Livre Blanc

Adoption des méthodes Agiles dans les projets IT

Plus de maîtrise, plus de valeur, plus tôt et plus vite

www.valtech.fr

Test drive

Agile contract

Planning gam

Retrospective meeting

Velocity

Scrum master

Product backlo

valtech

A propos de Valtech Technology

Valtech Technology est une division du groupe Valtech, société internationale spécialisée dans la formation, la réalisation de projets, le conseil en technologies, en e-business et en management. Valtech Technology est l'un des leaders en France dans l'implémentation des méthodes de développement Agiles (SCRUM en particulier).

Par son double savoir-faire de conseil et de conduite de projets informatiques, Valtech Technology apporte aux entreprises l'expertise et l'expérience des architectures et de l'industrialisation des développements. Spécialisé dans l'adoption des méthodes Agiles, Valtech dispose de plusieurs centres de services, notamment un centre de développement offshore en Inde à Bangalore. Impliqué de bout en bout dans la réalisation des projets, Valtech Technology est l'un des seuls cabinets de conseil et de service à faire concrètement le lien entre le conseil et la réalisation de projets.

Créé en 1993 et côté sur l'Eurolist d'Euronext, le groupe Valtech emploie aujourd'hui plus de 1 100 salariés à travers le monde (Etats-Unis, Europe et Asie) et a réalisé un chiffre d'affaires de 106,7 millions d'euros en 2007.

Remerciements

Je profite de ces quelques lignes pour remercier l'ensemble des consultants de VALTECH TECHNOLOGY qui se sont beaucoup investis durant ces 6 derniers mois sur la publication de ce Livre Blanc Agile. Ces efforts réunis ont permis de mettre en avant notre maîtrise des méthodes Agiles ainsi que nos nombreuses expériences terrain.

Je tiens à remercier particulièrement Elisabeth DUCARRE, Etienne CHARIGNON, Frédéric TREMEAU, Gilles MANTEL, Hélène GRANBOULAN, Nathalie LOPEZ et Stéphane LABATI qui ont su synthétiser leurs retours d'expérience dans des domaines aussi variés que les pratiques Agiles, la gestion des exigences, le test, la conduite de projet, la qualité, le Lean et le CMMI.

Enfin, nos remerciements vont également à nos clients qui, par leur confiance, ont accepté de nous associer à leurs défis, renforçant jour après jour notre expertise et notre capacité à les aider dans l'adoption de l'agilité.

Hubert GILLON, Delivery Manager, VALTECH TECHNOLOGY.

LIVRE BLANC **élaboré par la communauté Agile de Valtech Technology, forte de 30 consultants, seniors et experts, sous la direction de Hubert GILLON, Delivery Manager, responsable de l'activité projet offshore.**

valtech

Sommaire

Table des figures

Table des tableaux

1. Vision

1.1 Pourquoi adopter les méthodes Agiles ?

Dans plus de **80%** *des projets, la première cause d'échec est le fameux cycle en cascade...*

Dans plus de 80% des projets, la première cause d'échec est le cycle en cascade, qui prône un planning fixe et un enchaînement strict des différentes phases, des spécifications jusqu'à la validation logicielle. Cette vision rassurante est bien loin de la réalité des activités d'ingénierie qui ne sauraient se succéder strictement sans qu'aucun changement n'intervienne pour perturber un planning qui n'a de durée de vie que le temps de le prononcer.

Le rôle joué par le client dans un tel cycle intervient principalement au lancement du projet, à chaque jalon majeur, soit à des moments pouvant être espacés de plusieurs mois sur de gros projets, et surtout en fin de projet pour réceptionner et recetter le logiciel. Cet effet tunnel conduit à un logiciel inadapté et souvent de piètre qualité.

Cela est souvent renforcé par le mode contractuel forfaitaire qui :

- **durcit les relations** entre client et fournisseur,
- **rend le passage de témoin long et douloureux** à la fin du projet.

De plus, le passage de relais entre les phases successives dans lesquelles œuvrent des équipes différentes, généralise une relation de type client-fournisseur et n'encourage pas l'empathie ni l'esprit d'équipe, bien au contraire. A chaque transition, une perte en ligne survient qu'il s'agisse de temps, de connaissance, d'information ou de responsabilité. Elle rend inefficace ce cycle jusqu'à nos jours universel, et qui a néanmoins eu le mérite d'exister avant les autres et d'aider les organisations à standardiser leurs activités d'ingénierie de développement. Cette standardisation

a produit un nombre important de documents, rendus souvent pléthoriques sur les projets, détournant ainsi l'objectif premier qui est de réaliser une application répondant aux besoins d'un client, vers la production de documents complets, coûteux à élaborer, et souvent inutiles. Elle a également eu pour conséquence de créer un rôle qualité ayant pour fonction de vérifier l'application de ces standards ou référentiels applicables.

A contrario, **les méthodes Agiles préconisent l'adoption d'un cycle itératif et incrémental** permettant à un projet de s'adapter à son contexte et aux changements inhérents à la nature humaine, aux erreurs ou aux changements ne manquant jamais de survenir tout au long d'un projet.

Ce type de cycle de vie facilite la prise en compte des changements. Il a comme autre avantage de permettre de façon régulière (à chaque fin d'itération) et à une fréquence élevée (2 à 4 semaines) de recueillir le feedback du client et/ou des utilisateurs quant au devenir de l'application en cours de développement, annulant ainsi tout effet tunnel.

Un autre avantage est le fait de fixer des objectifs à court terme. Cela permet de maintenir une pression constante mais supportable sur l'équipe, alors qu'au début d'un cycle en V chacun a l'impression d'avoir suffisamment de temps devant lui et subit finalement une pression énorme à l'approche de la livraison.

De plus, **l'agilité prône la collaboration entre les personnes et l'intégration des équipes.** Elle combat les fameux passages de relais en rassemblant dans un même espace toutes les énergies et la compétence de personnes toutes centrées sur l'application à réaliser. Plus de barrières donc, et des tâches définies par l'équipe au meilleur moment c'est-à-dire quand on en a besoin, plutôt qu'au début du projet.

Enfin, **les méthodes Agiles mettent l'accent sur l'importance de développer le bon produit** en limitant la documentation à produire au strict nécessaire, et en concentrant l'essentiel de l'effort d'une équipe à réaliser une application de qualité, testée souvent, exempte d'anomalie et répondant à coup sûr aux vrais besoins des utilisateurs par le feedback régulier. Point besoin d'une personne dédiée à la qualité dans ce type de fonctionnement car toute l'équipe assure la qualité des activités par le partage, la collaboration, et l'industrialisation des processus centrée sur leur efficacité, notamment grâce à l'automatisation du déploiement, des tests et des revues de code.

> *Le succès des projets Agiles renforce jour après jour l'engouement des DSI et des équipes informatiques pour des pratiques remettant l'application et l'homme au centre du sujet. Un projet n'est-il pas d'abord une aventure humaine vécue par des hommes pour d'autres hommes ?*
> ***Hubert GILLON, Delivery Manager, VALTECH TECHNOLOGY***

1.2 Qui est concerné ?

Rappelons que l'agilité est basée sur les **4 grands principes** suivants :

- **priorité aux personnes et aux interactions** plutôt que sur les processus et les outils,
- **focus sur le logiciel à développer** plutôt que sur sa documentation (on privilégie le code et sa qualité),
- **collaboration avec le client** (feedbacks fréquents) plutôt que négociation et suivi du contrat,
- **réactivité au changement** qui consiste à accueillir tout changement de façon positive plutôt que de se réfugier derrière un planning détaillé initial qui est de fait, forcément faux.

Forts de ces principes, on voit qu'une organisation, un département, une business unit, un projet et même une équipe peuvent adopter l'agilité avec succès.

Mais qu'en est-il d'un projet déjà démarré ou en difficulté ? L'Agilité peut également dans ce cas, soit améliorer les résultats déjà obtenus, soit résoudre bon nombre des causes des difficultés vécues.

Elle va amener les personnes impliquées à :

- mieux collaborer,
- prendre du recul sur l'application en priorisant les actions et en remettant à plat le chiffrage initial,
- donner plus de visibilité aux clients et utilisateurs,
- éliminer l'effet tunnel induit par le cycle en V, en le remplaçant par des itérations courtes et maîtrisées.

> *L'idéal serait que toute l'organisation ait conscience de l'intérêt de fonctionner différemment et qu'elle mette dans sa stratégie l'adoption d'un ou plusieurs de ces principes.*
> ***Hubert GILLON, Delivery Manager, VALTECH TECHNOLOGY***

1.3 Quelles sont les pratiques Agiles les plus répandues ?

Les pratiques Agiles les plus répandues sont :

- l'implication forte du client à travers le rôle de "**Product Owner**",
- l'utilisation d'un "**Product Backlog**" pour établir une vision à terminaison des fonctions d'un produit et des priorités métier associées,
- l'utilisation d'un "**Iteration Backlog**" traduisant les fonctions d'un "product backlog" en tâches priorisées, chiffrées (en heures) et assignées à l'équipe projet,

- le "**Scrum Meeting**" appelé également "standup meeting", réunion quotidienne (maximum 15 min et sans compte rendu) rassemblant une équipe de maximum 10 personnes ayant pour objectif de communiquer ce que chacun a fait le jour d'avant, les problèmes rencontrés et les tâches du jour prévues,
- le "**Retrospective Meeting**" se focalise sur les événements survenus et l'analyse causale des dysfonctionnements, des pertes de productivité et de qualité. Un ou deux axes d'amélioration seront privilégiés de façon consensuelle et se traduiront par des tâches dans le backlog de l'itération suivante,
- le "**Sanity Check**", point à mi parcours d'une itération formalisant l'avancement de la réalisation des fonctions d'un produit afin de savoir si les objectifs de l'itération en termes de livraison pourront être atteints (en qualité et en quantité) et de décider en conséquence soit de réduire les objectifs en enlevant des fonctions de l'"iteration backlog", soit au contraire d'en ajouter,
- l'"**Iteration Planning**" en début d'itération, permet à l'ensemble de l'équipe projet d'identifier, d'estimer, de planifier et de s'affecter les tâches,
- la **Vélocité** qui est la mesure de la capacité du projet (ou d'une équipe) à livrer des fonctions en tenant compte de la capacité de production (taille de l'équipe et durée des itérations). Elle s'appuie souvent sur la taille des fonctions exprimée en points de fonction, points de cas d'utilisation ou poids de la série de Fibonacci (1, 2, 3, 5, 8...),
- l'**Intégration Continue** consiste à compiler, assembler, vérifier et tester l'ensemble du code source dès qu'un nouvel élément est mis à disposition, soit idéalement une à plusieurs fois par jour,
- le "**Test Driven Development**" **(TDD)** consiste à écrire les programmes de tests unitaires avant de programmer les composants eux-mêmes, puis de « refactorer » le code source testé unitairement jusqu'à obtenir un code de qualité,
- le "**Test Driven Requirements**" **(TDR)** permet de spécifier le logiciel par l'exemple – i.e. les exigences logicielles sont détaillées sous forme de cas de test,
- le "**Pair Programming**" consiste à programmer en binôme dans le but d'être plus efficient en termes de conception, de revue de code et de transfert de compétence.

1.4 *Comment adopter les méthodes Agiles ?*

Le fait de ne pas savoir de quelle façon démarrer une migration vers l'agilité freine vraisemblablement de nombreuses entreprises ou DSI, malgré l'intérêt évident qu'elle suscite.

Voici la démarche proposée par Valtech.

- **Veille active :** permet de tester sa motivation et sa détermination à vouloir adopter les méthodes Agiles à partir de séminaires, de salons spécialisés, de lectures techniques ou commerciales...

1 ou plusieurs petits projets de 3 à 6 itérations pour commencer.

- **Etude d'opportunité d'adoption de l'agilité :** consiste à vérifier au sein de son organisation s'il est opportun de lancer une démarche d'adoption en termes de stratégie, de connaissance, de disponibilité, de contraintes et de planning. Le recrutement ou le recours à des personnes expertes de l'agilité améliorera grandement les chances de succès de la démarche d'adoption.
- **Préparation du projet pilote :** former aux méthodes Agiles (2 jours), former le "product owner" (2 jours), établir les métriques de suivi de l'adoption (1 jour) et établir le "product backlog" du projet pilote (3 à 5 jours).
- **Projets pilotes (3 à 6 mois) :** il s'agit de réaliser 1 ou plusieurs petits projets de 3 à 6 itérations avec l'aide de consultants experts de l'agilité, en conseil ou même en tant qu'opérationnel intégré à l'équipe projet (cela est assurément plus efficace).
- **Rétrospective (1 à 2 jours) :** un bilan des projets pilotes s'impose alors, destiné à recueillir les métriques des projets et à en faire la consolidation, à analyser les événements ayant eu un impact négatif en termes de productivité, d'interface entre équipes distantes (nearshore ou offshore), de frais de structure et de suivi, pour ensuite définir un plan d'amélioration.
- **Planification du déploiement (1 jour) :** un plan de déploiement doit être élaboré afin d'orchestrer le déploiement des bonnes pratiques Agiles sur les différents projets. Il reste d'assez haut niveau (quelles pratiques sur quels projets et à quelle échéance) afin de laisser chaque projet libre de gérer ses priorités de développement via son backlog.
- **Déploiement de l'agilité dans l'organisation (1 à 2 ans) :** si possible, ce déploiement doit être guidé par le ROI d'où l'importance d'avoir défini et utilisé des mesures dans les étapes précédentes. A la fin de chaque itération de chaque projet, un "retrospective meeting" permettra de corriger les processus ne donnant pas satisfaction, en véritable outil d'amélioration continue de l'organisation devenue Agile.

1.5 Quels sont les accélérateurs de cette adoption ?

La meilleure façon d'accélérer l'adoption de l'agilité est d'éliminer petit à petit les freins à cette adoption.

Mis à part l'incontournable résistance au changement et les erreurs de mise en œuvre et de déploiement, voici les freins que nous avons répertoriés.

Du point de vue du projet

- Le fait que le projet pilote choisi, ne soit pas un projet assez important mais un petit projet non significatif, qui ne permettra pas de réellement tirer des conclusions transposables aux futurs projets : le projet sera donc de taille significative et durera de 3 et 6 mois.

- Le manque de sponsor réel met en danger la pérennité de la démarche d'adoption qui, par moment, nécessite un soutien inconditionnel de la direction et des décisions rapides : le "product owner" doit réellement être impliqué et moteur sur le projet vu l'importance de son rôle.
- La communication insuffisante vers le management de l'organisation.
- La non mise en place des rétrospectives, facteur incontournable de l'amélioration continue nécessaire au succès de l'adoption de nouveaux processus Agiles.

