Automne 10

Peut-on utiliser l'agile partout?

Le cas de l'application à une démarche de R&D autour de services innovants

Martin Bahier

Ce document est une thèse professionnelle ayant pour sujet principal l'agile, comment les méthodes issues de l'agile sont appliquées, et quel regard nous pouvons porter sur ces méthodes.

Cette thèse professionnelle tentera d'aborder la question autant d'un point de vue théorique que pratique, à la fois au travers de la littérature existant sur le sujet, mais aussi en s'appuyant sur des exemples concrets (expérience personnelle et retours d'expérience d'autres personnes).

Table des matières

I. Résumé et mots clés / Abstract and Key words

1. Version Française

a. Résumé

Dans cette thèse professionnelle, il est question de savoir si les pratiques Agiles peuvent être utilisées dans un autre contexte que le développement informatique. Après la définition d'un vocabulaire commun sur l'Agile et de quelques unes des méthodologies de gestion de projet « à planification forte » ayant cours actuellement seront abordées les conditions d'applicabilité de l'Agile d'une manière plus générale ainsi que ce que j'ai retiré de mon expérience personnelle chez Octo Technology dans un contexte de « R&D ». Cette thèse professionnelle va tenter de montrer que, si l'Agile ne peut pas s'utiliser dans absolument tous les domaines (pour diverses raisons), ont peut l'utiliser au moins partiellement dans des domaines qui ne sont pas nécessairement liés à l'IT.

b. Mots clés

Agile, Recherche et Développement, intérêt en dehors de l'informatique

2. English version

a. Abstract

This professional thesis is about trying to figure out if the various practices of agile methods can be used in a context that is different from computer software development. After the definition of a common vocabulary about Agile and some of the most common plan driven project management methodologies nowadays I will discuss about how Agile methods could be applied in a more general context and about my personal experience during my internship at Octo Technology concerning an "R&D" field. This professional thesis will try to demonstrate that, if it may not be possible to use Agile methods for any project (for various reasons), these methods can, at least, be partly used in fields that are not directly linked to IT.

b. Key words

Agile, Research and Development, purpose outside an IT perspective

II. Introduction

1. Intérêt du sujet et du stage

L'ensemble des méthodes agiles forme un corpus de méthodologies dont s'inspirent de plus en plus les sociétés qui cherchent une organisation et une gestion de projet plus réactives que les méthodologies classiques telles que le cycle en V à passe unique. Les méthodes agiles mettent l'utilisateur final au centre du processus et tendent, de ce point de vue, à dissiper la frontière entre client et « fournisseur ».

Mener une démarche de recherche et développement met en jeu, finalement, les mêmes acteurs que ceux que l'on peut retrouver sur des démarches plus en « aval » : il y a un commanditaire, des « exécutants », divers intermédiaires et, à la fin de la démarche, il est demandé une forme de « livrable » qui devra être recetté et dont, comme pour un produit classique, on devra gérer la vie après le processus de création.

Il est donc intéressant, à priori, de se poser la question de l'impact des méthodes agiles sur un processus de réflexion en amont, de ses avantages, ses inconvénients et des conséquences que cela peut éventuellement avoir sur la vie d'un produit une fois sorti de son département de R&D.

Mon stage chez Octo Technology m'a permis de pratiquer l'Agile au quotidien au milieu de consultants et d'architectes experts dans ce domaine. Le sujet de mon stage étant « Le Web 2.0 dans les telcos », j'ai été amené à suivre une démarche de R&D (veille, réalisation de prototypes …) encadrée par les méthodes Agiles tout au long de mon stage, et son organisation a été pensée pour que tous mes travaux soient encadrés en utilisant la panoplie d'outils mis à disposition par les méthodes « Agiles ».

2. Remerciements

Je tiens à remercier tout particulièrement Emilie Chabbouh et Yannick Martel de m'avoir confié ce stage. Et je les remercie aussi tant pour leurs conseils éclairés pendant mon travail à Octo, que pour l'aide précieuse qu'ils m'ont apportée, de façon formelle ou au détour d'une conversation, pour la rédaction de cette thèse professionnelle. Je tiens aussi à remercier Guillaune Duquesnay, Olivier Hoareau et Dorian Yahouédéou pour leur temps et leurs conseils à la fois techniques et méthodologiques. Enfin, un grand « merci » à tous les Octos avec qui j'ai passé ces derniers mois (dont Vincent Coste, Jean-loup Yu, William Montaz, Jean-François Grang, Cédric Pointel, Maxence Walbrou, Mikaël Robert, Florent David, Rémy Virin, Jonathan Scher, …) pour leur bonne humeur et leurs encouragements.

3. Avant-Propos

Le contenu de cette thèse professionnelle, est le résultat de recherches personnelles menées en parallèle et pendant le stage que j'ai réalisé chez Octo Technology. Il se fonde, à la fois, sur les conférences auxquelles il m'a été donné d'assister ainsi que sur les

discussions que j'ai pu avoir avec certains « Octos[1] » sur le thème de l'agile dans les projets. J'ai, par ailleurs travaillé avec la littérature existant déjà sur le sujet (dans des publications écrites comme sur le web). Bien que j'aie mené ces recherches de la façon la plus consciencieuse possible, il n'en reste pas moins que certains thèmes abordés peuvent faire l'objet de plusieurs interprétations. Je ne prétends pas que ce document (ou mon opinion d'ailleurs) puisse refléter la « vérité absolue » sur la gestion de projet et sur le rôle des méthodologies issues de l'Agile dans cette gestion. Ce document représente, tout au plus, une opinion, que j'espère la plus raisonnée et la plus cohérente possible, sur le sujet.

[1] Nom donné aux personnes travaillant chez Octo Technology

III. Synthèse et recommandations

Les méthodes agiles peuvent-elles être utilisées partout ?

"Sur le papier"

Le succès d'une équipe Agile semble énormément reposer sur le « pari de la communication » qui est fait par l'Agile et suppose que toutes les parties prenantes sur un projet Agile arriveront à communiquer efficacement et à bon escient pendant tout le long d'un projet. Divers outils sont mis à disposition des équipes à cet effet (kanban boards, techniques de réunion face á face, pair programming …) pour exploiter mais améliorer la communication entre les divers membres d'un projet. Certains outils vont aussi permettre de mitiger les effets d'un contexte projet « non idéal » pour l'Agile. Cependant, tous les outils ne s'adaptent pas toujours bien à tous les environnements il faut donc faire très attention au contexte du projet lui même avant de décider d'utiliser tel ou tel outil provenant de l'Agile ou de décider de l'adapter à son propre contexte.

De plus, si l'Agile peut parfaitement convenir à certaines personnes, d'autres ne seront peut être pas aussi à l'aise avec ou seront source de problèmes dans un contexte Agile (les personnes empêchant le groupe de s'exprimer comme des « héros » ou des « dominateurs » cherchant trop à être le centre de l'attention). Or l'un des objectifs de l'agile d'une manière générale est de mettre l'accent sur les personnes et les relations de travail qu'elles peuvent avoir, il est donc essentiel que chacun adhère aux principes de communication valorisés par l'Agile si l'on souhaite faire fonctionner une équipe selon ce mode.

Faire de l'Agile, comme utiliser n'importe quelle nouvelle méthode de gestion de projet, ne s'improvise pas, aussi une équipe qui ne connaît pas du tout les principes de l'Agile aura du mal à fonctionner à plein régime dans un premier temps puis, avec la pratique, finira par devenir efficace si elle adhère aux concepts et à la « philosophie » qui se cachent derrière l'Agile. Comme pour tout « gros » changement, il faut se préparer à un démarrage ou tout n'ira pas aussi bien qu'avant, pour voir progressivement la situation s'améliorer. Débuter en Agile a un coût d'entrée à ne pas négliger, ce n'est pas parce que l'on remplit globalement moins de documents que dans un projet géré selon les recommandations PMI qu'il est plus facile de faire de l'Agile.

Si l'Agile se destine principalement aux projets de développement informatique (elle est née des besoins de certains de ces derniers), il semble que ses aspects principaux (développements itératifs avec livraison fonctionnelle à chaque itération, cycle de feedback, mode de gestion de la complexité …) puissent être adaptés à d'autres contextes. Typiquement cela semble, au vu de mon expérience, possible pour certains travaux de R&D.

Dans la pratique

Les méthodes et outils issues de l'Agile peuvent se révéler très utiles lorsque l'on mène une démarche orientée R&D. Cependant, la panoplie complète des outils

agiles n'est pas forcément actionnable immédiatement. Typiquement pour des mesures de performance comme celles issues d'un backlog il faut disposer d'une certaine expérience ainsi que de plusieurs itérations « d'amorçage » pour en profiter à plein (l'expérience seule ne permet pas d'avoir immédiatement des mesures fiables si l'équipe est nouvelle ou change). On peut aussi se demander si le terme de « performance » a bien sa place quand on parle de R&D, en un sens on peut répondre que oui, quand par « performance » on veut dire « travaux retravaillés autant que nécessaire pour atteindre un standard de qualité fixé avec le client 'final' des travaux et livrés sous une forme convenue ». Ici, de la même façon que l'on mitige les risques de non-qualité sur un projet plus « concret », les itérations courtes et l'implication du client final peuvent (et doivent) aider à produire des travaux de recherche de qualité. On fera cependant attention à la nuance qui existe entre « montrer au client ce qu'il souhaite voir » et « faire l'exposé honnête aussi précis et exhaustif que souhaité d'un sujet de recherche ». De la même façon que Régis Medina dit qu'il faut en permanence se demander « qu'est-ce que je fabrique ?» et ne pas opiner à la moindre demande du client.

En ce qui concerne un contexte plus « standard » de l'agile (le développement logiciel), on peut se rendre compte qu'il y a de multiples facteurs qui influencent le bon déroulement d'un projet agile mais que, de tous ces facteurs, un revient constamment : la bonne communication des membres de l'équipe. Le contexte, les outils et, bien sûr, les personnes participant au projet contribuent au fait que la communication peut se faire dans de bonnes conditions et permettre aux outils agiles de fonctionner correctement.

Recommandations

Au vu de ce que j'ai pu lire sur le sujet et expérimenter par moi même, il semble que l'on puisse utiliser les outils caractérisant l'agile dans d'autres cadres que celui du développement logiciel (comme la R&D par exemple). Cependant, au vu du cadre « optimal » qui se dessine quand au bon fonctionnement d'une équipe Agile, on pourrait dire que, pour avoir une chance d'utiliser l'Agile (ou en tout cas la partie de ses outils pertinents pour ce que l'on cherche à faire), il faut :

- Une équipe de petite taille et qui est efficace dans la façon dont les uns communiquent avec les autres ;
- Avoir une vision de la planification qui consiste à enchaîner des cycles d'activité courts (2 mois maximum) prenant en compte un feedback extérieur pour les valider ;
- Un périmètre d'objectifs à atteindre « convertible » en User Stories : l'objectif est utile en soi, vient s'intégrer dans un « tout » plus imposant et peut être refait si nécessaire. Il est achevable dans la durée d'un cycle d'activité;
- Prioriser ces User Stories pour maximiser la valeur de ce qui ressortira de la démarche ;

- L'estimation des items à réaliser se fait en comparant la complexité (technique, recherches documentaires nécessaires …) de réalisation de ce dernier par rapport à d'autres qui font office de « référence » ;

Cela va typiquement désigner des projets où l'on peut utiliser des équipes de taille plutôt réduite (feedback et prise de décision rapides) et dont les lots vont pouvoir être organisés et réalisés itérativement puis retravaillés sans induire un surcoût trop important pendant les intervalles (courts) choisis, ce qui exclurait à priori de facto les grandes réalisations techniques comme la construction de moteurs à réaction par exemple où certaines pièces « prototypes » vont coûter une fortune et où il ne sera juste pas possible de tenir la cadence des itérations courtes d'un point de vue financier ...

Cependant, si l'on s'écarte un peu des considérations standard sur la durée des itérations en Agile, on peut se demander si une itération doit être « courte » quand on regarde son calendrier ou quand on regarde la durée totale du projet. Par certains aspects (livraisons itératives entre autres) des programmes comme Apollo se rapprochent de ce que l'on peut trouver dans le déroulement d'un projet agile (priorisation, réalisation par incréments successifs, de boucles de feedback …). C'est bien sûr un grand écart par rapport à ce que l'on trouve dans le développement logiciel agile, et est sans doute commun à d'autres façons de gérer projets et programmes « au long cours » car on a ici affaire à des pratiques qui vont favoriser l'innovation (de tels projets en ont justement besoin pour ne pas être complètement dépassés avant même d'être achevés) mais cela mérite d'être noté. Le principe même de l'Agile étant de fournir une « boîte à outils » et un certain nombre de principes pour en optimiser l'usage, on peut penser que, si certains domaines semblent « interdits » à l'agile, ils ne le sont pas tous. Il est évident qu'il faudra faire des compromis sur ces projets « particuliers » pour de l'agile (et qu'il faudra oser tenter l'aventure) mais « sur le papier » cela semble possible.

IV. Octo Technology

1. Présentation de la société

Fondé en 1998, OCTO Technology est un cabinet de conseil en systèmes d'information. Son activité principale se divise en plusieurs coeurs de métier : le conseil en architecture de systèmes, la conception de produits et l'accompagnement de projets informatiques. La plupart de ses clients sont des grands comptes dans les domaines de la banque, l'assurance, les médias, les opérateurs télécoms et l'industrie. OCTO Technology est actuellement coté en bourse sur le marché parisien Alternext et est dirigée par ses membres fondateurs.

En termes de positionnement stratégique, OCTO Technology se situe assez singulièrement entre le monde des sociétés de service conventionnelles et celui des cabinets de conseil. On pourrait la décrire comme une « SSII haut de gamme » centrée sur le conseil. OCTO Technology vend pour se démarquer une prestation d'excellence. La devise de l'entreprise est d'ailleurs "Apprenons à jouer l'excellence" et fait directement référence à l'expertise et le soin apportés sur chacune des missions. Dans le même temps, l'accent est mis sur le plaisir que prennent les consultants à effectuer leurs missions. Le crédo en vigueur chez OCTO Technology est qu'une équipe que le cabinet va aider sur un projet doit être capable de se débrouiller toute seule une fois les « OCTOs » partis. Ceci implique un grand niveau d'expertise de la part de ses consultants et une montée en compétences constante afin de pouvoir toujours apporter quelque chose de différent et de nouveau dans les sociétés qui lui demandent son expertise. Cette montée en compétences s'explique en partie par le fait qu'il existe une plage de temps réservée à la R&D chez OCTO, et ce pour tout un chacun : on est encouragé à participer au blog, à la rédaction de livres blancs, à travailler sur le dévelopement des outils internes à OCTO, à la formation à des nouvelles technologies, et à la contribution à des articles de presse.

Depuis sa création, OCTO Technology jouit d'une croissance annuelle de 20% de son chiffre d'affaire (15,4 M d'euros en 2009); cette tendance s'est confirmée même dans des périodes un peu moins évidentes telles que la crise qui a impacté le secteur informatique au cours des deux dernières années. Cette évolution constante se traduit notamment au niveau de l'organisation par une augmentation régulière du nombre de salariés. Au second trimestre 2010, l'effectif total se monte à 161 personnes, à mettre en balance avec l'équipe fondatrice de 1998, forte de 8 passionnés (d'où le nom OCTO Technology). Le siège de la société est basé à Paris, les bureaux étant situés sur les Champs-Elysées. Deux filiales, en Suisse et au Maroc, complètent le positionnement international de l'entreprise. Il faut enfin souligner une volonté marquée de se développer sur d'autres marchés en dehors de l'Hexagone. Une campagne de prospection a d'ailleurs démarré en Amérique du Sud et devrait aboutir à la mise en place d'une cellule locale.

En interne, l'organisation est segmentée suivant des LOBs (Line Of Business) qui reprennent les différents secteurs d'intervention des consultants : l'assurance, la banque, l'industrie, les médias, et les télécoms, mais aussi des LoBs plus transverses (« Usabilité » par exemple). A noter que ces secteurs sont ceux sur lesquels OCTO Technology est historiquement implanté, car l'entreprise a réussi a fidéliser la plupart de ses clients.

Les salariés représentent chacun des métiers du conseil en informatique. On retrouve donc en proportion quasi équivalente des consultants, des experts, des architectes juniors et séniors ainsi que quelques experts séniors. Enfin une équipe de communication est là pour promouvoir les actions de la société et mettre en avant les services qu'elle propose via, entre autres, le site internet de la société. Fait marquant, il existe une certaine mixité au sein de l'entreprise même si celle-ci est surtout favorisée par les équipes des ressources humaines et de la communication. Il y a néanmoins quelques consultantes en mission chez les clients.

En marge de son activité de SSII, assez classique, OCTO Technology organise de nombreuses manifestations, à destination de ses employés, de ses clients et même de consultants de d'autres sociétés. On peut ainsi mentionner les nombreuses formations dispensées tout au long de l'année qui couvrent des domaines aussi bien techniques que managériaux. Dans la même optique, des petits-déjeuners à thème et des conférences sont organisés régulièrement.

En interne, les consultants peuvent participer à des BOF (Bird Of A Feather), qui se présentent sous la forme d'un exposé technique autour d'un thème donné ou qui mettent en exergue un retour d'expérience de mission.

Des initiatives permettent également aux consultants de découvrir les spécificités du métier des autres LOBs : les écoles de la Supply Chain, de la Banque et de l'Assurance fournissent lors de présentations informelles des éléments introductifs, des clés de lecture du quotidien de ces univers.

Enfin, depuis 3 ans, l'entreprise met sur pied chaque année un évènement majeur : l'Université du Système d'Information (l'USI). Il s'agit d'un ensemble de conférences au cours desquelles diverses personnalités issues du monde de l'informatique, des chefs d'entreprise ou encore des intellectuels de divers horizons prennent la parole et exposent leurs idées sur différents thèmes. L'objectif est de fournir à l'auditoire des conférences de qualité sur des sujets articulés autour de quatre thèmes principaux. En 2010, les thèmes retenus sont "Innovant, Durable, Ouvert et Valeur". Cet évènement rassemble plusieurs centaines de personnes sur deux jours, début juillet.

2. Mon travail chez OCTO Technology

J'ai intégré OCTO Technology en tant que stagiaire au sein de la LoB TS[2] où j'ai immédiatement commencé à travailler en mode « agile » (livrables réguliers, utilisation d'un Kanban simplifié, rédaction et mise à jour d'un backlog ainsi que d'un burndown chart…) de sorte que j'ai très vite pu me rendre compte de ce qu'impliquait, au niveau méthodologique, de travailler en agile depuis l'étape de la R&D.

Mon stage contenait deux composantes essentielles :

- Une partie Veille / R&D « pure » où il était question de réunir un certain nombre d'informations en rapport avec la définition et les usages faits du « Web 2.0 » (de l'élaboration d'un vocabulaire commun à l'isolation de « bonnes pratiques » à destination des opérateurs Telcos français). C'est aussi durant ces périodes qu'ont été proposées des idées d'amélioration des services 2.0 existants chez les opérateurs telcos français, dont certaines ont été concrétisés avec la réalisation de prototypes ;
- Une partie « Élaboration de prototype » (Spécifications, étude de faisabilité technique, et production), basées sur les idées d'amélioration de l'existant ou de nouvelles fonctionnalités proposées pendant une des parties de « Veille / R&D » ;
- De façon plus transverse, j'étais chargé de gérer mon travail en mode « projet » :
    - Organiser les « points client » et les recettes du travail effectué ;
    - Mesurer mes performances d'une itération sur l'autre et pouvoir justifier auprès de mon « Product Owner » le fait de m'engager à effectuer certaines tâches pendant une itération ou de suggérer d'en découper certaines car les estimant « trop grosses » pour pouvoir être achevées en une seule.

