permis d'améliorer les processus de développement de nouvelles fonctionnalités adaptatives et correctives dans le département de la maintenance des applications Java sous la division des opérations. Avec l'utilisation de la méthode Scrumban et l'outil Jira, il est maintenant possible d'avoir une vue d'ensemble des applications et des urgences tout en étant priorisées et mis à jour de façon quotidienne et d'obtenir des données afin de mieux estimer. Il est aussi maintenant possible que tous les membres de l'équipe puissent être efficaces lors d'une urgence en production grâce aux transferts de connaissance lors des réunions hebdomadaires.

Le tableau suivant reprend les objectifs mentionnés à la section 4.1 et explique comment ils ont été atteints.

Tableau 7.1 - Liste des problèmes

Problèmes	**Objectifs**	**Atteinte des objectifs**
Absence de vue d'ensemble	L'équipe doit être en mesure d'obtenir l'information des demandes de changement dans un seul endroit commun.	L'équipe est en mesure d'obtenir l'information des demandes de changement dans un seul endroit commun avec l'aide de l'outil Jira.
Priorisation segmentée	Toutes les demandes de changement de toutes les applications doivent être priorisées en un seul endroit commun.	Toutes les demandes de changements de toutes les applications sont priorisées en un seul endroit avec l'aide de l'outil Jira.

Problèmes	Objectifs	Atteinte des objectifs
État d'avancement inconnu	L'équipe doit être en mesure d'obtenir l'état d'avancement à jour de chacune des demandes de changement de toutes les applications.	L'équipe est en mesure d'obtenir l'état d'avancement à jour de chacune des demandes de changements de toutes les applications avec l'aide des réunions hebdomadaires et de la gestion du temps dans l'outil Jira.
Estimation impossible	L'équipe doit être en mesure de donner des estimations pour chacune des demandes de changement.	L'équipe est en mesure de donner des estimations pour la plupart des demandes de changements avec l'aide de la banque de demande de changement dans l'outil Jira.
Informations des solutions non communiquées	L'équipe doit être en mesure de faire du support sur tous les changements et nouveautés technologiques déployés en production.	L'équipe est en mesure de faire du support sur tous les changements et nouveautés technologiques déployés en production avec l'aide de transfert de connaissance lors des réunions hebdomadaires.

L'information est plus claire et les estimations seront de plus en plus précises. Ainsi, les gens d'affaires auront une meilleure confiance en l'équipe. À l'aide des réunions hebdomadaires, il est aussi possible de constater plus rapidement les goulots, dont les erreurs techniques lors de la phase de développement, de mieux les comprendre et d'en trouver des solutions, en équipe, afin d'améliorer l'efficacité et la fluidité du processus. Ainsi, il est possible de déployer les changements en production dans un meilleur délai qu'auparavant. Finalement, il est possible d'en faire plus au bout de l'année et cela avec le même budget ou moindre.

Avant de terminer, il a été fortement recommandé à l'équipe de continuer à faire un suivi des données du processus à l'aide de l'outil Jira afin de toujours s'autoévaluer. En d'autres mots, il est recommandé de toujours faire de l'amélioration continue dans

le département afin de continuer à augmenter l'efficacité et la vélocité des processus même si ce n'est jamais facile à appliquer.

Finalement, l'amélioration continue n'est pas à négliger et tout le monde devrait en faire, peu importe le domaine professionnel. Les différentes méthodologies pratiquées ont démontré leur efficacité et leurs avantages. Les cadres comme Scrum, Kanban et Scrumban ont plusieurs avantages et il est fortement recommandé de les utiliser dans des contextes similaires.

La gestion de projets et la gestion des processus sont des sujets énormes et très complexes, et je pense qu'il est primordial de toujours s'informer des nouveautés, de lire et de participer à des séminaires. Cependant, n'oubliez jamais ceci : c'est bien de connaître la théorie, mais il faut aussi être capable de la mettre en pratique. Les retombées en valent vraiment la peine et cela fera de vous des gens professionnels encore plus connaissant qu'hier. N'est-ce pas la beauté de la vie de toujours grandir?

