Intégration progressive de composants des méthodes agiles : Retour d'expérience

Elena Kornyshova, Rebecca Deneckere, Adrian Iacovelli

Intégration progressive de composants des méthodes agiles : Retour d'expérience

Elena Kornyshova, Rebecca Deneckere, Adrian Iacovelli

Intégration progressive de composants des méthodes agiles : Retour d'expérience

Elena Kornyshova[1], Rébecca Deneckère[2], Adrian Iacovelli[2]

1. CEDRIC, Conservatoire National des Arts et Métiers
2, rue Conté, 75003, Paris, France
elena.kornyshova@cnam.fr

2. CRI, Université Paris 1 - Panthéon Sorbonne
90, rue de Tolbiac, 75013 Paris, France
Rebecca.Deneckere@univ-paris1.fr, adrian.iacovelli@gmail.com

RESUME. L'ingénierie des méthodes situationnelles cherche à construire des méthodes adaptées à une situation donnée, soit par construction à partir de composants de méthodes déjà définis, soit par l'adaptation d'une méthode existante (en utilisant diverses techniques : configuration, extension, réduction, etc.). Cependant, ces techniques sont peu utilisées dans l'industrie car elles sont considérées comme compliquées et très lourdes à implémenter. Dans cet article, nous décrivons deux expériences pratiques (dans une entreprise de développement et de recherche et une entreprise du secteur touristique) d'intégration progressive des composants de méthodes agiles au lieu d'une mise en place en une fois. Les deux cas ont été sélectionnés pour l'étude car ils possédaient une caractéristique en commun, à savoir le rejet de la mise en place des méthodes agiles. Nous présentons dans cet article les étapes d'intégration progressive dans les deux cas et analysons les résultats de ces expériences.

ABSTRACT. Situational Method Engineering aims at constructing methods adapted to a given situation, either by a construction from a set of predefined method components or by a customization of an existing method using different techniques: configuration, extension, reduction, and so on. However, these techniques are still limited in practice, as considered complicated and heavy to implement. In this paper, we describe two practitioner experiences (a development and research company and a travel company) of the progressive integration of agile method components instead of the one-time implementation. We present in this paper the gradual integration processes in two cases and we analyze the results of these experiments.

Mots-clés : Ingénierie des Méthodes Situationnelles ; Composants de méthodes ; Méthode Agile ; Intégration Progressive ; Rapport d'expérience

KEYWORDS: Situational Method Engineering; Method Component; Agile Method; Progressive Integration; Experience Report

DOI:10.3166/RCMA.25.1-n

1. Introduction

L'ingénierie des méthodes situationnelles (IMS) statue qu'une méthode de développement de système d'information doit être alignée avec le contexte du projet sur lequel elle s'applique. En effet, chaque situation est différente et nécessite un support méthodologique différent. Dans cet objectif, l'IMS propose des méthodes de construction spécifiques et adaptables selon la situation du projet en cours en réutilisant des parties de méthodes existantes, appelées composants, stockées dans des bases de méthodes. Bien que plusieurs approches de construction existent, leur implémentation dans le contexte industriel est difficile. Les entreprises reconnaissent le bien fondé de ces approches et de ces techniques mais trouvent leur implémentation difficile et couteuse.

Une autre manière d'utiliser les techniques d'IMS de manière plus douce est d'introduire les composants de manière progressive, un à la fois, et d'attendre que les utilisateurs soient à l'aise avec les premiers changements avant d'aller vers une autre modification. Dans cet article, nous appellerons cette introduction une « intégration progressive ». Nous présentons dans ce travail le résultat de deux expérimentations observées dans deux entreprises souhaitant améliorer leur processus de gestion de projets informatiques. Les deux cas d'étude ont eu lieu dans des entreprises très différentes par leurs tailles, secteurs, méthodes de travail, mais ayant un point en commun, qui était la résistance des membres de l'équipe à la mise en place des méthodes agiles. Pour cette raison, dans les deux cas, il s'agissait d'introduire certains éléments des méthodes agiles au lieu de faire une intégration des méthodes entières en une seule fois. Nous présentons dans cet article les étapes d'intégration (avec les composants associés) dans les deux cas et nous analysons les résultats d'intégration. Le premier étude de cas a déjà été décrit dans (Deneckère et al., 2016a).

La Section 2 aborde les fondements théoriques. Ensuite, nous présentons les deux cas d'étude : une entreprise de développement et de recherche (Section 3) et une entreprise du secteur touristique (Section 4). Ces sections décrivent le contexte organisationnel, les étapes d'intégration pour chaque cas, ainsi que les résultats obtenus. Dans la Section 5, nous analysons les avantages et les inconvénients de l'intégration progressive des composants de méthodes. Nous concluons et présentons les axes de notre recherche future dans la Section 6.

2. Fondements théoriques

Cette section propose un court état de l'art sur l'ingénierie des méthodes situationnelles, présente les méthodes agiles et leurs composants utilisés dans les deux expériences. Nous décrivons aussi les travaux existants en ce qui concerne l'application de l'ingénierie des méthodes aux méthodes agiles.

2.1. Ingénierie des méthodes situationnelles et composants de méthodes

Les méthodes de développement de Systèmes d'information (SI) se sont de plus en plus complexifiées, ceci dans le but d'obtenir une méthode adaptée à toutes les situations de projets rencontrées par les professionnels. Cependant, il est maintenant admis que le concept de méthode universelle n'est pas un concept viable (Henderson-Sellers et al., 2014) et plusieurs travaux se sont penchés sur la manière d'adapter les méthodes à la situation au moment de leur construction, ce qui permet d'obtenir directement la meilleure méthode pour le projet en cours. Ces méthodes adaptées à la situation sont appelées des *méthodes situationnelles* et reposent sur le principe que les méthodes sont décomposables en blocs indépendants appelés *composants de méthodes* et qui peuvent être réassemblés pour former une nouvelle méthode. Ces composants sont en général sélectionnés selon les besoins en fonction de l'adéquation de leur contexte et de celui du projet (Kornyshova et al., 2011b). Cependant, ces approches de construction sont souvent assez complexes. L'ingénierie des méthodes est considérée comme une démarche trop couteuse, consommatrice du temps et exige un niveau considérable d'expertise. L'ingénierie des méthodes situationnelles (IMS), introduite dans les années 80, tente de répondre à cette problématique en offrant diverses techniques permettant de créer ou d'adapter des méthodes selon la situation du projet en cours (Henderson-Sellers et al., 2014). En effet, il est maintenant admis que la situation d'ingénierie de chaque projet est différente et engendre donc un besoin de support méthodologique différent (Kumar et Welke, 1992). Pour permettre une adaptation efficace, l'IMS propose de réutiliser les méthodes en les considérant comme des ensembles de composants, ceux-ci étant réutilisables et pouvant être combinés les uns avec les autres. Ces composants sont stockés dans une base de méthodes, ce qui permet de les retrouver plus facilement. Plusieurs approches ont été proposées sur cette notion de composant modulaire : Approches d'assemblage de composants (Brinkkemper et al., 1998) (Firesmith et Henderson-Sellers, 2002) (Ralyté et Rolland, 2001) ; Approches de configuration (Karlsson et Agerfalk, 2004) ; Approche d'extension (Deneckere, 2001) ; Approches d'adaptation (Rossi et al., 2004) ; Approches dirigées par les modèles (Cervera et al., 2012) ; Approches orientées services (Guzelian et Cauvet, 2007) (Iacovelli, 2012) ; Approche de familles de méthodes (Kornyshova et al., 2011a) (Deneckère et al., 2014). Des études comparatives de ces différentes approches ont été effectuées dans divers travaux (Nehan et al., 2007) (Kuhrmann et al., 2014) (Henderson-Sellers et al., 2014).

2.2. Méthodes de développement agiles

Le concept de méthode agile est apparu au début du millénaire avec l'arrivée du manifeste agile en 2001 (Beck et al., 2001). Comme il est indiqué dans (Serena Software, 2007), alors que la publication de ce manifeste n'a pas entériné la ruée vers les méthodes agiles, celle-ci ayant commencé bien plus tôt, le manifeste signale tout de même l'acceptation de la philosophie agile dans le monde professionnel. Plusieurs méthodes agiles ont été définies et largement communiquées depuis lors, telles que

Lean Software Kanban (Anderson, 2003), Extreme Programming (Beck, 2000), Scrum (Schwaber et Beedle, 2002), Crystal (Cockburn, 2002) et DSDM (Stapleton, 1995) entre autres. Une étude de 2013 (VersionOne, 2013) signale que 57% des répondants travaillent dans des entreprises ayant plus de cinq équipes pratiquant l'agilité et 38% avec plus de 10 équipes. Ces chiffres indiquent que le mouvement agile est un succès d'adoption et qu'il est largement utilisé dans le monde professionnel.