Du point de vue des conditions de travail

- L'équipe est isolée du reste de l'organisation durant toute l'itération. L'idéal est que le product owner soit intégré à l'équipe : le scrum master est là pour éviter toute perturbation extérieure.
- La disposition des lieux doit favoriser les échanges : pas de cloison ni de box (« cubicle »).
- L'utilisation de tableaux blancs, de wiki, d'outils tels que Valtech Cockpit, RallyDev et TeamConcert, favorisent grandement la collaboration et l'efficacité des processus Agiles.
- L'absence d'outil collaboratif permettant de partager en temps réel les informations projet défavorise la collaboration au sein de l'équipe alors qu'il faut investir dans les actions destinées à créer un esprit d'équipe et à faire travailler les gens entre eux d'une manière naturelle, constructive et efficace.
- Le fait que certaines personnes interviennent à temps partiel sur de multiples projets nuit à la collaboration entre les acteurs. L'équipe doit être dédiée et composée de personnes compétentes et communicantes.

Du point de vue des processus

- Spécifier à l'avance un ensemble d'exigences tue la collaboration entre le "product owner", l'analyste et le développeur tout en rendant moins efficace le transfert de connaissance entre l'analyste et l'équipe de développement.
- Planifier toutes les itérations en détail sur une longue période de temps n'a aucun sens et constitue une perte de temps : seule l'itération courante doit l'être, avec une bonne vision des objectifs pour les itérations suivantes.
- Baser sa vision de l'avancement sur le % de tâches terminées plutôt que sur le reste à faire.
- Ne pas créer de tests fonctionnels automatisables trop tôt dans le cycle de vie.

L'essentiel à retenir

- Les méthodes Agiles préconisent l'adoption d'un cycle itératif et incrémental.
- L'agilité prône la collaboration entre les personnes et l'intégration des équipes.
- Les méthodes Agiles mettent l'accent sur l'importance de développer le bon produit.
- 4 grands principes : • priorité aux personnes • focus sur le produit
  • collaboration avec le client • réactivité au changement
- L'adoption de l'agilité doit être guidée par le ROI.
- L'élimination des freins à l'adoption est le meilleur moyen d'accélérer cette adoption.

2. Les méthodes Agiles "pour les nuls"

2.1 Témoignage d'un coach Agile Valtech

Agile, c'est pratique !

L'agilité peut se vivre au jour le jour sur des projets informatiques. Ce n'est pas un doux rêve inatteignable et dans bien des cas, c'est même assez facile.

> *Les sceptiques vous diront peut-être que l'agilité n'est qu'un concept, un rêve éloigné des réalités concrètes rencontrées au quotidien. N'entendons-nous pas souvent des "oui, mais chez nous ça n'est pas possible !", ou des "oui, mais dans la vraie vie ce n'est pas comme ça !" ? Ah, oui, mais pourtant, j'existe ! »*
> *« Avant de trouver des projets officiellement Agiles, j'ai fait pendant assez longtemps du développement logiciel Agile en sous-marin et, les pratiques que j'ai pu mettre en place ont toujours apporté énormément au projet, si ce n'est pas tout simplement le succès.*
> ***Etienne CHARIGNON, Consultant senior, Scrum master certified, VALTECH TECHNOLOGY***

Les pratiques comme le TDD (Développement piloté par les tests : voir en annexe) ou les tests de recette automatisés sont faciles à mettre en place à condition qu'on veuille bien s'en donner la peine.

Comme dit le proverbe « la qualité est gratuite à condition que l'on veuille bien en payer le prix ». Les pratiques que vous voudrez mettre en place vous feront gagner du temps sur le long terme, mais coûteront quelque chose au début.

Spécifications incrémentales et TDR

... Valeur métier noté entre 1 et 3, ...

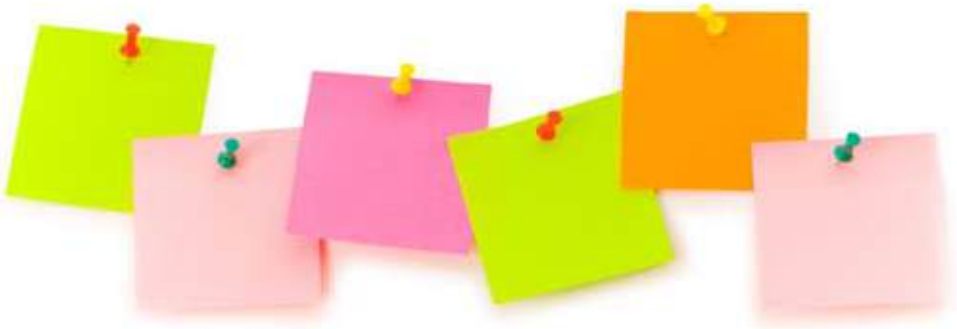
... difficulté technique notée de 1 à 13 suivant la suite de Fibonacci...

Comme on pourrait s'y attendre, les pratiques de développement sont bien plus faciles à mettre en place que, par exemple, celles qui font intervenir le client. Pour le faire, sur un projet sur lequel nous travaillons en ce moment, nous avons planifié une itération zéro de mise en route d'une semaine, durant laquelle nous avons entre autre défini la liste des scénarii d'utilisation mais aussi mis en place Fitnesse, un outil de TDR (Test-Driven Requirement: voir en annexe) basé sur un wiki.

Nous avons conseillé au client de ne plus détailler toutes ses exigences fonctionnelles avant de démarrer les développements. A la place, nous lui avons demandé la liste des scénarii d'utilisation (le "product backlog"). Puis pour chaque scénario, il a attribué une valeur métier noté entre 1 et 3, et nous en avons estimé la difficulté technique notée de 1 à 13 suivant la suite de Fibonacci. Un tel "product backlog" est ensuite plus facile à manipuler que directement une spécification détaillée.

Mais le travail de spécification ne s'arrête pas là. Au début de chaque itération, nous choisissons les scénarii qui doivent être réalisés pendant l'itération et en définissons les critères d'acceptation. C'est de cette manière que nous détaillons les spécifications. **Pour éviter le gaspillage, les détails ne sont pas prévus entièrement à l'avance** (pas de stock), mais seulement au moment où les développeurs sont prêts à les recevoir.

> *Ce flux tendu de spécification constitue la "sève du processus Agile".*
> ***Etienne CHARIGNON, Consultant senior, Scrum master certified, VALTECH TECHNOLOGY***



"War room"

La première chose à faire est de réunir les gens dans un même lieu, dédié au projet. La mise en place d'autres pratiques s'en trouve grandement favorisée :

- pour suivre le projet, on utilise **les post-its sur les murs,**
- pour faciliter le travail en binôme et la propriété collective du code, il est préférable d'avoir un **ensemble d'ordinateurs non affectés individuellement,** mais dédiés à un type de tâche : développement, bureautique... De plus, il est indispensable que les postes de développement soient homogènes.

Dès que vous aurez mis en place tous ces éléments vous aurez votre "war room". Il est évident qu'il est plus difficile de le faire lorsque les développeurs sont éparpillés dans un "open space".

Serveur d'intégration continue

Votre "war room" peut contenir une **machine dédiée à l'intégration continue** car il est assez facile de libérer une machine pour la dédier à cette activité étant donné que les développeurs travaillent chaque fois que nécessaire en binômes.

En quelques minutes, vous installez un logiciel comme Hudson ou CruiseControl. Ce type de logiciel sera capable d'aller lui-même chercher le code source dans le dépôt de votre système de gestion de version (par exemple Subversion ou ClearCase), de le compiler et d'exécuter une série de tests et de mesures de qualité avec des outils tels que CheckStyle ou Cobertura.

Il reste ensuite à **astreindre plusieurs fois par jours les développeurs à poster leur travail sur le dépôt central** (opération appelée "Commit").

2.2 *Résultats obtenus sur des projets réalisés par Valtech*

Valtech est la première société en France, en 2002, à avoir proposé une formation de conduite de projet dans un contexte itératif et incrémental (à l'époque sur la base de Unified Process). Mais très vite, cette démarche utilisée sur les projets de l'époque a été remplacée par la démarche SCRUM, plus Agile et moins contraignante du point de vue de la documentation projet et des livrables.

Cette adoption a été renforcée par l'intervention en 2003, en France mais également en Inde dans notre centre de développement à Bangalore, de Monsieur Craig Larman, auteur de nombreux ouvrages de référence en matière de processus d'ingénierie logiciel et d'agilité.

Il a eu la responsabilité de déployer l'ensemble des principes Agiles sur la totalité des projets, jusqu'à la réorganisation complète des bureaux organisés en « cubicals », transformés en espaces de travail ouverts. **Il a également certifié Scrum Master l'ensemble des chefs de projet,** et a initialisé la mise en place de nouveaux outils tels que Valtech Cockpit, la plateforme collaborative Valtech, utilisée sur les projets encore aujourd'hui.

Cet électrochoc a été réellement salutaire car les résultats sur les projets ont été grandement améliorés sur différents plans.

- Le **Respect des délais :** le time boxing et le fait de procéder par sprints successifs a permis aux équipes projet d'augmenter leur productivité.
- La **Qualité des livraisons :** les anomalies étant corrigées au fur et à mesure, avec une priorité plus importante que les fonctionnalités à livrer, la charge de travail de gestion de ces anomalies a diminué pour laisser place à plus de fonctions implémentées.

- La **Confiance de nos clients :** tout nos clients sont restés fidèles et possèdent des équipes à Bangalore dont le nombre s'accroît chaque année. De plus, de nouveaux clients intéressés par l'offshore et l'agilité nous ont mis en compétition sur des « Proof of concept » avec à la clef de nouveaux projets sur des domaines jusque-là inexplorés par Valtech, comme par exemple celui de testeur de puces électroniques.
- La **Satisfaction de nos collaborateurs :** le fait de collaborer plus étroitement avec des équipes distantes et d'une culture différente a motivé de nombreux consultants, les incitant à aller plus souvent en Inde ou à faire des missions de conseil sur l'adoption de l'agilité dans un contexte onshore mais également offshore.
- Un **Certain confort sur les projets :** souvent le terme projet est synonyme de stress, d'insatisfaction, de pertes financières… Avec la mise en place des pratiques Agiles, la maîtrise des projets s'accroît, le feedback du client et des utilisateurs est réel, la qualité des livraisons et la productivité ont nettement augmenté.

2.3 *Difficultés couramment rencontrées*

L'adoption de l'agilité est une démarche d'entreprise qui peut se limiter à un projet. Cependant de la même manière que le CMMI s'adresse à une organisation et non à un projet unique, l'adoption de l'agilité peut également avoir comme périmètre, l'ensemble des projets d'une organisation.

> *Une des premières difficultés est le « passage à l'acte » qui suppose que le management soit convaincu des bienfaits d'une telle approche pour les clients et pour les projets délivrant les applications ou les produits.*
> ***Hubert GILLON, Delivery Manager, VALTECH TECHNOLOGY***

Autre difficulté souvent rencontrée, celle d'adopter l'état d'esprit Agile, qui suppose de ne pas se contenter de faire son marché dans les pratiques Agiles. En effet, l'adoption de l'agilité est l'affaire de tous et non de certains acteurs qui mettraient en œuvre telle ou telle pratique Agile, pensant que cela limiterait les risques d'échec.

En supposant que tout le monde soit convaincu de l'intérêt d'adopter l'agilité, la mise en œuvre concrète des pratiques telles que planning games, rétrospectives et backlogs produit suppose une **bonne maîtrise des concepts** sous-jacents ainsi qu'une **expérience concrète des projets.**

20 personnes sur une itération de 4 semaines gèrent 1600 tâches de 2 heures en moyenne

- Si l'on prend l'exemple des tâches des backlogs d'itération, il arrive très souvent que les chefs de projet se retrouvent littéralement noyés par le nombre de tâches à gérer. La raison en est que la granularité des tâches gérées est souvent bien trop fine, ce qui induit évidemment un nombre déraisonnable de tâches : une équipe de 20 personnes définissant à chaque itération de 4 semaines des tâches de 2h en charge en moyenne, se traduit par 1600 tâches !
- Le fait de fonctionner en time boxing (durée d'itération fixe quoiqu'il arrive et quelque soit le résultat de l'itération), de livrer des documents partiellement réalisés (contraire à notre culture), ou d'identifier 3 voire 4 améliorations à réaliser sur la prochaine itération à l'issue d'une rétrospective, constituent autant de difficultés et de pièges qu'il faut à tout prix éviter, au risque de discréditer l'agilité à jamais.

Aussi est-t-il très important de faire intervenir un scrum master et des experts des pratiques Agiles pour se faire aider dans la mise en œuvre de l'agilité.

La granularité des tâches aurait oscillé entre 4h et 2j par tâche ce qui aurait conduit à 5 fois moins de tâches à gérer, le projet aurait été livré à l'heure, et le nombre d'améliorations aurait été de 1 par itération au maximum.

2.4 Opportunités de l'offshore

...Ne gérer que des tâches de 4 heures à 2 jours de charge...

... Ne retenir qu'une amélioration par itération au maximum...

Valtech est implanté en Inde depuis 2001 et possède aujourd'hui un centre de développement Agile à Bangalore. Ce centre est mis à contribution sur la très grande majorité des projets.

Son utilisation est un succès aujourd'hui grâce à la mise en place de l'agilité qui a permis :

- aux équipes distantes de mieux collaborer et de mieux communiquer,
- d'améliorer la visibilité sur les projets,
- de partager les estimations en utilisant systématiquement des méthodes telles que Function Point et Use-case Point, et donc d'aboutir à des engagements respectifs clairs,
- de programmer des voyages fréquents pour favoriser les transferts de connaissances et l'apprentissage de la culture du pays distant,
- de converger vers des modèles d'organisation projet et une définition d'acteurs clés tels que les Coordinateurs offshores en charge de faciliter la communication entre les équipes distantes,
- de responsabiliser les équipes offshores par la mise en place localement de planning games, de scrum meetings et de rétrospectives,
- de donner naissance à un véritable esprit d'équipe, renforcé par des activités extra professionnelles permettant de créer les liens très utiles lorsque surviennent certaines difficultés ou certains problèmes.

> *L'utilisation de nouveaux outils collaboratifs tels que Valtech Cockpit, et la mise en place de l'intégration continue entre les équipes distantes ont amélioré la productivité en offshore et la compréhension mutuelle des compétences et des responsabilités de chacun.*
> ***Hubert GILLON, Delivery Manager, VALTECH TECHNOLOGY***

On peut donc affirmer aujourd'hui que l'**agilité constitue un outil réellement efficace pour favoriser l'utilisation d'équipes distantes en nearshore ou en offshore,** capable de réduire la distance entre les hommes au niveau organisationnel, démarche projet et outil, et de créer l'esprit d'équipe si bénéfique à la réalisation des projets informatiques.