Bibliographie

Articles de Wikipédia

Canadien National dans Wikipédia. (2014). [En ligne] Récupéré le 15 octobre 2014 à l'adresse <http://fr.wikipedia.org/wiki/Canadien_National>

Capability Maturity Model Integration dans Wikipédia. (2016). [En ligne] Récupéré en 2016 à l'adresse <https://fr.wikipedia.org/wiki/Capability_Maturity_Model_Integration>

Cycle de développement logiciel dans Wikipédia. (2015). [En ligne] Récupéré le 16 novembre 2017 de <https://fr.wikipedia.org/wiki/Cycle_de_d%C3%A9veloppement_(logiciel)>

Locomotive dans Wikipédia. (2014). [En ligne] Récupéré le 15 octobre 2014 à l'adresse <http://fr.wikipedia.org/wiki/Locomotive>

Maintenance du logiciel dans Wikipédia. (2015). [En ligne]. [En ligne] Récupéré le 16 novembre 2017 de <https://fr.wikipedia.org/wiki/Maintenance_du_logiciel>

Article en ligne

Denning, S. (2012). What exactly is agile? Is Kanban agile? [En ligne] Récupéré en 2015 à l'adresse <https://www.forbes.com>

Livres

April, A. et Abran, A. (2016). Améliorer la maintenance du logiciel, Montréal, Loze-Dion.

Aubry, C. (2013). Scrum : le guide pratique de la méthode agile la plus populaire. Paris, Dunod.

Chalvin, D. (2011). Mieux vivre avec le stress, Montréal, Les Éditions Transcontinental.

Beck, K. et Andres, C. (2005). Extreme programming explained : embrace change. Boston, Addison-Wesley.

Beck, K. et Fowler, M. (2001). Planning extreme programming. Boston; Montreal, Addison-Wesley.

Boisvert, M. et Trudel, S. (2011). Choisir l'agilité : du développement logiciel à la gouvernance. Paris, Dunod.

Brechner, E., et Waletzky, J. (2015). Agile project management with Kanban. Redmond, Wash.: Microsoft Press.

Fernandez, A. (2011). Le chef de projet efficace, Paris, Éditions d'Organisation.

Gallo, C. (2011). Les secrets d'innovation de Steve Jobs : 7 principes pour penser autrement, Paris, Pearson.

Hammarberg, M., et Sunden, J. (2014). Kanban in action. Shelter Island, NY: Manning.

Kinlaw, D. (2005). Adieu patron! Bonjour coach!, Montréal, Les Éditions Transcontinental.

Kniberg, H., Skarin, M., Poppendieck, M., et Anderson, D. (2010). Kanban and Scrum. United States: InfoQ.

Morisseau, L. et Aubry, C. (2014). Kanban pour l'IT une nouvelle méthode pour améliorer les processus de développement. Paris, Dunod.

Pastor, P. et Bréard, R. (2011). Gestion des conflits (4e éd.), Paris, Éditions Liaisons.

Raizada, R. (2013). Are Scrum and Kanban Enough ?. ITNOW, Volume 55(3), 54-55.

Schwaber, K. and Sutherland, J. (2013). The Scrum guide. Scrum.Org and ScrumInc.

Mémoire

Counet, A. (2007). Amélioration du processus de la maintenance du logiciel par un système informatisé d'aide à la décision par ARNAUD COUNET. Namur, FUNDP – Namur.

Normes

International Standard - ISO/IEC 1074 IEEE Std (1997). Guide for Developing Software Life Cycle Processes. Genève, Suisse.

International Standard - ISO/IEC 14764 IEEE Std (2006). Software Engineering - Software Life Cycle Processes - Maintenance. Genève, Suisse.

International Standard - ISO/IEC TR 14471 IEEE Std (2007). Information technology - Software engineering – Guidelines for the adoption of CASE tools. Genève, Suisse.

International Standard - ISO/IEC 15939 IEEE Std (2008). Systems and software engineering – Software life cycle process. Genève, Suisse.

International Standard - ISO/IEC 15939 IEEE Std (2017). Systems and software engineering – Measurement process. Genève, Suisse.