2.3. Composants de méthodes agiles utilisés dans les deux expériences

Cinq composants ont été utilisés dans les deux expériences. Ces composants ont été définis dans nos travaux antérieurs sur les méthodes agiles et sur leur décomposition en composants de méthode (Iacovelli, 2012) (Deneckère et al., 2014) (Deneckère et al., 2016b). Chaque composant est détaillé de la façon suivante : son interface (composée de la situation - pré-condition à l'exécution du composant - et de l'intention à réaliser par le composant), sa description textuelle rapide, sa situation source (pré-condition), sa situation cible (post-condition) et son processus.

Le composant 1 « Planifier le projet avec les réunions de Sprint » (figure 1) explique les nouvelles fonctionnalités associées aux réunions de Sprint. Pendant chacune des réunions, l'ensemble des tâches du projet est étudié et un état est associé à chaque tâche. Des post-its colorés, manuels ou numériques, sont utilisés pour améliorer la visibilité et la différentiation entre les différents états. De nouvelles tâches peuvent être définies à partir des user stories (des récits) initiales et ajoutées à l'ensemble avec une estimation de leur durée d'intégration dans le projet.

Composant 1 – < {Besoins, Planification des tâches}, Planifier le projet avec les réunions de Sprint >
Description: L'objectif de ce composant est de définir les tâches au sein de chaque user story, d'indiquer l'état de l'exécution de chacune de ces tâches ainsi que l'avancement général du projet. Le but est d'aider à gérer les tâches en lui donnant une vision claire de leurs cycles de vie.
Situation Source : Besoins, Planification des tâches **Situation Cible :** Planification des tâches
Processus pour chaque réunion de Sprint : 1. Rapide évaluation des user stories déjà livrées. • Faire une démonstration des user stories achevées pendant le Sprint. • Modifier l'état de chaque tâche : Prête, Affectée, Terminée, Expirée, Transférée, Finie, Echouée. • Faire une rétrospective (analyse de l'amélioration continue) • Modifier l'état des user stories 'Affectée' en utilisant des couleurs de post-its différentes. 2. Evaluation des user stories à livrer. • Description des user stories à développer. Le client décrit ce qu'il veut dans le

Sprint suivant. L'équipe argumente avec le client pour écarter toute possible ambiguïté.

- Identifier un objectif de Sprint : une description en une phrase de l'objectif général du Sprint. Si une tâche n'est pas tout à fait reliée à l'objectif du Sprint, celle-ci ne doit pas être faite pendant ce Sprint.

3. L'équipe décide comment le travail doit être fait.
   - Planifier les tâches pour les nouvelles user stories avec une estimation de leur durée de développement.
   - Décrire le Sprint.

Figure 1. Composant 1: Planifier le projet avec les réunions de Sprint.

L'objectif du composant 2 « Planifier le projet avec le jeu de poker » (illustré à la figure 2) est d'intégrer un jeu dans la planification des tâches avec ce que l'on appelle le jeu du poker. Tous les besoins sont écrits sur des cartes et le jeu permet d'estimer la durée de leur développement. Chaque membre de l'équipe a l'opportunité de parler et de s'exprimer jusqu'à ce que tous les membres se mettent d'accord sur l'estimation. Les besoins sont alors regroupés en livrables.

Composant 2 – < {Besoins}, Planifier le projet avec le jeu de Poker>

Description : L'objectif est d'aider à estimer le temps nécessaire pour implémenter les besoins des utilisateurs et établir le premier calendrier des livrables. Ce composant utilise la technique du jeu de poker introduite par la méthode XP et utilisée plus tard dans d'autres méthodes dites agiles.

Situation source: Besoins

Situation cible : Livrables identifiés et planifiés

Processus :

1. Ecrire chaque besoin sur une carte. Les cartes habituelles utilisées dans ce type de jeu utilisent en général la suite de Fibonacci, incluant le 0 : 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 ; les autres types de cartes utilisent un type de progression similaire. La raison de cette progression est de refléter l'incertitude inhérente à l'estimation de grands éléments, ce qui signifie que plus l'estimation est longue, plus elle est incertaine.
2. Evaluer le temps idéal de développement de chaque tâche en utilisant les cartes. Un modérateur surveille le processus. L'équipe peut poser des questions pour clarifier certains points et évaluer les risques. Chacun pose une carte, face cachée, sur la table, avec son estimation (l'unité peut varier : jours, heures, points de story, etc.). Si nécessaire, les membres justifient leurs points de vue pour permettre une meilleure discussion (surtout pour les très grandes ou très faibles estimations). Ce processus est répété jusqu'à ce qu'un consensus soit atteint.
3. Regrouper les cartes sur la table pour constituer les livrables, sans ordonnancement à l'intérieur de chacun.
4. Calculer les dates de livraison en fonction des estimations.

Figure 2. Composant 2: Planifier le projet avec le jeu de Poker.

Le composant 3 « Planifier le projet avec le Backlog produits » (illustré à la figure 3) explique comment utiliser le Backlog pour planifier le projet. Le Backlog contient la liste des fonctionnalités nécessaires au projet. Il est premièrement rempli au début du projet mais son remplissage se continue pendant tout le développement. Le Backlog peut être considéré comme le référentiel de l'équipe sur le sujet des exigences du projet. Ce composant inclut trois sous-composants.

Composant 3 – < {Besoins}, Planifier le projet avec le Backlog produit >
Description : L'objectif de ce composant est d'aider le chef de projet et les clients à identifier et gérer les besoins. Le Backlog est alors utilisé pendant tout le projet pour avoir une vision globale des tâches de développement. Sous-composants : • Planifier le projet avec la réunion d'estimation • Planifier le projet avec la mise à jour continue du Backlog produit • Planifier les Sprints avec le Backlog produit
Situation source : Besoins **Situation cible :** Backlog produit
Processus général pour produire le Backlog : 1. Identifier les besoins de haut niveau au début du projet et les intégrer dans le Backlog 2. Mettre les user stories dans le Backlog 3. Tous les jours, si un besoin est identifié comme non nécessaire, le supprimer du Backlog. Au contraire, si un nouveau besoin est identifié, l'ajouter au Backlog. 4. Utiliser le Backlog régulièrement pour planifier les Sprints.

Figure 3. Composant 3: Planifier le projet avec le Backlog produit.

Le sous-composant 3.1 (illustré à la figure 4) explique comment utiliser le Backlog pour la planification initiale du projet. Le Backlog contient la liste de toutes les fonctionnalités nécessaires du projet et est défini au début de celui-ci avec le client. Le Backlog peut être considéré comme le référentiel de l'équipe concernant les exigences du projet.

Le sous-composant 3.2 (illustré à la figure 5) explique comment utiliser le Backlog pendant la durée totale du projet. Le Backlog contient la liste des fonctionnalités requises du projet et son élaboration peut être poursuivie pendant tout le développement.

Le sous-composant 3.3 (montré à la figure 6) explique comment utiliser le Backlog pour la planification des Sprints. Deux réunions par Sprint doivent être mises en place afin de comparer les résultats obtenus avec le reste à faire.

Sous-composant 3.1 – < {Besoins}, Planifier le projet avec la réunion d'estimation >
Description : L'objectif de ce composant est d'aider le chef de projet et les clients à identifier les besoins de haut niveau du système. En partant de la description du problème, l'équipe identifie les besoins de haut niveau qui seront affinés plus tard dans le processus. Il n'y a pas de formalisme prédéfini pour le Backlog ; il peut être représenté par un document texte, un tableur, une base de données ou même un ensemble de post-its. Chaque élément doit être décrit d'une manière atomique (seulement un besoin pour chaque élément). Le Backlog produit peut contenir des fonctionnalités, des erreurs, des travaux techniques ou juste de la connaissance.
Situation source : Besoins **Situation cible :** Backlog Produit
Processus pour initialiser le Backlog : 1. Identifier les besoins de haut niveau au début du projet et les mettre dans le Backlog. 2. Mettre les user stories dans le Backlog.