L'essentiel à retenir

Pour être Agile :

- Ne pas détailler toutes les exigences fonctionnelles avant de démarrer les développements.
- Pour éviter le gaspillage, les détails ne sont pas prévus entièrement à l'avance.
- Une machine est dédiée à l'intégration continue.
- Il faut astreindre plusieurs fois par jour les développeurs à poster leur travail sur le dépôt central
- Il faut adopter l'état d'esprit Agile.
- La mise en œuvre concrète des pratiques telles que les planning games, rétrospectives et backlogs produit suppose une bonne maîtrise des concepts sous-jacents ainsi qu'une expérience projet concrète.

Il est important de faire intervenir un Scrum master et des experts des pratiques Agiles pour se faire aider dans la mise en œuvre de l'agilité.

L'agilité constitue un outil réellement efficace pour favoriser l'utilisation des équipes distantes en nearshore ou en offshore.

3.Success Stories

3.1 Opportunités d'adoption de l'agilité

Etude d'opportunité

L'étude d'opportunité de l'adoption de l'agilité par une organisation est le moyen idéal pour s'initier en douceur au monde de l'agilité. Basée sur l'écoute et l'échange, cette étude permet d'apporter une solution personnalisée et adaptée aux attentes et besoins du client.

L'objectif de cette étude n'est pas de faire un audit projet, mais plutôt d'identifier les **pertes d'énergies** et les éventuels **effets d'inertie** de l'organisation projet ainsi que d'identifier de **nouvelles pratiques plus efficaces et plus Agiles.**

Les étapes de cette étude sont les suivantes :

- état des lieux sur les pratiques en cours,
- rédaction d'un document de synthèse sur l'existant,
- adoption de pratiques adaptées au contexte du client,
- rédaction et soutenance des pratiques retenues.

Déroulement des étapes

Etat des lieux sur les pratiques en cours

Cette étape se déroule sous forme d'**interviews** et de **workshops** qui visent à établir la **cartographie des pratiques en cours** auprès d'un échantillon de personnes intervenant sur tout le cycle de vie d'un projet (maîtrise d'ouvrage, maîtrise d'œuvre, cellule qualité, formateurs...).

Cette collecte d'informations est organisée par types d'activité projet telles que :

- le recueil du besoin,
- la gestion de projet,
- le transfert de connaissances,
- les spécifications logicielles (dossier d'analyse),
- la conception et l'implémentation,
- le test logiciel,
- la qualité logicielle,
- le déploiement et la mise en production.

Rédaction d'un document de synthèse sur l'existant

Le document de synthèse des interviews a pour but de mettre en évidence de façon objective et anonyme les différentes remontées d'informations effectuées lors de l'étape précédente.

Il fournit :

- le périmètre technique, fonctionnel et organisationnel du projet ou service candidat à l'agilité,
- les rôles et responsabilités de chaque personne interviewée au sein de l'organisation projet ou service,
- l'état des lieux de chacune des activités précédemment citées,
- les accélérateurs éventuels à l'adoption de l'agilité,
- les freins éventuels,
- les incontournables manquants (pratiques indispensables et pourtant absentes).

Ce document sert de base à un nouveau workshop d'échange destiné à l'**analyse détaillée et approfondie de ces freins et incontournables manquants.** A ce stade, d'autres workshops plus ciblés sont réalisés afin d'**identifier les pratiques les plus utiles,** en s'appuyant sur un catalogue de pratiques Agiles.

Adoption de nouvelles pratiques

Les pratiques identifiées précédemment sont synthétisées dans un document où :

- les risques et incontournables manquants sont rappelés avec les pratiques Agiles associées,
- les conditions d'entrée pour la mise en place de toutes les pratiques sont identifiées,
- la description et le mode opératoire de chaque pratique sont détaillés,
- les éventuels artefacts produits par chaque pratique sont identifiés,
- les conséquences et impacts sur l'organisation et les processus actuels sont décrits,
- les attendus opérationnels de la pratique sont décrits.

> *Il est important de présenter les pratiques pressenties à l'ensemble des acteurs et de les décrire. Mais il faut également discuter des impacts possibles sur l'organisation afin qu'un sous-ensemble de pratiques puisse être retenu par le client.*
>
> ***Frédéric TREMEAU, Chef de projet senior, VALTECH TECHNOLOGY***

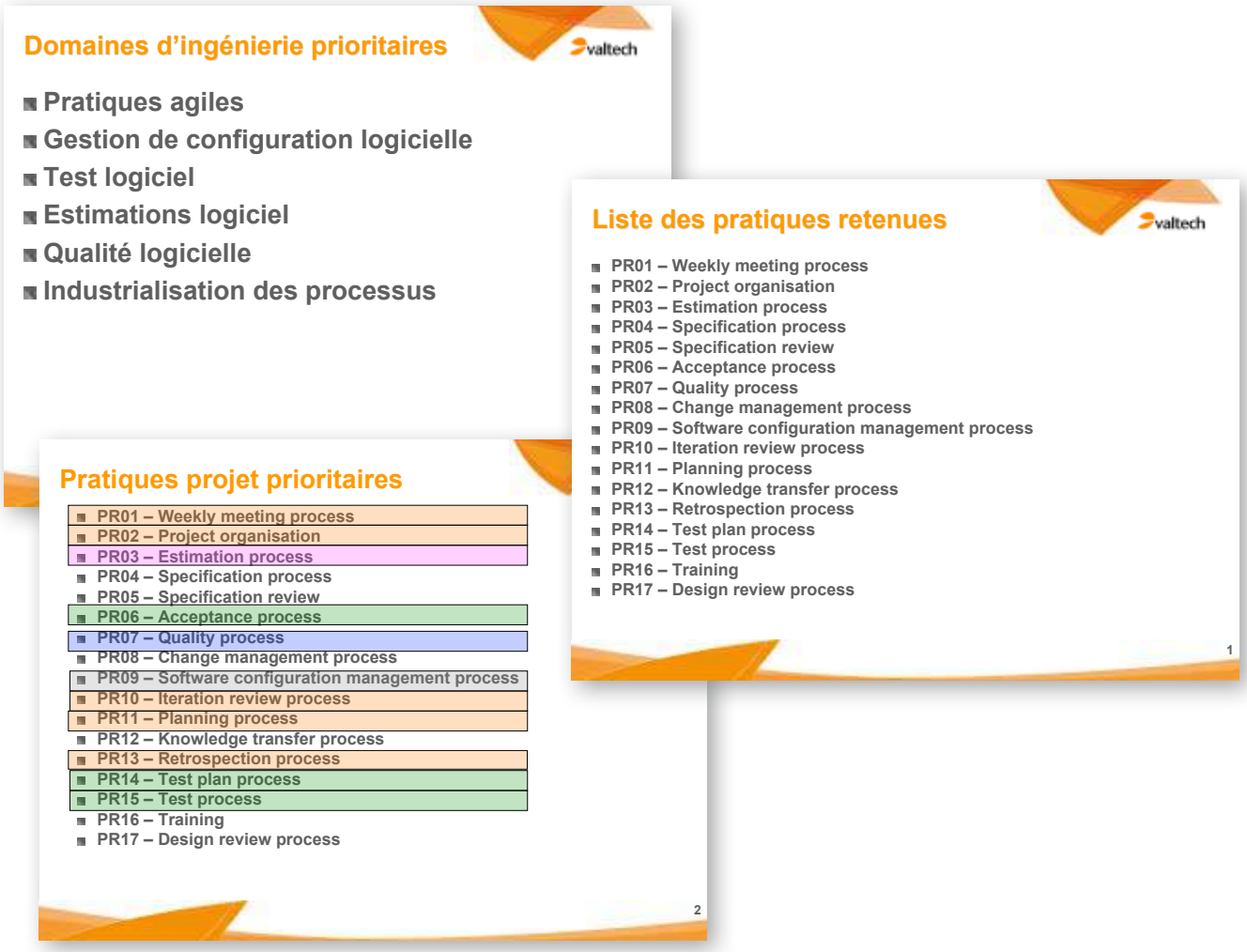


Figure 1 ***Exemple de pratiques d'ingénierie et de pilotage de projet (Source : Valtech Technology).***

Rédaction d'un document de synthèse sur les pratiques retenues

Valtech produit le document final de l'étude synthétisant les attendus du client, la démarche de l'étude et les pratiques retenues. Un plan d'action et une proposition de planning finalisent le document qui est présenté à l'ensemble des acteurs impliqués dans l'étude.

A l'issue de la soutenance, Valtech propose au client, soit :
- une prestation de production d'artefacts issus des pratiques Agiles retenues,
- une mission d'accompagnement dans la mise en œuvre des nouvelles pratiques Agiles sur le projet,
- une formation à l'agilité.

L'essentiel à retenir
- L'objectif de l'étude d'opportunité est d'identifier les pertes d'énergies et les éventuels effets d'inertie de l'organisation projet ou service ainsi que d'identifier de nouvelles pratiques plus efficaces et plus Agiles.
- L'étude d'opportunité est basée sur des interviews et des workshops qui permettent de définir la cartographie des pratiques en cours, pour ensuite identifier les freins à l'adoption de l'agilité, les incontournables manquants et les pratiques Agiles les plus utiles.
- Un plan d'action et une proposition de planning finalisent l'étude d'opportunité.

3.2 TDR au cœur des spécifications

Le Test-Driven Requirement, ou spécification dirigée par les tests, propose :
- de **centrer la description et la rédaction des besoins utilisateurs sur des exemples** qui constitueront autant de futurs cas de tests de recette,
- de **centrer la collaboration entre les équipes du projet sur la compréhension et l'enrichissement de ces exemples.**

Le contexte documentaire et collaboratif

Pour un des leaders de services en ressources humaines, Valtech a mis en place la pratique TDR pour un logiciel de gestion qui avait été créé en 1 an avec UML. La stratégie de développement avait été axée sur l'efficacité à court terme plutôt que sur la maintenabilité. Les activités projet étaient confiées à des équipes distinctes (développement / homologation / analyse-gestion de projet / MOA).

Après 6 mois d'exploitation, le client avait constaté les remarques suivantes.

- ++ Les fonctions majeures du logiciel étaient opérationnelles.
- − Chaque équipe disposait de ses propres documents, relatifs à son activité, sans les partager avec les autres équipes.
- − Les évolutions apportées après la mise en production étaient réalisées sans être spécifiées, engendrant un manque de visibilité sur le contenu fonctionnel réel du logiciel et rendant difficile l'intégration de nouveaux besoin.
- − Le périmètre de test déduit des spécifications était incohérent avec les fonctionnalités déjà implémentées.
- − L'homologation était de moins en moins souvent réalisée du fait de la difficulté à produire les cas de tests conduisant à une baisse de qualité.
- − La grande autonomie de chaque équipe se faisait au détriment de la communication.

C'est dans un tel contexte que Valtech a mis en place la pratique TDR.

Les enjeux du client

Compte tenu du constat réalisé pour parvenir à maîtriser les évolutions logicielles, il a été décidé de modifier les pratiques de spécifications.

Valtech a amené le client à privilégier la description concrète plutôt que la modélisation en incluant des exemples tels que ceux présentés ci-après.

Exemple de spécification TDR

Contexte de l'évolution

Il s'agit de modifier le cycle de vie d'une affaire afin de mettre à jour son état lorsqu'une convention est créée, sans être validée.

Jusqu'à présent, c'est seulement lorsque la convention est validée que l'état de l'affaire est mis à jour.

N.B. : une affaire est une opportunité commerciale détectée pour un compte donné, mais n'aboutissant pas immédiatement à la signature d'une convention avec ce compte. Lorsque l'opportunité commerciale se concrétise, l'affaire est transformée en convention. La convention est définitive à partir du moment où elle est validée. Cette évolution met donc en évidence à la fois le cycle de vie des affaires, celui des conventions et celui des comptes.

Les données initiales

Les "Comptes" et "Affaires" suivants ont été créés :

COMPTES				
Nom du Compte	**Matricule Responsable**	**Entité Responsable**	**État du Compte**	**Identifiant du Compte**
BONJOUR	0001	E1	1 (prospect sur affaire)	COMPT 01

AFFAIRES				
Nom de l'Affaire	**Matricule Responsable**	**Identifiant du Compte**	**État du Compte**	**Commentaire**
Affaire A	0002	COMPT 01	1 (ouverte)	Null

Tableau 1 ***TDR - Données initiales (Source : Valtech Technology).***

L'affaire est créée dans l'état par défaut « Ouverte » tandis que le compte est créé dans l'état « Prospect sur affaire ».

Transformer une affaire en convention

L'action consiste pour le responsable (matricule 0002 lié à l'affaire A) à transformer l'affaire A en convention.

AFFAIRES - RÉSULTAT				
Nom de l'Affaire	**Matricule Responsable**	**Entité Responsable**	**État du Compte**	**Identifiant du Compte**
Affaire A	0002	E1	2 (fermée)	Transformée en convention

Tableau 2 ***TDR - Affaire versus Convention (Source : Valtech Technology).***

L'affaire est fermée car transformée en convention. Elle ne peut plus être utilisée pour créer une convention.

COMPTES - RÉSULTAT				
Nom du Compte	**Matricule Responsable**	**Entité Responsable**	**État du Compte**	**Identifiant du Compte**
BONJOUR	0001	E1	1 (prospect sur affaire)	COMPT 01

CONVENTIONS - RÉSULTAT			
Nom de la Convention	**Matricule Responsable**	**Etat du Contrat**	**Nom de l'Affaire**
Convention issue de l'Affaire A	0002	4 (brouillon)	Affaire A

Tableau 2 *TDR - Affaire versus Convention (Source : Valtech Technology).*

La convention n'est pas encore validée d'où sa création dans l'état « brouillon ». L'état du compte reste inchangé.

Valider une convention

L'action consiste pour le responsable (matricule 0002) à valider la convention.

COMPTES - RÉSULTAT				
Nom du Compte	**Matricule Responsable**	**Entité Responsable**	**État du Compte**	**Identifiant du Compte**
BONJOUR	0001	E1	2 (client)	COMPT 01

CONVENTIONS - RÉSULTAT			
Nom de la Convention	**Matricule Responsable**	**Etat du Contrat**	**Nom de l'Affaire**
Convention issue de l'Affaire A	0002	1 (en cours)	Affaire A

Tableau 3 *TDR - Validation de convention (Source : Valtech Technology).*

La validation de la convention entraîne donc :
- le changement d'état du compte qui passe de « prospect sur affaire » à « client »,
- le passage de la convention de l'état « brouillon » à « en cours ».