À ceci est venu s'ajouter une participation au développement d'un outil interne à la société (Octopode NG), ou j'ai pu observer la mise en œuvre de quelques uns des outils « agiles » les plus répandus (TDD, Kanban & pair-programming essentiellement) dans une situation « réelle » (comprendre : travailler sur un projet complexe avec un client qui n'est pas son responsable direct et avec des contraintes de temps « non négociables »

[2] Line of Business Télécoms et Services Publics

pour livrer).

Chaque étape de mon stage a été gérée à la façon d'un petit projet avec des objectifs à atteindre entre deux itérations et des livrables dont j'étais responsable, les rôles de product owner et de client étant joués par les personnes chargées de m'encadrer. Ce « mode projet » m 'a permis de mettre en application ce que j'ai pu apprendre ces derniers mois durant mon cursus à Telecom Paris et à l'ESSEC soit d'une façon directe (stratégie, prospective, management de l'innovation, animation de réunions …), soit en me permettant d'assimiler rapidement les nouvelles notions méthodologiques qui m'ont été enseignées pendant le stage lui même, avec l'aide de mes responsables et en les utilisant.

V. L'Agile, c'est quoi ?

Définition d'un vocabulaire commun de ses grands principes et de ses outils

L'Agile est un terme qui désigne avant tout une philosophie, un corpus de méthodologies plus qu'une seule et même méthodologie universelle et un ensemble d'outils. Cependant, il y a un certain nombre de termes qui reviennent régulièrement et sont profondément ancrés dans la philosophie agile. La liste de définitions qui suit tente de décrire les plus emblématiques d'entre eux :

1. Lean Management

Pour Jim Womack (MIT, président du Lean Enterprise Institute) comme pour Emmanuel Chenu (Coach en développement logiciel chez Thales Aerospace), le Lean, c'est avant tout, dans l'idéal, résoudre les problèmes du client de la façon la plus complète et la plus exacte possible : il s'agit de comprendre avec lui de quoi il a exactement besoin afin de l'aider à mieux exprimer ce besoin pour pouvoir produire ce qu'il veut, le tout sans pour autant en faire trop . Cela doit se faire avec un minimum de gaspillage : Il faut, autant que possible, produire exclusivement des choses qui, une fois assemblées, ont une valeur « tangible » pour le client : on lui rend le service qu'il demande et pour lequel il accepte de payer mais en revanche on fait tout pour éviter des opérations comme, par exemple, du rework[3] sur des parties du projet qui sont censées avoir déjà été complétées, optimiser sa façon de travailler pour limiter les périodes d'inactivité de certaines équipes, travailler avec le moins de stocks possible (que l'on parle de stocks physiques ou de stocks d'idées « à réaliser »), ou encore du « gold plating[4] » qui, pour le coup, est juste inutile.

Le Lean c'est aussi produire ce que le client attend de l'équipe projet au moment ou il le souhaite et, pour certains projets, là où il le souhaite. C'est enfin réduire au minimum tous les aspects de « prise en main » (lead time d'utilisation) par le client : tout ce qui est « perte de temps » pour le client est un gaspillage à éviter.

Le Lean recommande d'être sur le terrain (« gemba ») pour voir comment cela se passe dans la « vie réelle » et trouver des axes d'amélioration en fonction de ce que les utilisateurs finaux parviennent ou pas à faire avec les produits que l'on crée pour eux. Le gemba doit aussi être l'occasion de s'interroger sur ses processus et de se demander si, oui ou non, chacun individuellement permet de générer de la valeur pour le client.

Selon Jim Womack, un process est dit « Lean » s'il est :

- « Valuable » : Il a de la valeur pour le client en tant que tel (celle que le client souhaite)
- « Capable » : Il produit le résultat escompté à chaque fois et ne requiert pas d'ajustements

[3] Retravailler un élément qui a déjà été livré

[4] Réalisations n'étant pas demandées par le client (que cela soit de façon explicite ou implicite)

- « Available » : Il est disponible quand le client le souhaite (Une fois livré, le produit fonctionne sans problème et est toujours disponible quand le client a besoin de s'en servir)

- « Adequate » : Le processus est capable de tenir la charge. Il n'existe aucune sur ou sous-capacité qui l'empêche de se dérouler de façon optimale à tout moment

- « Flexible » : Il est capable de s'adapter aux besoins du client

Le Lean comprend 5 étapes dans la façon que l'on a de mettre en place les techniques qui en sont issues :

a. Préciser la valeur dégagée par une famille de produits en se basant sur ce que souhaite le client final.

b. Identifier toutes les étapes contribuant au processus de création de valeur d'une famille de produits, tout en éliminant dès que possible les étapes ne créant aucune valeur.

c. Faire en sorte de créer des « paliers » de création de valeur, en les faisant s'enchaîner le plus rapidement possible de sorte à livrer le moins « brutalement » possible le produit fini au client.

d. Laisser les clients profiter de la valeur créée dans la dernière activité en cours au fur et à mesure que l'activité s'accomplit.

e. Identifier des enchaînements de processus permettant de créer de la valeur et retirer ceux qui n'apportent rien à la valeur finale du produit. A force d'itérer sur les processus et de retirer ceux qui ne créent pas de valeur, on se rapproche d'un enchaînement de processus « parfait » ou la valeur qui en est issue est crée sans gaspillages (on évite les gaspillages dans la « façon d'exécuter »).

2. Méthodes agiles

Dans le monde « agile », un certain nombre de principes ont été pensés comme étant absolument essentiels à la bonne marche d'un projet, comme l'alignement des équipes ainsi que la satisfaction du client par le dégagement au plus tôt d'un maximum de valeur. Ces principes ont été résumés sous la forme de 4 valeurs fondamentales et 12 principes dans le « manifeste agile » et constituent en quelque sorte une ligne de conduite à respecter pour tout « agiliste » :

Les quatre valeurs fondamentales Agiles sont :

- *Préférer l'interaction avec les personnes aux processus et aux outils.*
- *Préférer un produit opérationnel à une documentation pléthorique.*
- *Préférer la collaboration avec le client à la négociation d'un contrat.*
- *Préférer la réactivité au changement au suivi d'un plan.*

Les 12 principes

«

- *Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles.*
- *Le changement doit être accepté, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client.*
- *Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte.*
- *Les experts métier et les développeurs doivent collaborer quotidiennement au projet.*
- *Bâtissez le projet autour de personnes motivées. Donnez leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail.*
- *La méthode la plus efficace pour transmettre l'information est une conversation en face à face.*
- *Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet.*
- *Les processus agiles correspondent à un rythme de développement soutenable. Commanditaires, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment.*
- *Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité.*
- *La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle.*
- *Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent.*
- *À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens. »*

(Source : http://fr.wikipedia.org/wiki/Manifeste_agile)

Comme il est précisé dans chacune des valeurs, il n'est pas dit ici que les valeurs « à la droite des phrases » ne sont pas importantes, mais juste que, à choisir, on leur préférera celles de gauche.

L'agile va favoriser les interactions entre les acteurs d'un projet plutôt que la conformité à un « plan ». Cela ne veut pas dire que le plan n'est pas important, juste que l'on pense que la meilleure façon de réussir le projet est surtout de parvenir à faire travailler ensemble et de la façon la plus saine possible tous les acteurs d'un projet, plutôt que de suivre aveuglément un plan qui ne sera peut être plus si pertinent quelques mois plus tard.

Les méthodes agiles sont un moyen de faire discuter la MOA et la MOE sur toutes les étapes concernant le « concevoir » et le « réaliser » dans un projet. Elles sont aussi une certaine expression du lean management au niveau « projet » (au niveau de la conception d'un logiciel typiquement).

3. Kanban

Le Kanban est un outil du lean management qui, selon Taichi Ohno[5] est un des moyens permettant d'atteindre le JIT (Just In Time). Il permet à un ensemble de processus de fonctionner en flux tiré en mettant en place un certain nombre de règles que toutes les équipes travaillant sur un projet se doivent de respecter.

Le Kanban est, à la base, une façon de représenter graphiquement des informations sur des cartes. Il peut être utilisé au sein d'usines pour décrire des tâches ou surveiller des stocks (Dans les usines Toyota on a un système de Kanban de ce type pour surveiller les stocks, garder trace des cartes Kanban semble pouvoir être plus efficace que de garder trace des stocks eux mêmes), mais il peut également servir d'outil de gestion de l'état d'avancement d'un projet qui couvre alors toutes les activités (et tous les départements) de ce projet depuis le département marketing et les premières idées sur un nouveau produit jusqu'à la réalisation et enfin jusqu'à la livraison.

Derrière le Kanban se profilent 4 principes :

- Visualiser la production actuelle
- Identifier les goulots d'étranglement et les traiter si l'on en trouve pour lisser l'activité
- Améliorer les processus de production en continu par l'utilisation du flux tiré
- Atteindre le « Kaizen » (amélioration continue) à partir des 3 éléments précédents : il faut optimiser la façon dont se déroulent tous les processus pour obtenir l'activité la plus lissée possible et, de ce fait, éviter les à-coups de production et donc les gaspillages de toutes sortes (stocks, personnes inactives …)

Le Kanban sert à matérialiser les tâches en cours dans un projet et à savoir qui s'en occupe. Dans une usine, un Kanban peut prendre la forme de pancartes disséminées un peu partout.

Dans le cas de la gestion des tâches à effectuer (dans un projet informatique typiquement), cela prend en général la forme d'un tableau (un peu comme la représentation d'une usine et de ses différentes parties) recouvert de post-it symbolisant des User Stories à compléter. Ce tableau est sujet à plusieurs règles :

- Le kanban va en général être découpé en plusieurs zones, typiquement selon les équipes (ou les aspects « métier » concernés).
- Il existe des tâches « normales » et des tâches « critiques » (en général en retard). Pour chaque zone on peut décider de faire passer un certain nombre de tâches en statut « critique ». Ce nombre est désigné à l'avance et est en général fait pour ne pas tomber dans le travers de

[5] Ingénieur chez Toyota décédé en 1990, considéré comme le « père » du système de production Toyota

planification qui consiste à estimer que tout est critique (si toutes les tâches deviennent critiques, la notion de criticité n'a alors plus aucun sens)

- S'il y a une tâche « critique » pour une équipe, elle doit se concentrer exclusivement sur celle ci avant d'envisager autre chose.
- Il y a un nombre maximum de tâches que l'on peut faire entrer dans chaque zone du Kanban et ce afin d'éviter les goulots d'étranglement. L'idée sous-jacente est que les équipes de chacun des domaines couverts par le projet ne travaillent pas forcément à la même vitesse (parfois tout simplement parce que cela n'est physiquement pas possible). Cela amène chaque équipe, de façon incidente, à prioriser son travail de sorte à produire très tôt les éléments les plus importants (ceux produisant la plus forte valeur ajoutée) pour que la production à un instant *t* intègre la plus grande valeur ajoutée possible en fonction de ce que l'on souhaite produire.
- D'autres règles encore peuvent venir se greffer sur un Kanban mais elles sont en général spécifiques à un projet. L'idée est de mettre à disposition de tous un outil adapté plutôt qu'un frein.

Une telle représentation graphique permet à chaque personne qui travaille sur le projet de pouvoir facilement consulter son avancement ainsi que les éventuels points de blocage et ce sans être technicien ou spécialiste en méthodologie. Le principal but d'un tel outil est que l'état d'avancement du projet soit « abordable » pour n'importe qui, y compris quelqu'un ne faisant pas partie de l'équipe projet.

Une telle représentation fournit un support autour duquel toutes les équipes travaillant sur le projet peuvent discuter et, au besoin, se mettre d'accord sur les prochaines actions à effectuer avec l'avantage d'être un outil « concret » (on déplace les post-its, les reformule …), et qui oblige à formuler ce que chaque User Story[6] implique.

4. eXtreme Programming & TDD

a. eXtreme Programming (XP)

L 'XP est un ensemble de bonnes pratiques utilisées par les informaticiens pour mener à bien un projet. Les bonnes pratiques les plus notables sont :

- Travail en binômes (« deux cerveaux mais un clavier »), pour éviter les erreurs et favoriser l'amélioration continue et la maintenabilité d'une application (si deux personnes ont travaillé sur une portion très particulière de code et qu'une s'en va, il en reste toujours une qui sait très bien de quoi le code parle sans avoir à le reparcourir). Le pair programming est aussi utile en ce sens que les 2

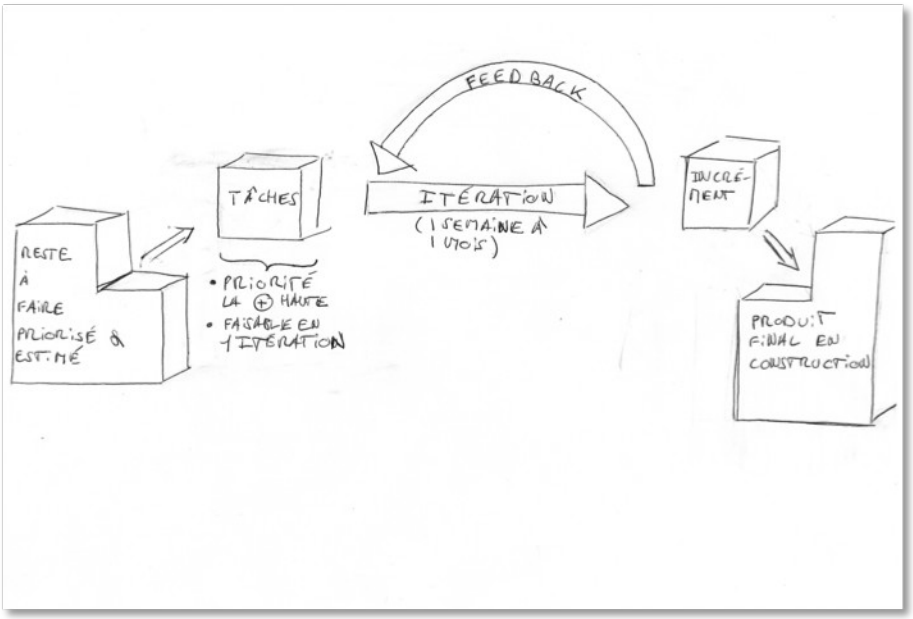
[6] Lot de fonctionnalités qui sert à remplir une fonction particulière du produit. C'est une « brique élémentaire » du projet.

personnes qui travaillent sur un même « bout de code » ne réfléchissent pas aux mêmes choses selon qu'elles sont au clavier ou pas. La personne au clavier va avoir tendance à se concentrer sur le « comment » (aspects purements techniques) et son binôme sur le « pourquoi » (voir l'image d'ensemble et empêcher son binôme de dévier)

- Refactoring régulier / Utilisation de tests automatisés (Assurer la qualité et la stabilité du code produit)

- Cycles de développement courts (entre 1 et 4 semaines). Objectif : livrer souvent, produire de la valeur ajoutée à chaque livraison, intégrer les nouvelles fonctionnalités de façon continue et obtenir un retour rapide sur le travail effectué pour corriger le tir très rapidement si quelque chose ne va pas.

Image 1 : Le développement itératif en un schéma



Le fait de livrer si régulièrement permet de toujours disposer d'une base « saine » de code au cas ou des changements seraient à inclure.

b. Test Driven Development (TDD)

Le TDD est une des composantes essentielles de XP car c'est elle en grande partie qui permet la mise en place et le maintient des tests automatisés. Le principe même du Test Driven Development est de commencer sa démarche d'écriture de programme par l'écriture des tests qui permettront de tester les fonctions à implémenter puis de mettre en place ces fonctions « pas à pas » en vérifiant toujours que les tests écrits pour la fonction « passent » (la fonction de test obtient le résultat attendu en utilisant le code fourni, l'aspect du code testé est implémenté et se comporte comme souhaité) une fois que ce qu'ils doivent tester a été implémenté. L'urgence absolue à traiter en priorité dans un tel type de développement est un test qui ne « passe » plus (régression) alors

qu'auparavant il ne posait pas de problème. Résoudre le problème à la source de la régression est bien plus important que n'importe quoi d'autre dans un tel contexte.

5. Mesure de la performance

Mesurer la performance dans un projet n'est pas toujours des plus aisés. Cela vient en partie du fait que l'on peut utiliser une foultitude de métriques différentes se basant sur à peu près tout ce que l'on peut quantifier facilement dans un projet (sa valeur actuelle, le nombre de jours/homme nécessaires, la date de fin de projet estimée à un instant t, le taux de bugs dans un programme, l'évolution du taux d'arrêt maladie depuis le début, le nombre de litres de café consommés par jour en moyenne …). Toutes ces mesures ne sont pas nécessairement pertinentes, d'autant plus qu'elles tendent à mettre de côté des aspects moins facilement quantifiables. C'est en particulier le cas de tous les facteurs « humains » d'un projet (montée en compétences, synergie de l'équipe …) qui sont parfois un peu « oubliés » ou en tout cas relégués à une sorte de « second rôle » en termes de performance dès lors que l'on peut difficilement les appréhender. À l'occasion de leur intervention à la conférence LeanSI de Telecom ParisTech le 10 Juin 2010, Elodie Aïdan et Marie-Noëlle Ninteya, en parlant de TWI (Training Within Industry) et en souhaitant montrer à l'auditoire à quel point cette méthode est performante quand il s'agit de faire apprendre un procédé nouveau , ont commencé à parler de taux d'erreurs, de temps moyen de « baisse de performance » à la suite de l'arrivée d'une nouvelle recrue ou encore de comment les matrices de performance de chaque membre de l'équipe. Ces matrices étaient rédigées puis conservées d'une fois sur l'autre pour que la personne puisse se voir progresser et, à partir d'un certain niveau de maîtrise, donner la possibilité de former les autres. Lors des questions, Michael Ballé (Telecom Paris) a posé cette question : « Combien de Marie-Noëlle avez vous formées ?». Et cette question, si on la comprend comme « en plus de développer les processus, êtes vous parvenues à développer les personnes ? » résume bien, en un sens, le problème de la mesure de la performance : Parfois, il n'est pas pertinent de définir une métrique de performance en se basant uniquement sur des indicateurs quantifiables liés au projet pour la performance actuelle d'une équipe et de faire des projections pour l'avenir, il faut parfois aussi prendre en compte des facteurs plus intangibles comme le développement des personnes qui pourra porter ses fruits au delà du projet. Il n'y a que dans les fiches de personnage des jeux de rôle que l'on peut placer un chiffre sur des compétences comme l'esprit d'équipe, le charisme ou encore le leadership d'une personne au sein de son équipe. Pour évaluer toutes ces caractéristiques chez une personne dans la « vraie vie », il faudra le plus souvent croiser plusieurs informations factuelles et s'en remettre à des avis extérieurs d'autres personnes ayant travaillé avec elle.