Figure 4. Sous-composant 3.1: Planifier le projet avec une réunion d'estimation.

Sous-composant 3.2 – < {Besoins, Backlog produit}, Planifier le projet avec la mise à jour continue du Backlog produit >
Description : L'objectif du composant est d'aider le chef de projet et les clients à gérer les besoins en temps réel. Après la définition initiale du Backlog, celui-ci est modifié sur une base journalière. Lorsque le Backlog devient d'une taille plus importante, il peut devenir nécessaire de séparer le Backlog en éléments à courts ou longs termes.
Situation Source : Besoins, Backlog produit **Situation cible :** Backlog produit
Processus pour utiliser le Backlog de manière journalière : 1. Si un besoin est identifié comme non nécessaire, le supprimer du Backlog. 2. Si un nouveau besoin est identifié, l'intégrer au Backlog.

Figure 5. Sous-composant 3.2: Planifier le projet avec la mise à jour continue du Backlog produit.

Sous-composant 3.3 – < {Backlog produit, User stories}, Planifier les Sprints avec le Backlog produit >
Description : L'objectif de ce composant est d'aider le chef de projet à planifier correctement les Sprints. Le Backlog produit est utilisé pour gérer les besoins et indiquer quelles tâches sont en cours, terminées ou encore à faire. Deux réunions sont introduites dans le processus au début et à la fin de chaque Sprint pour évaluer, avec le client, quelles user stories doivent être développées pour le Sprint en cours et ce qui a déjà été fait.
Situation source : Backlog produit, user stories

Situation cible : Backlog produit, Sprint
Processus pour planifier les Sprints avec le Backlog : 1. Au début du Sprint, faire une réunion de planification de Sprint pour prioriser les user stories avec le client. Cette réunion permet de définir l'objectif du Sprint : une courte phrase décrivant ce que l'équipe doit développer pendant le Sprint. L'équipe et le client l'écrivent conjointement. Le Backlog du Sprint est l'autre résultat de la réunion. C'est une liste d'éléments du Backlog que l'équipe s'engage à délivrer avec la liste des tâches nécessaires. 2. A la fin du Sprint, faire une réunion pour évaluer le travail fait et faire quelques démonstrations au client. Le succès du Sprint sera ensuite évalué pendant la réunion de fin de Sprint suite à la confrontation avec l'objectif du Sprint, plutôt qu'avec chaque élément sélectionné dans le Backlog produit.

Figure 6. Composant 3.3: Planifier le projet avec le Backlog produit.

Le composant 4 (illustré à la figure 7) explique comment introduire une réunion quotidienne dans le projet. L'équipe se réunit une fois par jour, toujours à la même heure, pour partager l'avancement du développement. La réunion est prévue pour une durée très courte. Si un point semble prendre trop de temps, l'équipe peut le mettre de côté pour en discuter en effectif réduit. Cette réunion permet de partager toutes les informations concernant le développement du projet. Plus que des exercices de cohésion d'équipe, la communication régulière et l'aide fournie aux autres membres permettent de renforcer l'unité d'une équipe.

Composant 4 – < {Planning des tâches, Sprint}, Gérer le projet avec la réunion journalière >
Description : L'objectif est d'aider le chef de projet à gérer le projet en organisant une courte réunion journalière (moins de 15 minutes). Toute l'équipe doit assister aux réunions. Toutes les choses importantes au sujet du développement doivent être partagées (écueils, nouveaux besoins). Les engagements journaliers permettent aux membres de l'équipe de communiquer plus efficacement et de coordonner leurs efforts pour résoudre leurs difficultés.
Situation source : Planning des tâches, Sprint **Situation cible :** Sprint
Processus : La réunion journalière ne doit pas excéder 15 minutes. Si un des éléments doit être approfondi, alors il est nécessaire de prévoir une autre réunion pour en discuter. La réunion prend place avec les participants debout pour que tout le monde se souvienne que la réunion doit être courte et efficace. Cette réunion doit avoir lieu tous les jours. Toute l'équipe est sensée assister à cette réunion mais la réunion n'est pas annulée si tout le monde n'est pas présent. 1. Tous les membres de l'équipe partagent leur avancement sur le développement. Ils parlent de leurs progrès depuis la veille, le travail prévu pour la journée et les problèmes rencontrés, ce qui permet de pouvoir demander de l'aide ou de collaborer

plus facilement avec les autres membres de l'équipe.

2. Les membres de l'équipe peuvent se poser trois questions : qu'ai-je fait hier ? Que ferais-je aujourd'hui ? Quels sont les obstacles qui m'empêchent de progresser correctement ?
3. Les obstacles sont écrits sur un tableau, visible par tous, qui identifie les obstacles et montre les progrès dans leur résolution. Ce tableau peut être modifié en dehors des réunions.

Figure 7. Composant 4: Gérer le projet avec la réunion journalière.

Composant 5 – < {Planning des tâches, Backlog}, Gérer le projet avec le graphique d'avancement >

Description : L'objectif est d'aider le chef de projet à gérer le projet avec une meilleure visibilité sur les travaux à venir. Le graphique d'avancement (burndown chart) est une représentation graphique des travaux encore à développer selon le temps restant. C'est une partie essentielle des projets agiles et un moyen de voir clairement les progrès réalisés pendant chaque Sprint.

Situation source : Planning des tâches, Backlog

Situation cible : Graphique d'avancement

Processus :

1. Construire le graphique :

   Le graphique est un graphique en deux dimensions. L'axe horizontal représente les Sprints. L'axe vertical correspond au montant de travail restant au début de chaque Sprint (dans l'unité préférée de l'équipe).
   a. Le départ du projet est le point le plus à gauche du graphique et arrive au jour 0 de l'itération.
   b. La fin du projet est le point le plus à droite et arrive au jour prévu de fin de projet.
   c. La ligne de travail idéale est une ligne droite qui connecte le point de départ et le point d'arrivée. Au point de départ, la ligne idéale montre la somme des estimations pour toutes les tâches qui doivent être complétées. Au point d'arrivée, la ligne idéale coupe l'axe x en montrant qu'il ne reste plus de travaux à réaliser.
   d. La ligne de travail réelle montre les travaux encore à effectuer. Au point de départ, la ligne est la même que la ligne idéale ; cependant, au fur et à mesure que le temps progresse, la ligne actuelle fluctue sous et au-dessus de la ligne idéale, selon les disparités entre les estimations et le temps de travail réel nécessaire pour le développement des tâches. En général, un nouveau point est ajouté à cette ligne tous les jours.
2. Interpréter le graphique
   a. Si la ligne actuelle est au-dessus de la ligne idéale, alors il y a plus à faire que ce qui était prévu et le projet est en train de dépasser le temps qui lui était alloué.
   b. Si la ligne actuelle est sous la ligne idéale, alors il y a moins à faire que prévu et l'équipe est en avance sur le développement du projet.

Figure 8. Composant 5: Gérer le projet avec le graphique d'avancement.

Le composant 5 (illustré à la figure 8) montre comment utiliser le graphique d'avancement pour avoir une meilleure visibilité sur les travaux à venir. Cela permet d'obtenir une liste des états des tâches toujours à jour et d'encourager l'équipe à affronter les difficultés plus tôt et de manière plus décisive.