Une double « nouveauté » : collaboration et format des exigences

Dorénavant, **la description des fonctionnalités est affinée de façon collaborative tout au long du processus de développement :**

- initiation par la MOA,
- enrichissement par les analystes,
- mise à jour au fil des remarques des équipes de développement et d'homologation.

Cette description s'appuie sur des cas d'exemples qui sont :

- les cas de tests des scénarii standards par la MOA,
- les cas de tests des scénarii d'exception (sans viser l'exhaustivité mais plutôt la vraisemblance) par l'homologation.

Les tableaux utilisés pas à pas dans notre exemple, suite à différentes actions opérateur, permettent de **visualiser concrètement les changements d'état successifs. Ils facilitent à la fois la compréhension du fonctionnel mais également l'identification des scénarii de test les plus pertinents.**

Finalement, l'ensemble des équipes projet a adhéré à la nouvelle approche TDR. Cette adhésion a été favorisée par le fait que les cas de tests décrits sous forme de tableaux étaient lisibles par des non-informaticiens.

Un retour d'expérience riche d'enseignements

Pour en faciliter l'appropriation par les équipes, la méthode TDR a été introduite progressivement sans la nommer, en incluant des exemples dans les spécifications existantes.

La démarche TDR est à l'origine des avancées suivantes.

1. L'équipe de développement a pris le réflexe d'enrichir les exemples fournis dans la spécification TDR. Ces exemples ont été utilisés en intégration continue et en homologation.
2. Une grande partie des ambiguïtés dans la description des règles métier est désormais levée avant le début des développements.
3. Le besoin se stabilise de plus en plus tôt dans le cycle de création d'une fonctionnalité.
4. Les erreurs de description ou d'implémentation sont détectées plus tôt et sont donc plus faciles à résoudre.
5. L'homologation se déroule désormais normalement.

Prochaine étape possible

> *La formalisation des spécifications selon l'approche TDR permet de produire facilement des spécifications exécutables. Couplé à un outil tel que FIT, Fitnesse ou GreenPepper, chaque exemple devient ainsi un cas de test automatique.*
>
> ***Hélène GRANBOULAN, Analyste senior, VALTECH TECHNOLOGY***

L'essentiel à retenir

- Le Test-Driven Requirement (TDR) propose de :
  - centrer la description et la rédaction des besoins utilisateurs sur des exemples,
  - centrer la collaboration entre les équipes du projet sur la compréhension et l'enrichissement de ces exemples.

- Dans une démarche TDR, il faut privilégier la description concrète plutôt que la modélisation.

- La description des fonctionnalités est affinée de façon collaborative tout au long du processus de développement.

- Les tableaux utilisés pas à pas, suite à différentes actions opérateur, permettent de visualiser concrètement les changements d'état successifs. Ils facilitent à la fois la compréhension du fonctionnel mais également l'identification des scénarii de tests les plus pertinents.

3.3 Retour d'expérience sur la gestion de projet

Contexte

Développement d'une application de gestion pour le suivi et la traçabilité des processus de fabrication pour un industriel français de l'aviation.

Mode de développement	**Duoshore avec une équipe de 5 analystes "'Onsite" sur site client et 25 développeurs 'offshores' dans notre centre de développement de Bangalore (Inde).**
Taille du projet	**9000 homme.jours**
Durée du projet	**2 ans**
Outils utilisés	**WSAD, Wiki Confluence, Jira**

Tableau 4 ***Contexte projet.***

Enjeux client

Les enjeux pour cette nouvelle application sont :

- d'offrir un **outil souple, robuste et facilement déployable,**
- de faciliter **le travail des utilisateurs** par une interface intuitive et simple d'utilisation,
- d'apporter une solution permettant de mieux maîtriser la traçabilité des procédés de fabrication,
- d'accroître **l'inter-opérabilité** du système avec les autres applications de gestion.

... 2 équipes Scrum ...

... 1 Product Backlog ...

... 1 outil collaboratif de type wiki ...

... 22 itérations de 4 semaines ...

Pratiques mises en œuvre

Les pratiques mises en œuvre sont :

- utilisation de la méthode d'organisation et de suivi Scrum sur les parties France et Inde **(2 équipes Scrum),**
- mise en place d'**1 Product Backlog,**
- utilisation d' **1 outil collaboratif de type wiki** pour communiquer entre les équipes de développement distantes et le client,
- mise en place d'un mode de livraison Inde / France basé sur de l'intégration continue (cf. Détail d'une journée). Les différents points de synchronisation France / Inde et MOA / MOE y sont identifiés ci-après.

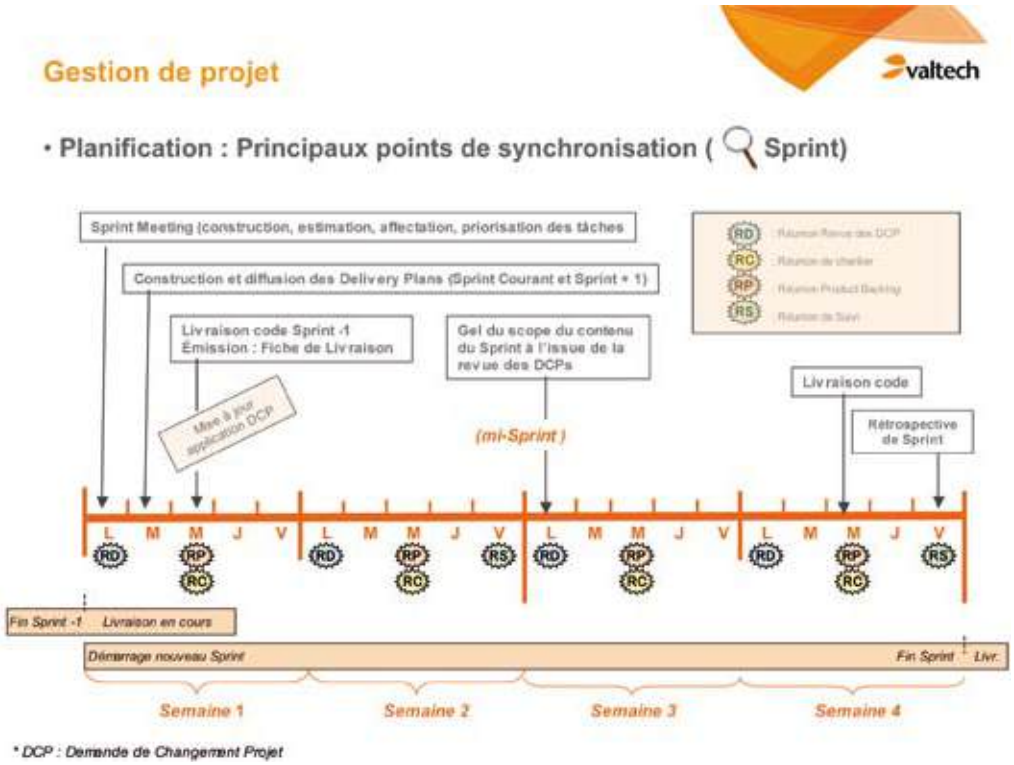


Figure 2 ***Planning détaillé d'une itération (Source Valtech Technology).***

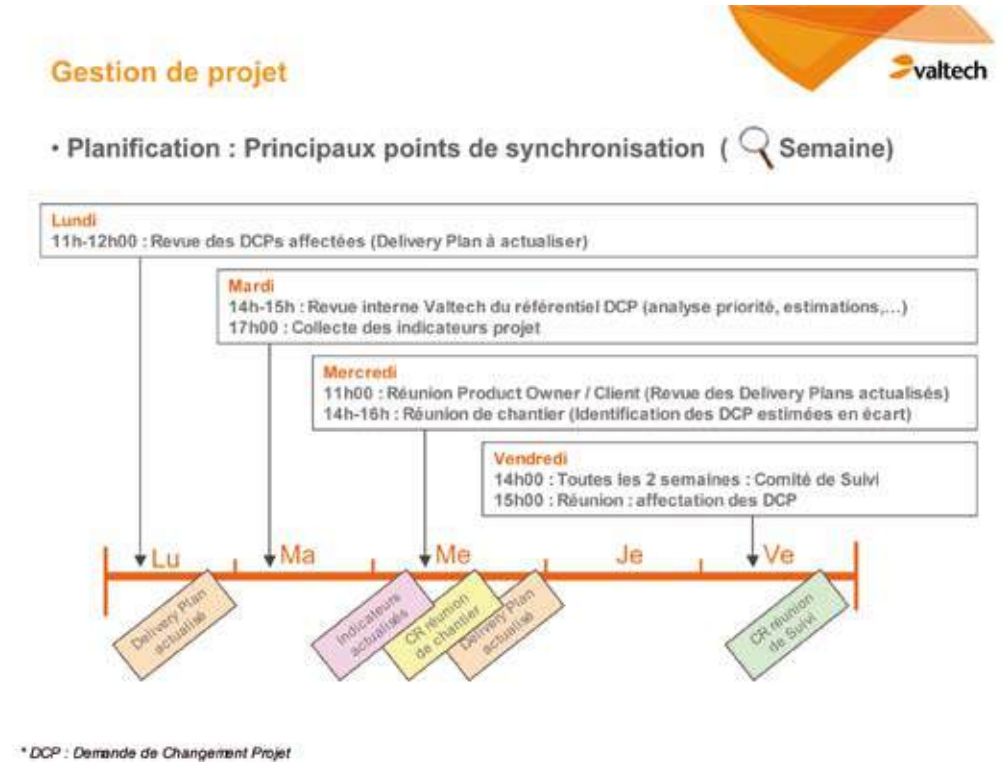


Figure 3 ***Planning détaillé d'une semaine (Source Valtech Technology).***

• Planification : Principaux points de synchronisation (Journée)

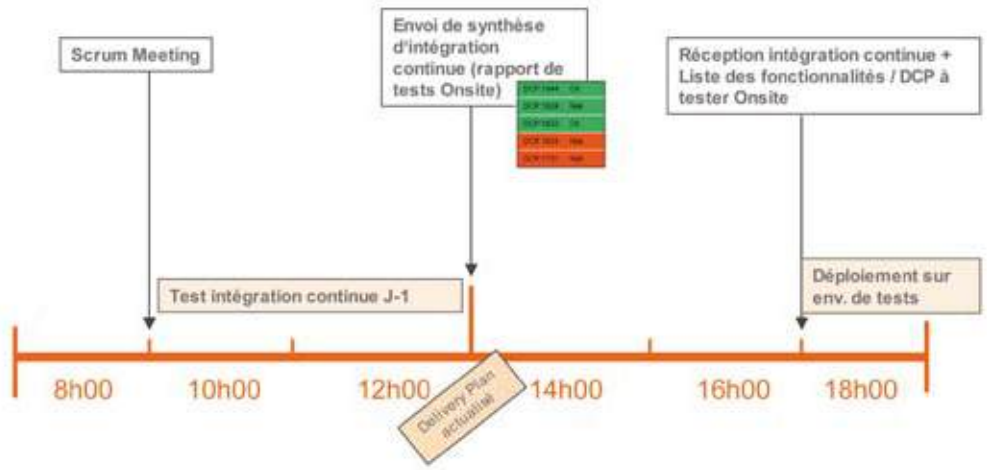


Figure 4 ***Planning détaillé d'un jour (Source Valtech Technology).***

Principe de répartition des responsabilités de la maîtrise d'œuvre :

Onsite (France) :

- relation avec le client,
- recueil des besoins,
- formalisation de documents d'analyses,
- suivi de la validation de ces documents,
- transfert de connaissance avec les équipes offshores,
- support fonctionnel aux équipes offshores,
- mise en place et contrôle des directives d'architecture du projet,
- contrôle des flux de traduction Français → Anglais (documents d'analyse) et Anglais → Français (document de conception),
- développement de fonctionnalités non délocalisables,
- vérifications des scénarii de test,
- coordination projet (car porteur de la responsabilité vis à vis du client).

Offsite (Inde) :

- formalisation des documents de conception,
- travaux de développement,
- élaboration des scénarii de tests,
- automatisation des tests,
- exécution des tests manuels/automatiques,
- exécution des contrôles qualité (PMD, RSA et FindBugs).

Difficultés rencontrées

Les difficultés rencontrées sont :

- la contrainte d'entrée sur le site et la planification longtemps à l'avance ne permettent pas de faire intervenir des ressources offshores sur le site client,
- l'obligation de planifier les réunions et les workshops longtemps à l'avance,
- la clôture de la documentation Word pour le client : en contradiction avec l'approche Wiki qui privilégie l'accès en ligne pour tous,
- la barrière de la langue (anglais) pour la maîtrise d'ouvrage,
- le souhait du client de raccourcir la prise en compte des demandes de changement d'une itération à l'autre,
- le cycle initial de validation des livrables documentaires était trop lourd et pas du tout adapté à l'approche itérative,
- les équipes Onsite et MOA ne communiquent que par mails.

Solutions apportées

Les solutions apportées sont :

- face à l'impossibilité de traiter les évolutions : **mise en place d'un volant de jours** pour traiter les évolutions : système d'enveloppe,
- pour réduire le nombre d'anomalies : **mise en place d'un circuit d'intégration continue** entre les développements réalisés en Inde et la plate forme d'intégration sur le site du client,
- l'accélération de la prise en compte des demandes client : **identification d'une provision de charges** pour traiter les demandes de changement en cours d'itération (jusqu'à la mi-itération),
- la réduction du backlog d'anomalies : **constitution d'une provision de charges** pour laisser le temps à l'équipe en Inde de corriger ses anomalies (interne / client) tout en produisant du fonctionnel,
- **le rapprochement physique des équipes Onsite avec la maîtrise d'ouvrage.**

Bénéfices obtenus

Les bénéfices obtenus sont :

- le volant de jours : **fin des tensions** concernant la qualification des anomalies / évolutions,
- **la confiance accrue du client** en la qualité des développements,
- **la visibilité totale sur le contenu du produit et des itérations** qui permet de planifier à l'avance les avenants contractuels pour traiter les nouvelles fonctionnalités majeures,
- **le respect des engagements** vis-à-vis des autres équipes impliquées dans le processus d'ingénierie (autres équipes de développement, intégration, qualification, tests de performances, ...),

- **réduction du nombre d'anomalies** grâce à une meilleure compréhension du fonctionnel par les équipes indiennes et à la découverte au plus tôt des anomalies (intégration continue),
- **nouvelles prises de commandes.**

L'essentiel à retenir

Les pratiques Agiles peuvent être appliquées dans un environnement non Agile même si celui-ci a de fortes contraintes. Il faut adapter ces méthodes au contexte du client en les adoptant de manière graduelle : une éducation préalable montrant les bénéfices attendus et une mise en évidence régulière des gains obtenus sont impératives.