Sites Web

CMMI Institue. (2016). [En ligne] Consulté en 2016 à l'adresse <http://cmmiinstitute.com>

CN - Transportation Services - Rail Shipping, Intermodal, trucking, warehousing and international transportation. (2014). Cn.ca. [En ligne] Consulté le 15 octobre 2014, à l'adresse <http://www.cn.ca>

Home - Railinc. (2014). Railinc.com. [En ligne] Consulté le 15 octobre 2014, de <https://www.railinc.com/rportal/web/guest/home>

IEEE. (2016). [En ligne] Consulté en 2016, à l'adresse <https://www.ieee.org>

Institut de recherche sur l'histoire des chemins de fer du Québec. (2014). Irhcfq.org. [En ligne] Consulté le 15 octobre 2014, à l'adresse <http://www.irhcfq.org/web_fr/bibliographie.html>

Internationa organization for standardization – ISO. (2016) [En ligne] Consulté en 2016, à l'adresse <https://www.iso.org/home.html>

Jira | Logiciel de suivi des tickets et des projets | Atlassian. (2017). Atlassian. [En ligne] Consulté le 14 janvier 2017 de <https://fr.atlassian.com/software/jira>

Kanban Project Management Software. (2014). Kanboard.net. [En ligne] Consulté le 1 novembre 2014, à l'adresse <https://www.kanboard.net>

Scrum.org. (2017). [En ligne] Consulté en 2017, à l'adresse <https://www.scrum.org>

S3M. (2016). [En ligne] Consulté en 2016 à l'adresse <https://www.s3m.org/fr/>

Trello. (2014). Trello.com. [En ligne] Consulté le 1 novembre 2014, à l'adresse <https://trello.com>

Annexe

CANADIAN NATIONAL RAILWAY

INFORMATION TECHNOLOGY

JAVA CORE SUPPORT PROCESS

WITH SCRUMBAN AND JIRA

V1.0

PREPARED BY

CARL SIMARD

Table of Contents

History of changes....81

Goal of this document....82

Process....82

Scrumban Framework....83

Jira....85

History of changes

Date	Name	Description	Version
2017-10-11	Carl Simard	New document	1.0

Goal of this document

The purpose of this current document is to explain the process of the Java Core Support team using Scrumban Framework and Jira.

Process

Before working on a new enhancement or a new change, a Change Request and Tasks on HP Service Manager must be created. To do so, the team lead has to create the new items, Change Request and Tasks, with Jira[18]. Then, Jira will automatically create them under HP Service Manager. If it has not already been done yet, the team lead or the developer has to prioritize the new change with the help of the business and update it, by moving the card to the right place, on Jira.

As soon the developer starts to work on the new change, the manager must move the item to the column *In Progress* on Jira. In theory, every day the developer must enter his time on Jira. Otherwise, he will receive the next morning an email from Jira saying that he must enter his time as soon as possible to stay up-to-date. This is really important for the team lead to follow the progress of the new changes.

When the new changes are ready to be deployed to the test environment, the developer can compile and build the new Baseline with Jenkins[19]. Once the deployment is done to DEV and UAT, the developer must do a smoke test and create the test cases before testing the systems and/or the applications that he has been working on. The next task is really important. All the tests must be done and completed by another person than the developer from the team or by QA team.

---

[18] Jira tool link: <https://jira.cn.ca:8443/secure/Dashboard.jspa>
[19] Jenkins tool link : <http://mtlh-clrca01ap.cn.ca/login?from=%2Fview%2FEMPS%2F>

While the testers are testing the systems and/or the applications, the developer can start to create the Implementation Plan for the promotion. Once all the tests are done and passed, the developer must add the test results to the Tasks on HP Service Manager and send a request to the business and the team lead for the approvals.

Promo team can now create a Change Control for the promotion. To do so, the developer must send an email to them with the date of the promotion. Once created, the developer can associate all the tasks to the Change Control. When ready, the developer can send the Implementation Plan to Promo team so they can add it to the Change Control.

When everything is all set, the team lead must verify everything and approve the Change Control before the end of the previous week of the promotion. At the beginning of next week, Promo team will finally verify and approve it too.

It is really important that the business and the users are aware of the promotion and all the new changes included in it. So, they can adapt they work to stay efficient in their day-to-day job while the system and/or the application is not available. To do so, the developer can send them an email the day before and another one with the result once the promotion is completed. In our team, the developer must coordinate the people before and during the promotion.

Scrumban Framework

Scrumbran Framework is a merged between some ideas of Scrum Framework and Kanban Framework. The idea behind it is to complete Kanban Framework. Because Scrum Framework is used for projects using agile methodology, some ideas are not adequate for maintenance. Let's take the example of an emergency change that has to be deployed right away to production. There is no way this can happen with Scrum Framework because the backlog of the iterations are immutable. However, the Scrum

meetings and some other ideas are really welcome in the maintenance mode. The goal here is not to explain Scrum Framework, except the fact this is a Framework that has been designed for agile methodology. However, for people who are interested to know more about it, there are a lot of books and websites on the Internet that discuss about Scrum Framework.