2.4. Application de l'ingénierie des méthodes aux méthodes agiles

Les méthodes de développement agiles sont en général définies comme des ensembles de bonnes pratiques et de bons comportements à adopter. Depuis le manifeste agile, énormément de livres et de documents ont été édités afin d'expliquer ces bonnes pratiques – par exemple (Shore, 2007) ou (Cohn, 2005) – mais tous ces documents manquent d'une explication claire du processus à appliquer, sur le fait qu'il faut être agile pour faire de l'agilité. Mais le fait qu'il n'y ait pas de processus formel ne signifie pas que les développements agiles ne sont pas structurés, uniquement qu'il y a une différence entre un processus complètement rigide et un processus formalisé. Dans (Meyer, 2014), Bertrand Meyer dit que « Le livre classique sur l'agilité est une succession d'observations générales alternant avec des anecdotes personnelles sur des sauvetages ou des échecs de projets. Ces anecdotes sont en général amusantes et parfois intéressantes, mais un cas d'étude n'est qu'un cas d'étude et nous ne savons jamais jusqu'où il est possible de généraliser ». Dans cette jungle de tâches, principes, conseils et recommandations, les nouveaux utilisateurs de méthodes agiles sont parfois perdus avec toutes les possibles alternatives qui leur sont offertes, avec une compréhension parfois très minimaliste des arguments en faveur ou contre celles-ci. Par exemple, (Meyer, 2014) mentionne que « toute équipe agile construit son propre cocktail de pratiques agiles, rejetant celles qui ne correspondent pas à ses besoins – jusqu'à présent cependant, chaque organisation, pour chacun de ses projets, doit répéter ce processus de tri ». Ce comportement est commun à d'autres champs d'expertise et l'utilisation d'autres types de méthodes (méthodes de développement, méthodes de déploiement, méthodes de conception, etc.) et le domaine de l'IMS peut peut-être aider à résoudre ce problème.

Les méthodes agiles ont déjà été étudiées plusieurs fois dans le domaine de l'IMS. (Qumer et Henderson-Sellers, 2007) ont montré dans un cas d'étude qu'une approche de méthodes situationnelles couplée à un cadre d'outils agiles (Agile Software Solution Framework - ASSF) peut être utilisée pour créer une méthode de développement hybride utilisable. Ceci peut être réalisé en combinant les pratiques agiles et formelles dans une situation donnée dans de grandes organisations. Dans (Karlsson et Agerfalk, 2008), une approche de configuration de méthodes a été proposée pour configurer eXtreme Programming. L'une des conclusions était que XP ne procure pas vraiment d'alternatives de processus et qu'il était donc assez ardu d'obtenir des possibilités intéressantes des développeurs. Cette étude n'utilisait cependant qu'une seule méthode agile, ce qui la rend difficile à généraliser. (Abad et al., 2010) (Abad et al., 2012) propose une approche de construction de méthodes situationnelles par assemblage spécifique aux méthodes agiles. Une base de composants de méthodes contient les fragments

nécessaires, conformes au méta-modèle SPEM 2.0., et utilisables avec des outils de méta-modélisation permettant d'implémenter ce méta modèle, incluant l'EPFC (Eclipse Process Framework Composer).

Les méthodes agiles sont en général définies comme des ensembles de bonnes pratiques et beaucoup de documents tentent de les expliquer. Ceux-ci manquent cependant d'explications claires sur le processus à appliquer, sur l'observation que l'on se doit d'être agile si l'on veut faire de l'agile. Nous avons élaboré un ensemble de composants de méthodes concernés par la représentation des méthodes agiles à un haut niveau ainsi que des vues détaillées de composants s'intéressant à la phase de lancement des projets dans les méthodes agiles : Scrum (Schwaber et Beedle, 2001), XP (Beck, 2000), DSDM (Stapleton, 1995) et Crystal Clear (Cockburn, 2002). Ces composants sont intégrés dans une famille de méthodes appelée Lancement de projets agiles (LPA). Cette famille de méthodes agiles propose un ensemble de composants organisés dans un processus clair et facile à comprendre spécialisé dans la phase de lancement des projets agiles. Ces composants de méthodes ont été décrits dans (Iacovelli, 2012) et développés plus avant dans (Deneckère et al., 2014).

3. Cas 1 : Entreprise de développement et de recherche

3.1. Contexte organisationnel

L'entreprise capitalise plus de 10 ans de recherche et développement dans le cloud computing et le big data. Elle est spécialisée dans le développement de systèmes d'informations complexes, particulièrement dans le domaine de la santé et de la recherche biomédicale.

L'entreprise travaillait sur plusieurs projets en même temps et utilisait une méthode de développement essentiellement basée sur une méthode de développement classique avec quelques éléments d'agilité. L'équipe avait une réunion hebdomadaire pour discuter des tâches à effectuer pendant la semaine. De nouvelles tâches pouvaient être ajoutées et les membres de l'équipe pouvaient discuter de leur faisabilité. L'équipe utilisait un googledoc pour sauver et partager les comptes-rendus des réunions. Une nouvelle version du googledoc était créée chaque semaine. L'outil Redmine[1] était utilisé pour gérer le projet mais son usage était limité à la définition des tâches de haut niveau et à la décomposition de ces tâches en sous tâches. La durée de réalisation des tâches était également intégrée dans cet outil.

L'organisation de la gestion de projet dans cette entreprise avait plusieurs problèmes.

a. La définition des tâches. Les tâches identifiées étaient de très haut niveau et pas assez détaillées. De plus, leur formulation était très informelle.

[1] http://www.redmine.org/

b. Le suivi de projet. Le temps de réalisation des tâches était spécifié dans le googledoc mais cela était fait à un niveau très haut et la hiérarchie des tâches n'était pas toujours à jour.

c. Le manque d'un outil spécifique pour aider dans les tâches de gestion de projet.

d. L'intégration des nouveaux besoins et les retours utilisateurs. Il n'y avait pas de réunion centrée utilisateurs. Chaque nouveau besoin ou retour utilisateur était soit traité en temps réel sans gestion des priorités, soit mis de côté pour une période indéterminée, ce qui engendrait des complications sur le suivi de ces tâches.

Tous ces problèmes étaient reliés au manque de méthode ou d'outil de gestion de projet. L'idée d'intégrer une nouvelle méthode de développement agile dans l'entreprise ne souleva pas l'enthousiasme dans l'équipe. Les membres de l'équipe pensaient que cela leur demanderait trop d'efforts et de temps pour un gain peut-être minime. En effet, la méthode utilisée n'était pas parfaite mais elle fonctionnait et donnait des résultats. C'est alors qu'un des membres de l'équipe a entamé l'intégration progressive des composants de méthodes agiles qui leur manquaient.

3.2. Ordre d'intégration

Pendant la période couverte par cette étude cinq composants de méthodes agiles ont été introduits progressivement. L'ordre d'intégration inclut donc cinq phases (étapes). Les composants agiles identifiés et correspondant aux besoins sont les suivants : Planifier le projet avec la réunion de Sprint, Planifier le projet avec le jeu de poker, Planifier le projet avec le Backlog produit, Gérer le projet avec la réunion journalière, Gérer le projet avec le graphique d'avancement. Les cinq composants détaillés plus haut sont concernés. L'ordre d'intégration établi est présenté sur la Figure 9.

L'ordre d'intégration des composants a été défini suivant deux exigences spécifiques évaluées en fonction des discussions informelles ayant eu lieu entre les membres de l'équipe de développement sur le sujet des concepts agiles :

- L'urgence de l'intégration d'une fonctionnalité spécifique et
- Le taux d'acceptation au changement des membres de l'équipe.

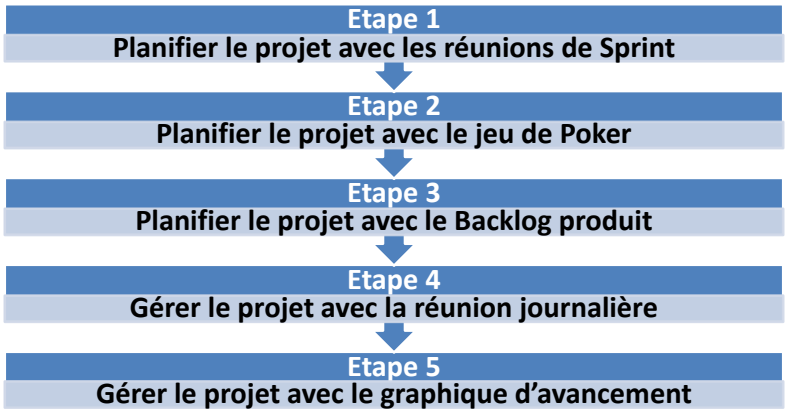

Figure 9. Ordre d'intégration des composants de méthodes agiles dans le cas de l'entreprise de développement et de recherche.

Etape 1: Planifier le projet avec les réunions de Sprint

Contexte : L'équipe utilisait un document Google pour planifier toutes les tâches des projets. Cependant, même si tous les membres de l'équipe partageaient ce document, le processus de gestion de celui-ci était relativement fastidieux, rempli après les différentes réunions ayant eu lieu. L'objectif ici était de trouver un outil qui permettrait de gérer un peu mieux cette tâche.