On peut donc se poser la question des axes à employer pour mesurer une telle performance, mais aussi du choix des métriques (les deux n'étant, bien sûr pas indépendants).

a. Quels axes ?

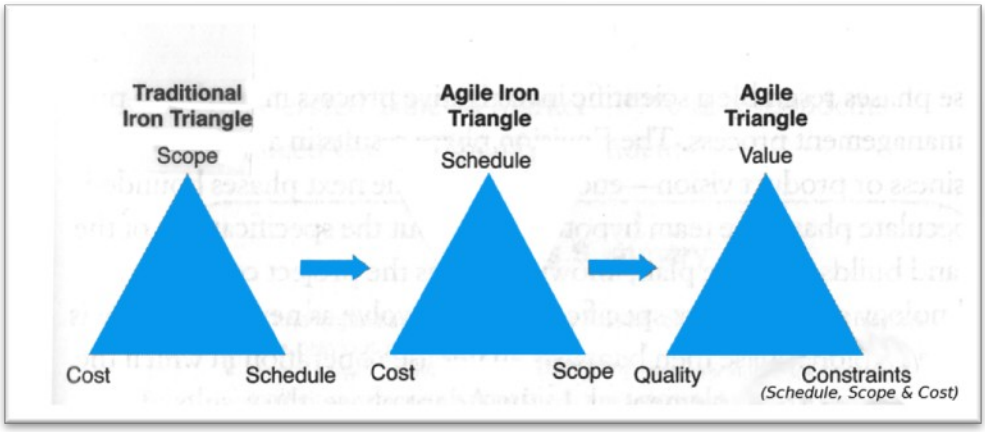
Traditionnellement dans la gestion de projet, un projet est conduit avec en tête 3 contraintes principales (Que l'on peut retrouver dans une approche PMI de la question par exemple):

- Le périmètre
- Les coûts
- Les délais

Et un bon chef de projet se doit de rester le plus possible « dans les clous » concernant ces 3 critères.

Lorsque l'on mène un projet « agile », ces critères sont, bien sûr, à prendre en considération, mais, en raison de l'objectif caché derrière tout projet agile (« apporter un maximum de valeur pour le client et favoriser le développement des personnes par rapport au développement des processus ») et de la nature itérative d'une gestion de projet de ce type, le « triangle de fer » agile ne se construit pas exclusivement autour de ces 3 seuls axes.

Image 2 : Les différents triangles de contraintes dans le management de projet



** Schéma original dans « Agile projet management » (Jim Highsmith – Addison Wesley, 2010)*

Avec le triangle de gauche qui représente la façon traditionnelle qui sert à s'assurer qu'un projet est, ou pas, conforme aux performances que l'on attend de lui (le périmètre est considéré inamovible (bien que dans les faits cela ne soit pas toujours le cas) car on l'admet fixé à l'initiation du projet, tandis que le coût et les délais sont sujets à variations), le second, la vision « agile » de ce triangle , pose les délais comme une variable inamovible, tandis que le périmètre, lui peut varier (ce qui impacte automatiquement les coûts).

Dans le triangle de droite (le triangle « agile »), l'accent est mis sur la

valeur dégagée par le projet pour le client (En rapport avec la valeur « Adapting to change over conforming to plan » dans le manifeste agile). Les contraintes (Délai, coûts et périmètre) étant ici prises en compte mais pas considérées comme des objectifs en eux mêmes pour l'équipe qui mène le projet, mais plutôt comme des moyens à disposition ou des « garde fous » selon comment on les considère.

b. Quelles métriques ?

Afin de permettre de mesurer la performance d'une équipe sans pour autant utiliser des métriques contraires aux principes mêmes qui régissent l'agile il a fallu trouver de nouveaux moyens pour mesurer la performance d'une équipe « agile ». L'utilisation de telles métriques peut se révéler utile lors de diverses étapes d'un projet (dont la phase de R&D). Les métriques utilisées en mode agile se concentrent toujours sur le même objectif : fournir le maximum de valeur possible à un instant t avec les moyens à disposition au client. A cet effet, un produit est découpé en scenarii que le client souhaite réaliser (« User Stories »), qui sont, par la suite découpées en listes de tâches (un peu à la façon dont, dans un projet aux normes PMI, on élabore d'abord une Work Breakdown Structure puis, une fois celle ci validée, on commence à écrire la liste des tâches à effectuer pour chacun des Work Packages contenus dans la WBS). Chaque User Story est priorisée par rapport aux autres mais il lui est aussi adjoint une valeur dite de « complexité » (pas une estimation du temps que cela prendra pour réaliser cette User Story mais bien une estimation faite par l'équipe de la complexité des tâches à effectuer pour la compléter) que les personnes qui seront en charge de cette User Story doivent évaluer. La complexité reflète, d'une part, la difficulté technique à réaliser une certaine tâche mais aussi la valeur que peut apporter cette User Story au client. Le couple « priorité / complexité » va permettre de déterminer quelles User Story sont à achever en priorité (grande complexité, priorité importante) dans le but que, si le projet s'arrête du jour au lendemain, le client, avec ce qui lui aura déjà été livré, aura obtenu la plus grande valeur possible pour son projet au regard des moyens et du temps investis. Pour définir cet ordre, c'est le client qui décide des aspects de priorité avec l'équipe qui se chargera de réaliser le projet et c'est l'équipe elle même qui jugera de la complexité d'une User Story.

En général, l'initiation d'un projet agile commence donc par la rédaction d'un backlog (en présence du client et du Product Owner), pour définir les User Stories à compléter et leur priorisation. Par la suite, l'équipe en charge du projet se fera fort de compléter ces User Stories aussi vite et bien que possible.

La valeur de « complexité » (qui n'a pas forcément en elle même de valeur pour le client) va servir à l'équipe pour mesurer sa « vélocité » et évaluer ses performances au fil des itérations (à l'aide d'un burndown chart[7] ou de

[7] Graphique ou sont reportés les points de complexité achevés lors des itérations successives afin d'évaluer la performance d'une équipe

graphes de vélocité par exemple). Plus on avance dans les itérations, plus ces informations deviennent précises et précieuses pour une équipe car elles leur permettent de savoir jusqu'à combien de User Stories elle peut gérer dans une même itération sans être surchargée, mais aussi de faire des projections sur la date de fin du projet ou l'impact d'un changement de périmètre. Dans une approche PMI, typiquement, on va corréler la performance d'une équipe à la valeur qu'elle est capable de fournir sur un projet en comparant le « réalisé » à un instant *t* par rapport au « prévu » en termes de valeur globale du projet. Si ces deux façons de mesurer semblent se valoir (le but est toujours de faire le maximum avec le temps et les moyens dont on dispose), on remarque que la vision PMI se focalise sur la valeur créée estimée d'un projet, tandis que la vision agile se base essentiellement sur la capacité du'ne équipe à résoudre rapidement des problèmes complexes (plus le total de points de complexité achevés dans une itération est important, plus l'équipe est performante). L'approche de la performance est ici très différente en fait, bien que le but soit le même.

En termes de R&D, ces instruments peuvent être utiles quand le temps est compté et que l'on veut préparer, à la suite de de travaux de recherche « pure », des livrables (documentation, prototypes …). Ces métriques permettent même, dans une certaine mesure de fixer, en se basant sur des travaux passés, des limites de temps pour chaque activité de R&D si ce n'est qu'au lieu d'évaluer le temps que prendra une User Story, on préférera estimer sa complexité, regarder si dans les itérations précédentes de la démarche on a déjà achevé une telle complexité dans une itération et éventuellement décider de découper la User Story en plusieurs « sous-histoires », qui seront elles mêmes priorisées entre elles, si jamais elle est trop importante pour être traitée en une seule itération.

On peut aussi ajouter que, d'un point de vue « humain » pour un manager, il est important de garder une trace de l'évolution des personnes au sein de son équipe et de s'assurer qu'elles arrives à se développer dans l'environnement qui leur est proposé. Une fois les besoins purement « pécuniers » satisfaits (la tranche « sécurité » dans la pyramide de Maslow), beaucoup de personnes vont s'orienter vers des besoins moins matériels (appartenance à un groupe, estime de soi et, ultimement, « réalisation de soi », toujours chez Maslow). Veiller à ce que ces besoins « non-basiques » puissent être satisfaits permet aux personnes travaillant sur un projet de se développer et de se sentir « bien » dans leur équipe, ce qui pourrait difficilement être néfaste à la performance globale de cette dernière.

c. Conclusion

La mesure de la performance dans un projet (qu'il soit agile ou pas) est nécessaire. Il faut toutefois faire preuve de discernement quand on choisit ses indicateurs car il y a fort à parier que d'un projet à l'autre certains

d'entre eux ne seront pas toujours aussi pertinents. Il faut, de plus, ne pas perdre de vue que certains indicateurs que l'on pourrait qualifier d' « inquantifiables » peuvent se révéler tout aussi importants que des indicateurs plus « classiques ». Un exemple simple de cela pourrait être le fait que les acteurs d'un projet sont capables ou pas de travailler ensemble : on peut avoir les meilleurs experts dans leur domaines respectifs sur un même projet, s'ils ne parviennent pas à travailler ensemble, il y a fort à parier que leur performance générale s'en trouvera impactée (et plus les interactions entre les personnes de ce projet devront être fréquentes plus grand sera le risque d'une performance pas aussi exceptionnelle que prévue).

VI. Les autres méthodologies

Dans la partie suivante, je vais aborder d'autres méthodes de gestion de projet et comparer ces différentes conceptions du déroulement d'un projet.

1. Le cycle en 'V'

Le cycle en V dérive lui même des premiers cycles de développement « waterfall » où un projet ou une activité industrielle était découpé en plusieurs phases qui s'enchaînaient de façon séquentielle depuis l'initiation jusqu'à la recette. On parle de cycle en 'V' à cause de la façon dont s'enchaînent les phases ainsi que de leur « niveau ». On distingue deux grandes « phases » lors d'un cycle en V : la prmière qui est celle de la définition du produit à obtenir, et se termine par l'implémentation même, et la seconde phase qui est celle de la vérification. Chacune de ces phases est découpée en plusieurs plus petites correspondant chaque fois à un niveau de conception différent, depuis l'idée générale jusqu'au design détaillé pour la phase de conception et de la maintenance post-livraison aux tests unitaires pour la phase de vérification.

La seconde « phase » du cycle en V a avant tout été pensée pour tenter d'apporter un peu plus de fiabilité au management d'un projet. Le cycle en V lui même est une méthodologie qui tente d'organiser un projet dans son ensemble, d'en minimiser les risques, d'en améliorer la qualité et de réduire les coûts induits par le projet et la vie du produit. Il a aussi pour vocation d'améliorer la communication entre les divers acteurs impliqués dans le projet, par l'introduction d'un vocabulaire commun.

2. L'approche PMI

Le PMI a, depuis plusieurs années, regroupé un ensemble de bonnes pratiques (dont certaines sont des outils utilisés dans les projets agiles), qui ont pour but de structurer leur déroulement et d'améliorer la performance des projets. C'est une approche qui couvre des aspects « techniques » (comment évaluer la durée de ses tâches, ou encore calculer l'état d'avancement d'un projet), mais aussi managériaux (comment de constituer et gérer son équipe), voire éthiques (comment se comporter lorsqu'on est un chef de projet certifié « PMP »).

Comme la plupart des autres méthodes de management de projet, l'approche PMI tend à essayer de découper chaque étape du projet en « phases » (définir le périmètre, choisir une équipe, ...) mais propose aussi des phases « transverses » telles que la gestion de la qualité (qui intervient entre chaque étape mais aussi pendant celles-ci). C'est aussi une méthodologie qui prend en compte l'aspect « humain » : en plus des aspects techniques enseignés ayant directement trait au management du projet, le manager certifié PMI doit aussi savoir gérer son équipe (conflits, mécontentements ...) et aborder divers sujets (théorie des besoins de Maslow, théorie de Herzberg ...) ou même des recommandations comme celle de prendre en compte les « soft skills » des gens pour avoir une équipe performante. L'approche PMI ne se contente pas de proposer des outils techniques au manager

de projet, elle lui donne aussi une « boîte à outils » la plus complète possible dans le domaine du management pour lui permettre d'avoir une réponse, ou en tout cas une « grille de lecture », face à tout problème qui pourrait surgir.

3. PRINCE2 (PRojects IN Controlled Environments)

La méthode PRINCE2 dérive d'une part de la méthodologie PRINCE mais aussi de PROMPT. PRINCE a été développée par le gouvernement de Grande Bretagne afin de pouvoir gérer ses projets informatiques, bien que cette méthodologie finisse par devenir la norme pour ses projets en dehors des domaines purement « IT ». PRINCE2 a été finalisé en 1996 avec une optique plus généraliste et conçue à la base pour couvrir tous types de domaines.

PRINCE2 est une approche structurée du management de projets. Elle propose une méthode efficace permettant de gérer des projets à périmètre bien défini. Elle encadre le projet du début à la fin et décrit des procédures portant sur sa conception et sa supervision et même sur ce qu'il faut faire si le projet ne se déroule pas comme prévu. Chaque processus est défini par des entrées et des sorties qui permettent à tout moment d'en vérifier l'état. La méthode PRINCE2 en elle même divise un projet en plusieurs grandes « phases » de sorte à essayer de contrôler le plus efficacement possible les ressources à disposition :

a. Démarrage d'un projet (Description du projet dans ses grandes lignes, du besoin, désignation de l'équipe projet et de son Project Manager)

b. Initiation d'un projet (Rédaction d'un business case, décisions concernant la gestion de la qualité et le contrôle du projet en général, estimation des risques)

c. Direction d'un projet (Process visant à déterminer l'influence du « project board » sur le projet)

d. Vérification d'une étape

e. Gestion du périmètre d'une étape

f. Clôture d'un projet

PRINCE2 est souvent citée comme une méthode de management difficilement adaptable à un projet de petite taille et ce à cause du nombre de documents considérables à remplir et à faire « vivre » tout au long du cycle de vie. Cependant, les défenseurs de la méthode affirment que celle ci supporte tout à fait le passage à l'échelle, tous les documents en vigueur dans un « gros » projet n'étant pas nécessairement requis dans un projet plus petit.

PRINCE2 emprunte beaucoup de notions à ce que l'on peut retrouver dans le PMBOK (Project Management Body Of Knowledge) du PMI (Project Management Institute) en mettant l'accent sur des notions comme l'amélioration continue, l'implication constante de toutes les parties prenantes ou encore le contrôle des processus par des revues.

4. Six Sigma / Lean Six Sigma

Six Sigma est une méthode de gestion de production inventée par Motorola en 1981 qui met la qualité au centre des processus en faisant adopter à ses praticiens une philosophie qui consiste à viser le « zéro défaut ». La principale mesure de performance est le nombre de produits défectueux qui sortent de la chaîne de production. La méthode Six Sigma tire son nom du modèle statistique qu'elle utilise pour évaluer la qualité de la production d'un projet.

La philosophie de ce modèle est que le taux de défauts des objets produits par un projet géré en Six Sigma ne devrait pas dépasser 3.4 produits défectueux pour 1 million (soit 0.00034% de taux de défauts sur l'ensemble de l'activité de production engendrée par le projet).

Cette méthodologie s'est, dans un premier temps, focalisée sur l'optimisation de la production industrielle de biens mais elle a par la suite été adaptée à l'optimisation de la qualité de processus ou de services fournis (des groupes comme la banque HSBC[8] l'utilisent). Elle met, de plus, l'accent sur le fait qu'il faut être en mesure de constamment évaluer la qualité de sa production, et que ceci est rendu possible par le fait que les processus de production eux mêmes ont des caractéristiques qui peuvent être mesurées et utilisées pour contrôler l'activité (cartes de contrôle, taux de retours …).

Selon Wikipedia (fr) :

«

Six Sigma repose sur les notions de client, processus et mesure ; il s'appuie en particulier sur :

1. *les attentes mesurables du client (CTQ - Critical To Quality) ;*
2. *des mesures fiables mesurant la performance du processus métier de l'entreprise par rapport à ces attentes ;*
3. *des outils statistiques pour analyser les causes sources influant sur la performance ;*
4. *des solutions attaquant ces causes sources ;*
5. *des outils pour contrôler que les solutions ont bien l'impact escompté sur la performance.*

La méthode se base ainsi sur 5 étapes qui se contractent dans l'acronyme DMAAC (ou DMAIC en anglais) pour « définir, mesurer, analyser, innover/améliorer (Improve en anglais) et contrôler ».

Plus qu'une méthodologie de gestion de projet en elle même, Six Sigma constitue

[8] Banque d'investissement et de détail

une démarche de gestion de la qualité. Cependant, par son exigence, elle exerce une forte influence sur la façon de mener un projet dans la mesure où elle ne tolère que peu d'erreurs en production et a donc un impact tôt lors de la conception d'un produit.

Le Lean Six Sigma se rapproche plus d'une méthodologie de gestion de processus puisqu'on combine, en fait, une méthodologie de gestion et d'amélioration des processus (le Lean Manufacturing), avec une méthodologie de gestion de la qualité (le Six Sigma). En ce sens, le Lean Six Sigma peut être un peu plus considéré comme une méthodologie de gestion de projets (par son influence forte sur les processus et la qualité).

5. Comparaison "Agile VS Reste du monde"

Sur la façon de mener un projet, on peut considérer que, si certaines pratiques que l'on retrouve dans l'agile peuvent se retrouver dans PRINCE2 ou les recommandations PMI (après tout, les pratiques utilisées dans ces méthodologies ont cours depuis des années, il est donc normal que l'on retrouve trace de celles qui ont fait leurs preuves un peu partout maintenant …), la façon de mener un projet est très différente puisqu'elle ne traite pas des aspects comme la mesure de la performance, la planification ou encore les personnes, de la même façon.

a. « Tenons nous en au plan … »

Ce que l'on remarque dans des approches comme PRINCE2, PMI ou même le cycle en V, c'est la volonté de mettre en place très tôt une stratégie (là où on n'a pas forcément le plus d'informations pertinentes pour aider ses décisions) et de tout mettre en œuvre pendant la durée du projet pour s'y tenir. Les déviations et autres changements de périmètre ne sont pas les bienvenus et sont considérés la plupart du temps comme néfastes et le résultat d'une mauvaise gestion ou d'une mauvaise planification. Au contraire, lorsque l'on aborde un projet agile, le changement est considéré comme « faisant partie du plan ». Il n'est pas forcément souhaitable mais il est admis que d'ici la fin du projet, le contexte aura peut être changé (conjoncture extérieure, difficultés ou facilités inhabituelles dans le projet lui même …) et qu'il faudra sans doute adapter le « plan » initial pour faire face à la nouvelle situation.