In our case, Kanban Framework is more applicable. The main goal of it is simply to monitor the velocity of the flow and the number of changes in the pipeline. This is usually done with a board named Kanban board and some cards that represent all the changes. The next figure gives a perfect example of a Kanban board in IT.

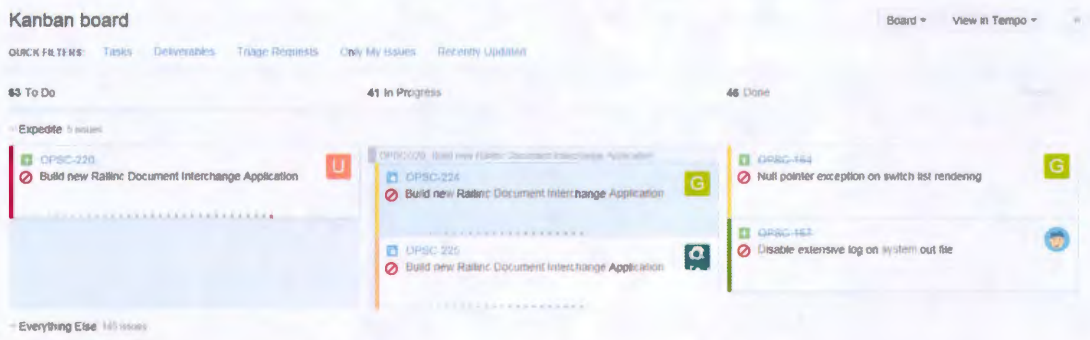


Figure 12 – Kanban board with Jira

In our example, there are three columns (*To do, In Progress* and *Done*) where the cards can be moved. Usually, the number of columns represents each operation in a process, but in our case, there is no need to have more. The new changes that have been created must be and prioritized in section *To Do*. Once a developer starts working of them, they are moved to section *In Progress* and then moved to section *Done* once they are deployed with success to production.

The last part to mention here is the fact that the developers must enter their time to the cards every day so the team lead can monitor the progress properly.

As mentioned, a weekly meeting is added in our progress to talk about our progress and difficulties. The goal is not to find a solution, because it would eat all the meetings. If necessary, another meeting can be organized to do so. Also, a knowledge transfer must be done every time a new change is deployed to production so all the members of the team are aware of it and understand the technology behind it.

Jira

Jira tool version v7 is a change requests and emergency tickets management tool that has been developed by Atlassian Corporation Pty Ltd[20]. The tool has already been installed by another department that had already analyzed it to be sure it follows our information technology department standards and norms. The next figures show an example of our Kanban board with Jira.