Les pratiques Agiles choisies et mises en place avec justesse permettent de préserver un des facteurs clés du succès du projet : la bonne collaboration entre les équipes du fournisseur et du client. N'oublions pas que cette collaboration entre équipes est au centre de la méthode Agile Scrum.

3.4 *Pratique de recueil de besoin - le Product Backlog*

Le manque de visibilité sur le contenu final probable d'un logiciel est souvent préjudiciable au client mais également aux équipes projet.

En effet, cette description permet entre autres :
- d'avoir une vision commune sur l'ensemble des fonctions ou cas d'utilisation définissant le scope du logiciel à développer,
- de comprendre l'intérêt et les enjeux des développements pour les utilisateurs (appelés également acteurs),
- d'estimer l'avancement du projet sur la base des fonctions ou cas d'utilisation livrés au client,
- de réaliser facilement des macro-estimations en utilisant par exemple la méthode des use case points,
- de préparer l'identification des tâches projet en les organisant autour des fonctions ou des cas d'utilisation.

Selon Valtech, une bonne manière de répondre à ce besoin de visibilité, est d'utiliser un artéfact fondamental à la méthode Agile Scrum qui s'appelle le Product Backlog.

Nous l'avons appliqué une fois de plus avec succès chez un de nos clients dans le domaine de l'assurance, pour formaliser de façon synthétique l'ensemble des fonctions attendues par les courtiers, et pour estimer la charge de réalisation de l'application.

La figure suivante présente un extrait du tableau obtenu après deux jours de workshop fonctionnel avec la maîtrise d'ouvrage et la maîtrise d'œuvre, ainsi que deux courtiers en assurance, futurs utilisateurs de l'application.

CAPABILITIES				FEATURES	
Capa.	Functional capabilities	Lot	Feat.	Features	Actors
	18				
C1	Quotation management (Lot 1)	1	F1-1	Quotation creation from scratch	User
		1	F1-2	Quotation creation from an existing version of quotation (same year)	User
		1	F1-3	Quotation creation from an existing version of quotation (previous year)	User
		1	F1-4	Quotation creation from a Petrus Program	User
C2	Date import (Lot 2)	2	F2-1	Select fields to import from loss files	User
		2	F2-2	Import Loss with developments - vertical	User
		2	F2-3	Import Loss with developments - horizontal	User
		2	F2-4	Import Loss files without developments - vertical	User
		2	F2-5	Import the LDFs	User
		2	F2-6	Import the indexes from the index database - European	User
		2	F2-7	Import the indexes from the index database - US	User

Figure 5 ***Exemple de Product Backlog (Source : Valtech Technology).***

On distingue donc les capacités de la future application avec pour chacune, une liste de fonctions appelées également **stories.** Chaque fonction est identifiée de façon unique pour pouvoir ensuite être facilement référencée.
De plus, chaque fonction se voit attribuer un **type d'utilisateur.**

A la fin de cet exercice, les capacités sont attribuées à des lots (appelés également releases).

Une fois ce premier travail réalisé, une priorité de développement a été estimée et affectée à chaque fonction. Elle est calculée à partir des estimations de complexité et de valeur métier (classées en High, Medium and Low). Cette démarche permet de limiter les risques en traitant en priorité les fonctions les plus complexes, et en majorant le ROI en livrant d'abord les fonctions apportant le plus de valeur aux utilisateurs. L'extrait du tableau suivant donne une idée du résultat.

FEATURES			PRIORITY			
Feat.	Features	Actors	Complexity	Business Value	Priority	UCP
	158			158		
F1-1	Quotation creation from scratch	User	M	H	Medium	10
F1-2	Quotation creation from an existing version of quotation (same year)	User				
F1-3	Quotation creation from an existing version of quotation (previous year)	User				
F1-4	Quotation creation from a Petrus Program	User	M	H	Medium	10
F2-1	Select fields to import from loss files	User	H	L	Medium	15
F2-2	Import Loss with developments - vertical	User				
F2-3	Import Loss with developments - horizontal	User				
F2-4	Import Loss files without developments - vertical	User				

Figure 6 ***Gestion des priorités et de la complexité dans le Product Backlog (Source : Valtech Technology).***

Un nombre de UCP (use-case point) a été attribué aux cas d'utilisation en fonction de leur complexité, en utilisant la méthode des use-case points, ceci afin de servir de base à une estimation globale du projet.

> *Finalement, nous avons été capables en moins de 5 jours, de décrire les besoins client sous la forme d'un product backlog, d'identifier l'ensemble des acteurs au sens UML, de définir les priorités d'implémentation, d'estimer la complexité des cas d'utilisation et de chiffrer le projet dont la taille représentait 3.000 homme.jours.*
>
> ***Frédéric TREMEAU, Chef de projet senior, VALTECH TECHNOLOGY***

L'essentiel à retenir

Le manque de visibilité sur le contenu final probable d'un logiciel est souvent préjudiciable au client mais également aux équipes projet.
L'utilisation product backlog peut se révéler très efficace à condition d'en maîtriser les concepts et de disposer des personnes compétentes et habilitées à prendre les décisions impactant le futur projet.

3.5 Pratique de gestion de projet - l'Iteration Backlog

L'Iteration Backlog a pour vocation de gérer à court terme l'ensemble des tâches identifiées, estimées et affectées à l'équipe projet sur chaque itération, afin de visualiser jour après jour l'avancement des travaux et d'atteindre les objectifs fixés pour l'itération. Les tâches en constituent donc la matière première. Sur les projets Valtech, les tâches sont associées à des fonctions ou à des scénarii de cas d'utilisation.

En début d'itération :

La granularité des tâches doit être entre 4 et 6 heures

- Les tâches sont identifiées pour prendre en compte de nouvelles priorités de livraison et pour minimiser le nombre d'itérations nécessaires à la livraison d'une fonction ou d'un cas d'utilisation. **La granularité des tâches doit être entre 4 et 16 heures.**
- Chaque tâche se voit affecter les informations suivantes :

- Un **Identifiant** unique (pour la traçabilité avec les fonctions ou cas d'utilisation),
- Un **Nom** explicite non générique mais spécifique au contexte et au sujet traité,
- Un **Responsable** en charge de l'exécuter,
- Une **Priorité** (basse, moyenne ou haute) qui permet au responsable de démarrer par les tâches les plus prioritaires,
- Des **Date de début et Date de fin estimées** permettant de lisser dans le temps la charge estimée pour la tâche et de déduire la cohérence de l'affectation des tâches à l'équipe en termes de capacité de production,
- Un **Effort estimé** en heures par le responsable de la tâche lui-même, validé par l'équipe lors du planning d'itération (exercice collégial de planification d'itération),
- Un **Poids estimé** représentant une unité de taille logiciel, utilisé pour la mesure de la vélocité de l'équipe (taille du logiciel qu'il est possible de livrer par unité de charge (h.j.)). Ce poids est estimé en utilisant la suite de Fibonacci (1, 2, 3, 5, 8, 13…). Nous nous limitons à l'intervalle 1-13 car par expérience, une valeur supérieure est souvent synonyme d'incertitude.

Chaque jour :

- Chaque membre de l'équipe projet renseigne son consommé et son reste à faire pour les tâches dont il a la charge. Une alternative consiste à nommer chaque semaine et à tour de rôle, un time tracker qui relève ces métriques pour l'ensemble de l'équipe.
- Un graphe **Burndown chart** est mis automatiquement à jour, imprimé et affiché dans l'espace de travail de l'équipe accessible également à distance via un wiki (y compris par le client). Il présente l'effort restant à accomplir en heures, pour finir les tâches allouées à l'itération, les tâches terminées et la courbe idéale de reste à faire.

Ce graphe prend la forme suivante :

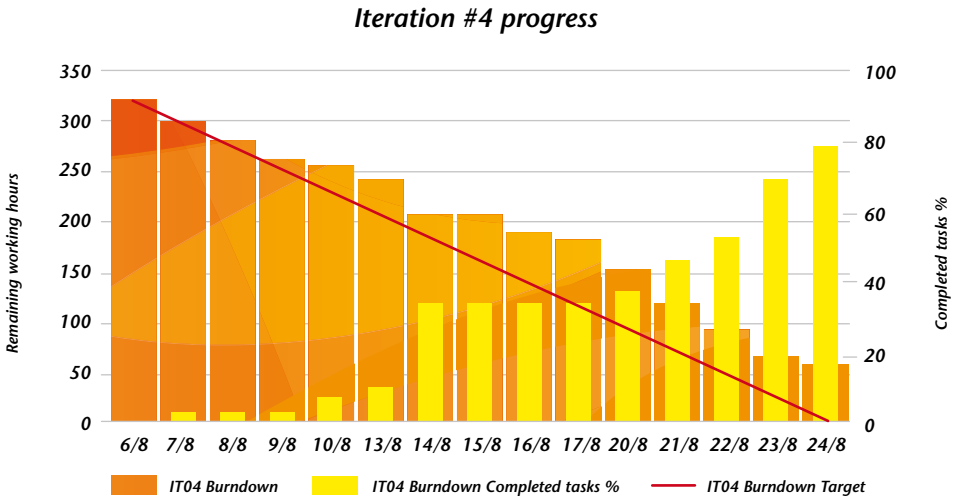


Figure 7 ***Exemple de « burndown chart » d'un Iteration Backlog (Source Valtech Technology).***

En fin d'itération, les métriques suivantes sont relevées et communiquées à l'équipe ainsi qu'au client :

- le **Schedule overrun** représentant le % de retard de livraison de l'itération. Il vaut normalement 0% si le time boxing est respecté,
- l'**Effort overrun** représentant le dépassement d'effort réalisé dans l'itération. Il vaut 0% si l'équipe n'a pas changé et que l'itération a duré le temps initialement prévu,
- la **Taille moyenne** de l'équipe calculée par le consommé sur le nombre de jours travaillés,
- l'**Avancement en % d'effort,** calculé par la formule universelle : Consommé/(Consommé + Reste à Faire),
- le **% du périmètre de l'itération réalisé.** Il est calculé par le rapport entre la taille du logiciel qui devait être livré (poids Fibonacci) et la taille livrée réellement,
- la **Vélocité** de l'itération, c'est à dire le rapport entre la taille du logiciel livré et la charge consommée associée. Elle mesure la capacité de l'équipe à livrer un certain nombre de fonctions. Plus elle est élevée et moins le projet durera longtemps.

> *Le Burndown chart est un outil très puissant pour maîtriser l'avancement des travaux réalisés par une équipe sur une période courte dont les objectifs ont été exprimés en tâches à réaliser.*
> ***Frédéric TREMEAU, Chef de projet senior, VALTECH TECHNOLOGY***

L'essentiel à retenir

L'Iteration Backlog a pour vocation de gérer à court terme l'ensemble des tâches identifiées, estimées et affectées à l'équipe projet sur chaque itération.
La mise en place de mesure pertinentes permet, jour après jour, de suivre l'avancement réel et de prendre les mesures correctrices, correctives et préventives appropriées. Attention toute fois à ne pas tomber dans l'excès de tracking avec des micro-tâches inférieures à 4 heures à gérer.

3.6 Retour d'expérience sur l'automatisation des tests

Contexte

L'agilité impose de nombreuses livraisons intermédiaires. Chacune de ces livraisons doit être intégralement testée ; dans le cas contraire, la perte de contrôle sur la qualité croît de manière exponentielle à chaque itération.

Un département de la DSI d'une grande banque de finance souhaite mettre en place l'agilité au sein de ses équipes de développement. Le département a la responsabilité du développement et de la maintenance d'un ensemble d'applications avec des technologies hétérogènes. Toutes les applications sont mises à jour en même temps en production à une fréquence trimestrielle. Chaque déploiement en production est précédé d'une campagne de tests utilisateurs de 2 à 3 semaines visant à s'assurer de l'absence de régression dans l'ensemble des applications.

Enjeux client

La durée des campagnes de tests de non-régression risque de masquer l'effet positif du déploiement des méthodes Agiles. Il est donc impératif de raccourcir la durée des campagnes et d'en accélérer la fréquence.

Pratiques mise en œuvre

Les pratiques mises en œuvre sont :

- automatisation des tests utilisateurs de non-régression, c'est-à-dire des tests fonctionnels sollicitant l'interface graphique,

... échec de tests unitaires en moins 5 min, échec de tests fonctionnels en moins d'une heure.

- mise en place de tests continus dans le processus d'intégration continue sur tous les niveaux de tests, depuis les tests unitaires jusqu'aux tests utilisateurs. Les niveaux de tests sont déclenchés à différentes étapes du cycle de vie du build afin de conserver un feedback optimal pour les équipes de développement : échec d'intégration détecté en moins de 2 min, échec de tests unitaires en moins de 5 min, échec de tests fonctionnels en moins d'une heure.

Difficultés rencontrées

- Les tests utilisateurs de non-régression, ceux qui sollicitent l'interface graphique, sont généralement coûteux à automatiser et à maintenir. La moindre modification d'une interface peut conduire à l'échec d'un script automatisé même si les services sous-jacents n'ont pas été modifiés :
- les équipes de développement n'ont pas toujours conscience de cette contrainte et introduisent des modifications sur l'interface sans reporter cette modification dans les scripts de tests automatisés,
- les tests sont automatisés trop tôt sur des pans fonctionnels non stables, ce qui en général entraîne des coûts prohibitifs de maintenance des scripts de test,
- l'interface graphique utilise des composants qui se prêtent difficilement à l'automatisation des tests. Ceci est d'autant plus vrai que certains composants proviennent d'éditeurs qui ne fournissent pas le procédé de test de leurs composants,
- le référentiel de données de l'environnement de tests est difficile à contrôler, les données proviennent de copies de la base de production. Les données de tests peuvent donc être perdues ou altérées et impacter le bon fonctionnement des tests de non-régression.

Solution apportée

Valtech s'est focalisé sur :

- **l'étude de compatibilité technique entre l'interface graphique et l'outil d'automatisation** afin d'identifier les composants graphiques problématiques et d'amener les développeurs à concevoir des interfaces testables,
- **l'étude de ROI** destinée à vérifier l'intérêt économique de l'automatisation (plus un test est exécuté, plus l'automatisation est rentable à court terme),
- **la définition de la stratégie d'automatisation** visant à :

- choisir quels tests automatiser pour maximiser la valeur des tests de non-régression par rapport à l'effort investi,
- définir la façon de gérer le référentiel de données.

> *L'automatisation des tests dans un contexte itératif et incrémental n'est pas une option, quel qu'en soit le prix. C'est une obligation .*
> ***Gilles MANTEL, Directeur de projet, VALTECH TECHNOLOGY***

NB : le fait d'itérer implique de rejouer systématiquement certains tests de non régression. Cela rend économiquement intéressante l'automatisation de ces tests, à condition toutefois que l'effort d'automatisation lié à la technologie utilisée ne soit pas rédhibitoire, d'où la nécessité de s'en assurer.