Martin Fowler[9] lui même quand il écrit sur l'agile et les autres méthodologies de management de projet, regroupe ces dernières sous le terme de « plan driven methodologies » pour les opposer à l'agile, ce qui montre bien la façon dont il perçoit la grande importance donnée à la planification d'un projet dans les méthodes « traditionnelles » par rapport à l'agile. Lors de sa keynote avec Neal Ford à l'USI 2010, il parlait même d'une vision « predictive » (traditionnelle) et d'une vision « adaptive »

[9] « Head of research » chez ThoughtWorks (cf annexe)

(agile) des projets et de la distorsion qui pouvait naître dans les esprits « prédictifs » de certaines équipes, qui estimaient que leur projet était un succès puisqu'il s'était déroulé conformément au plan. Certains, comme les auteurs du livre « Rework » vont même jusqu'à intituler un de leurs articles : « Planning is guessing » (« Faire un plan, c'est faire des prédictions »), en justifiant leur propos par le fait que l'avenir est tellement incertain (comment prévoir l'état de la concurrence ou du marché à plus de quelques semaines, par exemple ?) qu'il faut prendre les anticipations que l'on en fait pour ce qu'elles sont : une idée de ce qui pourrait se passer, mais certainement pas le seul et unique scénario possible. Avoir une idée la plus précise possible de là où on va et comment on compte faire pour y arriver est sans aucun doute un atout, mais le « plan » ne doit pas se transformer pour autant en « œillères » et faire ignorer un environnement de plus en plus changeant.

b. « … et cette tâche, combien estimez-vous qu'elle va prendre de temps ? »

De façon connexe à ce qui vient d'être dit, les méthodes de management « classiques » telles que l'approche PMI ont en général besoin d'évaluer dès le départ la durée de chaque tâche, ce qui va se faire de diverses manières, allant de la moyenne pondérée des bonnes pratiques PMI à une estimation GDM (« Gros Doigt Mouillé ») selon les pratiques en vigueur dans l'entreprise qui met en place ce projet. Par essence, une fois ces durées estimées et rentrées dans un diagramme de flot (Gantt par exemple), avec d'éventuelles réserves de temps, elles sont censées ne plus varier sauf cas exceptionnel (dans les bonnes pratiques PMI, par exemple, cela donnera lieu à une « change request » qui devra être approuvée par des responsables du projet).

L'approche agile de l'estimation est sensiblement différente, du fait principalement que la relation au plan n'est pas de même nature. Typiquement, en agile, pendant la rédaction du backlog[10], l'équipe va fournir une estimation comparée de complexité des tâches (et non pas de durée). On peut aussi se poser la question de la bonne façon d'évaluer cette complexité ou ce temps. En ce qui concerne la personne qui devrait faire l'estimation, la méthodologie PMI est d'accord avec l'agile : autant que faire se peut, c'est à la personne qui devra se charger de la réalisation d'une tâche d'en fournir une estimation de temps ou de complexité. Cela ne sera pas nécessairement une estimation « au plus court » mais plutôt une vision réaliste prenant en compte la capacité de travail de la personne ou de l'équipe voire ses antécédents sur le sujet. Elle n'ira clairement pas aussi vite si c'est la première fois qu'elle accomplit une certaine tâche dans un projet que si elle s'en charge depuis 10 ans dans la même société et avec les mêmes équipes. Toutefois, l'avantage que va offrir l'utilisation d'évaluations de complexité est sa simplicité de réalisation, puisqu'on ne s'engage pas immédiatement sur une durée mais bien sur une complexité estimée, rapportée au reste du projet. Quand on a utilisé un backlog, on

[10] Liste de User Stories priorisées entre elles

peut aussi dire que cela est plus simple et surtout pertinent de comparer deux User stories en fonction de leur complexité plutôt qu'en fonction de leur temps de réalisation : une tâche très complexe à réaliser mais qui requiert peu de temps apportera, à priori, beaucoup plus de valeur ajoutée au projet qu'une tâche très simple prenant le même temps, voire plus. L'estimation de complexité permet aussi, d'une certaine façon, de « renverser » la vision des tâches à effectuer : Les défis techniques prennent le pas sur les éléments de planning et les délais. D'une certaine façon, on passe d'une approche de planification à une approche de réalisation.

Ici, quelle que soit la façon d'évaluer (et quel que soit ce que l'on estime), on peut se dire que la meilleure façon d'estimer une tâche en durée ou en complexité reste encore de demander son avis à la personne qui va se charger du travail .

c. « De quel ROI parle-t-on ? »

Si l'on prend l'exemple de la mesure de la performance d'un projet, on se rend compte que des méthodologies comme celles du PMBOK se focalisent sur la valeur d'un projet à un instant *t*. Ce type de mesure n'est peut être pas toujours pertinent si l'on considère un projet avorté. Pour reprendre en la détournant la critique les détracteurs de la théorie de la valeur travail de Marx (« la valeur de la marchandise est proportionnelle au temps de travail humain ») : le projet a beau coûter cher, si l'on ne peut pas de se servir de ce qui a été produit, pour le client, il ne « vaut rien ». Dans le cas de projets qui finissent par avorter, le critère de la valeur actuelle n'a pas la même pertinence selon la façon dont le projet a été mené ; et en particulier, ce qui a été produit d'utilisable pour le client final.

L'agile tente de répondre à ce problème en organisant les phases de travail différemment de ce que l'on peut trouver dans des méthodes de gestion de projet plus traditionnelles (découpées en phases). Surtout, la performance mesurée l'est aussi par rapport à l'équipe (a-t-elle gagné en productivité ? les personnes se sont elles améliorées ? …), et pas seulement par rapport à ce qui « sort » du projet. Si l'on considère une gestion de projet agile, le résultat produit sera bien sûr au centre de l'évaluation de « valeur », à commencer par son succès ou son échec, mais, dans la valeur produite, on fera aussi valoir la montée en compétences des personnes et aussi l'adaptabilité ce qui a été produit.

En effet, la question de l'amélioration des équipes n'est pas neutre. Si une équipe rencontre des difficultés lors d'un projet et qu'elle « explose » les coûts additionnels, ce ne sera pas pour autant nécessairement un échec si le travail supplémentaire a permis de débusquer des problèmes de fond (liés à la façon même qu'a l'équipe de gérer ses projets par exemple) et de capitaliser ces nouvelles connaissances pour les projets futurs. Ici, le budget d'un projet pourrait bien en souffrir mais permettre dans le même temps de mettre à jour de graves dysfonctionnements organisationnels.

Emmanuel Chenu nous décrit, d'une certaine façon, ce phénomène lorsqu'il nous dit que l'on ne s'attache en général pas assez à l'analyse des « causes profondes » d'un problème : « Quand j'ai un problème sur mon projet, il peut être facile de le régler, mais si on ne comprend pas d'où il vient exactement, alors ce problème reviendra de manière récurrente miner le travail de l'équipe ».

Les processus de capitalisation des bonnes pratiques ne sont pas pour autant absents des méthodologies de gestion de projet dites « à planification forte ». On peut par exemple les retrouver dans la rédaction des « Lessons Learned » PMI qui interviennent au moment de la recette. Il faut, en revanche, remarquer que, pour le PMI par exemple, il s'agit là d'un processus particulier qui intervient à un moment précis du projet, et non d'une démarche continue. On peut par contre « reprocher » aux méthodologies qui ne prennent pas en compte l'évolution personnelle des individus de passer à côté de quelque chose qui, est important pour ces personnes (évoluer dans son travail, bien s'y sentir, monter en compétences …) et aux organisations qui les adoptent de négliger la « valeur ajoutée humaine » . Des personnes motivées et heureuses de travailler seront efficaces sur leur projet en cours mais aussi sur les projets à venir. Dans sa conférence « Vaincre le Plouf » (USI 2009), Guillaume Duquesnay illustre ceci par l'expérience d'un de ses amis : il a quitté une des sociétés pour laquelle il travaillait précisément parce qu'elle n'a pas pris en compte ses remarques et ses questions sur un projet qu'il devait faire démarrer. Au bout d'un moment, le projet a été abandonné (on l'a laissé mourir), l'ami Guillaume est parti sur un nouveau défi mais il a fini par quitter sa société à cause du manque de motivation qu'elle a suscité chez lui. C'est une « fausse économie » de négliger le « ROI humain » et de ne pas encourager les personnes d'une équipe à évoluer.

d. Tableau comparatif des approches de plusieurs méthodes de gestion de projet

VII. Conditions de mise en œuvre de méthodes Agiles

	Agile	Cycle en V	PMI	PRINCE2
Approche de la planification et des changements	• Planification faite par priorisation • Changement intégré dans le processus • Itérations courtes qui limitent l'impact des changements	• Planification selon des phases prédéfinies • Changement de plus en plus dangereux au fil du projet	• Planification selon des phases • Changements pris en compte et contrôlés (dangerosité qui monte au fil du projet)	• Planification selon des phases • Changements pris en compte et contrôlés (dangerosité qui monte au fil du projet)
ROI visé	• Atteindre les objectifs fixés • Développer les personnes	• Projet complété dans les temps avec le budget et les critères de qualité souhaités	• Valeur actuelle du projet terminé conforme aux projections • Critères de périmètre, délais, coûts et qualité respectés	• Projet terminé dans les temps avec le budget prévu •
Estimation de charge	• Découpe des actions à effectuer en « User Stories » • Complexité	• Temps	• Temps (moyenne pondérée) • Découpe des actions en « Work Packages »	• Temps
Livraison(s)	• Une par itération • Livraison fonctionnelle (partie de la livraison finale) utilisable par le client	• Une seule	• Une seule	• Une seule
« Voix du client »	• Omniprésente	• En début et fin de projet	• En début et fin de projet	• En début et fin de projet
Amélioration continue	• Omniprésente	• Pas explicite	• Rédaction des « Lessons learned » de fin de projet	• Importance certaine, même philosophie que le PMI • Item « Improve » dans les processus PRINCE2

dans un projet

1. Nature du projet

a. Caractère innovant

Que permettent les méthodes agiles ?

L'aspect innovant d'un projet (nouveaux concepts, utilisation d'outils de pointe …) n'est pas un frein pour la mise en place de l'agile ; il peut au contraire donner une plus grande flexibilité lorsque l'on a besoin d'expérimenter pour parvenir à un résultat.

Le fait d'utiliser des itérations de taille fixées et plutôt courtes en prenant en compte l'avis du client final à l'issue de chacune d'entre elles permet d'expérimenter, éventuellement de se tromper, et de changer d'orientation si besoin avant que le retour en arrière ne devienne trop coûteux (en temps ou en argent). L'agile permet donc une plus grande souplesse au niveau de la définition du périmètre d'un projet (qui n'est pas gravé dans le marbre dans son intégralité). Cependant, cette « qualité » peut avoir les inconvénients de ses avantages : si l'on ne parvient pas à prendre des décisions sur le cap à adopter, on finit par ne plus avancer, ce qui peut se révéler tout aussi néfaste que de ne pas avoir choisi la meilleure direction.

Sur des projets « nouveaux » qui pourront nécessiter des ajustements pendant la phase de développement, l'agile peut donc faire merveille (à condition d'être discipliné et de savoir où l'on va et ce que l'on fabrique). On peut cependant se poser la question de l'utilité de telles méthodes pour des projets fondés sur des technologies et des processus fiabilisées depuis longtemps (quand on souhaite adapter un outil mais que l'on ne ressent pas le besoin d'expérimenter pour le faire évoluer par exemple, où encore quand on se trouve dans une démarche d'optimisation de l'existant plutôt que de création de quelque chose de nouveau).

En permettant de rectifier le tir facilement et à intervalles très réguliers, l'agile peut donner un avantage stratégique à une équipe lorsque, par exemple, plusieurs projets du même nature et visant le même type de clients potentiels sont en compétition. L'agile, par son approche facilitantant les changements de cap en cours de route permettra à une équipe de s'adapter plus facilement à son environnement (technologique ou organisationnel). De son côté, le client pourra affiner certains points de sa demande en prenant en compte l'évolution de son marché et de ses contraintes (insertion, re-priorisation ou retrait de certaines fonctionnalités typiquement).

b. Niveau de risque

Quelles différences l'Agile montre-t-elle par rapport aux méthodes plus conventionnelles ?

La méthodologie agile, et en particulier le développement incrémental, permettent de fournir à intervalles réguliers un produit fonctionnel même si celui ci ne dispose pas forcément de toutes les fonctionnalités attendues du produit final. On peut donc considérer que, à valeur égale produite et à moyens égaux mis à disposition, un projet agile « avorté » aura plus de chances de produire quelque chose d'utilisable par le client (et donc valorisable commercialement) s'il décide d'arrêter le projet. De ce fait, l'agile aide, au moins d'une certaine façon, à minimiser le risque financier. De plus, le fait de devoir livrer régulièrement et à intervalles rapprochés permet de garantir, un certain niveau de qualité (on y reviendra), mais aussi de permettre, par cette gestion « au plus près » de la qualité, de minimiser le risque de tomber trop tard sur un défaut critique suffisamment important pour le faire avorter (impasse technique, coût de réparation trop important …).

L'agile peut donc bien s'adapter à des projets ou l'on a du mal à évaluer le risque. L'idée sous-jacente est que, d'une part et ce quel que soit le moment où l'on décidera de s'arrêter, on aura optimisé l'utilisation des ressources disponibles et que, d'autre part, on aura mieux maîtrisé les risques liés à l'environnement du projet (capacité à réagir rapidement). On peut donc penser que l'agile offre une approche différente de la gestion des risques financiers (en essayant de maximiser le ROI d'un projet, y compris s'il avorte) mais aussi contextuels (changements dans l'environnement du projet sur lesquels l'équipe n'a pas de contrôle).

Il est en revanche permis de penser que, pour un projet ou les risques sont connus et ou des plans de contingence sont prêts et rôdés pour traiter ceux que l'on estime les plus vraissemblables, il n'est peut être pas nécessaire de mettre en place tout un arsenal agile (itérations courtes avec points clients à leur fin entre autres), en tout cas pas spécifiquement comme une politique de gestion du risque. Il pourra se révéler utile d'avoir des plans de contingence concernant les risques plus « improbables » mais à priori pas d'avantage que lorsque l'on gère un projet avec une méthodologie à planification forte.

Autre point, la gestion de projet agile, comme toute autre méthodologie, ne s'improvise pas. Décider subitement de travailler en agile pour un projet à venir peut donc, en soi, représenter un risque si les personnes qui vont y travailler n'ont pas l'habitude de le faire en mode « agile », ou tout simplement pas l'habitude de travailler ensemble. Des équipes comme celle d'Emmanuel Chenu ont mis plusieurs années avant d'atteindre leur vitesse de croisière (pour l'équipe d'Emmanuel Chenu, il s'est écoulé plusieurs années avant que le niveau de qualité des projets par rapport à « avant l 'agile » ne devienne significativement meilleur avec des coûts d'intégration et un rework moindres).

c. État d'avancement (projet nouveau ou en cours)

Nouveau projet

Sur un nouveau projet, plusieurs facteurs vont déterminer s'il est possible et surtout bénéfique d'utiliser l'agile.

La familiarité des équipes avec les méthodes agiles va jouer un rôle déterminant dans le déroulement du projet. Si les personnes ont tout à apprendre des méthodologies agiles, il ne faudra pas négliger le « ticket d'entrée » que cela va représenter pour le projet. L'agile paraît beaucoup plus « libre » en termes de méthodologie, de processus à mettre en route et de documents à renseigner mais elle n'en demande pas moins une grande rigueur.

Un autre élément à sans doute prendre en compte est le fait que les méthodes agiles sont avant tout « taillées » pour faire face à un contexte incertain (des spécifications sujettes à changements en cours de réalisation ou besoin de réactivité pour gagner un avantage stratégique typiquement). Pour mettre en place un projet qui va essentiellement constituer une mise à jour de quelque chose d'existant et de bien connu il n'est pas forcément pertinent d'utiliser l'agile si les personnes qui vont être en charge du projet ne sont pas particulièrement à l'aise avec cette démarche et l'on peut parfaitement s'en tenir à la méthodologie qui a cours habituellement si elle fonctionne bien avec les personnes qui vont travailler sur le nouveau projet.

Projet en cours

Travailler en agile nécessite la mise en place d'un certain nombre d'outils et de principes pris dans la « boîte à outils » agile pour bien fonctionner. Cela demande d'avoir une équipe prête à travailler en dehors de ses habitudes et surtout avec une vision de la performance (et donc une façon d'en collecter les indicateurs) qui est très différente de ce que l'on va trouver dans d'autres méthodologies de gestion de projet.

On peut donc dire que, comme pour n'importe quelle méthodologie avec ses règles et ses modes de fonctionnement particuliers, il est très difficile, voire impossible, de passer sans « casse » d'une méthodologie de gestion de projet « standard » à une agile et inversement (beaucoup de changements à la fois méthodologiques, techniques, administratifs et humains à gérer), dès lors que le projet a dépassé les phases d'initialisation (on n'en est plus à se demander comment on va gérer le projet). On peut, de plus, se demander si aborder le choix d'un changement de méthodologie de gestion de projet alors que celui-ci a démarré n'est pas, d'une certaine façon, un aveu d'échec puisque, dans un contexte de « plan driven methodology », ne pas suivre le plan est déjà un problème en soi. Aborder la question de changer de méthodologie en plein milieu d'un projet est finalement un peu la même chose que de trouver un défaut de conception dans le produit que l'on fabrique, plus on s'en rend compte tard, plus il peut devenir complexe et coûteux (en temps comme en argent) de le réparer.

Introduire de l'agile dans un projet ou un programme n'est cependant pas impossible, bien que cela exige certains efforts ou concessions.On peut, par exemple, imaginer qu'une équipe projet se mette à l'agile au sein d'un programme, dans un périmètre bien défini. Un exemple qui illustre bien cette organisation est celui de l'équipe d'Emmanuel Chenu (chargée d'écrire le programme pour une centrale inertielle d'avion Airbus) qui a adopté progressivement l'agile pour ses développements dans un monde ou la méthode de gestion de projet privilégiée reste le cycle en V dans tous les domaines. On remarquera cependant que, si l'on considère l'exemple d'Emmanuel Chenu et de son équipe, cela fut un travail de longue haleine qui s'est étalé sur plusieurs projets (dont certains n'ont pas apporté des résultats équivalents à ce qui se faisait auparavant) et sur plusieurs années. Ce n'est pas une décision qui fut prise sur un coup de tête mais une décision de l'équipe, mais aussi du « upper management » qui a laissé aux collaborateurs le temps qu'il fallait pour que cela puisse se faire. Maintenant, pour l'équipe d'Emmanuel Chenu, un des « cauchemars » associé au développement de logiciels embarqués pour l'aviation , à savoir l'intégration, s'est envolé. Une fois l'équipe passée à Scrum et divers autres outils pour développer les logiciels embarqués qu'on lui demandait, les coûts d'intégration sont maintenant systématiquement tenus et même souvent minorés alors que, dans le domaine, ils ont tendance à systématiquement exploser (trop de « rework » induisant retards et pénalités).

En revanche, l'agile étant, par essence, une « boîte à outils » dans laquelle on peut aller piocher à sa convenance un outil ou l'autre (kanban board, stand up meetings …), il semble donc possible, alors que l'on utilise une méthodologie de gestion de projet classique, d'utiliser certains d'entre eux pour agrémenter ce qui est déjà en place (on n'est pas en train de « faire de l'agile », mais on s'en inspire …). Dans ce cadre, les outils fournis par l'agile pourraient servir d'éléments facilitateurs pour la gestion d'un projet. Néanmoins, l'aspiration à tout changer pour des outils « agiles » peut parfois révéler l'existence d'un « malaise méthodologique » plus profond. Changer de méthodologie ne sera alors efficace que si d'autres changements sont également opérés pour éradiquer les causes plus profondes du malaise.