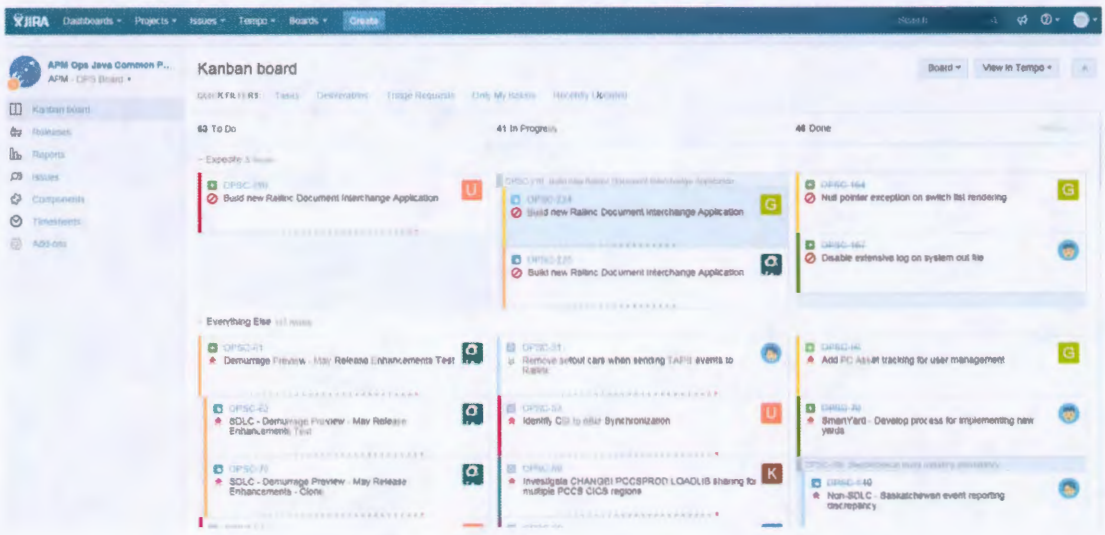


Figure 13 - Kanban board with Jira

[20] Jira [En ligne] <https://www.atlassian.com/software/jira>

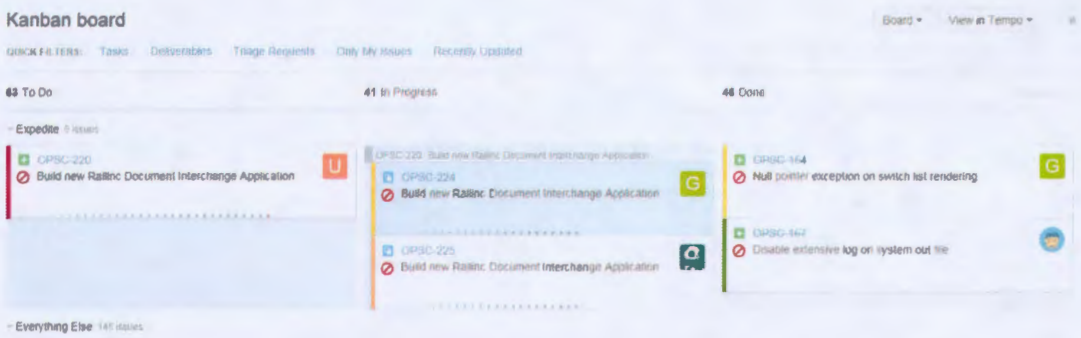


Figure 14 - Kanban board with Jira (zoom)

As mentioned, there are three columns in our Kanban board (*To Do, In Progress* and *Done*). This does not mean that all departments in information technology must have three columns. It depends of your process and needs. It is also possible to see that there are 63, 41 and 46 changes in the columns and they are prioritized from the top to bottom. Let's take the change OPSSC-224 as an example. The next figure shows the change OPSC-224.

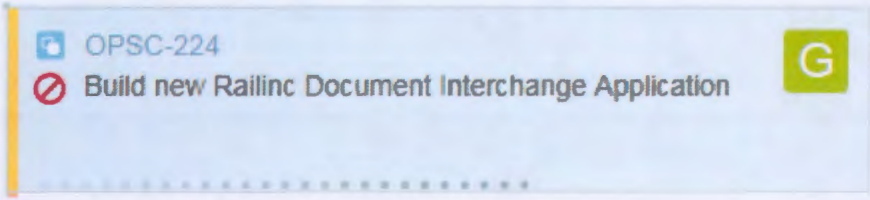


Figure 15 - Change Request OPSC-224

The G logo and the yellow colour at the left of the card mean that the card is assigned to Gustavo. Let's open it to see the details of its parameters.

The following figures show many parameters of the change OPSC-224 as the description, the people, some details and the time. The first figure is not really clear but it gives a pretty good idea of the screen. The next figure show information about what kind of work it is, the estimation, the budget code, the number of the change

request and the task one that are on HP Service Manager. The figure 7 shows the description of the change and finally, the figure 8 show the assigned people, the date of creation and update and the number of hours to complete the change.