Bénéfices obtenus

Avec l'automatisation, la durée d'un cycle de test utilisateur est de 2 heures.

Les campagnes de tests utilisateurs sont progressivement passées d'une fréquence trimestrielle à une fréquence quotidienne. La durée d'un cycle de tests utilisateurs a été réduite de 1 semaine à 2 heures.

La notion de "campagnes de tests" perd progressivement de son sens pour laisser place à la notion de "tests en continu".

L'essentiel à retenir

Les tests manuels peuvent être comparés à une force de frottement : plus la vitesse et la pression s'accentuent sur un corps en mouvement, plus importants les frottements sont, réduisant l'efficacité des efforts concédés pour accélérer le mouvement.

3.7 TDD au cœur des développements

Contexte

Le contexte s'annonce a priori défavorable, voire hostile à l'agilité : la méthodologie interne prône le cycle en V, totalement ancrée dans la culture industrielle de l'entreprise.

Ce projet industriel de 3 homme.ans dont il s'agit concerne le développement d'une application java "stand alone" avec une interface Swing et diverses autres parties écrites en C++.

Enjeux client

L'objectif du client consiste à améliorer la qualité et la sécurité des applications sans augmenter les coûts de développement.

Pratiques mise en œuvre

Valtech a aidé le client à mettre en place :
- une approche de **développement dirigée par les tests** (TDD),
- une démarche **d'automatisation des tests de recette.**

Difficultés rencontrées

Le projet de développement a suivi le cycle classique en V. Valtech est intervenu à partir de la phase de développement (le bas du cycle en V) et a ainsi hérité d'un document de spécification, fruit de plus d'un an de travail.
Il a, dès lors, été impossible de faire réaliser les tests par le client.

Certaines parties de l'application étant développées en Java et d'autres en C++, deux environnements de développement différents et deux outils de tests unitaires ont été utilisés - Eclipse et Visual Studio d'une part, JUnit et CPPUnit d'autre part. Cela a induit un double effort de mise en place des frameworks de tests unitaires.

La couverture des tests unitaires sur la partie C++ s'est avérée difficile à calculer.

Solutions apportées

Test Driven Development

Les parties en C++ étant des librairies utilisées par le code java et JUnit plus faciles à utiliser, le code C++ a majoritairement été testé depuis l'environnement Java avec JUnit. Malgré tout, certains tests ont dû rester dans la partie C++, de manière à garder un feedback court.

Tests de recette automatisés

L'équipe de développement a écrit les tests de recette, au moyen d'une librairie externe uispec4j, permettant, de "scripter" des scénarii d'utilisation de l'interface graphique Swing.
Par ailleurs, un simulateur sous MS Windows a permis de simuler les interactions de l'application avec un équipement propriétaire. L'outil AutoIt a été utilisé pour piloter ce simulateur depuis la suite de tests en java.

Bénéfices obtenus

> *Le projet a été livré le jour prévu, sans augmentation des coûts. L'équipe de développement s'est montrée très flexible vis à vis des diverses modifications de spécification, le tout sans perte de qualité ni apparition de régression.*
>
> ***Etienne CHARIGNON, Consultant senior, Scrum master certified, VALTECH TECHNOLOGY***

L'essentiel à retenir

La mise en œuvre des pratiques Agiles de TDD permet :
- de rester maître de la complexité des développements réalisés,
- de livrer dans les temps et le budget impartis.

4. L'agilité façon « Lean »

Lean et Agilité ne sont pas antinomiques dans le monde du développement logiciel, bien au contraire ! En effet, s'engager dans une démarche d'amélioration continue Lean permet aux maîtrises d'ouvrage (MOA) et aux directions informatiques de concrétiser les bénéfices des méthodes Agiles en conciliant de façon optimale leurs impératifs de qualité et de réactivité.
Elisabeth DUCARRE, Expert LEAN et CMMI, VALTECH TECHNOLOGY

Lean est connu dans le monde automobile par l'expérience Toyota, devenu le premier constructeur automobile, reconnu à la fois pour la qualité et l'innovation de ses produits. Tout le monde s'accorde à reconnaître que ce succès est du à son système de production Lean.

Cette approche vise à la fois :
- à améliorer la qualité et les délais,
- à réduire les coûts en tirant le meilleur parti des ressources tant humaines que matérielles,
- à éviter toute forme de gaspillage.

Le Lean est basé sur les principes fondamentaux suivants :
- déterminer ce qui **crée de la valeur** pour le client,
- identifier le **flux de valeur**,
- le rendre **continu**,
- rechercher la **perfection**.

Ces principes ont été, eux-mêmes, déclinés dans Lean Software Development ([Réf.2]) par les principes suivants :
- éliminer les gaspillages,
- favoriser la connaissance,
- construire la qualité intrinsèque,
- reporter la décision,
- livrer rapidement,
- respecter les personnes,
- optimiser le système dans son ensemble.

Pour répondre à ces principes, il est facile de constater que les méthodes Agiles et les pratiques associées sont très appropriées.

Lean Software Development considère toutes les méthodes Agiles comme valides pour appliquer le « Lean Thinking » au monde du logiciel, et en particulier le déploiement de la méthode Scrum.
Jeff SUTHERLAND, Coinventeur avec Ken Schwaber de la méthode Scrum.

« Favoriser la connaissance », « Reporter la décision », « Livrer rapidement » peuvent se traduire par la segmentation en itérations courtes des projets Agiles, ce qui apporte aux entreprises clientes la prédictibilité dont elles ont besoin sur la disponibilité des fonctionnalités en cours de développement. Elles peuvent ainsi planifier l'intégration continue des nouvelles fonctions à leur rythme et en fonction des ressources disponibles, sans surcoût ni surcharge de travail qui induisent de nombreuses sources d'erreur (Concept Lean **« Just in time »**).

Un exemple concret de la mise en œuvre du principe « Construire la qualité intrinsèque » (découlant du principe « Rechercher la perfection ») peut se traduire par la technique TDD associée à l'automatisation des tests, qui constitue une approche efficace d'amélioration de la qualité au plus tôt dans le cycle de développement. Cela permet de savoir immédiatement si ce qui est produit possède le bon niveau de qualité et de corriger les défauts au plus tôt.

Nous répondons également au concept Lean **« Stop the line »**, en corrigeant les anomalies dès qu'elles sont détectées, ce qui évite de gaspiller des ressources en poursuivant le développement de l'application sur des bases non fiables à 100%. En effet, la priorité de correction des anomalies doit être supérieure à celle d'ajouter de nouvelles fonctions au logiciel.

Enfin un important concept Lean est le **« Visual Management »**, axe largement soutenu dans les méthodes Agiles par une organisation de l'espace de travail, qui permet à toutes les parties impliquées (développeurs, testeurs, chefs de projets, représentants du métier) de visualiser instantanément l'état d'avancement du projet, et ceci en termes simples.

Grâce aux méthodes Agiles, les développeurs sont en mesure de générer quotidiennement une version compilée de l'application en cours de développement, afin de détecter au plus tôt les anomalies potentielles.

Le résultat de la compilation est visible (par exemple par l'allumage d'une lampe, un avertisseur sonore ou un écran coloré bien situé en fonction du résultat) et permet de réagir au plus vite. L'ensemble des informations de suivi de projet sont affichées grâce à des codes couleurs, sur un ou plusieurs tableaux à la vue de tous (nombre de fonctions testées et validées, nombres d'anomalies en cours de résolution, etc...).

L'ensemble des pratiques citées ci-dessus répondent également au premier principe **« Eliminer les gaspillages »** : réduire les retards (itérations courtes), se concentrer sur les défauts (TDD), mieux comprendre les exigences (TDR), etc...

Enfin, un des principes des méthodes Agiles est de se focaliser sur le produit plutôt que sur la documentation.

En conclusion, ce petit aperçu montre que **les pratiques Agiles sont un support efficace au déploiement d'une démarche Lean.**

Lean met le produit au coeur de son concept, ce qui permet une fois de plus, de faire converger ces deux approches.

L'essentiel à retenir

- Le Lean est basé sur les principes fondamentaux suivants :
  - déterminer ce qui crée de la valeur pour le client,
  - identifier le flux de valeur,
  - le rendre continu,
  - rechercher la perfection.

- Grâce aux méthodes Agiles, les développeurs sont en mesure de générer quotidiennement une version compilée de l'application en cours de développement, afin de détecter au plus tôt les anomalies potentielles.

- Les pratiques Agiles sont un support efficace au déploiement d'une démarche Lean.

5. L'agilité face aux standards Qualité

5.1 Qualité du produit ou des processus : Agilité ou ISO ?

Les méthodes Agiles se décrivent souvent par une approche peu formelle où le focus est mis sur la collaboration entre les personnes et sur le produit à réaliser. A contrario, les méthodes « ISO » sont souvent qualifiées de procédurières voire de contraignantes vis à vis des règles applicables, et plus orientées processus que produit. Ceci semble les opposer, pourtant ...

Le management par la qualité vise à définir des objectifs qualité au niveau de l'entreprise qui se déclinent ensuite par nature d'activité. Il s'agit de planifier et de définir les méthodes nécessaires à la maîtrise des prestations et des produits, et de mesurer le niveau d'efficacité atteint. On traduit souvent cette approche par le concept de « PDCA » (Plan, Do, Check, Act).

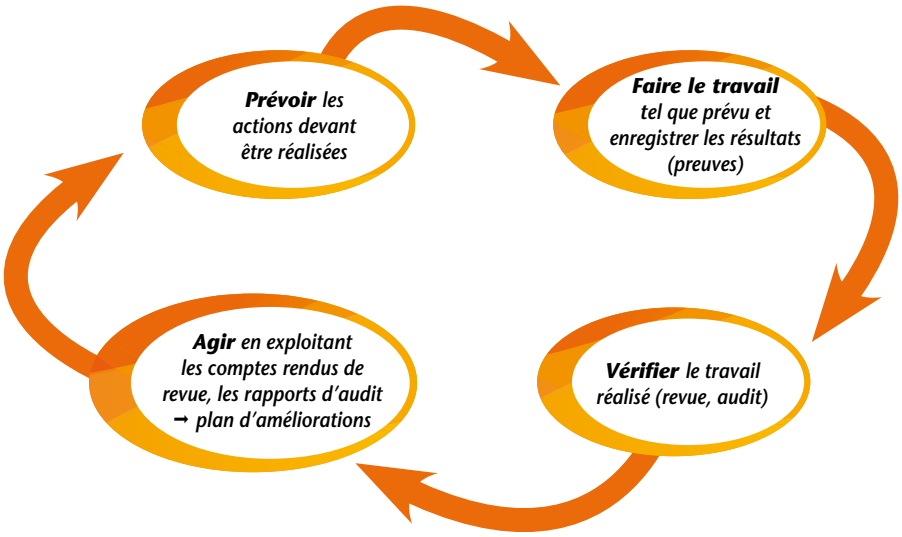


Figure 8 *Cycle de Deming PDCA (Plan, Do, Check, Act) (Source : Valtech Technology).*

Ce mode de management est assez proche de la logique d'itération en Agile avec notamment sa planification des itérations, ses bilans d'itération pour en mesurer les résultats et ses rétrospectives pour décider des améliorations à réaliser dans les itérations futures. **« Agilité » et « ISO » se rejoignent donc sur des valeurs communes de planification opérationnelle et d'amélioration continue.**

Ensuite, la dimension majeure de l'ISO 9001 est son approche processus qui vise à maîtriser l'ensemble des activités d'un organisme en cohérence avec la politique qualité de sa Direction et les exigences métier et réglementaires pour satisfaire

au mieux les clients. Les méthodes Agiles apportent cette même dimension de maîtrise projet et produits avec également une forte implication des clients.
« Agilité » et « ISO » se retrouvent donc à nouveau sur ces valeurs.

> *Agilité et ISO ne sont pas opposables car elle ne se contraignent pas et concourent même à rendre les pratiques projet plus efficaces au bénéfice des produits livrés et de la satisfaction client.*
> ***Hubert GILLON, Delivery Manager, VALTECH TECHNOLOGY***

5.2 Mettre de l'agilité dans une démarche CMMI

Qui a dit « inconciliable » ?

On a souvent tendance à opposer Agilité et CMMI en prétendant qu'être Agile signifie faire l'impasse sur tout processus prédéfini qui constituerait un carcan nuisible à la créativité et la productivité. Les détracteurs des modèles qualité tel que CMMI ont coutume d'affirmer que mettre en place des pratiques Agiles c'est « Assurer » la qualité du produit, alors que se conformer à un modèle tel que le CMMI n'a pour effet que de « Rassurer » ceux qui l'appliquent.

C'est oublier un peu vite que le fondement du modèle CMMI est d'inciter une organisation à s'améliorer de façon continue en structurant cet effort autour d'un ensemble de processus et avec une approche progressive par niveau de maturité. Le cycle d'amélioration continue IDEAL (et plus généralement toute démarche de type PDCA) sous-jacent au modèle CMMI se conforme donc bien au paradigme d'itération prôné par les méthodes Agiles.

Par ailleurs, **CMMI impose des objectifs (« generic goals » et « specific goals ») visant à garantir que la qualité des produits réalisés ne dépende pas de l'héroïsme des équipes mais de l'efficacité de ses processus.** Pour atteindre ces objectifs, CMMI propose un certain nombre de pratiques qui sont des recommandations sur le « quoi faire », et non sur le « comment faire ».
Il y a donc toute liberté à mettre en œuvre des pratiques Agiles pour atteindre ces objectifs.

Des pratiques CMMI Agiles ?

Divers travaux ont cherché à apparier pratiques du CMMI et pratiques Agiles sans montrer d'incompatibilité notoire. Parmi les points d'achoppement potentiels, on trouve par exemple la « traçabilité bidirectionnelle ». Une idée reçue est que, pour être conforme au CMMI, il soit nécessaire d'assurer une traçabilité totale entre les exigences, le produit final et tous les produits intermédiaires du travail. Pour cela, des outils industriels tels que Doors et Reqtify, s'avèrent indispensables.

Néanmoins, **il est tout à fait possible d'être conforme au CMMI** en assurant cette traçabilité uniquement entre les exigences et le produit final grâce…

- Aux outils de gestion de configuration, dès lors qu'ils :
- intègrent une gestion des demandes de changement : native (®IBM Rational UCM avec ClearCase et ClearQuest ou la nouvelle offre Team Concert), ou via une interface (Subversion + Trac par exemple),
- offrent un pont direct avec la gestion des exigences (®Serena Dimensions CM+RM).
- A l'emploi de techniques et d'outils de Test Driven Requirements (une fixture dans Fitnesse ou Greenpepper).