Par expérience, sur la question de savoir si l'on peut passer à une approche agile du management de projet sur un projet en cours, Guillaume Duquesnay estime, pour sa part, qu'il est tout de même conplexe d'intervenir sur des projets à planification forte déjà lancés. En général, les interventions qui sont faites par des agilistes sur ce type de projet consistent dans un premier temps à remettre les choses à plat (ce qui peut aller jusqu'au « reboot » du projet pour partir sur de nouvelles bases) puis à essayer d'introduire des optimisations issues du Lean sur un périmètre réduit (typiquement un lot correspondant à un périmètre fonctionnel précis) avant

d'agrandir ce dernier au fur et à mesure, d'améliorer le système pas à pas, de promouvoir la communication au sein de l'équipe mais aussi et enfin d'inciter à découper les unités de travail à la façon de User Stories. De ce point de vue, cela confirme ce qui a été dit en première approche : il est très difficile de passer d'une méthodologie « plan driven » à de l'agile pour gérer le même projet sans « casse » (au sens que l'on est obligé de reculer sur certains points et de changer complètement certains modes de travail).

Il n'y a pas de « recette miracle » selon Guillaume Duquesnay, chaque équipe projet est différente et l'agile est tellement centré sur la bonne entente des personnes travaillant ensemble que l' « alchimie de groupe » est cruciale pour déterminer les étapes à suivre. Encore plus dans un contexte aussi particulier que le changement d'une méthodologie de gestion alors qu'un projet a déjà été lancé.

d. Degré de complexité

Technique

D'une façon générale, on peut considérer que la complexité est l'unité de travail en Agile. Une User Story est estimée en fonction de la complexité que lui accorde l'équipe et doit pouvoir être réalisée d'une itération sur l'autre. Si elle devient trop complexe à réaliser (l'équipe n'est pas certaine de pouvoir compléter toute la User Story en une itération), la bonne pratique est de redécouper les itérations trop longues (de sorte à rester au contact avec son client grâce à des points fréquents pour éviter les déviations). De plus, la mise en place d'itérations trop longues rapproche, d'une certaine façon, les méthodes agiles et des méthodes de management de projet à planification forte. En ce sens, l'agile risque de perdre un des avantages qui la caractérise : l'implication de bout en bout de toutes les parties en présence .

Si l'on suit ce raisonnement, il reste toutefois possible d'utiliser l'agile dans des projets ou programmes très complexes techniquement. Mais que faire si l'on doit faire face à des User Stories qui, sont trop complexes de sorte à entrer dans une itération? Bien que l'on puisse considérer qu'il « suffise » de rallonger la taille des itérations pour que les plus petites User Stories entrent dedans, il ne faut pas perdre de vue que, ce faisant, on perdra certains avantages (comme le contrôle régulier des déviations par des livraisons rapides). On peut donc en déduire que, la possibilité ou pas de pouvoir découper suffisamment les User Stories pour pouvoir travailler dessus pendant une itération courte standard (de 1 semaine à jusqu'à deux mois) sera le facteur de choix déterminant.

On peut, enfin, se poser la question de ce qu'est une itération « courte ». Est-ce une itération qui dure entre une semaine et un mois, quel que soit le projet, oui faut-il apprécier sa durée en fonction de celle du projet et de la taille moyenne des livrables ? Si l'on peut se dire que, sur un projet de développement informatique, des

itérations de 2 semaines sont tout à fait concevables, il n'est pas certain qu'il en soit de même pour un des projets du dernier EPR par exemple où certaines tâches doivent exigent sans doute des mois, à la fois pour des raisons de grande complexité mais aussi parfois de sécurité (qui peut induire, justement, un nouveau niveau de complexité), voire éventuellement de faisabilité physique.

Organisationnelle

La mise en place d'équipes Agiles nécessite en général de fonctionner avec des petits groupes co-localisées, ce qui se traduit par un effectif maximum de 12 personnes sur le projet afin de conserver la dynamique collaborative (temps de décision et de circulation de l'information). Aussi, quelle que soit la taille de l'entreprise qui souhaite mener un projet en mode Agile, il est préférable que ce projet ne soit pas soumis à des niveaux de hiérarchie ou de dépendance vis à vis d'autres entités de l'entreprise trop importants. Une équipe agile a besoin d'être autonome. Aussi, une organisation trop lourde avec une pression hiérarchique importante ne manquera pas de compromettre le succès à un projet agile. Dans une organisation qui gère ses ressources par silos fonctionnels, une équipe agile aura extrêmement de mal à se mettre en place si ses membres ne sont pas affranchis de cette organisation. Au contraire, dans une organisation qui utilise une gestion par projets à la manière de ce qui est décrit comme l'environnement « idéal » du project manager par le PMI, , une équipe agile disposera de toute l'autonomie nécessaire pour être efficace.

L'agile peut cependant bien s'intégrer dans des projets ou des programmes de grande envergure. Si l'on prend l'exemple d'équipes comme celle d'Emmanuel Chenu on se rend compte qu'une équipe qui tend à être agiliste peut arriver à s'intégrer dans un environnement de projets où la règle est le cycle en V à passe unique. Cependant, on remarque que cette intégration a pu se faire parce qu'elle concerne une équipe particulière assignée à un projet précis. Cette équipe est la seule à intervenir sur son périmètre, et c'est elle qui va voir le client pour évaluer ses besoins, et prioriser son travail en conséquence, afin de lui fournir en continu les éléments dont il a besoin pour valider le travail de l'équipe et oeuvrer sur des prototypes les plus proches possibles du produit final. Enfin, préserver des équipes de petite taille permettra de conserver la réactivité que l'on demande à une équipe travaillant en agile et lui éviter de s'enliser dans des processus de décision à n'en plus finir.

2. Communication entre les différentes parties prenantes

(Quels sont les points dans la communication entre les personnes travaillant sur un même projet qui vont faciliter ou entraver la mise en place de méthodes « agiles » / «L'agile comme un facilitateur ? »)

a. Type de projet (« maison », sous-traitance, externalisation …)

Plus encore que le type de projet, c'est la qualité de la communication dans l'équipe qui va compter (Notamment si elle est éclatée dans plusieurs lieux). Dans un monde idéal, on peut penser qu'il faudrait co-localiser tous les acteurs d'un projet pour qu'ils puissent interagir de la façon la plus efficace possible à tout moment, mais cela n'est pas toujours possible. Dans le cas où certaines personnes de l'équipe ne sont pas présentes à plein temps sur un projet, il faudra réussir à co-localiser celles qui « comptent » aux instants clés (c'est pour cela par exemple que souvent on regroupe plusieurs rituels comme la livraison d'une itération et sa rétrospective[11]) car, il n'est jamais facile de concilier l'agenda de tout le monde.

Si l'organisation en elle même n'est pas un problème, comme pour tout autre type de gestion de projet, la façon dont sont remontées les informations reste cruciale. En général on préfèrera des structures avec peu d'intermédiaires par qui transitent les consignes pour éviter ce genre de déformations. Il faudra donc éviter leur multiplication et leur cloisonnement (absolument tout le monde doit « jouer le jeu »).

b. Communication entre les services

Le manque de communication, la mauvaise volonté des acteurs d'un projet, l'implication de plusieurs départements dans une même société, l'éclatement de l'équipe projet ou encore sa trop grande taille peuvent donc se révéler être un « poison » d'autant plus virulent quand la communication ne se fait pas et que l'on cherche cependant à travailler en Agile. Si cela est vrai pour la plupart des méthodologies de gestion de projet (si personne ne souhaite travailler en équipe, un projet aura du mal à aboutir quoi qu'il arrive) l'Agile y est particulièrement sensible.

Le risque quand plusieurs services doivent communiquer entre eux est que le plus souvent l'ensemble des personnes ne sera pas forcément disponible ou ne disposera pas forcément de l'intégralité des « canaux de communication existants » (face à face, téléphone, e-mail …). Certains outils utilisés par l'agile essayent de combler ces lacunes en permettant à chacun de transmettre simplement des informations sur ce qu'il fait, comme par exemple de mettre à disposition du reste de l'équipe toutes les informations nécessaires pour comprendre où en est le projet (c'est à cela que va servir un tableau Kanban par exemple).

Tous les outils « communicants » ne fonctionnent pas toujours comme on pourrait l'espérer. Lors de la réalisation de projets avec des équipes agiles

[11] retour de l'équipe sur comment s'est passée la dernière itération, (quels problèmes se sont posés, y a-t-il eu sur ou sous-performance de l'équipe …) afin de capitaliser et de mettre en oeuvre des actions d'amélioration pour les itérations suivantes (but poursuivi : atteindre le « kaizen »).

réparties sur la Grande Bretagne, les Etats Unis et l'Inde, les équipes de ThoughtWorks ont, par exemple, tenté d'utiliser des applications de type « wiki » pour régler les problèmes de communication qui peuvent naître entre deux équipes séparés fortement au niveau géographique (pas les mêmes bureaux et pas la même heure). Les premières tentatives ont « tenu » une ou deux semaines selon Martin Fowler et ont fini par s'améliorer quand ThoughtWorks a décidé d'adapter l'outil existant à ses besoins. Ce qui est à retenir ici, c'est qu'un outil « communicant », même de grande qualité, ne sera pas d'une grande aide s'il n'est pas adapté au contexte de l'équipe projet.

La communication au sein d'un projet agile est à la fois un prérequis (essayer d'utiliser l'agile en étant persuadé que l'e-mail est la seule manière efficace de faire passer des informations ne fonctionnera pas), mais aussi une conséquence (si les bons outils sont utilisés et que tout le monde « joue le jeu » il y a de bonnes chances pour que le projet fonctionne bien et que la communication soit naturelle). L'ensemble des outils que l'on associe à l'agile tels le tableau kanban ou encore les stand-up meetings peuvent se révéler d'une grande efficacité mais encore faut-il que les personnes participant au projet sachent pourquoi elles le font et aient la conviction que c'est efficace.

c. Localisation des équipes

Comment l'éclatement trop prononcé d'une équipe peut empêcher les outils fournis par l'Agile de fonctionner à fond ?

Une des valeurs fondamentales du manifeste Agile est que la communication face à face est un des meilleurs moyens de transmettre efficacement l'information. On peut donc dire, par extension, que la co-localisation d'une équipe qui souhaite travailler de façon Agile est un élément important, puisque cette dernière va favoriser les échanges « face à face » (en comparaison d'une équipe éparpillée sur plusieurs lieux). On peut, de plus, affirmer qu'il existe, d'une certaine façon, différentes formes de co-localisation, selon la manière dont l'équipe est organisée ainsi que ses besoins. Typiquement, la première co-localisation à laquelle on pense est celle de l'équipe elle même (celle qui va se charger de la réalisation du projet et qui souhaite fonctionner en mode Agile), mais on peut aussi penser à la co-localisation avec le client final (si l'équipe en charge du projet a des besoins de retours réguliers par exemple).

On peut donc considérer qu'il existe au moins deux types de co-localisations « utiles » en agile : la co-localisation des membres de l'équipe qui va mener le projet et celle entre l'équipe et le client final. De ces deux formes de co-localisation, chacune est importante à son niveau mais n'est pas un prérequis pour l'autre. Typiquement, une équipe qui va être en contact régulier avec son client final (celui qui va utiliser le produit) pour ajuster le déroulement du projet afin de livrer un produit correspondant à ses besoins pourrait gagner à être installée chez son client final d'une manière ou d'une autre. Cela n'est pas forcément le cas de toutes les équipes.

Colocalisation de l'équipe

Idéalement, une équipe voulant travailler en mode agile a besoin de plusieurs choses dont le fait d'être co-localisée.

Ceci renforce son aspect « task force » et permet de faire fonctionner à plein les outils de management visuels tels que le tableau Kanban ou encore les « stand up meetings » de début de journée. Le manifeste agile nous rappelle aussi que la façon la plus simple et la plus efficace de transmettre une information est de le faire face à face, et que la communication orale est, de ce point de vue, à privilégier dans un contexte agile.

Guillaume Duquesnay (Coach à Octo Technology), dans sa conférence USI 2009 (« Vaincre le Plouf »), va même plus loin en insistant sur le fait qu'une équipe n'est, d'une certaine façon, pas totalement co-localisée si elle est coupée en deux groupes situés à plusieurs mètres de distance dans un même openspace. Pour lui, ces deux « bouts » d'équipe ne vont déjà plus communiquer comme si tous étaient les uns à côté des autres et, selon son expérience, on va commencer à voir la différence sur des sujets aussi triviaux que de décider quand partir en pause déjeuner …

La façon dont on va répartir les membres de l'équipe projet n'est donc, en ce sens, pas à prendre à la légère.

Colocalisation avec le client final

Être chez le client va permettre, d'un côté, au client de pouvoir profiter des outils utilisés par l'équipe pour suivre le travail accompli au fil du temps mais aussi pouvoir plus simplement s'investir dans le projet.

Le fait d'être chez le client va aussi permettre de simplifier la mise en œuvre de principes comme le PDCA (Plan, Do, Check, Act) quand on a besoin d'un retour du client final régulier (par exemple quand cela concerne l'utilisation du produit fini). Régis Medina a discuté de cet aspect lors de la présentation qu'il a faite au séminaire Lean SI du 10 Juin dernier à Telecom ParisTech, lorsqu'il a abordé le sujet de la « double malédiction » de l'informaticien par rapport à son client : une « malédiction du savoir » (il lui est impossible de se détacher totalement de ce qu'il a fait, ce qui peut l'empêcher de bien saisir les problèmes de ses utilisateurs), mais aussi une « malédiction de l'ignorance » (il lui est difficile de connaître le métier de son client aussi bien que ce dernier, et donc de comprendre exactement ce qu'il souhaite et ce dont il a besoin) . Ici, être au contact régulier du client final peut se révéler être un réel avantage car cela va permettre de mieux connaître le métier de son client, d'en comprendre peut être un peu plus facilement les enjeux concernant le projet, ou même de pouvoir plus facilement analyser ce qui ne « plait pas » dans ce qu'a produit à un instant *t* l'équipe projet.

En revanche, il faut avoir tête qu'il n'est pas toujours réaliste et

pertinent, d'aller voir les « clients finaux » : pour une équipe de développement d'application pour Smartphone « grand public » il n'est à priori pas réaliste de penser pouvoir faire de la co-localisation avec le client final, par exemple.

L'éclatement d'une équipe où comment le courant peut mal passer avec la distance :

Lors de son intervention à l'USI 2010 Guillaume Duquesnay, nous expliquait comment le fait qu'une équipe (projet agile, bénévoles …) ne soit pas co-localisée, par contrainte ou par choix, réduisait les options disponibles de communication entre ses membres. Nous avons tendance à essayer de reconstituer des éléments que l'on n'a pas quand un canal est « fermé » et cela peut nous amener à mal interpréter une information donnée par une autre personne. En guise de conclusion, il recommandait, d'un côté, d'être le plus détaché possible du support mais aussi d'observer la plus grande rigueur possible quand aux moments ou l'on choisit de communiquer sur un canal « dégradé » (on aura plus volontiers tendance à être distrait si l'on n'est pas face à ses interlocuteurs pour discuter avec eux mais, par exemple, derrière son clavier). Il ajoutait que, si les technologies actuelles permettent de plus en plus de pallier le fait que l'on n'est pas dans la même pièce, ne pas être physiquement face à face fait, de toute façon, perdre au moins une partie de ce que l'on pourrait communiquer à l'autre (langage corporel, intonations …). Cela a parfois un effet distrayant, réduisant l'efficacité de la communication. La technologie ne dispense pas d'être vigilant et présent dans le cas de communications dégradées. Au contraire, elle devrait inciter à être encore plus « affuté » que dans un cadre de communication face à face.

Lorsqu'il parle de la façon dont les équipes de ThoughtWorks (USA, Grande Bretagne et Inde) travaillent entre elles, Martin Fowler fait le même constat : travailler sans pouvoir se voir est un réel problème. Pour l'éviter à ThoughtWorks, quand une équipe est répartie sur plusieurs sites, l'entreprise fait régulièrement voyager des développeurs et project managers d'un site à un autre (aussi bien depuis la Grande Bretagne vers l'Inde que le contraire par exemple). Cela permet à chacun de mieux comprendre comment les autres fonctionnent et, de cette façon, de créer un lien plus fort qu'une simple relation de travail. L'idée derrière ces voyages est que, pour ThoughtWorks, le gain d'efficacité compense largement les dépenses en billets d'avions, ces voyages contribuant beaucoup à la formation de chacun et à l'esprit d'équipe mais aussi au « qu'est-ce que je fabrique ? » dont parle Régis Medina. Cela permet une implication de tous par la connaissance des objectifs de la société au delà du projet lui même, mais aussi du « comment » celui-ci s'intègre dans quelque chose de plus grand (projet, programme ou stratégie globale de l'entreprise cliente).

L'éclatement d'une équipe où comment on arrive à travailler en agile avec la

distance :

L'absence de possibilité de co-localisation ne signe pas nécessairement l'arrêt de mort du développement en agile d'un projet. Il a été observé par des coaches agiles que, pour certaines équipes qui sont obligées de travailler à distance, on voit parfois apparaître diverses mesures pour compenser le manque de telle ou telle part de l'équipe. On l'a vu plus haut avec la façon dont ThoughtWorks organise son offshoring en Inde mais il existe aussi d'autres moyens qui vont permettre à une équipe agile éparpillée de rester efficace. Quand deux équipes sont séparées et qu'elles souhaitent travailler en agile malgré cela, elles vont en général développer des moyens leur permettant de contourner le problème que peut représenter l'absence de co-localisation. Dans certaines équipes projet, cela va se traduire par la duplication de certaines fonctions pour pouvoir « combler le vide » et travailler même quand il n'est pas possible d'obtenir de l'aide de l'autre (indisponible ou tout simplement trop chronophage pour elle). De cette façon on va, par exemple, voir se développer des compétences de MOA dans des équipes qui ne sont pas autant en contact avec le Product Owner qu'elles en auraient besoin. Il faut toutefois noter que cela requiert un certain niveau de maturité et des équipes qui, globalement, communiquent bien : il ne s'agit pas de monter deux équipes concurrentes mais bien de continuer de travailler main dans la main avec la ou les autres équipes qui sont sur le même projet.

d. Turnover des équipes

D'une façon générale, le turnover dans une équipe peut poser problème. Il faut le temps que la nouvelle personne arrive, s'intègre, soit formée … Cela prend du temps et monopolise en partie celui de l'équipe déjà en place qui risque donc de perdre en efficacité. Il existe diverses panoplies d'outils pour parer le plus possible à cet effet « néfaste » du « nouveau » remplaçant une personne partie du projet. Des méthodes comme TWI[12], spécialisées dans la formation rapide, sont là, en partie, pour mitiger la perte d'efficacité que peut induire un nouvel élément dans une équipe. L'agile avec ses outils, peut aussi permettre de limiter l'impact du turnover, cependant cela ne fait pas tout.