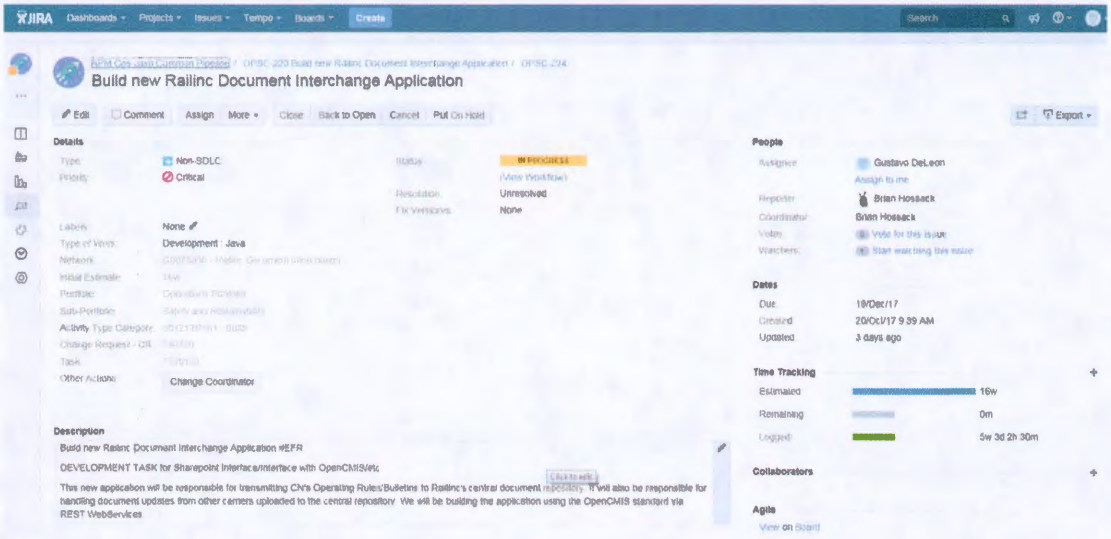


Figure 16 - Detail of the change OPSC-224 1/4

Figure 17 - Detail of the change OPSC-224 2/4

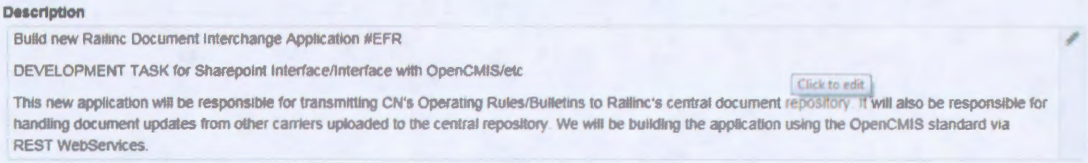


Figure 18 - Detail of the change OPSC-224 3/4

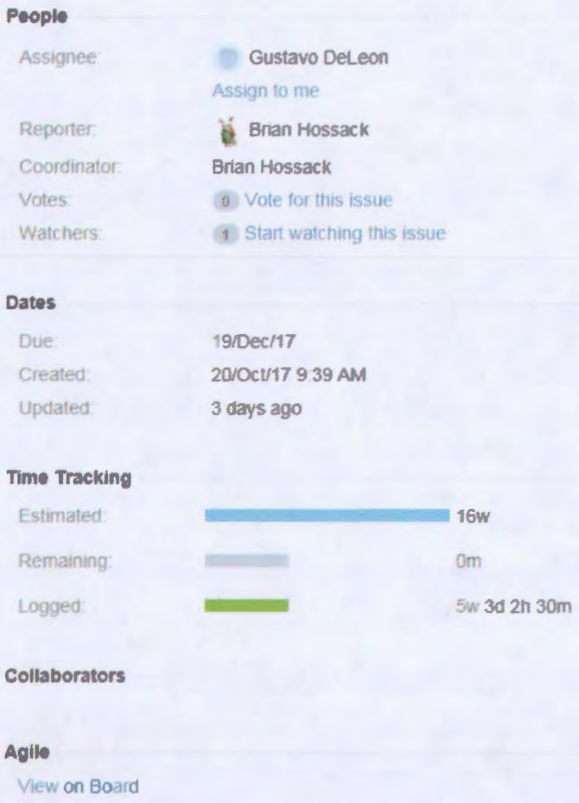


Figure 19 - Detail of the change OPSC-224 4/4

To log hours on Jira, there is an option in the top menu in the task called *Log Work*. A new window will open where you can enter all your information. The next figures show an example.