Qu'est-ce qu'un « Référentiel Qualité » Agile ?

Tout d'abord, rappelons que l'« Assurance Qualité » au sens du CMMI couvre à la fois les processus et le produit. Concentrons-nous ici sur le volet « processus ».
L'« Assurance Qualité » consiste à s'assurer que l'organisation se conforme systématiquement aux dispositions du « Référentiel Qualité », autrement dit au cadre méthodologique et aux procédures afférentes.

Les méthodes Agiles introduisent un certain nombre de pratiques dont le succès passe par une application rigoureuse, comme par exemple les différentes « cérémonies Scrum » : planning de l'itération, réunions d'équipe quotidiennes, revue d'itération, rétrospective. Le « contrôle qualité » incombe au Scrum Master qui reste néanmoins avant tout un facilitateur dans la mise en place et l'adoption des différentes pratiques Agiles. Sa sensibilité « processus » se double donc généralement de compétences techniques pour aider à la mise en place des outils, support de ces pratiques (Test Driven Requirements, Intégration Continue...).

> *Le référentiel qualité d'un projet Agile peut tenir dans un simple classeur et non, remplir quelques armoires de dossiers poussiéreux et jaunis mais en parfait état parce que personne n'ose jamais y toucher.*
> ***Stéphane LABATI, Expert CMMI, VALTECH TECHNOLOGY***

Le référentiel qualité.

- Il identifie les diverses **méthodes et pratiques appliquées** sans pour autant les re-détailler. Il suffit pour cela de se référer à l'**abondante bibliographie** qui existe sur le sujet. Chaque projet Scrum pourra simplement indiquer la durée de ses itérations puis utiliser les outils communs de gestion de backlog et de suivi des risques.
- Il contient divers **modèles de documents** – « Organisation Agile ne signifie pas zéro documentation » – mais **en nombre réduit.** Le principe est de ne produire que la documentation utile - qui apporte de la valeur - et au bon

... Organisation Agile ne signifie pas zéro documentation ...

moment. Rédiger un volumineux dossier de conception n'offre aucun intérêt si l'on se conforme déjà à des standards d'architecture et de codage. L'utilisation de frameworks peut aider à imposer l'architecture et la rendre transparente. Il est par ailleurs relativement facile d'automatiser la vérification de telles règles avec des outils d'analyse statique. Par contre, il est important de garder trace des réflexions qui ont amené à ces choix d'architecture et de frameworks, afin de transférer la connaissance du projet et éviter ainsi de se reposer plus tard les mêmes questions.

Un vrai référentiel qualité « Agile » :

- décrit un **ensemble de pratiques et de procédures,**
- contient uniquement les **procédures qui nécessitent d'être détaillées** afin d'être réellement exploitables,
- voit sa bonne **application contrôlée en permanence** par les Scrum Masters.

Qui occupe la tour d'ivoire du SEPG ?

Un autre des principes Agiles, prôné en particulier par Scrum, consiste à **responsabiliser totalement l'équipe** en lui conférant le pouvoir d'auto-organisation et d'auto-détermination. Par exemple, ce n'est plus un chef de projet qui affecte les tâches aux membres de l'équipe, mais ceux-ci qui se les approprient, en fonction de leurs compétences techniques ou fonctionnelles.

De même, on attend de chacun, lors des **rétrospectives**, qu'il identifie les points d'amélioration possibles sur les processus en vigueur et fournisse, si possible, les solutions associées.

D'aucun considère qu'en appliquant le CMMI, on se repose sur quelques éminents qualiticiens pour promulguer les bonnes pratiques à respecter. Or ce cénacle, le « Software Engineering Process Group » (SEPG), n'est rien d'autre que le rassemblement temporaire de quelques membres des équipes projets, légitimés par leur expérience, sous réserve évidemment qu'ils aient collecté au préalable les remarques de tous les membres de leurs équipes. Il s'agit donc là encore d'une certaine forme d'auto-détermination.

> *Toute l'équipe projet participe à l'amélioration des pratiques et constitue, à ce titre, le SEPG d'une organisation Agile.*
> ***Stéphane LABATI, Expert CMMI, VALTECH TECHNOLOGY***

Qu'apporte le CMMI au-delà des pratiques Agiles ?

A travers les rétrospectives, la méthode Scrum cherche à améliorer de façon continue ses pratiques, mais **en se focalisant uniquement sur le périmètre du projet, même si certaines améliorations touchent à l'organisation autour du projet.**

Une démarche CMMI vise à faire bénéficier de ces améliorations l'ensemble de l'organisation et pas seulement le projet qui les a suscitées. De plus, généraliser les améliorations à l'organisation entière apportera globalement plus de valeur que d'optimiser localement un projet.

Finalement, mettre en place des pratiques Agiles dans le cadre d'une démarche CMMI sert les objectifs d'amélioration continue du Lean Thinking, tout en préservant l'organisation des gaspillages liés à une sur-formalisation des processus.

L'essentiel à retenir

- « Agilité » et « ISO » se rejoignent sur des valeurs communes de planification opérationnelle et d'amélioration continue.

- CMMI impose des objectifs visant à garantir que la qualité des produits réalisés ne dépend pas de l'héroïsme des équipes mais de l'efficacité de ses processus. Pour atteindre ces objectifs, CMMI propose un certain nombre de pratiques qui sont des recommandations sur le « quoi faire », et non sur le « comment faire ».

- L'« Assurance Qualité » consiste à s'assurer que l'organisation se conforme systématiquement aux dispositions du « Référentiel Qualité », tandis que le « Contrôle qualité » incombe au Scrum Master qui reste avant tout un facilitateur dans la mise en place et l'adoption des différentes pratiques Agiles.

- Un vrai référentiel qualité « Agile » :
  - décrit un ensemble de pratiques et de procédures,
  - contient uniquement les procédures qui nécessitent d'être détaillées,
  - voit sa bonne application contrôlée en permanence par les Scrum Masters.

6. La contractualisation Agile, une affaire de bon sens

Inspirés des méthodes Agiles, nos contrats Agiles visent à construire un climat de confiance durable entre le client et le fournisseur, sur la base de trois engagements pris en commun :

- celui de collaborer pour atteindre l'objectif commun de livraison du logiciel attendu,
- celui d'être transparent sur les capacités, les performances et les difficultés rencontrées,
- celui de s'adapter aux changements métier du client, induits par un marché de plus en plus évolutif.

> *Là où le contrat au forfait fixe les objectifs et les modalités d'application, le contrat Agile définit les objectifs et les modalités de collaboration.*
> ***Nathalie LOPEZ, Directeur général adjoint, VALTECH TECHNOLOGY***

6.1 A quoi sert un contrat ?

Le contrat est un outil indispensable pour la conduite des projets informatiques. Entre un client et un fournisseur, il sert :

- à partager les risques,
- à se protéger des tentatives de l'un pour exploiter l'autre,
- à définir l'objectif visé, les exigences fonctionnelles, technologiques et de pilotage.

Dans de nombreux cas, cet outil est aussi à double tranchant, notamment lorsqu'il s'agit d'un contrat complet ou rigide qui cherche à anticiper de manière définitive, sans qu'on ait besoin d'y revenir, tous les scénarii possibles du projet et les réponses appropriées.

Typique des contrats de projets informatiques au forfait, cette approche se résume à essayer de **fixer par avance les délais, les coûts, le périmètre fonctionnel et le niveau de qualité du projet.** Tout changement de l'une de ces données impose une renégociation contractuelle qui détourne les intervenants de l'objectif opérationnel et induit alors des tensions entre le client et le prestataire. Le projet au forfait constitue donc un frein au changement et ne permet pas au client de s'adapter facilement aux besoins du marché, ce qui est pourtant vital pour lui.

Plus important encore, le contrat au forfait retire à la fois au client et au fournisseur toute possibilité réelle de pilotage, c'est à dire d'ajustement en fonction de la réalité rencontrée sur le terrain. De façon assez paradoxale, c'est le contrat qui est aux commandes, et qui condamne aussi bien le client que le fournisseur à une relation insatisfaisante, voire à l'échec.

6.2 Une nécessaire révolution des mentalités

> *La contractualisation Agile prend le contre-pied de cette approche : au lieu de chercher à tout prévoir dès le départ, le contrat Agile vise à établir et à maintenir un dialogue constant au service du projet et tout au long de la vie de celui-ci.*
> ***Nathalie LOPEZ, Directeur général adjoint, VALTECH TECHNOLOGY***

Là où le contrat au forfait strict se révèle être un obstacle, le contrat Agile sait n'être qu'un moyen d'atteindre l'objectif principal d'un projet, livrer aux utilisateurs finaux l'application qu'ils attendent. La mise en œuvre d'un contrat Agile passe ainsi par une révolution des mentalités, aussi bien côté client que côté fournisseur.

L'un et l'autre ont à apprendre à travailler ensemble, à créer et maintenir un climat de confiance par un dialogue transparent et constant.

6.3 Règles de collaboration client-fournisseur

Un contrat Agile doit déjà être le résultat d'une réelle collaboration, qui se traduit par la formalisation :

- des **pratiques de collaboration** entre le client et son fournisseur,
- des **moyens de restitution** des éléments de pilotage de son projet informatique,
- des **possibilités de réajustement** de ses objectifs en fonction des nouveaux besoins éventuels.

Pour l'entreprise cliente, des seuils d'anomalies très faibles, de l'ordre de 20, se traduiront forcément par un gain de temps ...

Tout d'abord, le client et le fournisseur établissent ensemble la liste des fonctionnalités qui feront l'objet du contrat et la réajuste au fur et à mesure du projet. C'est aussi en commun, qu'ils définissent l'ordre de priorité de chaque fonctionnalité basée sur sa valeur ajoutée métier et qu'ils font une estimation de sa complexité.

La constitution de l'équipe projet est aussi le résultat d'une collaboration. Côté client, elle sera composée au minimum d'un responsable produit. Mais le client doit aussi être pleinement impliqué dans la sélection des ressources côté fournisseur.

C'est également par le dialogue que le client et le fournisseur définissent les indicateurs de bonne marche du projet. Dans ce domaine, la meilleure stratégie consiste à privilégier des indicateurs qualité car la qualité fait levier sur la productivité. **Pour l'entreprise cliente, des seuils d'anomalies très faibles, de l'ordre de 20, se traduiront forcément par un gain de temps** dans toutes les étapes futures de la vie de l'application en développement comme en maintenance.

Enfin, le fournisseur s'engage sur un niveau de qualité adapté au besoin de son client, plutôt que sur une productivité beaucoup plus arbitraire. En effet, la productivité est un critère plus contractuel par nature et plus difficile à mesurer et expose à retomber dans le piège de défiance des contrats classiques au forfait.

6.4 Méthodes et outils permettant de travailler de manière transparente

Travailler en toute transparence vise à créer et maintenir la confiance, ainsi qu'à éviter tout rapport de force déséquilibré. Lorsque le contrat est signé, client et fournisseur s'engagent à travailler ensemble en toute transparence.

Cela nécessite une bonne maîtrise des pratiques Agiles :

- le fournisseur délivre à intervalles réguliers (itérations de 2 à 4 semaines) un jeu de fonctionnalités exploitables par le client,
- à chaque itération, client et fournisseur choisissent en commun le périmètre de livraison, en tenant compte des priorités métier,
- les indicateurs et outils d'avancement sont partagés et permettent d'ajuster le niveau de qualité et d'acceptabilité des livrables : l'effet boîte noire est ainsi supprimé,
- le client voit son produit se construire progressivement à chaque itération en ayant conscience des risques et des difficultés rencontrées, mais aussi en mesurant l'efficacité des équipes.

Certains contrats Agiles subdivisent le projet en itérations courtes de 2 à 4 semaines, qui font l'objet de mini-forfaits.

Mini-forfaits de 2 à 4 semaines

Chaque itération aboutit à la livraison au client d'une ou de plusieurs fonctionnalités exploitables et pouvant faire l'objet d'une recette partielle. Ce n'est qu'à l'acceptation de ce livrable que le fournisseur sera rémunéré pour le travail effectué durant l'itération.

La contractualisation Agile préserve ainsi la relation client-fournisseur de tout rapport de force. A chaque fin d'itération, les deux parties analysent ensemble la qualité et la vitesse d'exécution du projet. Puis elles évaluent en toute transparence leur capacité à atteindre l'objectif fixé pour l'itération suivante.

Le contrat Agile fait ainsi passer la relation client-fournisseur d'un mode « perdant-perdant » dans lequel le client est insatisfait et le fournisseur peine à rentabiliser sa contribution, à un mode véritablement « gagnant-gagnant ».

6.5 Flexibilité pour tenir compte des changements fonctionnels

Dans un contrat Agile, le client est d'abord libre de changer d'avis, ou plus exactement de faire évoluer le périmètre fonctionnel selon son besoin ou pour saisir une opportunité technologique.

Ni le client, ni le fournisseur ne reste prisonnier d'un cahier des charges qui peut être très vite dépassé. Tout au long du projet, le client peut intégrer de nouveaux besoins fonctionnels et supprimer des fonctionnalités potentiellement inutiles.

Le fournisseur s'engage sur sa capacité et sa réactivité à prendre en compte les changements en se basant sur des pratiques d'ingénierie appropriées : usine logicielle, architecture et suivi de projet Agile.

Les impacts opérationnels et financiers de chaque évolution sont partagés de manière à permettre au client de faire des choix en termes métier et financier.

6.6 Réalisme du contrat Agile

Outre qu'il permet au client de rester maître de son projet de bout en bout, le contrat Agile présente le double avantage de renforcer la confiance au fur et à mesure des itérations livrées et acceptées, et de maintenir le niveau de qualité attendu, la rémunération du fournisseur étant directement liée à la satisfaction du client.
Le contrat Agile est beaucoup plus réaliste, car à chaque itération, client et fournisseurs peuvent tenir compte des progrès réalisés et des difficultés rencontrées pour réévaluer sans attendre l'adéquation des ressources et des objectifs.

Le contrat Agile repose sur un triple engagement mutuel du client et du fournisseur : collaboration, visibilité et flexibilité qui le rend particulièrement adapté à la conduite de projets offshore et multisite.

Nathalie LOPEZ, Directeur général adjoint, VALTECH TECHNOLOGY

En effet, plus les équipes appelées à collaborer sont éloignées, plus grande est l'importance de maintenir le dialogue et la confiance. Le fournisseur s'engage, mais la visibilité sur l'ensemble du projet lui permet de conserver l'autonomie qui lui est nécessaire pour garantir la qualité et la rentabilité de sa contribution. Quant au client, il reste à tout moment le seul maître de son projet, sans craindre des rapports de force déséquilibrés avec ses fournisseurs.