Processus : Une extension de l'outil RedMine permet de gérer des post-its numériques correspondant aux tâches spécifiées dans RedMine. L'outil permet de lier les user stories aux tâches de haut niveau identifiées au commencement du projet. Cette extension a été testée sur place et intégrée au processus de projet. Aucune autre intégration n'a été effectuée jusqu'à ce que ce changement n'ait été complètement accepté par les membres de l'équipe (ce qui a pris 3 à 4 mois).

Résultat : La planification est passée d'une gestion d'un document Google à la définition des tâches à effectuer pendant les réunions. Une conséquence intéressante est que ces réunions, au lieu de devenir plus longues à cause du remplissage fait auparavant de manière ultérieure, sont devenues plus courtes ! La nouvelle fonctionnalité de l'outil a été considérée comme très efficace pour récupérer toutes les informations nécessaires au développement. De plus, elle a permis à l'équipe d'être plus efficace dans la gestion des estimations de temps qui n'étaient pas prises en compte dans le document Google d'origine.

Etape 2: Planifier le projet avec le jeu de Poker

Contexte : Ce composant était fortement désiré dès le début du projet par l'équipe de développement. En effet, il s'agit de la technique la plus connue dans les méthodes agiles. Il a donc été le second composant a être intégré dans la méthode d'origine. L'utilisation d'un outil était également quelque chose de nécessaire car un membre de l'équipe travaillait à distance et il fallait utiliser quelque chose qui permette de travailler sur des cartes numériques, plutôt que sur de réelles cartes physiques.

Processus : L'outil RedMine utilisé dans l'entreprise possédait déjà un outil appelé 'Jeu du poker' donc il n'était finalement question ici que de l'introduire auprès de l'équipe, d'expliquer le processus de priorisation et de superviser la première utilisation dans le projet.

Résultat : L'adaptation au changement de l'équipe pour cette nouvelle manière de planifier les tâches a été très rapide. En effet, étant donné que cette technique faisait partie de leurs besoins initiaux, l'équipe était vraiment très avide de pouvoir l'utiliser. L'ensemble des membres de l'équipe était très satisfait de cette nouvelle manière de prioriser les tâches.

Etape 3: Planifier le projet avec le Backlog Produit

Contexte : L'équipe voulait avoir la possibilité de gérer les tâches sur un plus long terme. Les user stories étaient créées semaine après semaine, sans avoir de réunion récapitulative permettant de gérer l'estimation de la durée de développement. Les tâches de haut niveau étaient créées au début du projet mais il manquait un niveau de granularité entre les tâches de haut et de bas niveau. La fonctionnalité manquante était la gestion correcte et complète d'un Backlog.

Processus : L'outil utilisé avait une fonctionnalité de Backlog déjà intégrée mais qui n'était pas utilisée par l'équipe. Les tâches non terminées sont maintenant intégrées dans le Backlog, ainsi que les user stories non planifiées pour la semaine suivante. Le Backlog est rempli au début du projet directement avec le client lors d'une réunion spécifique dédiée à cette étape du processus de développement.

L'introduction de ce composant a été décomposée en trois phases (correspondant aux trois sous composants) : premièrement, le Backlog a été uniquement utilisé pour les exigences de haut niveau définies au départ du projet (cette phase a duré au moins 6 mois). Ensuite, les user stories ont été intégrées dans le Backlog d'une manière régulière dès qu'elles faisaient leur apparition, ceci d'une manière assez informelle (6 mois). Finalement, le Backlog a été utilisé pour planifier les Sprints (2 mois).

Résultat : Après plus d'un an, le Backlog est maintenant utilisé complètement. Il y a plusieurs types de réunions qui ont été introduites dans le processus de développement :

- Réunion d'estimation (avec le client) pour créer les user stories (avec la technique du jeu de poker), donner les informations sur la vélocité du projet et faire des démonstrations.
- Réunion de début de Sprint pour prioriser les user stories avec le client.
- Réunion de fin de Sprint pour évaluer le travail fait et faire quelques démonstrations au client.

Sous-composant 3.1: Planifier le projet avec la réunion d'estimation

Contexte : Les tâches de haut niveau étaient créées au début du projet mais étaient uniquement intégrées dans le document Google, accompagnées d'une description informelle. Comme l'objectif était de supprimer ce document, il était nécessaire de trouver un autre moyen de stocker les exigences du projet.

Processus : L'outil utilisé par l'entreprise avait déjà une fonctionnalité de Backlog qui n'a eu qu'à être intégrée dans le processus de développement. Le Backlog est maintenant rempli directement au début du projet - avec le client - dans une réunion spécifique dédiée à cette étape du développement.

L'introduction de ce composant a été assez longue (6 mois) car l'équipe était assez réticente à utiliser cette fonctionnalité. Il a donc été nécessaire de l'introduire doucement et sans trop de changements dans le processus.

Résultat : Une réunion d'estimation avec le client est maintenant mise en place pour créer les user stories (avec la technique du jeu de poker).

Sous-composant 3.2: Planifier le projet avec la mise à jour continue du Backlog produit

Contexte : L'équipe utilisait le Backlog uniquement au début du projet pour stocker les besoins de haut niveau. Elle voulait avoir la possibilité de gérer les tâches sur un plus long terme, en ajoutant un niveau de granularité entre les tâches. Le chef de projet devait avoir un accès à une utilisation plus poussée du Backlog.

Processus : Les user stories sont maintenant intégrées dans le Backlog de manière régulière, dès qu'elles font leur apparition, de manière informelle. De nouveau, cette intégration a été assez longue. Même à ce moment alors que l'équipe avait fini par accepter l'utilisation du Backlog pour le départ du projet, l'équipe était toujours réticente à utiliser cette fonctionnalité de manière régulière.

Il a donc été nécessaire de les habituer à cette approche pendant environ 6 mois avant de pouvoir aller encore un peu plus loin.

Résultat : Cette fonctionnalité a beaucoup aidé l'équipe à gérer les user stories qui apparaissaient pendant le développement. Le Backlog est maintenant utilisé pour toutes les tâches et le lien entre elles est beaucoup plus clair pour tous les membres de l'équipe.

Sous-composant 3.3: Planifier les Sprints avec le Backlog produit

Contexte : L'équipe remplissait le Backlog de manière régulière mais ils ne l'utilisaient pas complètement pour planifier les Sprints.

Processus : Le Backlog était utilisé essentiellement pour intégrer les besoins en début de projet. Il est maintenant utilisé pour y intégrer les tâches non terminées, ainsi que les user stories non planifiées pour la semaine suivante. Il a été nécessaire d'intégrer cette fonctionnalité sur deux mois pleins pour assurer une intégration efficace.

Résultat : Après l'intégration des trois sous composants (3.1, 3.2, 3.3), le Backlog est maintenant utilisé d'une manière complète. Deux réunions ont été également intégrées dans le processus : la réunion de début de Sprint et la réunion de fin de Sprint. Les Sprints sont maintenant en général de deux semaines et le client a été intégré d'une meilleure manière dans le processus de développement.

Etape 4: Gérer le projet avec la réunion journalière

Contexte : L'équipe de projet était assez petite et se réunissait souvent pour discuter autour de la machine à café. Cependant, comme l'un des membres de l'équipe travaillait à distance, il était difficile d'avoir une communication réellement efficace entre tous les membres de l'équipe. L'outil utilisé a déjà permis d'améliorer cette communication sur l'avancement des tâches mais certains détails importants étaient laissés de côté, seulement évoqués d'une manière informelle en dehors des réunions officielles du projet.

Processus : Les réunions ont été planifiées pour améliorer la communication entre les membres de l'équipe, avec un accès internet direct pour celui travaillant à distance.

Résultat : Les réunions ont beaucoup amélioré la communication entre les membres de l'équipe, essentiellement en regard du membre travaillant à distance qui est maintenant à même de connaître tous les détails du projet.

Etape 5: Gérer le projet avec le graphique d'avancement

Contexte : Il était difficile pour l'équipe du projet d'avoir une vue à long terme sur les Sprints à venir. L'information était présente mais disséminée dans les documents lorsqu'une tâche était comprise dans plusieurs Sprints. Il était donc ardu d'estimer correctement les différents Sprints. Les progrès d'un développement peuvent être suivis en utilisant un graphique d'avancement mis à jour à chaque fin de Sprint.