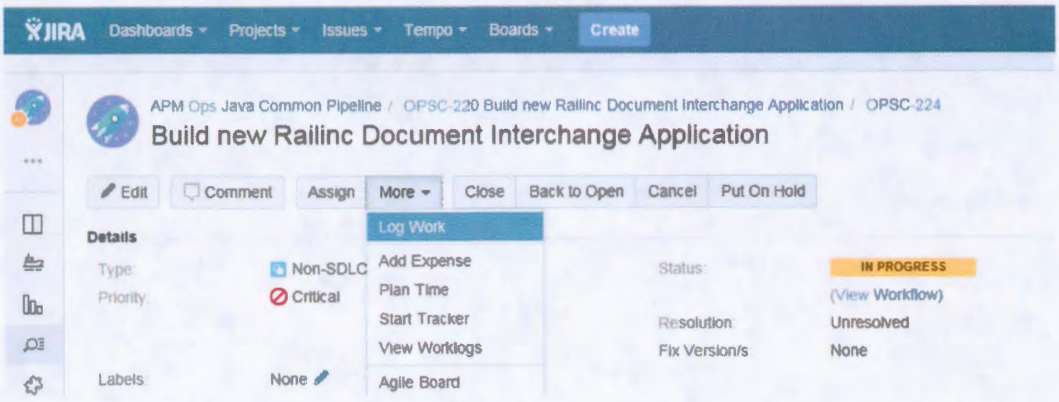


Figure 20 - Log work menu example

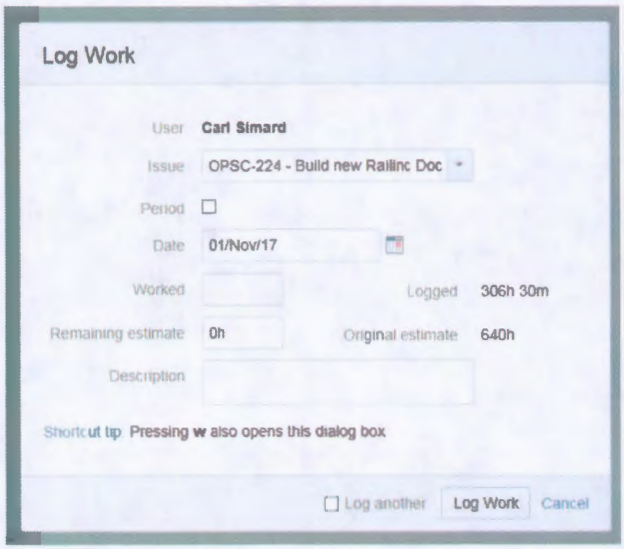


Figure 21 - Log work window example

To see your timesheets or to enter more hours on Jira, there is an option on the top menu called *Timesheets* under *Tempo* tab. The next figure shows an example.

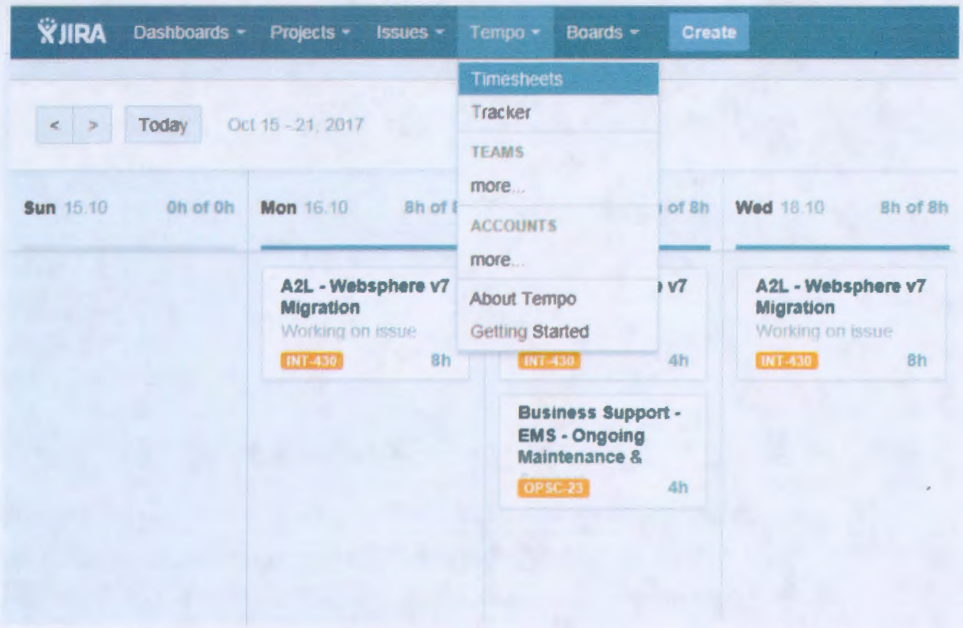


Figure 22 - Timesheets example

Jira tool is complete and professional. There are a lot of other functionalities to learn and play with like the velocity rapports, the burndown graphics and the workflows.