Pendant mon stage j'ai travaillé sur l'ERP Octo, un projet qui existe et qui livre des nouvelles versions depuis environ quatre ans maintenant. Il est géré avec la « panoplie agile » : il y a un client, un Product Owner, un Scrum Master et nous utilisons divers outils comme le TDD et le pair programming pour cadrer le développement. J'ai pu m'intégrer sans soucis particuliers à l'équipe et commencer à travailler. Le projet avance bien et comporte un nombre assez important de fonctionnalités, toutes opérationnelles. C'est d'autant plus impressionnant quand on sait que cet outil est réalisé par des Octos volontaires sur leur temps « libre »

[12] Training Within Industry

(intercontrat), mais c'est aussi cette particularité qui contribue, à mon sens, à une partie de la dette technique du projet : chacun ayant sa lecture des normes utilisées (et la connaissance de cette norme se diluant), on tombe parfois sur des portions de l'application qui fonctionnent mais qui devraient être complètement réécrites car peu maintenues et plus à jour. Au regard de ce projet, on peut dire que, si les outils fournis par l'agile vont clairement l'aider à ne pas dévier (l'ERP « maison » Octo fonctionne toujours après 4 ans à ce « régime »), conserver les mêmes personnes pendant sa durée de vie est sans doute un atout (Les règles implicites en place perdurent, la conscience de ce que l'on fabrique et des objectifs est commune …). D'une certaine façon, on peut même parler d'un engagement certainement plus fort lorsque l'on a « materné » le projet sur une base régulière que lorsque l'on a travaillé dessus de façon épisodique. De plus, si j'ai pu me mettre à travailler rapidement sur cet ERP à partir du moment où j'en ai su juste assez sur son architecture et les outils utilisés pour ne pas tout « casser », je suis conscient que certains problèmes que j'ai pu rencontrer auraient pu être évités si j'avais pu bénéficier de l'expérience d'une équipe « à demeure ». D'une certaine façon, un projet sur lequel travaillent des volontaires qui vont et viennent est la « pire » forme de turnover qui existe (pas d'effectif fixe pour ainsi dire) et ce n'est réellement pas facile à appréhender pour un « nouveau ».

Au final, ce que montre mon expérience ici est que, comme on pourrait s'y attendre, les outils ne font pas tout et qu'une équipe soudée, qui sait comment fonctionne le projet et en partage une vision commune, sera plus capable de mener l'intégration de « nouveaux » (en leur faisant faire le « tour du propriétaire ») , d'intervenir efficacement sur le projet mais aussi de prendre des décisions pertinentes le concernant.

3. L'Agile convient-il à tous ?

a. Les individus face à l'agile (existe-t-il des « profils » plus compatibles que d'autres avec les méthodes agiles ?)

Les indicateurs de personnalité de type MBTI[13] ont, selon Guillaume Duquesnay, un problème : leur fiabilité dépend grandement du type de projet sur lequel la personne doit travailler. En ce sens, il est compliqué de s'en servir comme d'un indicateur infaillible pour se dire si telle ou telle personne sera à l'aise dans un projet mené en Agile. Il souligne toutefois que, l'on peut repérer certains signes pour savoir à quel point telle ou telle personne va s'intégrer à une équipe et y participer productivement ou pas.

L'agile c'est, comme dit plus haut, un « pari de la communication et de la confiance » : il faut que tous les membres de l'équipe soient à même de pouvoir s'exprimer. Cela ne veut pas dire que toutes les personnes travaillant sur un projet doivent être des clones taillés sur le même modèle, s'écoutant tous les uns les autres, ne provoquant jamais de conflit ... Les personnes critiques sur ce qui a été décidé constituent

[13] Meyers Briggs Type Indicator

toujours un atout car ils mettent à l'épreuve les idées incubées par l'équipe. La plupart des personnes ouvertes d'esprit vont être capables d'appréhender l'agile et ses valeurs et donc d'interagir sainement dans cet environnement.

En revanche, un héros excellent partout et omniprésent ne constituera pas forcément un atout et risquera d'étouffer les autres membres de l'équipe en les empêchant de s'exprimer. D'une façon générale, si l'on s'aperçoit dans un projet qu'un de ses membres y est surreprésenté (position de « leader », il est considéré comme un référent et intervient systématiquement sur pratiquement tout ce qui touche au projet sans trop laisser de place aux autres), c'est en général qu'il y a un problème de communication qui, s'il ne s'est pas posé, risque de surgir à l'avenir.

L'agile, par essence, est une façon de gérer un projet qui prend en compte le changement. Les personnes averses au changement (dans le sens qu'elles aiment quand tout est « balisé » et avec un fort niveau de contrôle) rencontreront aussi certainement plus de difficultés à travailler en mode agile. Dernier point à prendre en compte et qui peut être un obstacle : les méthodologies employées. Si, par exemple dans le cadre d'un développement logiciel, les membres des équipes de développement sont amenés à travailler en binômes (Pair Programming) il vaut mieux, pour être efficaces, que chaque membre d'un binôme arrive à comprendre (et éventuellement accepter) les objectifs et les bénéfices de ce genre de la méthode. Il peut parfois arriver sur des projets que le point sensible soit en fait une personne faisant un blocage sur la méthodologie même du travail en binôme, plus qu'un « héros ».

Dans les équipes qui fonctionnent bien, il y a en général une autorégulation du groupe qui finit par se faire. Il ne s'agit pas juste de prendre les meilleures personnes chacune dans son domaine, les experts, de les mettre dans une salle, et d'attendre qu'un produit parfait sorte de cette rencontre comme par enchantement.

Quoi qu'il en soit : « La confiance ça s'apprend. Ce n'est qu'au fil du temps (et du ou des projets) que les membres d'une équipe apprendront à travailler ensemble et à se faire confiance mutuellement, ce n'est pas inné et la situation peut changer d'un projet à l'autre » souligne Guillaume Duquesnay.

b. L'agile peut-il s'adapter aux individus ? Dans quelle mesure ?

Comme il est dit plus haut, l'agile et tous ses outils se basent avant tout sur une grande communication entre les membres d'un projet. On peut donc légitimement supposer que, dans le cadre d'une équipe composée de personnes ouvertes, plus que la méthode, c'est l'équipe qui va s'ajuster selon les besoins. De plus, et surtout finalement, l'agile est une boîte à outils dans laquelle on va piocher ce qui paraît être le bon outil (technique ou méthodologique) pour atteindre à ses objectifs (à priori, un projet livré

dans les temps, qui correspond à ce que le commanditaire souhaitait et qui a permis de « développer » les personnes y ayant participé). On peut donc dire que l'agile, « s'adapte » aux besoins d'une équipe projet sur ce point.

4. Gestion de la complexité

Comment la façon qu'a l'agile de gérer la complexité dans un projet peut être un frein ou au contraire un accélérateur pour son déroulement

Avec l'agile, la complexité se gère en isolant les fonctionnalités permettant à l'utilisateur final du produit d'accomplir une certaine action. Ces ensembles de fonctionnalités sont appelés « User Stories ». Celles-ci servent en partie à gérer la complexité d'un problème en forçant l'équipe et le product owner à regrouper certaines fonctionnalités que devra avoir le produit final dans un seul objet (la User Story) et de faire en sorte que sa réalisation puisse être contenue dans une itération. En procédant de façon incrémentale, on développe à chaque itération, une ou plusieurs User Stories en ne se focalisant que sur celles-ci et aucune autre. Les User Stories précédemment terminées ont normalement déjà atteint le niveau de qualité requis par le projet tandis que les suivantes ont été jugées moins prioritaires et ne devront donc pas être abordées dans l'immédiat. Si l'on se rend compte que des user stories moins prioritaires devraient être réalisées avant celles en cours, c'est sans doute qu'il y a eu une erreur d'estimation pendant la phase de rédaction du backlog. Les problèmes techniques complexes sont abordés en en restreignant le plus possible le périmètre. En ce sens, l'agile permet de gérer la complexité en adoptant une approche de « diviser pour régner ».

5. Gestion de la qualité

a. Le coût de la non-qualité

La posture même du lean management et de l'agile en particulier est, plus encore que de limiter les dégâts que pourraient occasionner des défauts remarqués trop tard dans un projet, de les empêcher d'y entrer en premier lieu. Le fait d'utiliser des itérations très courtes où un livrable partiel doit être fourni permet de remarquer très vite les défauts qui ont pu être introduits. L'état d'esprit reflété par le TPS (Toyota Production System), le manifeste agile ou encore des méthodes de développement comme TDD, permet de le faire. Une des manières de mesurer l'interrêt d'une telle approche c'est encore de s'attarder un instant sur le coût de la « non-qulité ».

Si l'on considère une gestion de projet traditionnelle (cycle en V, bonnes pratiques PMI, …) il est admis que le coût des défauts grimpera exponentiellement au cours d'un projet (plus on se rend compte tardivement qu'il y a un problème, plus il sera cher à corriger). Dans le cas

d'un projet informatique mené selon les « canons » de l'Agile, le coût observé de la non-qualité augmente bien au fil du temps, mais de façon beaucoup moins prononcée (augmentation logarithmique plutôt qu'exponentielle). La raison majeure à cela est qu'en utilisant des itérations courtes, on a plus de chances de se rendre compte rapidement de l'existence d'un défaut et donc de pouvoir le réparer simplement (et au moment ou cela coûte le moins : juste après qu'il soit apparu). Le coût sera sensiblement plus élevé si l'on a du attendre plusieurs mois une validation (par exemple l'action « perform quality control » des bonnes pratiques PMI, qui fait partie du processus transverse « Manage and Control »). Le coût d'un défaut a aussi ce potentiel de pouvoir rester « marginal » du fait qu'un projet agile va se dérouler de façon incrémentale : on ne passe pas d'une phase d'une certaine nature à une autre qui serait différente, on boucle toujours sur la même tant qu'on n'a pas produit le résultat final. De ce point de vue, « revenir à la planche à dessin » est moins coûteux que de devoir éventuellement retourner plusieurs mois en arrière dans une ancienne phase.

Image 3 : Le coût du changement sur un projet "traditionnel" en fonction du temps

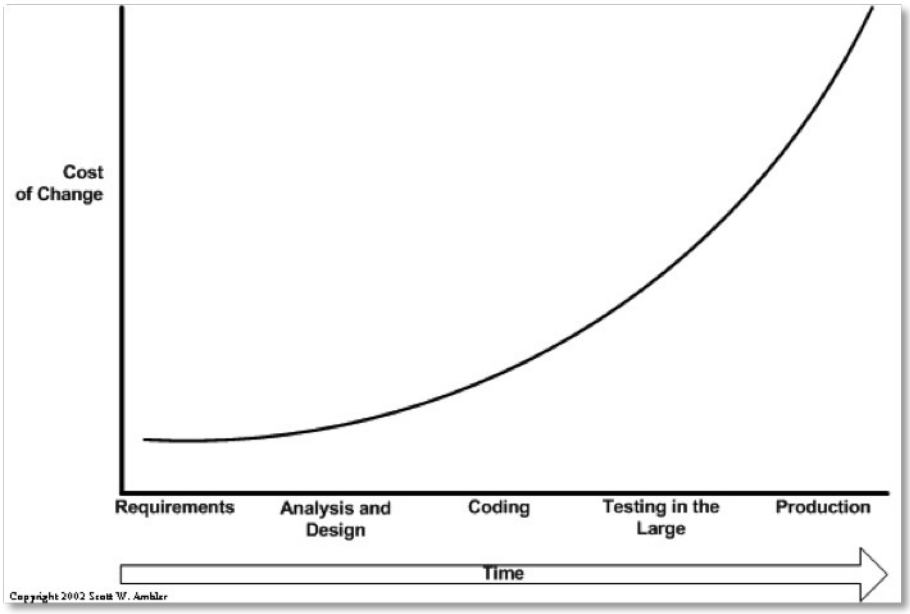


Image 4 : Le coût du changement dans un projet Agile "idéal" en fonction du temps

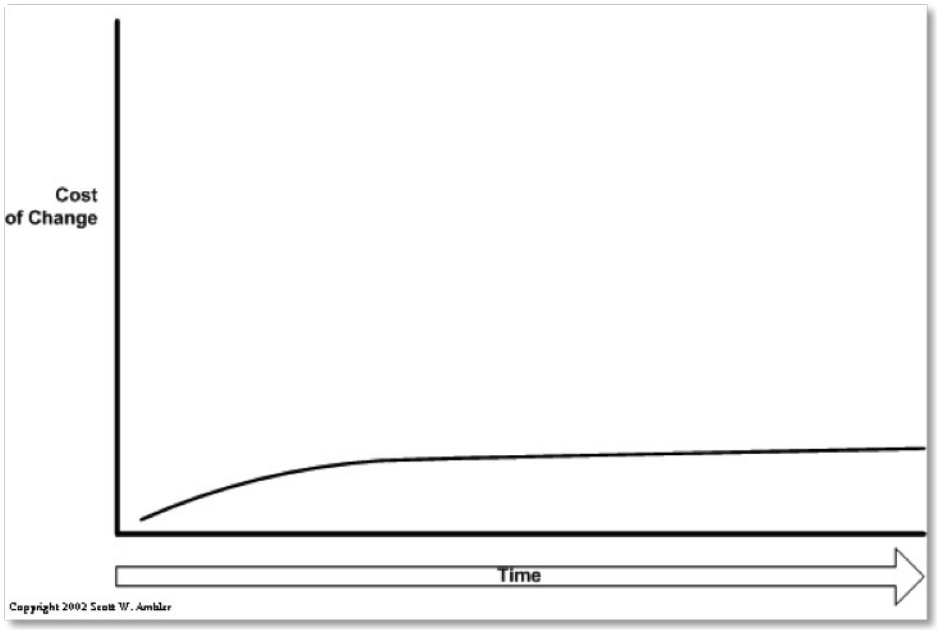
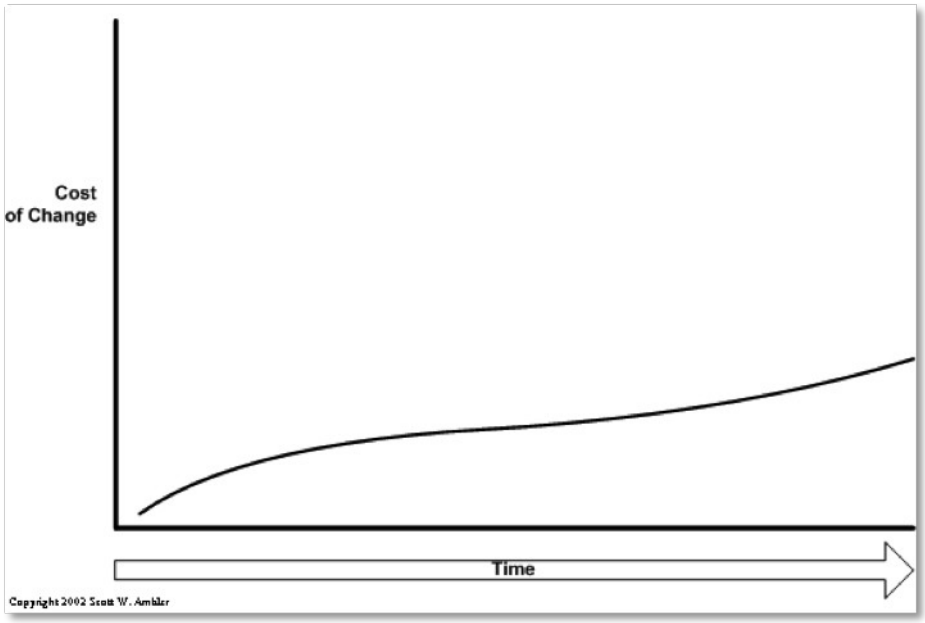


Image 5 : le coût du changement dans un projet Agile "standard" au fil du temps

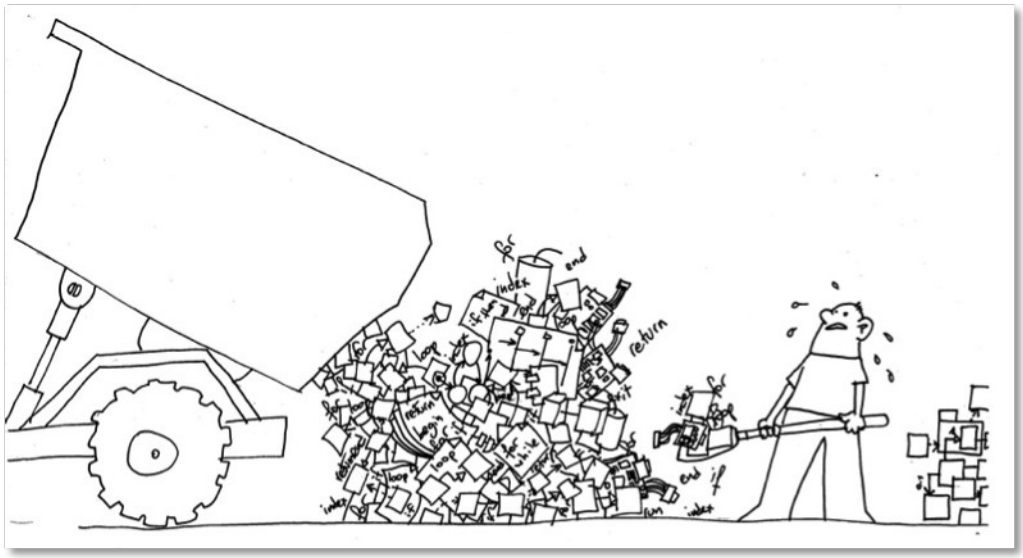


b. Criticité de la qualité et risques induits dans certains projets

Dans un autre ordre d'idée, il y a des domaines dans lesquels on a besoin d'un haut niveau de qualité du produit fini et où les modèles industriels tels que le cycle en V à passe unique posent problème. Lors de son intervention à la conférence Lean SI du 10 Juin 2010, Emmanuel Chenu nous a fait part du chaos que peut représenter l'intégration de logiciels embarqués dans un équipement critique pour un produit plus grand (dans ce cas : une centrale inertielle à embarquer dans des avions de ligne). Lorsque l'on suit la méthodologie classique qui a encore cours dans le domaine de l'aviation (modèle industriel) et où l'intégration du logiciel se fait à la fin, faute de disposer le matériel qui va servir à l'intégration en

amont, il est en général nécessaire de procéder à du rework (parfois en profondeur) et on peut légitimement se poser des questions sur la fiabilité d'un produit dont on assemble toutes les pièces au dernier moment en espérant que l'appareil « tombe en marche » du premier coup après assemblage ... Ce n'est pas tout à fait l'image de rigueur à laquelle on pourrait s'attendre de projets traitants d'éléments critiques. Emmanuel Chenu a d'ailleurs fait référence à ce mode d'intégration par le nom « Intégration Big Bang » lors de sa présentation (et ici, « Big Bang » n'est pas tant là pour décrire l'intégration comme « source de tout ce qui est » mais bien plutôt comme « grand chaos résultant de l'accumulation de tous les problèmes qui n'ont pas pu être traités en amont pour de diverses , en général mauvaises, raisons »).

Image 6 : L'intégration "Big Bang" illustrée par Emmanuel Chenu



Lorsque la qualité est critique pour un projet (au sens que si le produit final est de mauvaise qualité il représente un danger pour ses utilisateurs), il constitue un risque à ne pas négliger pour toutes les parties prenantes, un risque financier d'une part mais aussi souvent un risque associé à l'image des acteurs de ce projet.