Processus : RedMine avait déjà cette fonctionnalité dans sa boite à outil. Cependant, l'équipe ne l'utilisait pas puisque le planning n'était pas correctement défini au début du projet. Maintenant que l'information concernant la planification est cohérente et indiquée dans l'outil, la fonctionnalité du graphique d'avancement peut être utilisée de manière régulière et efficace.

Résultat : Ce composant a été intégré assez récemment dans l'entreprise mais les premiers retours sont très encourageants. L'équipe a une meilleure vision de l'estimation des tâches encore à effectuer et est capable de créer une meilleure planification.

3.3. Conclusion de l'intégration

Durant la période couverte par cette étude, les composants de méthode agile ont été introduits progressivement. Cette intégration a été faite en cinq étapes correspondant aux cinq composants, de manière consécutive, et n'a pas suivi l'ordre logique indiqué dans les méthodes agiles initiales. Nous pouvons dire que les composants sélectionnés venaient essentiellement de la méthode SCRUM mais certains sont également présents dans d'autres méthodes agiles. La séquence logique de SCRUM et des autres méthodes suggère l'intégration de ces cinq composants lors de la même étape alors que dans ce cas l'intégration a été découpée en autant d'étapes que de composants, avec succès dans ce cas où la résistance au changement était assez forte.

4. Cas 2 : Entreprise du secteur touristique

4.1. Contexte organisationnel

La deuxième entreprise est une compagnie internationale du secteur touristique de grande taille (plus de 10000 employés partout dans le monde). Elle est spécialisée dans l'organisation des voyages et existe depuis plusieurs décennies.

Le système d'information de cette entreprise est très riche et contient des applications anciennes. L'étude a été réalisée dans une équipe travaillant sur le périmètre des applications du backoffice du domaine financier-comptable. Il s'agit d'applications existantes qui sont déjà en production. Les employés travaillent sur

des projets de maintenance applicative : traitement des anomalies, petites évolutions (inférieures à cent mille euros), changements de version pour les outils éditeurs, changements d'interface. La durée moyenne d'un projet est de 3 à 4 mois. L'objectif principal des membres de l'équipe est de s'assurer du bon fonctionnement des applications.

L'équipe est composée d'une dizaine de personnes, chacune étant spécialisée sur un sous-ensemble d'applications (sur un périmètre bien défini pour chaque membre de l'équipe) et doit gérer plusieurs projets en même temps. Le temps de travail est divisé en temps pour les projets planifiés et en temps pour la gestion des imprévus. Les membres de l'équipe ont des compétences dans des technologies différentes et ont des différents niveaux de compétences. Ils ne sont pas toujours interchangeables les uns avec les autres. Certaines personnes travaillent sur un poste donné depuis plus de dix ans.

Les problèmes majeurs de cette équipe sont liés au manque de travail en équipe, aux difficultés de la gestion des imprévus en l'absence des personnes gérant telle ou telle application, à la gestion difficile de la charge des membres de l'équipe et au manque de visibilité sur le travail réalisé par chacun puisqu'aucun outil de suivi n'est mis en place pour la gestion de projets. Au moment de l'étude, une nouvelle organisation a été mise en place, à savoir que les équipes ont été redéfinies. Le nouveau chef d'équipe a souhaité mettre en place quelques éléments de la méthode SCRUM en mode progressif - sans parler des méthodes agiles ouvertement car les membres de l'équipe étaient réfractaires à ce type de changement.

4.2. Ordre d'intégration

L'étude réalisée au sein de cette entreprise est de courte durée (moins d'un an). Le processus d'intégration inclut trois étapes. En premier lieu, le chef de projet a entamé la mise en place de deux composants : *Gérer le projet avec la réunion journalière* et *Planifier le projet avec la réunion de Sprint*. En deuxième lieu, le composant *Planifier le projet avec le Backlog produit* (le sous-composant Planifier le projet avec la mise à jour continue du Backlog produit) a été introduit. En troisième lieu, le chef d'équipe a relancé l'intégration du composant *Planifier le projet avec la réunion de Sprint* (l'étape en cours au moment de l'étude). Les composants utilisés sont : Composant 1, Composant 4 et Composant 3 (3.2) de la liste des composants donnée plus haut. L'ordre d'intégration des composants a été défini par le chef d'équipe en se basant sur les priorités d'amélioration du fonctionnement de l'équipe. L'ordre d'intégration du deuxième cas d'étude est présenté sur la Figure 10.

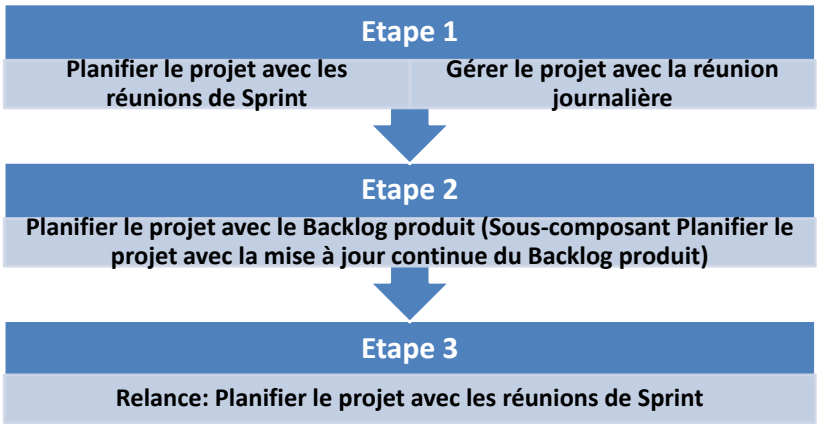

Figure 10. Ordre d'intégration des composants de méthodes agiles dans le cas de l'entreprise du secteur touristique.

Etape 1: Planifier le projet avec la réunion de Sprint et Gérer le projet avec la réunion journalière

La première étape est caractérisée par la mise en place simultanée de deux composants : *Planifier le projet avec la réunion de Sprint* et *Gérer le projet avec la réunion journalière*. L'objectif global de cette étape est de pallier le manque de visibilité et de stimuler le travail en équipe. En effet, les membres de l'équipe ne comprennent pas l'intérêt du travail en équipe et pensent que ce mode n'est pas adapté à leur contexte caractérisé par les projets en mode RUN sur les applications existantes avec une spécialisation forte entre les membres. Les besoins clés identifiés à cette étape étaient d'avoir une meilleure réactivité, remonter les alertes en temps réel sur le tableau, indiquer le statut des tâches en temps réel, anticiper sur l'imprévu.

Composant 1. Planifier le projet avec la réunion de Sprint

Contexte : Afin de stimuler le travail en équipe et d'améliorer la façon de distribuer les tâches entre les membres de l'équipe, la motivation de ces réunions consistait à identifier les remplaçants en cas d'absence prévue ou imprévue des membres de l'équipe pour assurer la continuité du travail, surtout sur les applications critiques. Le périmètre d'applications étant celui des finances-comptabilité, certaines applications sont très sensibles et les problèmes doivent être résolus rapidement, par exemple, lorsqu'il s'agit de la clôture de la période comptable ou bien des paiements des fournisseurs.

Processus : Une réunion de Sprint a été instaurée toutes les deux semaines. Il s'agissait de suivre les projets planifiés et de prendre en compte les imprévus les

plus considérables. Un tableau a été mis en place pour représenter les tâches à l'aide des post-its. L'état d'avancement (prévu, en cours, en attente, terminé) a été pris en compte en utilisant les différentes zones du tableau.

Résultat : La mise en place de ce composant a duré quatre mois. Le résultat est négatif car les réunions du Sprint n'ont pas été maintenues à la fin de cette période. Le problème majeur rencontré était la pertinence des informations remontées lors des réunions. Environ 50% de gens ont joué le jeu mais les autres étaient réticents à partager le travail qu'ils réalisaient : un des objectifs étant de rendre les gens remplaçables ils avaient donc peur de perdre leur travail.

Composant 4. Gérer le projet avec la réunion journalière

Contexte : Le besoin principal pour l'installation des réunions journalières est d'améliorer la gestion des alertes (des imprévus) au sein de l'équipe. Lorsqu'une alerte survient, il est indispensable d'analyser sa criticité et, en fonction de cette dernière, d'établir les délais de résolution du problème.