L'essentiel à retenir

- Le contrat est un outil indispensable pour la conduite des projets informatiques.
- L'approche contractuelle forfaitaire se résume à essayer de fixer par avance les délais, les coûts, le périmètre fonctionnel et le niveau de qualité du projet.
- Avec la contractualisation Agile, le client et le fournisseur apprennent à travailler ensemble, à créer et maintenir un climat de confiance par un dialogue transparent et constant.
- Le contrat Agile formalise :
  - les pratiques de collaboration entre le client et son fournisseur,
  - les moyens de restitution des éléments de pilotage projet informatique,
  - les possibilités de réajustement des objectifs en fonction des nouveaux besoins éventuels.
- Dans un contrat Agile, le client est libre de faire évoluer le périmètre fonctionnel selon son besoin ou de saisir une opportunité technologique.
- Le fournisseur Agile s'engage sur sa capacité et sa réactivité à prendre en compte des changements.
- Les impacts opérationnels et financiers de chaque évolution sont partagés.
- Le contrat Agile est beaucoup plus réaliste car à chaque itération, client et fournisseurs peuvent tenir compte des progrès réalisés et des difficultés rencontrées.

ZOOM • Pratiques Agiles

Agilité n'est pas synonyme de désordre, au contraire, le monde de l'agilité comprend un certain nombre de méthodes très formalisées tel que Scrum ou Extreme Programming (XP) pour ne citer que les plus connues.

Voici un schéma qui est inspiré de la représentation en 3 niveaux de Ron JEFFRIES (un des fondateurs de l'Extreme Programming) agrémenté d'autres pratiques Agiles que nous utilisons régulièrement chez Valtech :

- le cercle extérieur contient les pratiques « liées au » client,
- le cercle intermédiaire contient les pratiques de management,
- le cercle intérieur contient les pratiques de développement.

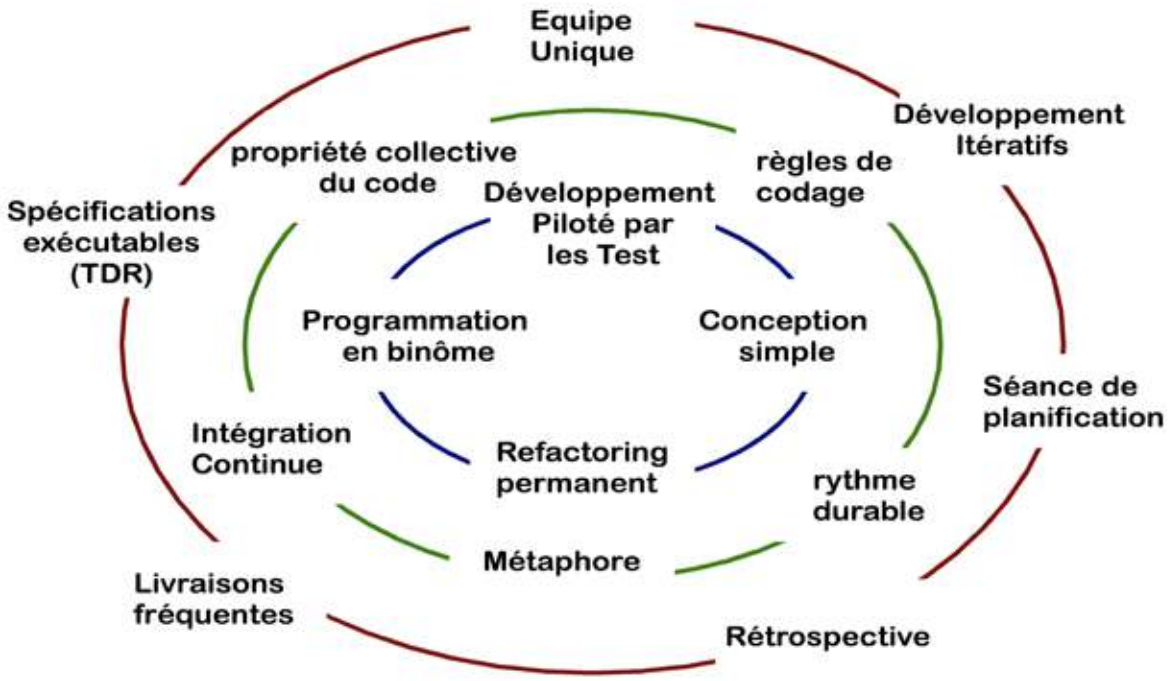


Figure 9 ***Les pratiques Agiles issues de XP et Scrum (Source Valtech Technology).***

Cercle Client

Développement Itératif : l'activité de développement est organisée en cycle dont la durée est fixée une fois pour toute pour le projet. On appelle ces cycles des itérations ou sprints. Ils définissent la pulsation cardiaque du projet.

Equipe Unique (whole team) : il s'agit ici de casser le modèle cloisonné Spécificateur-Développeur-Testeur. Les participants d'un projet de développement forment une seule équipe ou tout le monde participe à la hauteur de ses compétences, et dans le cadre d'un rôle identifié, vers un but commun. Il est important de souligner que cette pratique consiste aussi à rassembler l'équipe sur le plan géographique (dans la mesure du possible).

Livraisons fréquentes (Small Releases) : les livraisons sont effectuées souvent, toutes les 2 à 4 semaines. Cette pratique permet de réduire notamment les difficultés parfois rencontrées au moment de la mise en production.

Rétrospective : à chaque fin d'itération, un temps est prévu pour réfléchir au déroulement de l'itération passée et chercher les moyens d'améliorer l'efficacité pour les itérations futures.

Séance de planification (Planning game) : la structure des séances de planification est très codifiée, avec un certain nombre d'étapes identifiées à réaliser avec tous les membres de l'équipe. Ces séances de planification reviennent cycliquement en début de chaque itération, définissant ainsi les spécifications de l'application à produire petit à petit, plutôt qu'en un seul grand coup de canon en début de projet. C'est lors de ces séances que l'on définit ce que l'on appelle dans Scrum le Sprint Backlog ou Iteration Backlog, à partir de la liste des fonctions / scénarii rassemblés dans ce que l'on appelle, le Product Backlog, et qui suit l'application d'itération en itération.

Tests client automatisés (TDR) (Customer Tests) : on parle aussi de spécifications exécutables ou Test Driven Requirements. Le client définit les critères d'acceptation des scénarii fonctionnels sous forme de cas de test.

Cercle Management

Intégration Continue (Continuous Integration) : le système développé est intégralement assemblé et testé plusieurs fois par jour. Aujourd'hui, cette pratique est grandement facilitée par des outils dédiés (Hudson, CruiseControl…).

Métaphore (Metaphore) : l'équipe doit rechercher et utiliser une analogie comme modélisation du système à développer. Cette technique très répandue dans le monde informatique permet aussi à l'équipe de se construire un vocabulaire commun, notamment pour la communication entre les profils fonctionnels et techniques.

Propriété collective du code (Collective Code Ownership) : tout le code de l'application est accessible en modification par tous les membres de l'équipe. Il n'y a pas de domaine réservé. Cette pratique qui à l'avantage de résoudre le syndrome de l'autobus (qu'est ce qu'on fait si une personne de l'équipe se fait renverser par un autobus), permet aussi de fluidifier le travail quotidien de l'équipe. Elle suppose d'avoir une communication très développée sur les détails techniques de

réalisation. Les tests unitaires intensifs et la programmation en binôme soutiennent cette pratique de manière significative.

Règles de codage (Coding Standard) : l'équipe se fixe des règles de codage de manière à ce que le code soit homogène et facilement lisible par tous.

Rythme durable (Sustainable Pace) : cette pratique initialement intitulée par les américains "40 heures par semaine" qui n'a jamais signifié grand choses dans la culture française, recommande de ne pas faire d'heures supplémentaires plus de deux semaines de suite. Les membres de l'équipe doivent être en forme pour donner le meilleur d'eux-mêmes. La généralisation des heures supplémentaires est le fléau des équipes mal organisées. L'optimisation des processus apportée par ce type de méthode permet d'améliorer notablement la productivité sans augmenter la charge de travail.

Cercle Développement

Conception Simple (Simple Design) : à tout moment, le design de l'application est le plus simple possible qui puisse répondre aux exigences rencontrées jusque là. La simplicité ne sous-entend pas de prendre des raccourcis sur la qualité. Le code doit être concis, modulaire, cohérent, lisible et doit passer tous les tests.

Développement Piloté par les Test (TDD) (Test-Driven Development) : l'activité de programmation suit le processus suivant : Ecrire un test -> Ecrire le code le plus simple qui puisse compiler -> refactorer pour introduire l'abstraction nécessaire et éliminer d'éventuelles duplications.

Programmation en binôme (Pair Programming) : elle se caractérise par le fait que le code est écrit par deux personnes, un pilote et un co-pilote. Les rôles au sein du binôme et les binômes eux même changent régulièrement ce qui permet à l'équipe d'avoir une meilleure connaissance du code de l'application.

Refactoring permanent (Mercyless Refactoring) : pratique de développement qui consiste à améliorer le code sans en changer son comportement.

Glossaire

CMMI Capability Maturity Model Integration

DSI Direction des Systèmes d'Information

MOA Maîtrise d'Ouvrage

MOE Maître d'œuvre

PDCA Plan Do Check Act

PMD Outil d'analyse des flux de données et de recherche des copier/coller, du code mort et des constructions complexes en Java.

ROI Return On Investment (retour sur investissement)

RSA IBM-Rational Software Architect

TDD Test Driven Development

TDR Test Driven Requirements

Documents de référence

RÉF.	DOCUMENT	AUTEUR	EDITEUR	EDITION
Réf. 1	Agile Software Development with Scrum	Ken Schwaber	Pearson Education	2008
Réf. 2	Implementing Lean Software Development, From Concept to Cash	Mary and Tom Poppendieck	Addison Wesley	2007
Réf. 3	Gestion de projet : vers les méthodes Agiles	Véronique Messager Rota	Eyrolles	2007
Réf. 4	Agile estimating and planning	Mike Cohn	Prentice Hall	2004
Réf. 5	Gestion de projet – Extreme Programming	Jean-Louis Bénard	Eyrolles	2004
Réf. 6	User stories applied	Mike Cohn	Addison-Wesley Professional	2004
Réf. 7	Test-Driven Development	Kent Beck	Pearson Education	2003
Réf. 8	Agile and Iterative development	Craig Larman	Addison-Wesley Professional	2003

valtech

Nos références

ABN AMRO - Airbus - Alcatel - ATP - Atx - Audi - Axalto - Bech - Bruun - Betfair - BMW - BofA - Brasserie Kronenbourg - Bull - Cap Gemini - Carrefour - Casino - Cofinoga - Continental - Dagens Nyheter - DanBolig - Danisco - Dassault - Dyrup - Ecco - EDS - EMSI - European Commission - Falck Alarm - Fedex Kinko's - Fortum Power & Heat - France Télécom - GFU Softec - Grundfos - Hedra - IBM - Iflex Solutions - JC Decaux - JPCM - Karolinska Institutet - KG Knutsson - Kungliga Tekniska Högskolan - Kuoni Scandinavia - L&G - La Poste - LIFFE - A Little World - London Underground - Louis Vuitton Malletier - LVMH (Kami, Sephora, Vuitton, Dior) - MACIF - MAE - Man Investments - Metro - Mindlance Inc - Mindtree consulting - Nationwide - NESA - Novo Nordisk - Omgeo - Orange - PBS - Pegasus Solutions - Peugeot Citroën Automobile - Predica - Rigshospitalet - Sanef - SG - Siemens - Skanska - Softlab - Sony Ericsson - Statskontoret - Sydbank - Telia Sonera - Thales Avionics - Thyssen - Krupp - T-Mobile - Travelocity - UBS - Unisys - United Biscuits - Vivarte - Voca - Vodafone - Warner Chappel Music Inc…

valtech

France

Paris
Siege social du groupe
80, avenue Marceau
75008 Paris
Tél. : + 33 (0)1 53 57 71 00

Paris, La Défense
Cœur Défense A
92931 Paris la Défense Cedex
Tél. : + 33 (0)1 41 88 23 00

Puteaux
11, quai de Dion Bouton
92800 Puteaux
Tél. : + 33 (0)1 41 44 65 30

Toulouse
Immeuble Tersud – Bâtiment B
5, avenue Marcel Dassault
31500 Toulouse
Tél. : + 33 (0)5 62 47 52 00

Allemagne

Düsseldorf
Bahnstraße 16
40212 Düsseldorf
Tél. : + 49 (0)211 179237-0

Frankfort
Werner-Heisenberg - Straße 2
63263 Neu-Isenburg
Tél. : + 49 (0)6102 88468-0

Hambourg
Kurze Mühren 1 / Spitalerhof
20095 Hamburg
Tél. : + 49 (0)211 179237-0

Munich
Zweigstraße 10
80336 München
Tél. : + 49 (0)89 893242-0

Danemark

Aarhus
Longhøjevj 1, True 30-32
8381 Tilst
Tél. : + 45 32 88 20 00

Copenhague
Kanonbådsvej 10
1437 København K
Tél. : + 45 32 88 20 00

Etats-Unis

Bryan-College Station
700 University Drive East
Suite 104
College Station, TX 77840
Tél. : +1 (979) 595 99 20

Dallas
5080 Spectrum Drive
Suite 700 West
Addison, Texas 75001
Tél. : + 1 (972) 789 12 00

Houston
17240 Huffmeister Road
Cypress, TX 77429
Tél. : +1 (832) 497 10 40

New York
100 Wall Street
Suite 2215
New York, NY 10005
Tél. : + 1 (212) 688 99 00

Raleigh
8601 Six Forks Road
Suite 400
Raleigh, NC 27615
Tél. : + 1 (919) 878 66 90

Inde

Bangalore
30/A, 1st. Main Road
J. P. Nagar 3 rd. Phase
Bangalore - 560 059
Tél. : + 91 80 26 07 90 00

Royaume-Uni

Londres
102 Aldersgate Street
London EC1A 4JK
Tél. : +44 (0)20 70 14 08 00

Manchester
9th Floor
8 Exchange Quay
Manchester M5 3EJ
Tél. : +44 (0)16 18 73 63 00

Cwmbran
Springboard Innovation Centre
Llantarnam Park
Cwmbran
Torfaen NP44 3AW
Tél. : + 44 (0)16 33 64 78 75

Suède

Stockholm
Tegnérgatan 23
111 40 Stockholm
Tél. : + 46 8 615 33 00

Corée

Séoul
19F Gateway Tower
12 DongzaDong
YongsanGu Seoul 140-709
Tél. : + 82 27 27 56 00

www.valtech.com