Pour la société qui fournit le logiciel, sa non-qualité est un risque financier car cela va induire du rework qui ne pourra pas toujours être facturé au client puisqu'étant « de la faute » de la société. La non qualité, de plus, représente aussi un risque en termes d'image. Si jamais un produit « critique venait à provoquer un accident grave à cause de problèmes de qualité cela aurait un impact négatif indéniable sur l'image de l'entreprise, tant parmi ses pairs dans son secteur d'activité, que dans l'opinion publique.

Pour le projet dont Emmanuel Chenu est responsable, la mise en place de

méthodologies agiles dans les étapes de développement logiciel a permis, de « tranquilliser » ses supérieurs avec une équipe qui, une fois rôdée aux nouvelles méthodologies employées, est parvenue à tenir les délais négociés en début de projet y compris pendant la phase d'intégration (pourtant la plus complexe à gérer et sujette à impondérables). Les outils agiles et le TDD ont, de plus, posé les bases leur permettant de mettre en place de nouveaux paradigmes de programmation qu'il leur étaient impossible d'utiliser jusque là à cause de la méthode de gestion de projet elle même. Le TDD par exemple leur a permis de gagner en efficacité en introduisant « l'inversion de dépendances » dans leur projet (le logiciel développé n'était plus directement dépendant du matériel, dont le fonctionnement était simulé).

La phase d'intégration engendre généralement divers retards éventuellement assortis de pénalités pour les sous-traitants des avionneurs et constitue donc un risque non négligeable que les méthodes agiles peuvent permettre de réduire à défaut de les faire disparaître. Du côté de l'avionneur, le risque du retard d'un de ses sous-traitants peut aussi représenter un risque vis à vis de ses clients si la pièce sous-traitée est nécessaire à l'avancement des commandes. Le fait qu'une société puisse livrer à temps et ne pas prendre des retards dont personne ne connaît la durée est donc appréciable. De plus, si l'on s'attache à un aspect plus sécuritaire, les pièces « critiques » des avions (y compris les « pièces logicielles ») ont en général besoin de l'agrément de la FAA (Federal Aviation Administration) ou de l'EASA (European Aviation Safety Agency). Ces autorités sont rassurées de savoir, par exemple, que le logiciel qui gère la centrale inertielle du prochain Airbus a été testé des centaines de fois et que la probabilité de trouver des « bugs » après sa mise en service est très faible.

Ici, l 'agile permet de gérer à la fois à un risque lié au projet (refus de l'agrément d'une autorité de certification, qui impliquerait de revoir en profondeur sa copie, voire de repartir de zéro) et aussi celui qui se manifeste pendant la vie du produit (la sécurité des passagers). Il n'est pas question de dire que dans une conduite de projet classique, le produit n'est pas testé de façon approfondie avant d'être mis en service (dans le cas d'un appareil aussi critique et sensible qu'une centrale inertielle il ne recevrait pas son agrément s'il n'était pas jugé entièrement sécurisé de toute façon), mais la philosophie agile fait des tests une partie intégrante de tout le processus de « réalisation » et est moins vue comme une série de « checkpoints » à passer au fur et à mesure mais plutôt comme un processus continu visant à empêcher les défauts de pénétrer dans le futur produit.

c. L'agile : une méthodologie « sans concessions » au niveau de la qualité du livrable ?

L'agile, par ses outils, peut faire penser que c'est une sorte de « super méthodologie » de gestion de projets qui livre toujours à temps et ne fait

jamais de compromis sur la qualité de ses livrables. D'une certaine façon, c'est en partie vrai. Le fait que le pilotage des projets se fasse en tenant compte exclusivement de ce qui a été réalisé oblige les équipes à maîtriser la question de ce qui est fait. Cependant, le fait que l'agile permette de modifier, somme toute, assez simplement le périmètre d'un projet (et d'en prioriser les lots) tout en pratiquant la livraison itérative fonctionnelle a une raison assez simple : il est toujours possible de décider d'abandonner des lots non prioritaires pour à la fois tenir les délais et les critères de qualité souhaités (les délais n'étant en général pas renégociables). L'idée ici est que l'agile permet de faire des concessions maîtrisées quand cela est nécessaire et que, quoi qu'il arrive, le positionnement de l'agile face à la qualité de ce qui est livré est de faire preuve de rigueur de sorte à assurer la qualité et la pérennité du produit.

VIII. Cas personnel (démarche de R&D dans les services innovants)

Entre le TPS dans les usines Toyota et eXtreme Programming … la R&D

Afin de faire une expérience supplémentaire au sujet des domaines d'applications possibles de l'agile, il m'a été donné l'opportunité de faire un stage plutôt orienté « R&D » chez Octo Technology. Cela m'a permis d'expérimenter l'applicabilité de l'agile à des domaines peut être moins contrôlés que des projets qui appliquent, justement, les résultats de ces phases de R&D (la créativité ne se commande pas et qu'il est à priori difficile de gérer une démarche de R&D exactement de la même façon que l'on gèrerait un autre projet).

1. Méthodologie et outils utilisées pour encadrer mes démarches :

Tout au long de mon stage, j'ai été amené à travailler sur des aspects différents de ce que l'on peut qualifier comme de la R&D : je suis passé successivement par des phases de veille puis de prototypage et une des grandes « constantes » entre ces phases a été la façon dont je m'organisais et les interactions avec mes responsables.

Lors de mon travail, mes responsables et moi même avions plusieurs facteurs en tête que nous souhaitions maîtriser le plus possible :

- *Temps*

  Le temps a été un élément important de ce stage et ce pour trois principales raisons :

  - Mes travaux étant essentiellement exploratoires, il fallait néanmoins qu'ils prennent fin, dernier délai, lorsque mon stage se terminerait ;
  - Une autre raison à cela était que le temps « présentiel » (où ma responsable pouvait venir discuter avec moi de mon avancement et de la suite des évènements) pouvant être réservé à mon encadrement était compté et qu'il fallait donc en faire bon usage ;
  - Enfin, l'ensemble des travaux faits pendant mon stage devait, idéalement, se terminer et être consultables et utilisables par les autres consultants du cabinet lorsque je partirais.

  D'une façon générale, organiser des livrables y compris sur les phases

exploratoires de mon travail a permis de me contraindre à finir les travaux qui m'étaient demandés les uns après les autres (et ainsi éviter de tout « empiler » pour devoir tout livrer la dernière semaine). Pour les personnes qui ont évalué mes travaux, cela a, d'une certaine façon, permis de « lisser » l'activité de relecture.

- *Performance*

Il n'est pas toujours aisé de mesurer la performance de travaux de R&D, toutefois, pour avoir une idée de la quantité de travail que je pourrais effectuer d'ici la fin de mon stage il nous fallait trouver une façon de la mesurer. Les travaux de « pure » R&D étaient donc sujets, comme les activités plus orientées « réalisation » de mon stage, à des évaluations « de complexité ». J'essayais donc, d'un côté, de me rappeler si je disposais déjà des informations dont j'allais avoir besoin pour écrire tel ou tel document ou s'il allait falloir faire des recherches, et dans ce cas essayer d'estimer si les informations après lesquelles j'allais « courir » seraient plutôt simples à trouver ou au contraire difficiles.

- *Pertinence*

Étant quelque peu nouveau dans le monde des telcos, j'avais besoin de retours sur mes travaux de la part de mes responsables afin d'éviter d' « enfoncer des portes ouvertes » et de me focaliser sur les éléments les plus importants.

A ces fins, pour remplir les objectifs de temps, performance et pertinence, nous avons mis en place un certain nombre d'outils pour m'aider dans mon travail au quotidien, mais aussi pour être capables de pouvoir estimer, à un instant t, ou j'en étais et le travail que j'étais capable de fournir à horizon d'une ou plusieurs itérations. J'ai ainsi effectué mon stage en m'appuyant sur plusieurs outils et concepts :

- Itérations courtes

Rapidement, nous nous sommes mis d'accord sur une base d'itérations (avec point et présentation de mon travail à leur fin) de deux semaines, de sorte que je devais donc régulièrement présenter la partie de mes travaux sur laquelle je m'étais engagé à travailler pendant l'itération et qui devait être finalisée et éventuellement utilisable immédiatement après le point. Ces itérations courtes m'ont permis d'obtenir un feedback « formel » (parce que programmé) en plus du feedback que j'ai pu recevoir pendant les itérations elles mêmes, que cela soit de la part des personnes m'encadrant ou de consultants intéressés par mon travail ;

- Kanban, Backlog & Burndown chart

Afin de me permettre de m'organiser, nous avons rédigé un backlog très tôt et priorisé les User Stories de sorte que j'aie produit le maximum possible de « valeur » (livrables sous forme d'articles, de slides, de prototypes …) avec mes travaux d'ici la fin de mon stage. Le backlog permettait, au début d'une itération, de définir les « livrables » à fournir d'ici le prochain point avec ma responsable, le Kanban, quant à lui, lui permettait de pouvoir consulter facilement ou j'en étais pour chacun d'entre nous. Le burndown chart nous a permis d'évaluer la quantité de travail que je pouvais accomplir entre deux itérations et donc de décider de la charge de travail pour les suivantes en tenant compte de cela ;

- Feedback régulier / gemba

D'une certaine façon, ce feedback est une application du concept de « Gemba » ou l'on consulte régulièrement le client de son projet pour voir si ce que l'on est en train de produire convient toujours à ce qu'il souhaite ou si l'on commence à dériver. En utilisant des itérations de deux semaines, j'ai pu disposer du feedback de plusieurs personnes à intervalles réguliers et continuellement travailler en évitant à mon projet de dévier.

Le concept de « gemba » a été particulièrement mis à profit lors des phases de développement de prototypes (mais n'a pas été ignoré durant les recherches plus classiques), avec les personnes intéressés par mes recherches qui me servaient de « cobayes » et potentiels utilisateurs finaux, et me permettaient de déterminer les bonnes choses et les moins bonnes choses dans le travail que je produisais ;

- Kaizen (amélioration continue)

Le fait de se réunir régulièrement et de pouvoir demander de l'aide autour de moi m'a permis, d'une part, de ne pas dévier dans mon projet (il y avait toujours quelqu'un pour me remettre « sur les rails » au besoin), mais m'a aussi de m'améliorer d'un point de vue méthodologique et technique voir d'apprendre de nouvelles choses. Ceci m'a permis d'être plus efficace par la suite (meilleure maîtrise de mon organisation et plus grande efficacité technique) et donc de pouvoir achever des travaux de plus en plus complexes pour moi dans une seule itération.

2. Résultats attendus / obtenus

a. Sur les « trois facteurs »

Si l'on considère les trois facteurs « clés » de ce projet (temps, performance, pertinence), voici ce que j'ai pu observer au cours de mon stage :

- Temps

Le Kanban a été d'une grande aide d'un point de vue organisationnel. Bien que n'étant pas utilisé au maximum de ce qu'il a à offrir, il m'a permis, ainsi qu'aux personnes m'encadrant, de rapidement prendre connaissance de l'avancement global de mon projet à un instant t.

Les indices de performance calculés à partir du backlog ont, eux, permis d'effectuer des projections sur la date de fin « idéale » du projet (fin du projet ou j'ai le temps de livrer toutes les User Stories), et selon les cas, de modifier le scope (ajouter ou enlever des User Stories) ou ne rien toucher.

Ce que l'on attendait du Kanban était qu'il permette à toutes les personnes m'encadrant ou même simplement me suivant de loin de savoir où j'en étais. Il a rempli son œuvre ;

- Performance

Le backlog ainsi que le burndown chart et les calculs de vélocité ont permis d'estimer de façon de plus en plus précise le travail que je pourrais accomplir pendant une itération. Cependant, il est toujours difficile d'évaluer la complexité de tâches de R&D, surtout lors des premières itérations (qui servent avant tout à « amorcer la pompe » aussi bien en termes de recherches que d'évaluation de ce que ces travaux peuvent représenter en travail), aussi les projections fournies par ces outils n'ont pas été utiles immédiatement, il leur a fallu plusieurs itérations pour que les mesures qui en sortaient permettent effectivement de faire des projections un peu plus précises ;

- Pertinence

L'approche même de l'agile m'a donné l'opportunité de me mettre dans une posture d'amélioration continue qui m'a permis de fournir des résultats en accord avec ce que souhaitaient mes responsables (en termes de qualité méthodologique et technique ainsi que de sujets abordés). L'objectif de l'approche Agile était ici d'explorer un maximum de pistes en rapport avec mon sujet de stage d'ici sa fin, cet objectif a, nous le pensons, été atteint : j'ai été capable de me focaliser sur les sujets que nous avons estimé les plus « critiques » en priorité et ce en évitant de dévier grâce aux points réguliers et à la taille des itérations, les livrables requis à chaque itération obligeant à fixer des idées, les soumettre et éventuellement les retravailler autant de fois que nécessaire de sorte à délivrer le maximum de valeur « globale » en fin de période.

L'agile a aussi permis, dans une certaine mesure, à tous nous aider à mieux définir mon périmètre de recherche et, en particulier, de l'ajuster « à chaud » quand cela était nécessaire (nouvelle « bonne idée » ou, au contraire, idée à la réflexion pas si bonne que cela). Cet aspect relevant de l'adaptabilité de l'agile en contexte « incertain » a plutôt bien fonctionné. En particulier sur la réalisation de prototype, la souplesse conférée par ces méthodes a permis de retravailler certains aspects ou de changer le scope sans heurts ;

b. L'exemple (proto)type :

Lors de mon stage, j'ai donc réalisé un prototype « proof of concept » pour illustrer une idée sortie de mes phases de R&D. Ce « projet dans le projet » a lui aussi été mené en mode agile (User Stories dans le Kanban, compte rendu de progression et livraison fonctionnelle à tous les points de stage ou ce projet était d'actualité …). Ce projet a donné lieu à des modifications de scope diverses (l'idée était là mais nous ne savions pas encore bien au début ce qui était techniquement faisable ou pas) et à fini par aboutir à une réalisation fonctionnelle qui ne contient pas tout ce qui était prévu à la base (déscopages divers, retours en arrière …), mais contient aussi de nouvelles fonctionnalités (ajouts au scope initial), pour servir un objectif double :

- « Prouver que ça marche » (principe même du « proof of concept ») en montrant suffisamment de fonctions sans rentrer dans un développement complexe qui aurait demandé plus de personnes ou de jours disponibles de ma part ;
- Pouvoir être déployé sans trop de difficultés pour être montré

Si, dans l'ensemble, les outils à ma disposition ont plutôt été d'une aide appréciable, il est clair que pour les parties de réalisation ou j'ai, pour l'essentiel, travaillé seul (et après avoir observé plusieurs équipes « agiles » travaillant dans les mêmes locaux que moi et le fait d'avoir, en travaillant sur l'ERP d'Octo, travaillé en équipe), je pense que le fait de ne pas avoir été seul m'aurait sans doute permis d'être plus performant et de, par moments moins prendre le risque de dévier.

c. Octopode NG (ONG) : ERP et « centre de formation agile »

En parallèle des activités liées à mon stage, je me suis porté volontaire pour travailler sur l'ERP d'Octo, développé par des Octos[14] et pour les Octos. Mon objectif était, d'un côté, d'apprendre « de l'intérieur » comment fonctionnait

[14] Surnom donné aux personnes travaillant chez Octo Technology

la société, et de l'autre de pouvoir côtoyer d'autres Octos et ainsi travailler en équipe en mode agile sur un projet en cours (cela fait environ quatre ans que le projet existe et est mis à jour). Il ressort de cette expérience que je suis passé par à peu près toutes les grandes étapes du cycle d'un développement logiciel « agile » :

- eXtreme Programming / TDD : j'ai programmé en binôme avec une autre personne travaillant sur ONG, il nous a fallu respecter les standards de programmation déjà présents et donc, entre autres, mettre les tests au centre de notre démarche ;

- « gemba » : en mettant en place les fonctionnalités demandées par mon PO, il a fallu régulièrement demander des retours de nos « clients finaux » pour voir si ce que nous avions produit correspondait bien à leur désir, et recommencer une itération pour faire mieux dans le cas contraire ;

- livraisons incrémentales, itératives et fonctionnelles : chaque livraison (suivie d'une mise en production si l'on avait le « go » du client et du PO) contenait au moins une nouvelle fonctionnalité qui pouvait être soit un nouveau service rendu par l'ERP, soit un « fragment » de ce dernier, en attente les prochains développements pour être complété.

En plus des nouvelles fonctionnalités que j'ai aidé à développer, j'ai aussi été amené à m'interroger sur le fonctionnement interne de l'application et donc à jeter un œil aux « entrailles » d'ONG et y ai constaté plusieurs choses :

- Oui, un projet agile peut s'étaler sur plusieurs années. Les projets menés en mode agile ne sont pas condamnés à durer quelques semaines maximum ;

- Non, les outils seuls fournis par l'agile ne permettent pas de s'affranchir de la dette technique générée par un projet (il existe maintenant des portions de l'application qu'il faudrait presque ré-écrire « de zéro »), au mieux, ces outils vont permettre d'en réduire les impacts les plus néfastes (régressions de fonctionnalité) et vont servir de « tuteur » à son bon déroulement, mais ce qui va faire le projet, c'est une équipe « unique » (le moins de personnes possible qui ont et viennent).

IX. Table des figures

X. BIBLIOGRAPHIE / WEBOGRAPHIE, CONFERENCES ET INTERVIEWS

- « Agile Project Management 2nd edition » (Jim Highsmith – Addison Wesley 2010)
- www.lean.org (définitions et matériaux autour de ce qu'est le Lean Management)
- Conférence LeanSI de juin 2010 (Telecom ParisTech - intervenants : Régis Medina, Emmanuel Chenu, Marie-Noële Ninteya et Elodie Aïdan)
- Wikipedia français et anglais
  - définitions du cycle en V à passe unique
  - traduction française du manifeste Agile
  - définitions de la méthode de management de projet PRINCE2
  - définitions du Kanban dans le TPS
  - définitions de XP
  - définitions de la méthode Six Sigma
- www.brighthub.com
  - http://www.brighthub.com/office/project-management/articles/67087.aspx?p=1 (tentative de comparaison entre l'agile et plusieurs méthodologies de gestion de projet à planification forte)
  - http://www.brighthub.com/office/project-management/articles/62040.aspx (présentation de la méthode PRINCE2)
  - http://www.brighthub.com/office/project-management/articles/2441.aspx?p=1 (présentation de la méthode six sigma)
- www.universite-du-si.com
  - keynote de Michael Ballé (USI 2010)
  - keynote Martin Fowler & Neal Ford *(ThoughtWorks)*(USI 2010)
  - keynotes de Guillaume Duquesnay « Vaincre le Plouf » (USI 2009) et « Recracher la pilule bleue, avaler la pilule rouge » (USI 2010)

- http://www.agilemodeling.com/essays/costOfChange.htm (Le coût du changement et l'impact de l'agile dessus)
- http://martinfowler.com/articles/agileOffshore.html (« Using an Agile Software Process with Offshore Development »)
- http://martinfowler.com/articles/newMethodology.html
- http://martinfowler.com/articles/designDead.html
- « Rework : change the way you work forever » (J. Fried, D. Heinemeyer Hansson)
- « OCTO Talks ! » (http://blog.octo.com)
- … et l'expertise des personnes comme Guillaume Duquesnay ou Yannick Martel qui ont eu la gentillesse de m'accorder un peu de temps pour discuter sur un sujet ou l'autre abordé dans ce document !