Processus : Le chef de l'équipe a entamé la mise en place de ce composant rapidement car le besoin était urgent. Lors de ces réunions, les tâches à faire ont été énumérées et les membres de l'équipe pouvaient choisir les tâches qu'ils voulaient prendre en charge. Le chef d'équipe avait pour rôle de guider cette sélection. L'objectif principal était de gérer les imprévus et d'affecter les tâches urgentes.

Résultat : La mise en place de ce composant s'est heurtée aux problèmes de motivation des employés. En effet, même si certaines personnes ont présenté leur avancement par rapport aux tâches assignées, les informations remontées n'ont pas été toujours pertinentes. Une partie du personnel était réfractaire aux changements et ne voulait pas que leurs collègues aient la visibilité sur leur travail en cours ; les autres jouaient le jeu. Les tentatives de mise en place des réunions journalières ont duré deux mois. Au final, les réunions journalières avec toute l'équipe se sont transformées en réunions personnelles (réunions tête-à-tête entre le responsable et une personne à la fois) et ont lieu une fois par semaine. Cela a permis d'avancer sur le problème de visibilité sur les tâches en cours et d'améliorer l'affectation de nouvelles tâches y compris des imprévus. Cependant le rythme n'est pas celui qui était attendu (hebdomadaire au lieu du journalier) et cela n'est pas fait en travail d'équipe.

Etape 2: Planifier le projet avec le Backlog produit

Contexte : L'objectif de cette étape est de mettre en place le Backlog produit pour être utilisé par le chef d'équipe et par les membres de cette équipe. Avant la mise en place du Backlog, rien n'avait été fait pour planifier et suivre la réalisation des projets de façon partagée et standardisée. Chaque membre de l'équipe utilisait sa propre façon de suivre ses activités s'il en avait besoin.

Processus : Afin de mettre en place le Backlog produit, un panneau mural a été installé pour positionner les post-its correspondant aux tâches de haut niveau. En effet, les tâches plus détaillées ne peuvent pas être suivies sur un tableau car elles sont trop nombreuses. Le niveau de granularité sélectionné est le sujet, de type « mise à jour de l'interface de l'application A ».

Résultat : La phase d'intégration a duré quatre mois. Les membres de l'équipe trouvaient la méthode bien dans sa globalité. Cependant uniquement la moitié des employés de l'équipe partageaient les informations sur leurs activités pour compléter le panneau. Les autres donnaient des informations incomplètes, parfois fausses, en argumentant sur le fait qu'ils ne voyaient pas l'intérêt de les partager avec tout le monde. Au résultat, le panneau existe mais seul le chef de l'équipe l'utilise en le complétant suite aux réunions individuelles. Etant donné que ce panneau n'est pas utilisé par toute l'équipe, nous pouvons constater la non-réussite de la mise en place de ce composant agile.

Etape 3: Relance: Planifier le projet avec la réunion de planification du Sprint (en cours)

Contexte : Le problème d'identification des personnes capables de remplacer les autres sur les applications et les projets clés n'étant pas résolu, le chef d'équipe a décidé de relancer les réunions du Sprint.

Processus et Résultat en cours : Cette fois-ci l'accent était mis sur la conduite du changement plus que sur les aspects méthodologiques de l'agilité. La recherche d'un outil convenable était en cours ainsi que l'étude d'une solution pour la gestion agile des imprévus. Le chef d'équipe souhaitait intégrer un outil collaboratif de type « mur virtuel » pour accompagner les membres de son équipe dans la mise en place des réunions du Sprint. Un travail explicatif a été mené afin d'assurer le personnel impliqué que les reformes n'avaient pas pour objectif de remettre en cause leur travail.

4.3. Conclusion de l'intégration

Malgré quelques éléments positifs (le progrès au niveau des interactions entre le chef d'équipe et le personnel, une meilleure visibilité du chef sur le travail réalisé et l'affectation des tâches urgentes contrôlée, même si le temps réel n'est pas toujours respecté), les résultats de l'intégration restent négatifs. Cela peut s'expliquer par les a priori du personnel, mais aussi par la nature des projets de l'équipe qui sont incompatibles avec l'agilité. Le travail en équipe qui est indispensable pour gérer les projets en mode agile n'est pas très développé au sein de l'équipe (environ 50% seulement y adhèrent). Nous avons constaté également que les composants de méthodes agiles sont mieux perçus par le personnel que les méthodes agiles car ils peuvent apporter des solutions aux problèmes en place sous condition d'une conduite de changement plus élaborée.

5. Leçons apprises

5.1. Résultats positifs

L'intégration progressive des composants de méthodes est possible. L'intégration des composants prévue dans la méthode d'origine a réussi complétement dans un cas et échoué dans le deuxième. L'échec dans le deuxième cas d'étude est dû aux problèmes organisationnels plutôt qu'aux composants agiles mêmes.

Acceptation des composants de méthodes. Il semble qu'une intégration graduelle amène à une bonne acceptation du changement, potentiellement meilleure qu'une intégration de méthode entière. Dans le premier cas, un changement doux, progressif, continu sur une longue période de temps, a permis d'introduire les changements petit à petit, l'un après l'autre, jusqu'à ce que tous les changements soient introduits (ou jusqu'à ce qu'il y ait une résistance forte au changement, résistance qui devra donc être gérée correctement avant de pouvoir continuer l'intégration). A la fin de cette expérience, les fonctionnalités manquantes de SCRUM avaient toutes été introduites dans la méthode de développement de l'équipe. Dans le deuxième cas, les gens n'ont pas rejeté les composants agiles comme ils le faisaient avec les méthodes agiles. Même s'il y a des personnes réfractaires aux changements ils ne l'ont pas exprimé ouvertement et l'intégration a pu avoir lieu. Nous pouvons conclure que l'intégration progressive a été mieux acceptée qu'une intégration classique dans ces deux cas avec des contextes organisationnels très différents. Il est également à noter que l'outil utilisé dans le premier cas a certainement été d'une grande aide lors de l'intégration des composants puisqu'il était déjà accepté par l'équipe de développement avant notre travail et que nous avons simplement implémenté les composants de méthodes en choisissant d'utiliser le même outil, ce qui a restreint les possibilités de résistance.

Meilleure considération des priorités de l'équipe. Dans le premier cas, nous avons étudié les fonctionnalités qui manquaient dans la méthode utilisée, mais nous avons aussi suivi de près la résistance au changement de l'équipe de projet. Nous avons alors priorisé les composants en prenant en compte l'importance de chaque fonctionnalité en fonction de l'acceptation potentielle des changements à intégrer.

5.2. Limitations

Base commune. Ce type d'intégration est seulement possible s'il n'y a qu'un seul projet en route au même moment, ou si les équipes de développement acceptent de modifier la méthode de développement pour tous les projets en même temps. Il y a des Sprints pour chaque projet et il serait vraiment difficile d'utiliser différentes méthodes en même temps pour les mêmes développeurs. Une base commune est nécessaire pour gérer correctement les développements. Cela a été

confirmé dans le deuxième cas, car la nécessité de gérer un grand nombre de projets implique de les considérer comme faisant partie d'un programme général et d'appliquer les méthodes agiles au plus haut niveau de granularité.

Contexte technico-organisationnel. Le rôle du contexte organisationnel et technologique est primordial pour la réussite ou non-réussite de l'intégration des méthodes agiles et/ou de leurs composants. Nous avons observé dans le deuxième cas qu'un certain nombre de facteurs technico-organisationnels (tels que la spécialisation technique des membres de l'équipe, le mode RUN, les délais différents des projets, le cloisonnement des personnes dans les différents bureaux, la réfraction aux changements, etc.) empêchaient la réussite de l'intégration progressive.

6. Conclusion et Travaux futurs

Ces expériences ont montré qu'une introduction progressive de nouveaux composants de méthodes dans une méthode de développement est possible et peut être plus facilement acceptée par les équipes de développement, mais peut avoir des difficultés liées à la conduite du changement.

Dans le premier cas, l'intégration a été faite de deux manières. Premièrement, les tâches de planification et d'estimation on été intégrées d'une façon très lente et progressive. Les changements améliorèrent les réunions hebdomadaires puisqu'elles sont passées de 3 heures à 2h, puis 1h30 (pour une équipe de 5-8 personnes). Cette amélioration est essentiellement due au fait qu'il n'y a plus de discussions sur des sujets où tout le monde est déjà d'accord. Il ne s'agissait finalement que de formaliser dans l'outil ce qui existait déjà dans le document Google. Deuxièmement, il a été intégré les problématiques de Backlog et de graphique d'avancement. Comme les premiers changements avaient été acceptés sans trop de difficulté, il était un peu plus facile d'introduire de nouvelles techniques dans le processus de développement. Ces changements améliorèrent l'estimation des tâches à moyen terme, induisirent une meilleure relation clientèle et un partage plus efficace et plus rapide des informations.

Dans cette expérience, l'intégration a amené à une amélioration de l'efficacité du travail de l'équipe. L'entreprise peut maintenant annoncer qu'elle utilise la méthode SCRUM dans le développement de ses projets. L'équipe est heureuse de travailler maintenant avec un processus plus formalisé qui permette une meilleure gestion du travail d'équipe, de la durée des réunions, de l'estimation des tâches et de la relation clientèle. Dans le premier cas, les clients ont un accès à une partie de l'outil utilisé et sont beaucoup plus impliqués dans le processus de développement.

Dans le deuxième cas, les résultats de l'intégration progressive sont plus complexes. Premièrement, la mise en place de certains composants agiles a pu avoir lieu malgré la réticence de certaines personnes aux méthodes agiles. Le progrès a été atteint en ce qui concerne la visibilité du travail et de l'affectation des tâches urgentes.

Cependant, pour cette étude, l'intégration progressive peut être considérée comme un échec à cause de l'esprit d'équipe peu présent parmi les participants. Cependant, l'intégration progressive se poursuit dans cette entreprise afin d'introduire un outil virtuel pour supporter les réunions du Sprint et de mener au mieux la conduite du changement pour identifier les remplaçants sans que le personnel ne se sente menacé.

Nos travaux futurs seront de gérer de nouvelles expérimentations dans d'autres entreprises pour confirmer les résultats de ces deux études et voir s'il est possible d'améliorer les composants identifiés. Contrairement aux expériences réalisées, les nouvelles expériences doivent être menées par un ingénieur de méthodes afin de pouvoir utiliser les techniques d'assemblage pour intégrer les nouveaux composants avec les méthodes en place. Nous supposons que la participation d'un ingénieur de méthode améliora les résultats d'intégration. Une attention particulière sera accordée aux facteurs organisationnels et technologiques qui peuvent accélérer ou ralentir l'intégration progressive. De nouvelles expérimentations nous permettraient de comprendre pour quels types de projets l'intégration progressive est appropriée et quels sont les facteurs d'une intégration réussie. Nous avons l'intention d'organiser de nouvelles expérimentations en utilisant un protocole d'évaluation formalisé dans le but de récupérer des informations sur l'efficacité de l'approche proposée.

References

Abad Z., Sadi M., Ramsin R. (2010), *Towards tool support for situational engineering of agile methodologies* In APSEC 2010, Asia Pacific

Abad Z., Alipour A., Ramsin R. (2012), *Enhancing Tool Support for Situational Engineering of Agile Methodologies in Eclipse*, Studies in Computational Intelligence, Vol 430, pp 141-152

Anderson D. (2003), *Agile management for software engineering: Applying the theory of constraints for business results*,

Beck K. (2000), *Extreme programming explained: embrace change*, Addison-Wesley, UK

Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R. C., Mellor S., Schwaber K., Sutherland J., Thomas D. (2001), *Manifesto for Agile Software Development.* Agile Alliance

Brinkkemper S., Saeki M., Harmsen F. (1998). Assembly Techniques for Method Engineering. *CAiSE 1998*, pp. 381-400, LNCS 1413, Springer Verlag.

Cervera M., Albert M. , Torres V. , Pelechano V. (2012), *A Model-Driven Approach for the Design and Implementation of Software Development Methods*. International Journal of Information System Modeling and Design (IJISMD), vol. 3(4), pp. 86-103.

Cockburn A. (2002), *Agile software development*, Addison-Wesley, London, UK

Cohn M. (2005), *Agile estimating and planning*, Prentice Hall editors

Deneckere R. (2001), *Approche d'extension de méthodes fondée sur l'utilisation de composants génériques*. PhD thesis. University of Paris 1 Panthéon-Sorbonne

Deneckere R., Kornyshova E., Ralyté R. (2014). *Famille de méthodes: la flexibilité au cœur du processus de construction de méthodes*. Journal Ingénierie des Systèmes d'Information, vol. 19(1), pp. 67-95

Deneckère R., Kornyshova E., Iacovelli A. (2016a). *Progressive integration of method components: A case of agile is development methods.* In: Proceedings of IEEE Research Challenges in Information Science (RCIS 2016), Grenoble, France, pp. 1-10

Deneckère R., Kornyshova E. (2016b) Evaluation du concept de Famille de Méthodes et de son Application aux Méthodes Agiles, Journal Ingénierie des Systèmes d'Information, vol. 21(2), pp. 95-121

Firesmith D.G., Henderson-Sellers, B. (2002). *The OPEN Process Framework: An Introduction*. Addison-Wesley, London, UK, 330 p.

Guzélian G., Cauvet C. (2007), *SO2M: Towards a Service-Oriented Approach for Method Engineering*. In: Proceedings of IKE'07, Las Vegas, Nevada, USA

Henderson-Sellers B., Ralyté J., Agerfalk, P. Rossi M. (2014) *Situational Method Engineering*, Springer

Iacovelli A. (2012). *Approche orientée service pour la configuration de méthodes outillées.* PhD thesis. University Paris 1 Panthéon-Sorbonne

Karlsson F., Ågerfalk P.J. (2004), *Method Configuration: Adapting to Situational Characteristics While Creating Reusable Assets*. Inf. and Soft. Technology, vol. 46(9), pp. 619–633.

Karlsson F., Ågerfalk P. J. (2008), *Method Configuration: The eXtreme Programming Case*, in XP 2008, Limerick, Ireland, pp 32-41

Kornyshova E., Deneckère R., Rolland C. (2011a) Method Families Concept: Application to Decision-Making Methods, Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD), London, United Kingdom.

Kornyshova E., Deneckere R., Claudepierre B. (2011b). Towards Method Component Contextualization. *International Journal of Information System Modeling and Design (IJISMD)*, IGI Global, 2 (4), pp.49-81.

Kumar K., Welke R.J. (1992). Methodology EngineeringR: A Proposal for Situation Specific Methodology Construction. *Challenges and Strategies for Research in Systems Development*, Cotterman, W. and J. Senn (eds.), J. Wiley, Chichester, UK, pp. 257-266.

Kuhrmann M., Mendez-Fernandez D., Tiessler M. (2014), A mapping study on the feasibility of method engineering, Journal of Software: Evolution and Process, Volume 16, Issue 2, pp. 1053-1073.

Meyer B. (2014), *Agile!: The Good, the Hype and the Ugly*, Springer Publishing

Nehan Y-R., Deneckere R., (2007) Component-based situational methods, ME'07, Genève, Suisse

Qumer A., Henderson-Sellers (2007), *Construction of an Agile Software Product-Enhancement Process by Using an Agile Software Solution Framework (ASSF) and Situational Method Engineering*, In COMPSAC 2007, Beijing, China, pp 539-542

Ralyté J., Rolland C. (2001), *An Assembly Process Model for Method Engineering*. In: Proceedings of CAISE 2001, LNCS 2068, Springer, Berlin, pp. 267-283.

Rossi M., Ramesh B., Lyytinen K., Tolvanen J.-P. (2004), *Managing evolutionary method engineering by method rationale*. Journal of the AIS, vol. 5(9), pp. 356-391.

Schwaber K. , Beedle M. (2002), *Agile Software Development with Scrum*, Prentice Hall PTR, Australia

Serena Software (2007), *An Introduction to Agile Software Development*, Inc. Serena, Mariner, TeamTrack, http://www.serena.com (accessed in march 2015)

Shore J. (2007), *The art of agile development*, O'Reilly Media Editors

Stapleton J. (1995), *DSDM −Dynamic system development method*. Addison-Wesley, UK,

VersionOne (2013), *8th annual state of agile development survey*, http://www.versionone.com (accessed in Nov. 2014)