XII. ANNEXES

1. Compte rendu Conférence Lean SI Telecom ParisTech

1ère partie :

Intervenant : Régis Médina

Comment mieux comprendre les attentes du client à partir d'une démarche lean ?

Constat : Le développement logiciel est parfois un peu comme une « theillère diabolique » (le service est rendu mais c'est une horreur pour se servir du logiciel au quotidien).

Écueils du cycle en V dans le développement logiciel :

- On essaye de prendre des décisions de design au moment ou l'on a le moins d'informations sur ce qu'on veut (utilisateurs finaux comme informaticiens), c'est à dire au début.
- Budget consommé intégralement (pas de réserves pour l'inévitable rework)
- Logiciel pas forcément facilement modifiable

Ceci a entrainé le début de la mise en place de pratiques plus « agiles » (écoute du client, livraisons plus fréquentes…). Mais dans la pratique le « miracle » ne se produit pas.

On dit qu'il faut toujours écouter le client, mais parfois, ne pas l'écouter peut être une bonne chose, typiquement quand il a des demandes qui ne coulent pas de source (par exemple quand il demande une fonction pour importer des données et les marquer immédiatement pour effacement), en général à cause d'une habitude pas toujours cohérente prise il y a longtemps avec l'outil précédent.

En général, il y a plusieurs clients sur un projet, il est donc très difficile de tous les satisfaire, sans compter le fait qu'il peut arriver que l'on réalise parfaitement ce qu'a demandé le client mais qu'il n'en soit pas satisfait pour autant (besoin mal exprimé par exemple).

Tous ces problèmes de « non qualité » perçue par le client amènent à se poser une question pas si anodine : « Qu'est-ce que je fabrique ? ». C'est cette interrogation en partie qui conditionne le moment ou l'on doit remettre en question ce que demande le client.

Problème introduit sous forme de blague : PDD (Problem Driven Development). Ou comment organiser son pilotage de projet pour résoudre complètement le problème du client. Cela comprend d'un côté intégrer les demandes techniques spécifiques mais aussi que ce que l'on met en place fonctionne dès qu'on en a besoin et ne soit pas une perte de temps pour l'utilisateur (un minimum voire aucun blocage de l'activité et si possible aucune formation sur le nouvel outil).

Pour faire court : il faut produire la valeur ajoutée sans le « gras » autour pour le client (réduire au maximum le lead time d'utilisation). Exemple de Google : le client souhaite le service que Google lui rend mais ne souhaite pas rester dans le système (sur les pages de résultats de recherche). Aussi le moteur de recherche est optimisé pour rendre un service de recherche le plus pertinent possible afin que l'utilisateur sorte rapidement du

système (suggestions, rapidité, pertinence, propositions de corrections …).

L'ère de la course au features est terminée. Le succès d'un développement, pour le client, se mesure en examinant le lead time d'utilisation (« est-ce plus rapide qu'avant ? »). Et ici, plus rapide ne veut pas seulement dire « plus performant techniquement».

Autre difficulté évoquée :

- Malédiction du savoir (difficile pour un développeur de se mettre dans la peau d'une personne ne connaissant pas du tout le système construit)
- Malédiction de l'ignorance (difficile, voire impossible, pour un développeur de bien connaître le métier de ses clients)

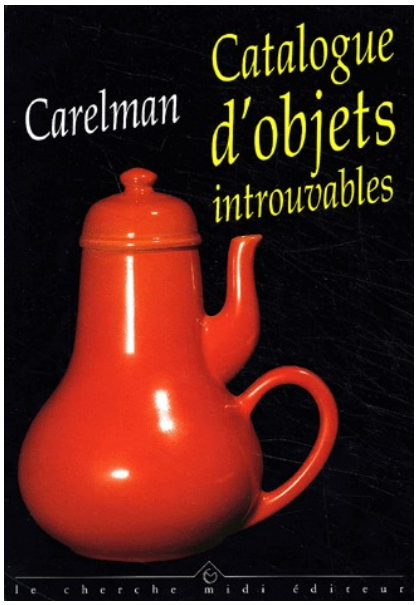
Un des moyens évoqués, en plus d'essayer des solutions que l'on imagine, pour remédier au problème est tout simplement d'aller s'asseoir près d'un utilisateur et de l'observer en silence pour voir comment cela se passe pour un utilisateur « lambda » (il faut se taire, prendre des notes, et laisser parler la personne qu'on observe, si elle en a envie). Essayer de résoudre ses problèmes par des réflexions à l'emporte pièce sans prendre en compte les utilisateurs ne fonctionne pas. Il vaut mieux établir une cartographie des étapes « clé » de l'utilisation d'un logiciel et d'identifier ces étapes (qui sont en fait les points de « souffrance » des utilisateurs quand elles sont mal décrites et / ou implémentées), et travailler avec les utilisateurs pour produire le logiciel qui, non seulement fonctionne, mais en plus est adapté aux utilisateurs. Il y a des choses auxquelles on ne peut tout simplement pas penser en réunion (position des boutons dans une IHM par exemple).

On ne peut pas savoir à l'avance si une évolution sera bien accueillie, et c'est pour ça qu'il faut faire des expérimentations structurées (PDCA).

On peut donc se retrouver avec, au final, deux systèmes de contrôle des actions à effectuer (le système classique que l'on a dans son projet et aussi celui qui permet de gérer les soucis que peuvent rencontrer les utilisateurs).

Le gemba est un atout dont il ne faut jamais se passer. Il va permettre dans le cas d'un développement logiciel par exemple, d'être au contact des utilisateurs et, par itérations successives, de trouver les approches qui permettent à l'utilisateur de pouvoir utiliser le logiciel et d'en améliorer son lead time d'utilisation.

Figure 1 : La théière diabolique de Régis Médina ... qui, selon Carelman, est en fait une cafetière pour masochistes ...



2nde partie :

intervenant :*Emmanuel Chenu*

Pb du dévelopement logiciel dans des systèmes critiques en avionique : le logiciel est la seule partie « molle » et il doit absorber tous les problèmes que pourra poser le matériel.

Les avionneurs ont tout délégué à leurs fournisseurs et sont de plus en plus exigeants, en demandant des choses toujours plus complexes.

Le problème des cycles de développement pour l'avionique c'est qu'on est encore beaucoup dans le cycle en V à passe unique. Certains développeurs, comme l'équipe d'Emmanuel, tendent à instaurer des pratiques lean pour résoudre les problèmes que ces méthodologies leur posaient jusqu'ici.

Avant, pour les fournisseurs des avionneurs, on procédait à une intégration « Big Bang » : intégration de tout le produit par petits morceaux d'un seul coup (divergence + retard de livraison + matériel pas fonctionnel à la livraison). Les projets tiennent les couts / délais jusqu'à la phase d'intégration et là, tout explose (défauts cachés, gaspillages divers, rework, produit pas opérationnel voire instable).

Dans une planification « optimiste », l'intégration représente à elle seule 30% du coût initial du projet voire plus …

Une des causes possibles : l'intégration arrive beaucoup trop tard

Une autre : hardware développé en parallèle du logiciel ce qui pose problème (dans un cycle en V ça explose à la toute fin).

Expérimentations menées pour résoudre les problèmes :

- XP + Scrum avec un RàF priorisé sur le ROI client (il doit pouvoir s'en servir). On passe de phases de développement à une activité continue. Pour ceci, on élabore avec le client des MMF pour définir les priorités et on utilise des Kanbans pour détailler / visualiser les activités et la production.
- Découplage du logiciel du matériel : Utiliser du TDD, procéder à une inversion de dépendances pour que le logiciel soit à 99% indépendant du matériel.
- Mise en place d'un procédé d'intégration continue : les développeurs sont encouragés à livrer plusieurs fois par jour (machine de build qui surveille le repo, joue les tests et build à intervalles réguliers avec jusqu'à plusieurs dizaines de builds / jour)

Tout ceci permet plus de souplesse (le client peut avoir des versions intermédiaires, les développeurs peuvent réutiliser du soft au lieu de réinventer la roue pour chaque nouveau projet …)

Les coûts d'intégration sont passés à 5% du budget initial (et ne sont plus dépassés).

Les avionneurs et les autorités de certification sont ok avec cette façon de faire.

Autre problème : la gestion des défauts

Piste : défauts mal détectés ou détectés trop tard.

Solution envisagée : politique de détection préventive des défauts via des cycles courts (20 jours : plusieurs tests, builds réguliers…), pilotage par les tests automatisés (qui servent de véritable « tuteur » au produit). De cette façon, les tests servent bien à empêcher les défauts de rentrer dans le produit. De plus les tests sont joués sur les machines de développement et le serveur d'intégration (au cas ou les différentes architectures feraient apparaître des bugs).

La partie « rapide » des tests est jouée plus de 100 fois / jour, les « longs » 4.

Dans le cas ou cela ne va plus : « Stop the line », les développeurs s'organisent pour régler le problème, et la production reprend seulement une fois qu'il est résolu.

La couverture de tests a aussi son importance (attention portée aux aspects non testables)

Regret pour l'instant : pas assez d'analyse des causes profondes quand un problème « hors normes » survient.

Programmation par contrat (Bertrand Meyer) pour éviter les mauvaises utilisations de code.

Pair programing (un codeur / un relecteur) pour éviter d'injecter certains défauts

Conséquences :

- 99.9% de couverture, taux de défauts divisé par 10 (5 défauts / 10Klc maintenant)
- Pratiques peu à peu standardisées et diffusées dans les équipes Thalès.

Avantages dans un contexte avionique :

- code testé très souvent et non sujet à interprétation
- lissage de l'activité
- gestion préventive des défauts
- devient un critère de choix pour les avionneurs (sûreté du code testé en permanence, intégré en continu et dont la livraison s'effectue sans « surprises »)

La démarche de résolution des problèmes est une démarche continue, longue, sans fin mais extrêmement motivante (progression de tout le monde, apprentissage, gratifiant), mais il n'y a pas franchement de fin au cycle (syndrome du « ah oui mais là ça marche pas ! » : on n'est jamais en pause).

3ème Partie : TWI

Intervenants : C. Ignace / E. Aïdan / M-N. Ninteya

TWI (Training Within Industry) : Pratique en cours chez Nokia Siemens Networks (NSN).

Le TWI est « né » aux États Unis pendant la Seconde Guerre Mondiale dans le but de permettre à de la main d'œuvre non qualifiée de monter rapidement en compétences sur un processus industriel donné (typiquement : les femmes des soldats américains mobilisés qui ont pris leur place dans les usines pour continuer de faire tourner l'économie du pays).

Le but de cette technique se base sur le constat qu'une équipe commence à être « au top » en termes d'organisation et de performances quand le projet est fini et qu'elle se sépare. Il faut donc mettre en place des pratiques permettant de rendre l'équipe plus efficace plus rapidement. La méthode TWI se focalise sur le transfert de connaissances et comment le faire de façon efficace et pérenne

La démarche TWI se décompose en 3 étapes :

1. Identifier les connaissances essentielles de ce que l'on cherche à transmettre

2. Les diffuser (Préparer, montrer, faire faire, puis consolider)

3. Les améliorer

TWI s'appuie sur des « modes opératoires » pour décrire les étapes d'une opération à réaliser (description complète, illustrations … etc). L'élaboration de ces modes opératoires nécessite l'arrêt de la production pour se focaliser sur des ateliers permettant d'isoler les composantes essentielles des tâches à accomplir. Ces modes opérationnels, une fois élaborés, ne restent pas figés, ils évoluent s'il y a matière à les enrichir ou les corriger.

La formation TWI se focalise avant tout sur le « faire » avec des exercices pour les formés (encadrés par un formateur ou à faire eux mêmes). La formation s'organise en plusieurs séances d'entrainement collectif ou tous les « formés » regardent faire le formatteur puis sont interrogés sur tel ou tel « mode opératoire » enseigné. Les « modes opératoires » étant des fiches rassemblant toutes les informations nécessaires à la réalisation d'une tâche spécifique.

TWI donne lieu à la création d'une matrice de compétences qui permet d'évaluer chacun selon des grandes zones de connaissances, d'évaluer l'équipe de la même façon mais aussi de pouvoir, pour un manager, identifier les personnes capables de former les autres.

Il est à noter que le TWI n'a pas spécialement pour but l'amélioration d'un produit ou processus existant, il est avant tout là pour rendre des « non-experts » sur un sujet opérationnels et autonomes le plus vite possible et s'assurer de leur montée en compétences sur le sujet.

- Bilan chez NSN après 4 ans (dans le service chargé des offres mobiles prépayées):
- Equipes plus stables
- Besoin d'experts moins criant
- Nouveaux intégrés sans trop de couacs
- Qualité au rendez-vous (rigueur + vision plus claire de l'objectif)

TWI permet potentiellement de former beaucoup de personnes dans la mesure ou cela fonctionne chez NSN où les effectifs peuvent jusqu'à doubler d'une année à l'autre (service des offres mobiles prépayées).

Question en suspens :

- Adaptable dans des domaines de création comme la conception & le développement logiciel ?

Fait étrange : Lors des questions à la fin, à la question « Combien d'experts sont capables de former les autres depuis la mise en place TWI ? » il a été répondu : « avant il y en avait 8, maintenant on est 3 » ce qui peut paraître étrange.

2. Portraits des personnes citées

Cette anexe rassemble des courtes desciptions des personnes que j'ai pu citer dans ma thèse professionnelle, elles sont divisées en 3 catégories :

- « Gourou » (A contribué grandement au sujet, en est un « théoricien »)
- Professionnel (Utilise de façon professionnelle les outils dont il parle)
- Académique (Enseigne sur le sujet, publie des articles dessus)

Selon l'orientation principale de cette personne vis à vis de son sujet de prédilection. L'orientation indiquée n'est qu'une indication, celle que j'ai jugée dominante au moment d'écrire sur ces personnes qui peuvent combiner plusieurs orientations mais à des niveaux différents.

Emmanuel Chenu

Actuellement : ScrumMaster & coach en développement agile / lean chez Thales Avionics (Valence - France)

Profil : Professionnel

Emmanuel Chenu est un ingénieur diplômé de l'ISEP (Institut Supérieur d'Électronique de Paris), il travaille actuellement chez Thales Avionics dans le domaine du développement de logiciels critiques embarqués, et donne des cours de génie logiciel Agile à des élèves ingénieurs de 5ème année. Il est, de plus, Scrum Master certifié. A l'occasion il prend part à des conférences comme le séminaire Lean/SI de Telecom ParisTech pour venir partager sa propre expérience et les enseignements qu'il en a tirés.

Au fil des projets auxquels il a participé, il a aidé à la mise en place de pratiques lean et agiles dans un monde ou le model industriel « classique » du cycle en V règne sans partage, pour aider à augmenter la performance et la qualité des résultats de son équipe, mais aussi la qualité de leur environnement.

Perfectionniste par nature et jamais vraiment capable, de son propre aveu, de « s'assoir et de juste regarder la machine tourner » une fois venu au bout d'un projet, Emmanuel Chenu cherche en permanence à améliorer la façon de faire les choses au sein de son équipe pour la rendre plus efficace projet après projet.

Jim Womack

Actuellement : Directeur du LEI (Lean Enterprise Institute)

Profil : Académique

James P. Womack a été le directeur de recherche de l'IMVP (International Motor Vehicle Program) au MIT, et est le fondateur du LEI qui est une organisation à but non commercial qui a pour but de diffuser la pensée « lean » dans le monde et d'y réfléchir. Il est devenu connu mondialement en 1990 avec la publication de son livre « The machine that changed the world » qui a introduit le terme « lean production ».

De 1975 à 1991 Jim Womack a mené plusieurs études comparatives sur les pratique de production ayant cours dans le monde dont la plus importante était celle pour l'IMVP. Il a quitté le MIT peu âpres la publication de cette dernière puis a fondé le LEI en 1997.

Martin Fowler

Actuellement : « Head of Research » chez ThoughtWorks

Profil : « Gourou »

« *Martin Fowler est un auteur, conférencier, informaticien et consultant américain dans la conception de logiciels d'entreprise. Ses centres d'intérêts principaux sont la programmation orientée objet, la refactorisation (refactoring), les patrons de conception (design patterns), UML et les méthodes de programmation agile où il est un pionnier et une référence.*

[…]

Il est membre de Agile Alliance et est co-auteur du Manifesto for Agile Software Development. »

(*source : Wikipedia.fr*)

Jim Highsmith

Actuellement : Cadre chez Information Architects Inc.

Profil : « Gourou »

Jim Highsmith est un ingénieur informaticien américain et l'auteur de livres traitant de méthodologies de développement logiciel. Il est le créateur de la méthodologie de gestion de projet « adaptive », décrite dans son ouvrage de 1999 « adaptive software development », et a remporté le prix Jolt en 2000 ainsi que le prix Stevens en 2005. Il est aussi l'auteur de « Agile project management » (2004).

(*source : Wikipedia,org – traduit de l'anglais*)

Régis Médina

Actuellement : Coach Lean

Profil : Professionnel

Régis Médina est un coeach Lean spécialisé dans les projets informatiques. Il les aide à mieux servir leurs clients et à réduire leurs coûts en éliminant les gaspillages.

Il est un des pionniers du développement Agile en France et a co-écrit le premier ouvrage français traitant de l'eXtreme Programming.

(*source : regismedina.com*)

3. Octopode NG

Octopode NG (ONG) est un outil interne développé par les Octos qui permet d'enregistrer et de présenter un aperçu de l'activité du cabinet à un instant t.

Il permet d'enregistrer l'état des missions faites par le cabinet (signée, en cours, terminée …) mais aussi de gérer les CRA (Compte Rendu d'Activité) de tous les Octos.

Octopode NG possède d'autres fonctionnalités encore, permettant de centraliser une bonne partie des informations nécessaires au fonctionnement de la société. L'objectif, à terme, semble d'ailleurs d'avoir un outil « personnalisé » capable de gérer tout ce qui pour l'instant se fait de façon pas toujours satisfaisante (comme des données qui seraient consignées dans des fichiers Excel) pour éviter entre autres d'avoir plusieurs copies pas toutes à jour qui circulent.

ONG est aussi utilisé comme un outil de formation à disposition des Octos qui souhaitent se former soit sur les technologies de la plateforme, soit sur les outils utilisés pour développer (TDD, pair programming, planification en mode agile …)

Avec ONG, j'ai eu l'occasion d'utiliser plusieurs outils issus de l'agile dans un contexte de développement logiciel et de faire mes premières armes sur des concepts comme le TDD, ainsi que d'en réaliser l'efficacité sur des projets comprenant une base de code importante et des effectifs changeants. Pendant mon stage j'ai participé à diverses améliorations de la plateforme sous la forme, d'une part, du co-développant une nouvelle fonctionnalité, mais aussi en prenant en compte les demandes d'amélioration ou de correction de certains services déjà fournis par la plateforme (correction de bugs, ajouts de fonctionnalités supplémentaires mineures …).